

PROFESSIONAL TRAINING REPORT

at

**Sathyabama Institute of Science and Technology
(Deemed to be University)**

Submitted in partial fulfillment of the requirements for the award of Bachelor of
Technology in Information Technology

By

Jagadees.M

REG. NO. 39120042



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600119, TAMILNADU

APRIL 2022



SATHYABAMA
I NSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade “A” by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

CHENNAI– 600119

www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of Jagadees.M

(Reg. No: 39120042) who carried out the project entitled “**Recognition of objects with convolutional neural network**” under my supervision from FEBRUARY 2022 to APRIL 2022.

Internal Guide

DR.V.MARIA ANU M.E., Ph.D.

Head of the Department

DR.R.SUBHASHINI M.E., Ph.D.

Submitted for Viva voce Examination held on

Internal Examiner

External Examiner

DECLARATION

I, **Jagadees.M** hereby declare that the project report entitled **Recognition of objects with convolutional neural network** done by me under the guidance of **Dr. V. MARIA ANU** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.



DATE: 28-04-2022

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D, Dean**, School of Computing, **Dr. R. Subhashini, M.E., Ph.D. Head of the Department** of **Information Technology** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.V.MARIA ANU** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



CERTIFICATE OF PARTICIPATION

THIS CERTIFICATE IS PROUDLY PRESENTED TO

Jagadees

participated in "Artificial Intelligence-Personifwy"
from 11th Nov, 2021 to 11th Jan, 2022
and successfully completed the program.

12-Jan-2022

DATE



PAUL MATHEW. I
OVERALL COORDINATOR

T-IITB-2201000599

ABSTRACT

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

In recent years there have been remarkable advances in areas such as machine learning and pattern recognition, both using convolutional neural networks (CNNs). It is mainly due to the increased parallel processing power provided by graphics processing units (GPUs). In this chapter, the reader will understand the details of the state-of-the-art algorithms for object detection in images, namely, faster region convolutional neural network (Faster RCNN), you only look once (YOLO), and single shot multibox detector (SSD). We will present the advantages and disadvantages of each technique from a series of comparative tests. For this, we will use metrics such as accuracy, training difficulty, and characteristics to implement the algorithms. In this chapter, we intend to contribute to a better understanding of the state of the art in machine learning and convolutional networks for solving problems involving computational vision and object detection

TABLE OF CONTENTS		
CHAPTER No.	TITLE	PAGE No
	ABSTRACT	06
	LIST OF FIGURES	09
	LIST OF TABLES	
1	INTRODUCTION	10
2	OBJECT DETECTION TECHNIQUES	11
	2.1 SINGLE SHOT MULTIBOX DETECTOR	11
	2.2 YOU ONLY LOOK ONCE	12
	2.3 FASTER REGION CONVOLUTIONAL NEURAL NETWORK	13
3	CONCEPTS INVOLVED	15
	3.1 TENSOR FLOW	15
	3.2 KERAS	16
	3.2.1 SEQUENTIAL MODEL	
	3.2.2 FUNCTIONAL MODEL	
	3.3 MATPLOTLIB	19
	3.4 SUPERVISED LEARNING	19
	3.5 COMPUTER VISION	21
	3.6 DEEP LEARNING	22
	3.7 CONVOLUTIONAL NEURAL NETWORK	23
4	SYSTEM SPECIFICATION	25
	4.1 HARDWARE SPECIFICATION	25
	4.2 SOFTWARE SPECIFICATION	
5	SYSTEM IMPLEMENTATION	26
	5.1 SYSTEM ARCHITECTURE	26
	5.2 PROJECT IMPLEMENTATION	27
6	RESULTS AND DISCUSSION , PERFORMANCE ANALYSIS	28
7	CONCLUSION	30

REFERENCES	31
APPENDIX A : SNAPSHOTS	32
APPENDIX B : SOURCE CODE	38

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
2.1	SINGLE SHOT MULTIBOX DETECTOR	11
2.2	YOLO	12
2.3	F-RCNN	13
2.3.1	GRAPH OF RCNN	14
2.3.2	RCNN LAYER	14
3.1	DIFF TENSORFLOW	15
3.2	KERAS LAYER	17
3.2.1	SEQUENTIAL MODEL	18
3.2.2	FUNCTIONAL MODEL	18
3.3	MATPLOTLIB	19
3.4	DEEP LEARNING TYPES	20
3.4.1	SUPERVISED LEARNING	21
3.5	COMPUTER VISION	21
3.7	CNN	23
3.7.1	LAYERS OF CNN	24
5.1	SYSTEM ARCHITECTURE	26
5.2	SPLITTING OF DATASET	28

CHAPTER 1

1.INTRODUCTION

There are fascinating problems with computer vision, such as image classification and object detection, both of which are part of an area called object recognition.

For these types of issues, there has been a robust scientific development in the last years, mainly due to the advances of convolutional neural networks, deep learning techniques, and the increase of the parallelism processing power offered by the graphics processing units (GPUs). The image classification problem is the task of assigning to an input image one label from a fixed set of categories.

This classification problem is central within computer vision because, despite its simplicity, there are a wide variety of practical applications and has multiple uses, such as labelling skin cancer images, use of high-resolution images to detect natural disasters such as floods, volcanoes, and severe droughts, noting the impacts and damage caused .

The performance of image classification algorithms crucially relies on the features used to feed them. It means that the progress of image classification techniques using machine learning relied heavily on the engineering of selecting the essential features of the images that make up the database. Thus, obtaining these resources has become a daunting task, resulting in increased complexity and computational cost. Commonly, two independent steps are required for image.

classification, feature extraction, and learning algorithm choice, and this has been widely developed and enhanced using support vector machines (SVMs).

The SVM algorithm, when considered as part of the supervised learning approach, is often used for tasks as classification, regression, and outlier detection. The most attractive feature of this algorithm is that its learning mechanism for multiple objects is simpler to be analyzed mathematically than traditional neural network architecture, thus allowing to complex alterations with known effects on the core features of the algorithm. In essence, an SVM maps the training data to higher-dimensional feature space and constructs a separation hyperplane with maximum margin, producing a nonlinear separation boundary in the input space.

Today, the most robust object classification and detection algorithms use deep learning architectures, with many specialized layers for automating the filtering and feature extraction process. Machine learning algorithms such as linear regression, support vector machines, and decision trees all have its peculiarities in the learning process, but fundamentally they all apply similar steps: make a prediction, receive a correction, and adjust the prediction mechanism based on the correction, at a high level, making it quite similar to how a human learns. Deep learning has appeared bringing a new approach to the problem, which attempted to overcome previous drawbacks by learning abstraction in data following a stratified description paradigm based on a nonlinear transformation. A key advantage of deep learning is its ability to perform semi-supervised or unsupervised feature extraction over massive datasets.

The ability to learn the feature extraction step present in deep learning-based algorithms comes from the extensive use of convolutional neural networks

(ConvNet or CNN). In this context, convolution is a specialized type of linear operation and can be seen as the simple application of a filter to a determined input. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in the input by tweaking the parameters of the convolution. The network can adjust itself to reduce the error and therefore learn the best parameters to extract relevant information on the database.

Many deep neural network (DNN)-based object detectors have been proposed in the last few years. This research investigates the performance of state-of-the-art DNN models of SSD and Faster RCNN applied to a classical detection problem where the algorithms were trained to identify several animals in images; furthermore to exemplify the application in scientific research, the YOLO network was trained to solve the mice tracking problem. The following sections describe the DNN models mentioned earlier in more details.

CHAPTER 2

OBJECT DETECTION TECHNIQUES

2.1 SINGLE SHOT MULTIBOX DETECTOR:

The single shot multibox detector is one of the best detectors in terms of speed and accuracy comprising two main steps, feature map extraction and convolutional filter applications, to detect objects.

The SSD architecture builds on the VGG-16 network, and this choice was made based on the strong performance in high-quality image classification tasks and the popularity of the network in problems where transfer learning is involved

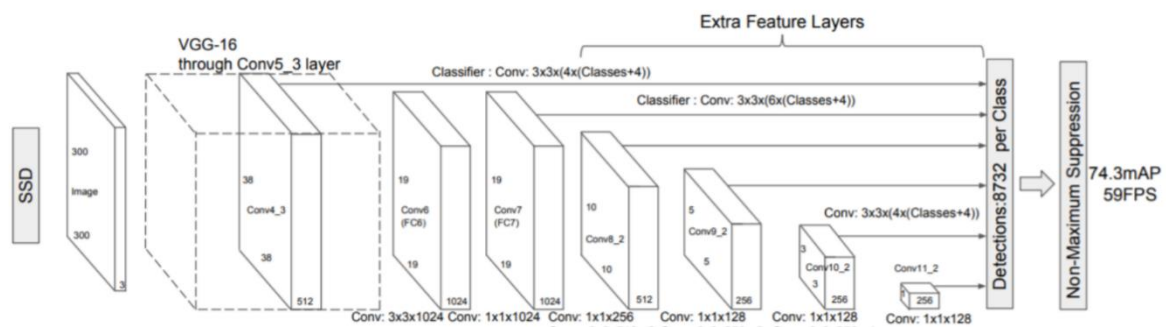


FIG 2.1

Progressively decreasing convolutional layers decreases the feature map size and increase the depth. The deep layers cover larger receptive fields and construct more abstract representations. This is helpful in detecting larger objects. Initial conv layers cover smaller receptive fields and are helpful in detecting smaller objects present in the image.

2.2 YOU ONLY LOOK ONCE:

You only look once is a state-of-the-art object detection algorithm which targets real-time applications, and unlike some of the competitors, it is not a traditional classifier purposed as an object detector.

YOLO works by dividing the input image into a grid of $S \times X \times S$ cells, where each of these cells is responsible for five bounding boxes predictions that describe the rectangle around the object. It also outputs a confidence score, which is a measure of the certainty that an object was enclosed. Therefore the score does not have any relation with the kind of object present in the box, only with the box's shape.

For each predicted bounding box, a class it's also predicted working just like a regular classifier giving resulting in a probability distribution over all the possible classes. The confidence score for the bounding box and the class prediction combines into one final score that specifies the probability for each box includes a specific type of object. Given these design choices, most of the boxes will have low confidence scores, so only the boxes whose final score is beyond a threshold are kept. Eq. states the loss function minimized by the training step in the YOLO

algorithm

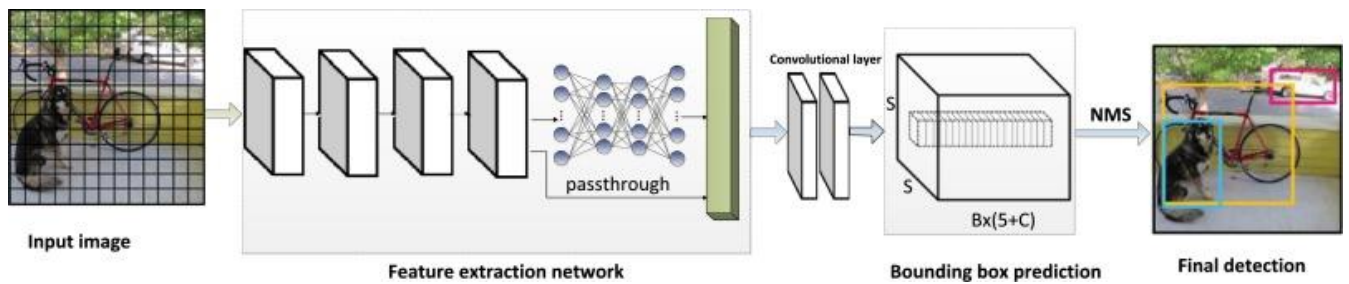


FIG 2.2

YOLO, Also Known as You Only Look Once is one of the most powerful real-time object detector algorithms. It is called that way because unlike previous object detector algorithms, like R-CNN or its upgrade Faster R-CNN it only needs the image (or video) to pass one time through its network.

YOLO is orders of magnitude faster(45 frames per second) than other object detection algorithms. The limitation of YOLO algorithm is that it struggles with small objects within the image, for example it might have difficulties in detecting a flock of birds. This is due to the spatial constraints of the algorithm.

2.3 FASTER REGION CONVOLUTIONAL NEURAL NETWORK:

The faster region convolutional neural network is another state-of-the-art CNN-based deep learning object detection approach. In this architecture, the network takes the provided input image into a convolutional network which provides a convolutional feature map. Instead of using the selective search algorithm to identify the region proposals made in previous iterations, a separate network is used to learn and predict these regions. The predicted region proposals are then reshaped using a region of interest (ROI) pooling layer, which is then used to

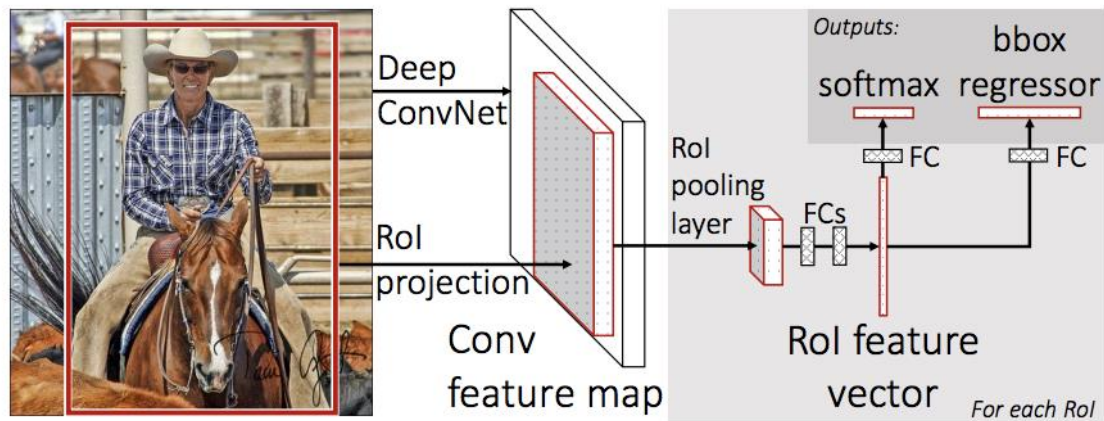


FIG 2.3

The same author of the previous paper(R-CNN) solved some of the drawbacks of R-CNN to build a faster object detection algorithm and it was called Fast R-CNN. The approach is similar to the R-CNN algorithm. But, instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map. From the convolutional feature map, we identify the region of proposals and warp them into squares and by using a RoI pooling layer we reshape them into a fixed size so that it can be fed into a fully connected layer. From the RoI feature vector, we use a softmax layer to predict the class of the proposed region and also the offset values for the bounding box.

The reason “Fast R-CNN” is faster than R-CNN is because you don’t have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image and a feature map is generated from it.

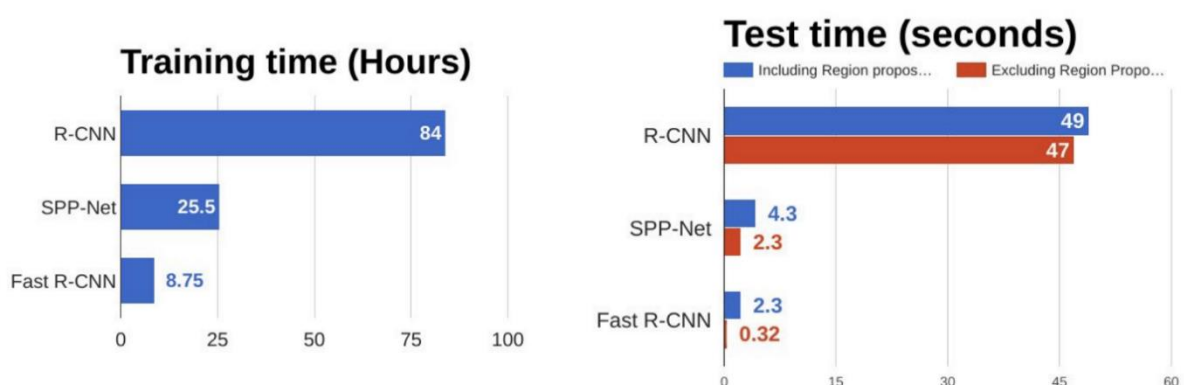


FIG 2.3.1

From the above graphs, you can infer that Fast R-CNN is significantly faster in training and testing sessions over R-CNN. When you look at the performance of Fast R-CNN during testing time, including region proposals slows down the algorithm significantly when

compared to not using region proposals. Therefore, region proposals become bottlenecks in Fast R-CNN algorithm affecting its performance.

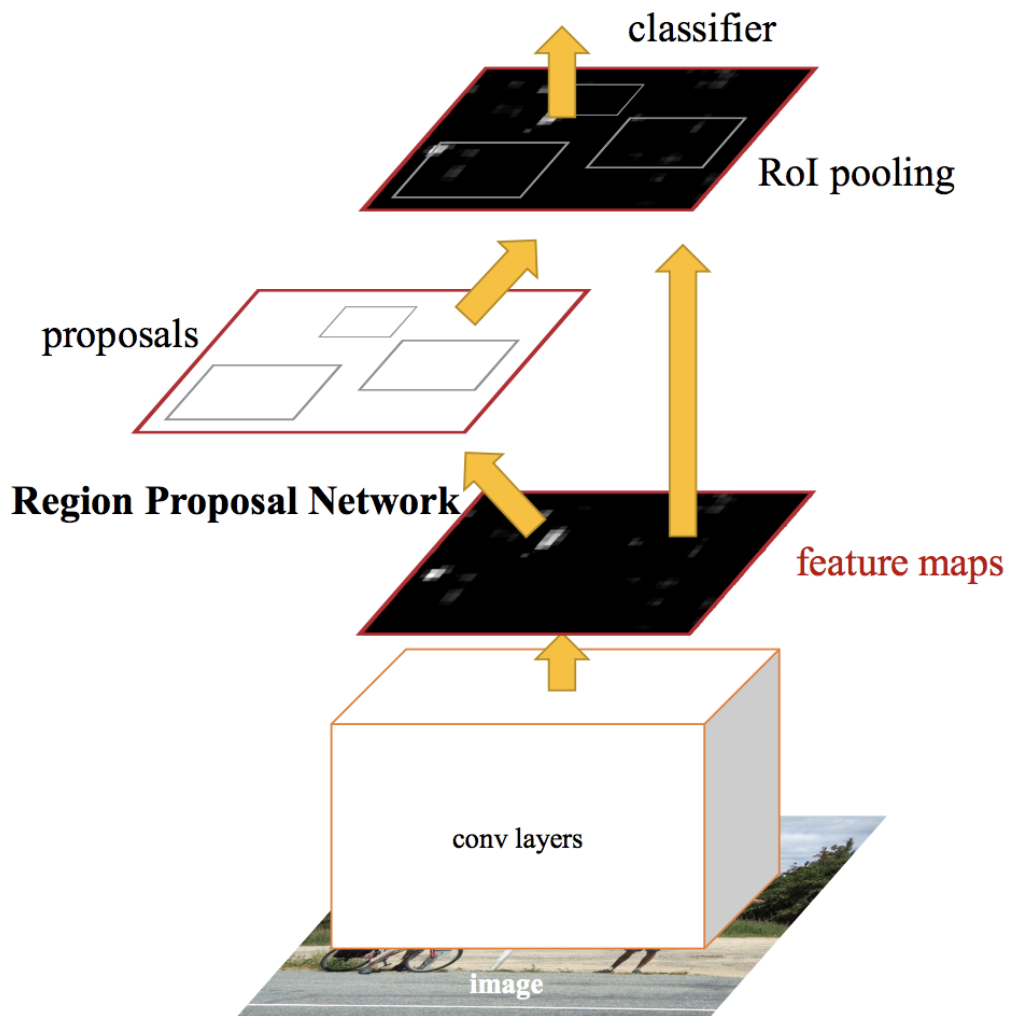


FIG 2.3.2

CHAPTER 3

CONCEPTS INVOLVED

3.1 TENSORFLOW:

TensorFlow is an end-to-end open source platform for machine learning

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML, and gives developers the ability to easily build and deploy ML-powered applications. TensorFlow provides a collection of workflows with intuitive, high-level APIs for both beginners and experts to create machine learning models in numerous languages. Developers have the option to deploy models on a number of platforms such as on servers, in the cloud, on mobile and edge devices, in browsers, and on many other JavaScript platforms. This enables developers to go from model building and training to deployment much more easily.

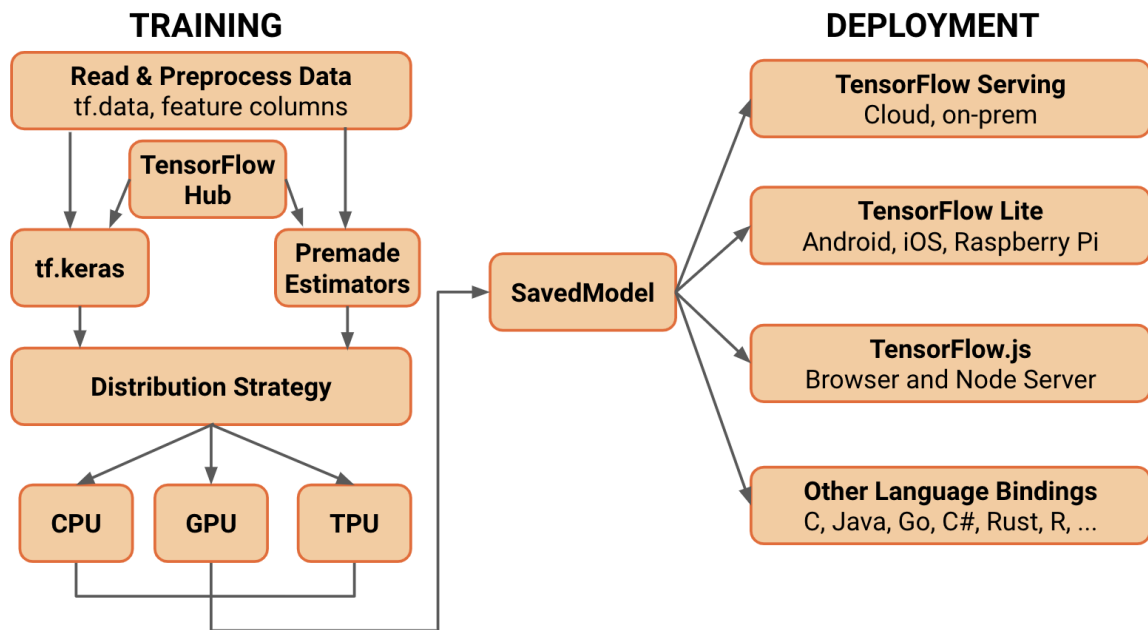


FIG 3.1

- Tensorflow bundles together MACHINE LEARNING and DEEP LEARNING models and algorithms.
- It uses Python as a convenient front-end and runs it efficiently in optimized C++.
- Tensorflow allows developers to create a graph of computations to perform.
- Each node in the graph represents a mathematical operation and each connection represents data. Hence, instead of dealing with low-details like figuring out proper ways to hitch the output of one function to the input of another, the developer can focus on the overall logic of the application.

- The deep learning artificial intelligence research team at Google, Google Brain, in the year 2015 developed TensorFlow for Google's internal use. This Open-Source Software library is used by the research team to perform several important tasks.
- TensorFlow is at present the most popular software library. There are several real-world applications of deep learning that makes TensorFlow popular. Being an Open-Source library for deep learning and machine learning, TensorFlow finds a role to play in text-based applications, image recognition, voice search, and many more. DeepFace, Facebook's image recognition system, uses TensorFlow for image recognition. It is used by Apple's Siri for voice recognition. Every Google app that you use has made good use of TensorFlow to make your experience better.

3.2 KERAS:

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

Keras is:

Simple -- but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.

Flexible -- Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.

Powerful -- Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, and Waymo.

The core data structures of Keras are layers and models. The simplest type of model is the Sequential model, a linear stack of layers. For more complex architectures, you should use the Keras functional API, which allows to build arbitrary graphs of layers, or write models entirely from scratch via subclassing.

Keras is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides

essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2: you can run Keras on TPU or on large clusters of GPUs, and you can export your Keras models to run in the browser or on a mobile device.

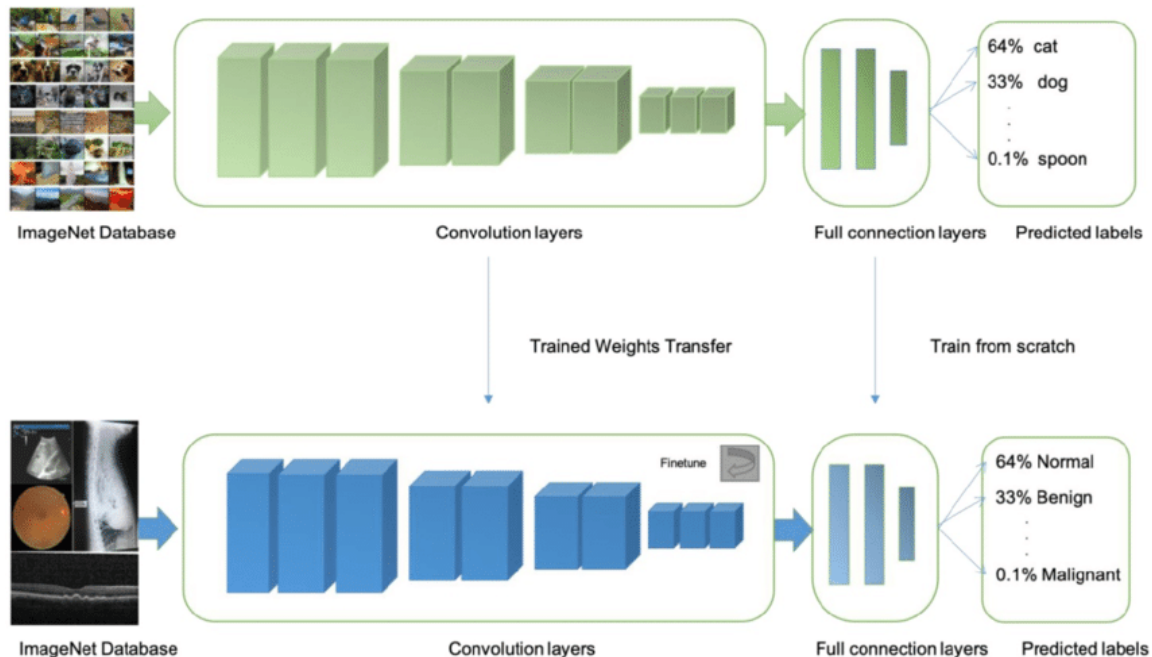


FIG 3.2

3.2.1 SEQUENTIAL MODEL:

A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer. Each layer has weights that correspond to the layer the follows it. We use the 'add()' function to add layers to our model.

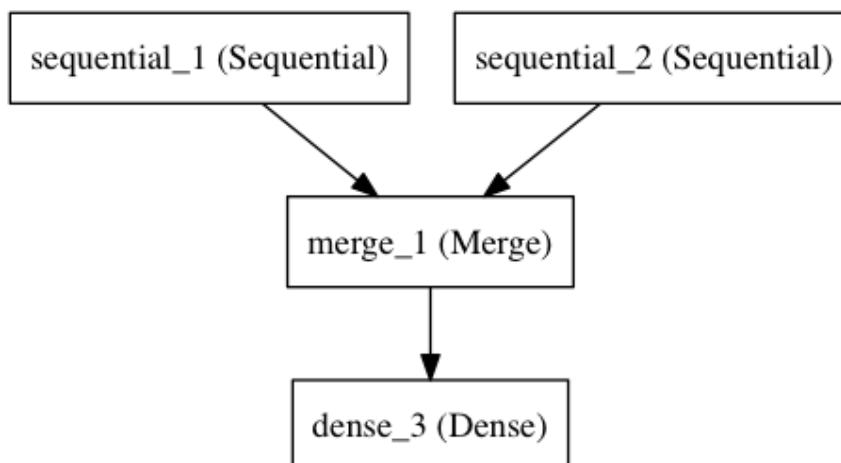


FIG 3.2.1

3.2.2 FUNCTIONAL MODEL:

It is a more flexible API and is more advanced in nature. It is an alternative model against the SequentialModel that enables you to create models in a more complex manner. This model helps you to define multiple numbers of input and output.

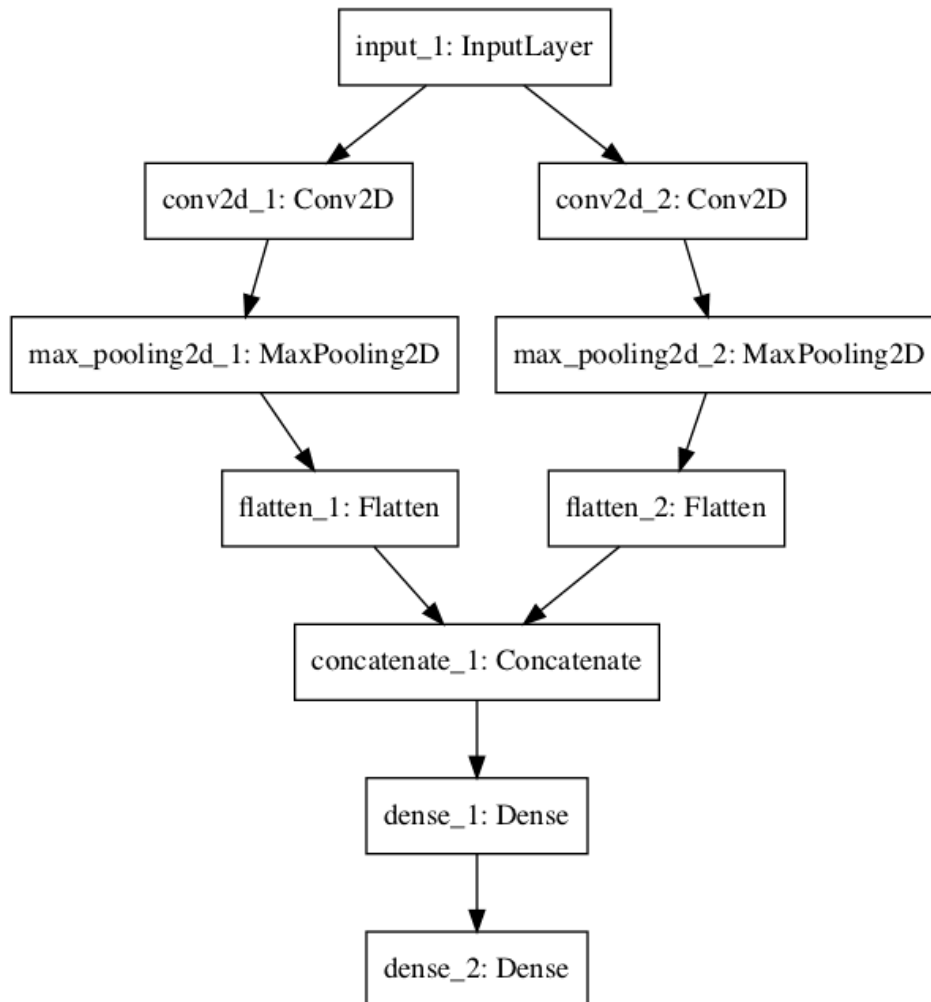


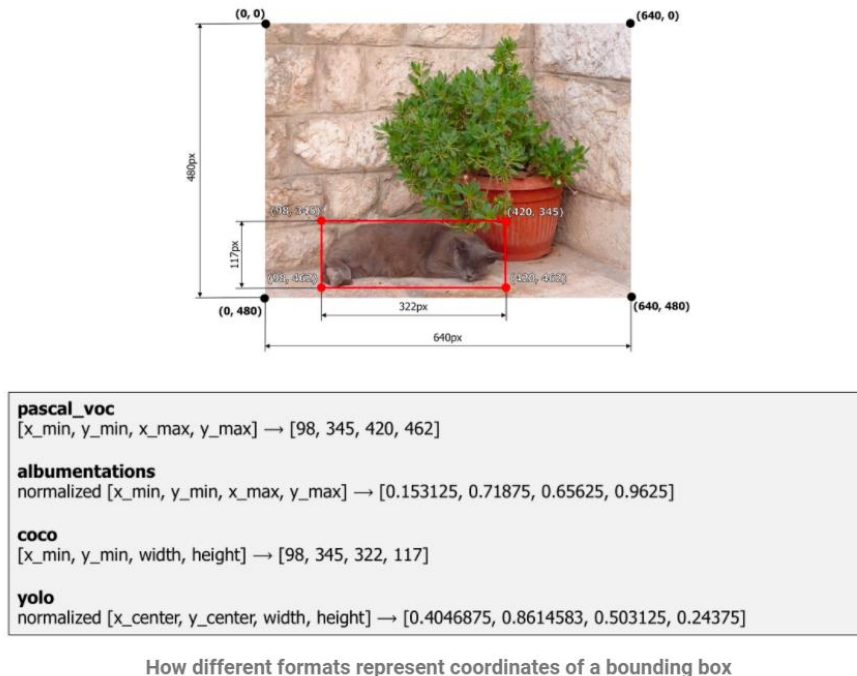
FIG 3.2.2

3.3 MATPLOTLIB:

Matplotlib is a Python 2D plotting library that produces high-quality charts and figures, which helps us visualize extensive data to understand better. Pandas is a handy and useful data-structure tool for analyzing large and complex data. And pyplot function makes some changes to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

One of Matplotlib's most important features is its ability to play well with many operating systems and graphics backends. Matplotlib supports dozens of backends and output types, which means you can count on it to work

regardless of which operating system you are using or which output format you wish.



How different formats represent coordinates of a bounding box

FIG 3.3

3.4 SUPERVISED LEARNING:

Object detection is a supervised machine learning problem, which means you must train your models on labeled examples. Each image in the training dataset must be accompanied with a file that includes the boundaries and classes of the objects it contains.

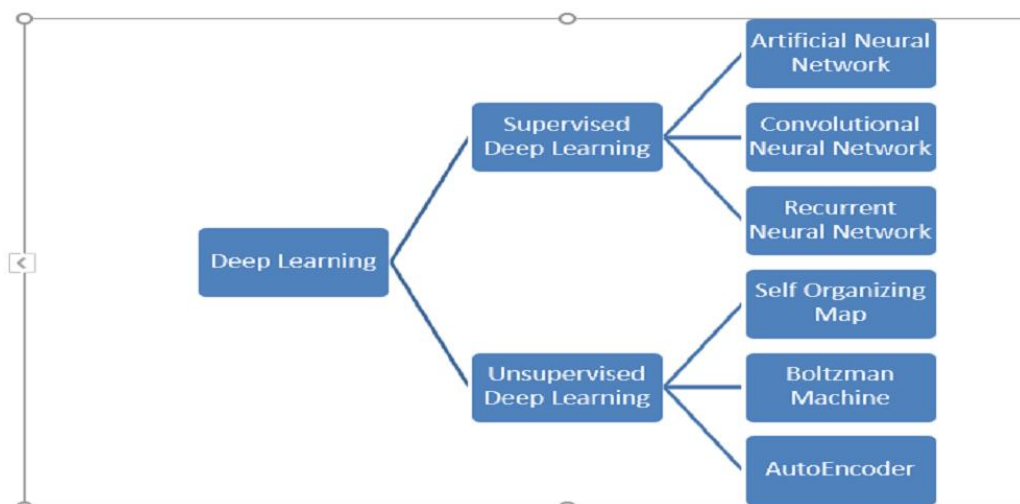


FIG 3.4

Supervised learning is the most common form of machine learning. With supervised learning, a set of examples, the training set, is submitted as input to the system during the training phase. Each input is labeled with a desired output value, in this way the system knows how is the output when input is come. For example consider some experimental observations that can be classified into N different classes. So we have a training set (a sequence of pairs) $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ where x_i is the input and y_i is the class of corresponding output. The training is performed by the minimization of a particular cost function which represent the bind from input x_i and the desired output y_i .

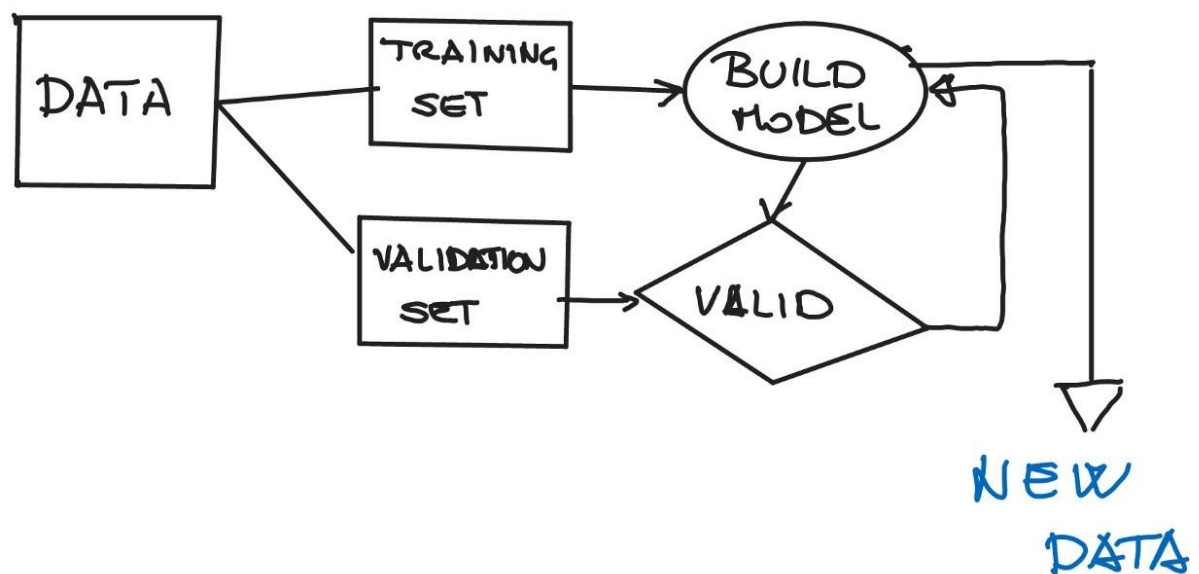


FIG 3.4.1

3.5 COMPUTER VISION:

Computer vision goes under the names of CV or AI vision. It is a field of artificial intelligence that handles image and video analysis. Computer vision systems include a set of AI techniques. The latter allows a computer to have a human-like vision that extracts information from the visual world.

It seeks to automate tasks that human vision can achieve. This involves methods of acquiring, processing, analyzing, and understanding digital images, and extraction of data from the real world to produce information.

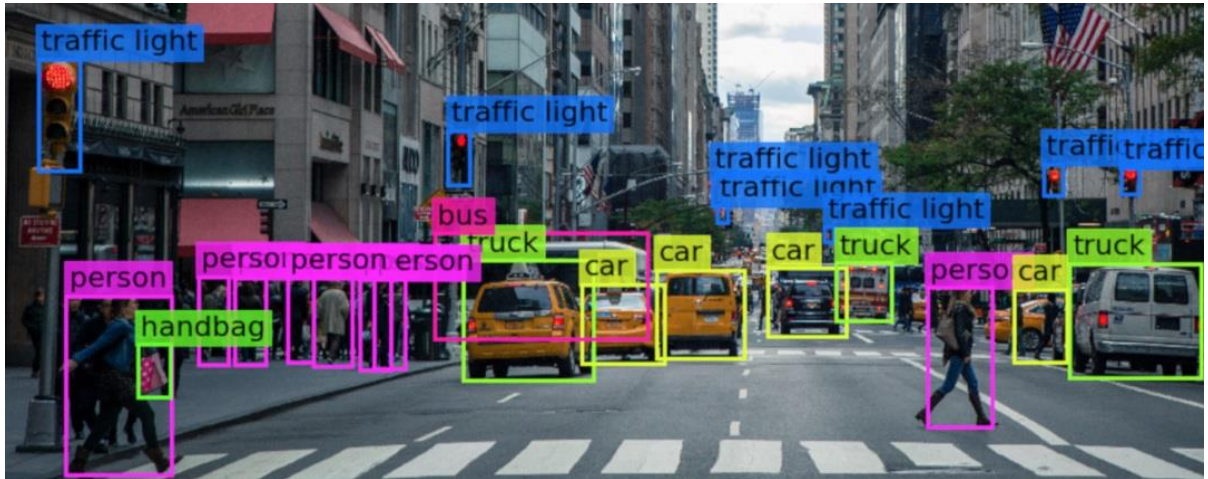


FIG 3.5

Computer vision also plays an important role in facial recognition applications, the technology that enables computers to match images of people's faces to their identities. Computer vision algorithms detect facial features in images and compare them with databases of face profiles. Consumer devices use facial recognition to authenticate the identities of their owners. Social media apps use facial recognition to detect and tag users. Law enforcement agencies also rely on facial recognition technology to identify criminals in video feeds.

Computer vision also plays an important role in augmented and mixed reality, the technology that enables computing devices such as smartphones, tablets and smart glasses to overlay and embed virtual objects on real world imagery. Using computer vision, AR gear detect objects in real world in order to determine the locations on a device's display to place a virtual object. For instance, computer vision algorithms can help AR applications detect planes such as tabletops, walls and floors, a very important part of establishing depth and dimensions and placing virtual objects in physical world.

3.6 DEEP LEARNING:

Deep learning is a type of machine learning and artificial intelligence (AI) that imitates the way humans gain certain types of knowledge. Deep learning is an important element of data science, which includes statistics and predictive modeling. It is extremely beneficial to data scientists who are tasked with collecting, analyzing and interpreting large amounts of data; deep learning makes this process faster and easier.

At its simplest, deep learning can be thought of as a way to automate predictive analytics. While traditional machine learning algorithms are linear, deep learning algorithms are stacked in a hierarchy of increasing complexity and abstraction.

Deep learning neural networks:

A type of advanced machine learning algorithm, known as an artificial neural network, underpins most deep learning models. As a result, deep learning may sometimes be referred to as deep neural learning or deep neural networking.

neural networks, artificial neural networks and feedforward neural networks, and each has benefits for specific use cases. However, they all function in somewhat similar ways -- by feeding data in and letting the model figure out for itself whether it has made the right interpretation or decision about a given data element.

Neural networks involve a trial-and-error process, so they need massive amounts of data on which to train. It's no coincidence neural networks became popular only after most enterprises embraced big data analytics and accumulated large stores of data. Because the model's first few iterations involve somewhat educated guesses on the contents of an image or parts of speech, the data used during the training stage must be labeled so the model can see if its guess was accurate. This means, though many enterprises that use big data have large amounts of data, unstructured data is less helpful. Unstructured data can only be analyzed by a deep learning model once it has been trained and reaches an acceptable level of accuracy, but deep learning models can't train on unstructured data.

3.7 CONVOLUTIONAL NEURAL NETWORK:

CNN is a supervised type of Deep learning, most preferable used in image recognition and computer vision.

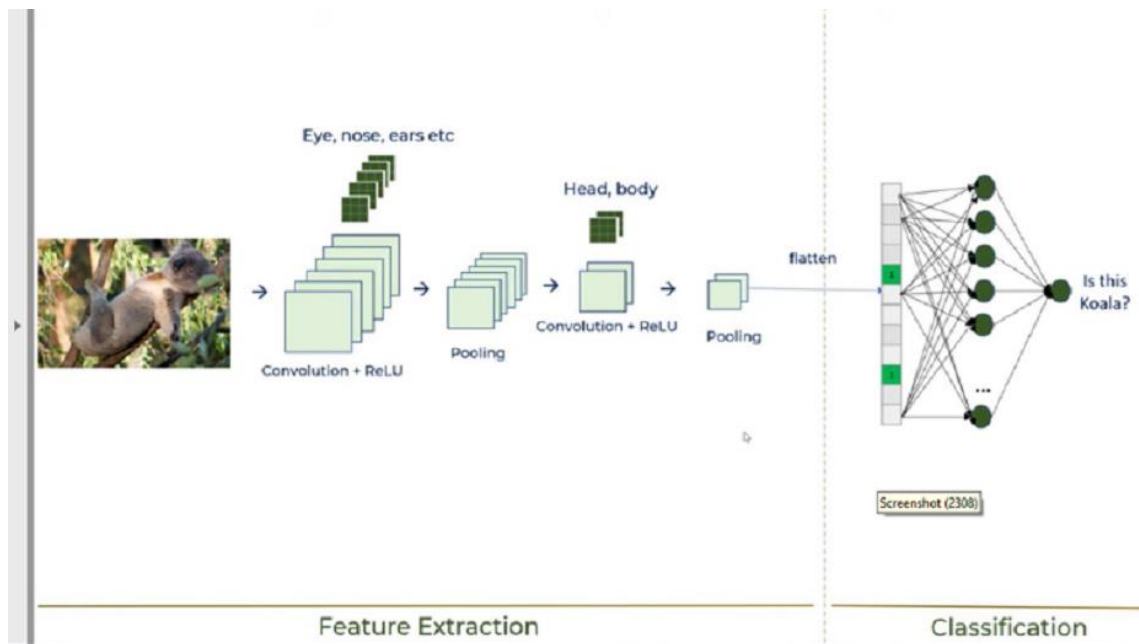


FIG 3.7

CNN has multiple layers that process and extract important features from the image. There are mainly 4 steps to how CNN works

Step : 1 Convolution Operation with Relu Activation Function

The objective of the Convolution operation is to find features in the image using feature detectors to preserve the special relationship between pixels. Relu activation function is used to break linearity and want to increase non-linearity because images are themselves are highly non-linear.

Step: 2 Pooling

Pooling is a down-sampling operation that reduces dimensions and computation, reduces overfitting as there are fewer parameters and the model is tolerant towards variation and distortion.

Step: 3 Flattening

Flattening is used to put pooling output into one dimension matrix before further processing.

Step: 4 Fully Connected Layer

A fully connected layer forms when the flattening output is fed into a neural network which further classifies and recognized images.

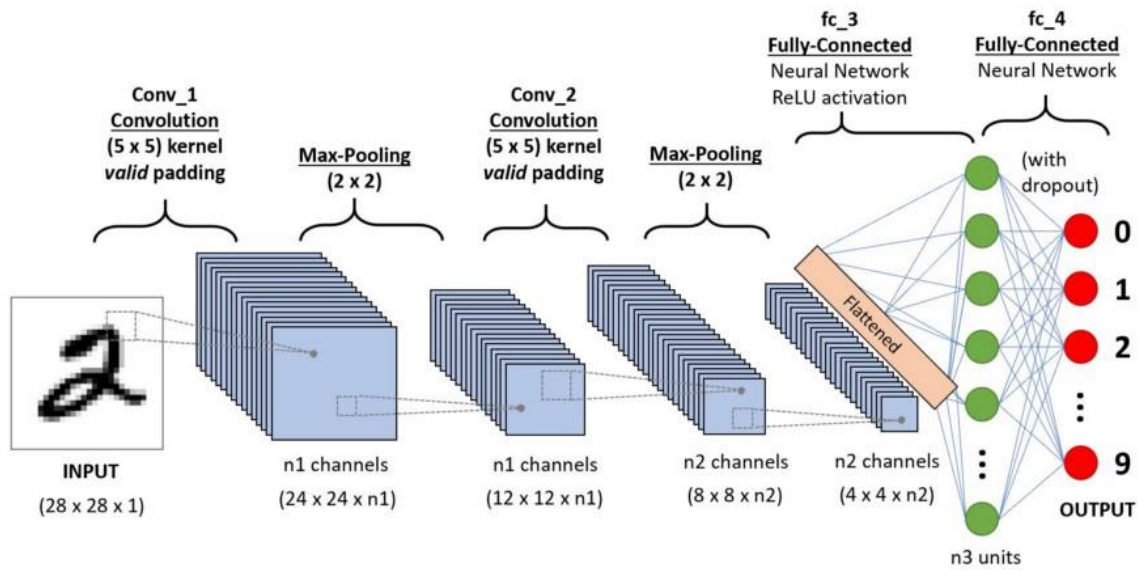


FIG 3.7.1

Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. When you input an image in a ConvNet, each layer generates several activation functions that are passed on to the next layer.

The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc.

CHAPTER 4

SYSTEM SPECIFICATION

4.1 HARDWARE SPECIFICATION:

The algorithms that are shown throughout this paper are designed for a specific target, NVIDIA Jetson TX1. This system-on-a-chip contains a CPU module and a GPU module [9]. The CPU module is a quad-core ARM Cortex-A57 CPU. The GPU module is a Maxwell GPU that has 256 cores, evenly distributed onto two Stream Multiprocessors. It was selected because of its low power consumption which is estimated lower than 20W when fully utilized, yet its high computational capacity per watt. Moreover this system-on-a-chip supports standard and widely used frameworks such as Compute Unified Device Architecture (CUDA). In addition it supports 16-bit floating point operations which may enable to reach a better performance if precision is not the most important aspect.

4.2 SOFTWARE SPECIFICATION:

OPERATING SYSTEM : windows,Mac

LANGUAGE : PYTHON,R-PROGRAMMING,JAVA,JAVA SCRIPT

SOFTWARE : Anaconda,Jupyternotebook,Tensorflow,Python packages

ANACONDA:

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

JUPYTER_NOTEBOOK:

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Pérez and Brian Granger.

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text.

Jupyter Notebook provides you with an easy-to-use, interactive data science environment across many programming languages that doesn't only work as an IDE, but also as a presentation or education tool. It's perfect for those who are just starting out with data science!

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 SYSTEM ARCHITECTURE

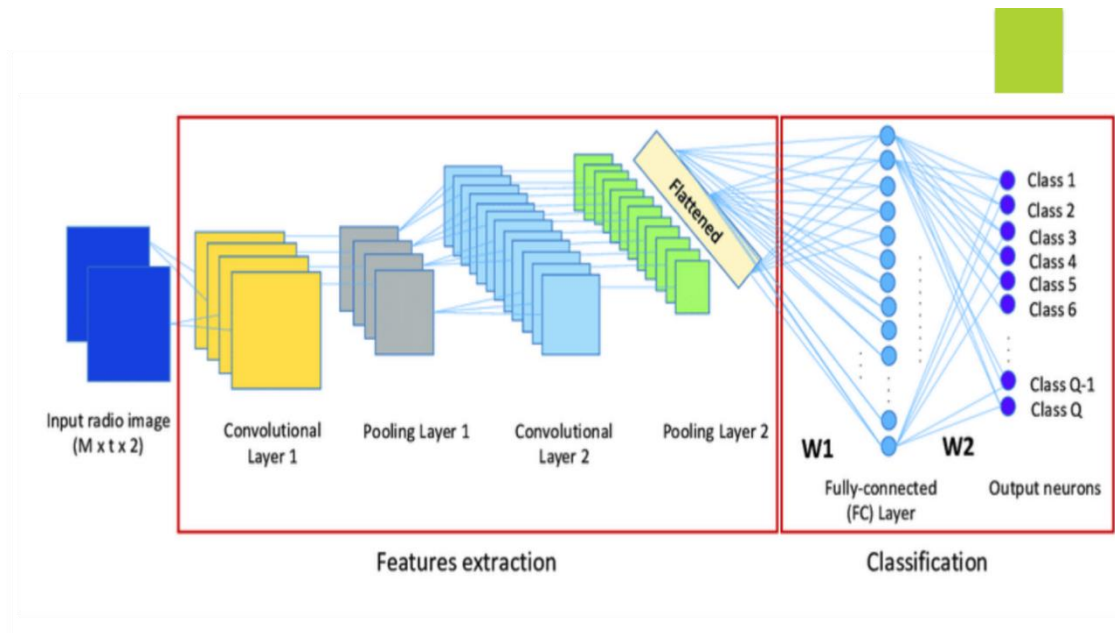


FIG 5.1

STEP 1: CHOOSE A DATASET

STEP 2: PREPARE DATASET FOR TRAINING

STEP 3: CREATE TRAINING DATA

STEP 4: SHUFFLE THE DATASET

STEP 5: ASSIGNING LABELS AND FEATURES

STEP 6: NORMALISING X AND CONERTING LABELS TO CATEGORICAL DATA

STEP 7: SPLIT X AND Y FOR USE IN CNN

STEP 8: DEFINE,COMPILE AND TRAIN THE CNN MODEL

STEP 9: ACCURACY AND SCORE OF MODEL

5.2 PROJECT IMPLEMENTATION:

STEP 1: Import all libraries , packages

STEP 2: Load the data from tensorflow or keras

STEP 3: Split the data

STEP 4: Pre-process the data into the appropriate form

a) Normalize

b) One-hot encoding

STEP 5: Start creating the model

STEP 6: Add all the layers sequentially

STEP 7: Compile,train and validate

STEP 8: Predict and print the predicted labels with the respective objects

LOADING DATASET:

`Tf.keras.datasets.cifar10.load_data()`

Or

`Keras.datasets.cifar10.load_data()`

SPLITTING OF DATASET:

Further the data is splitted into training and testing set based on size of the dataset.

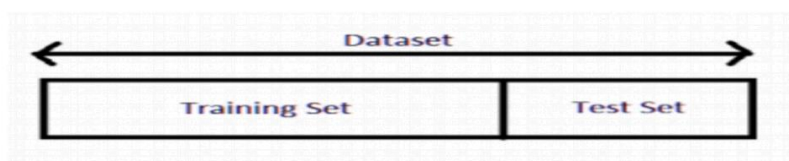
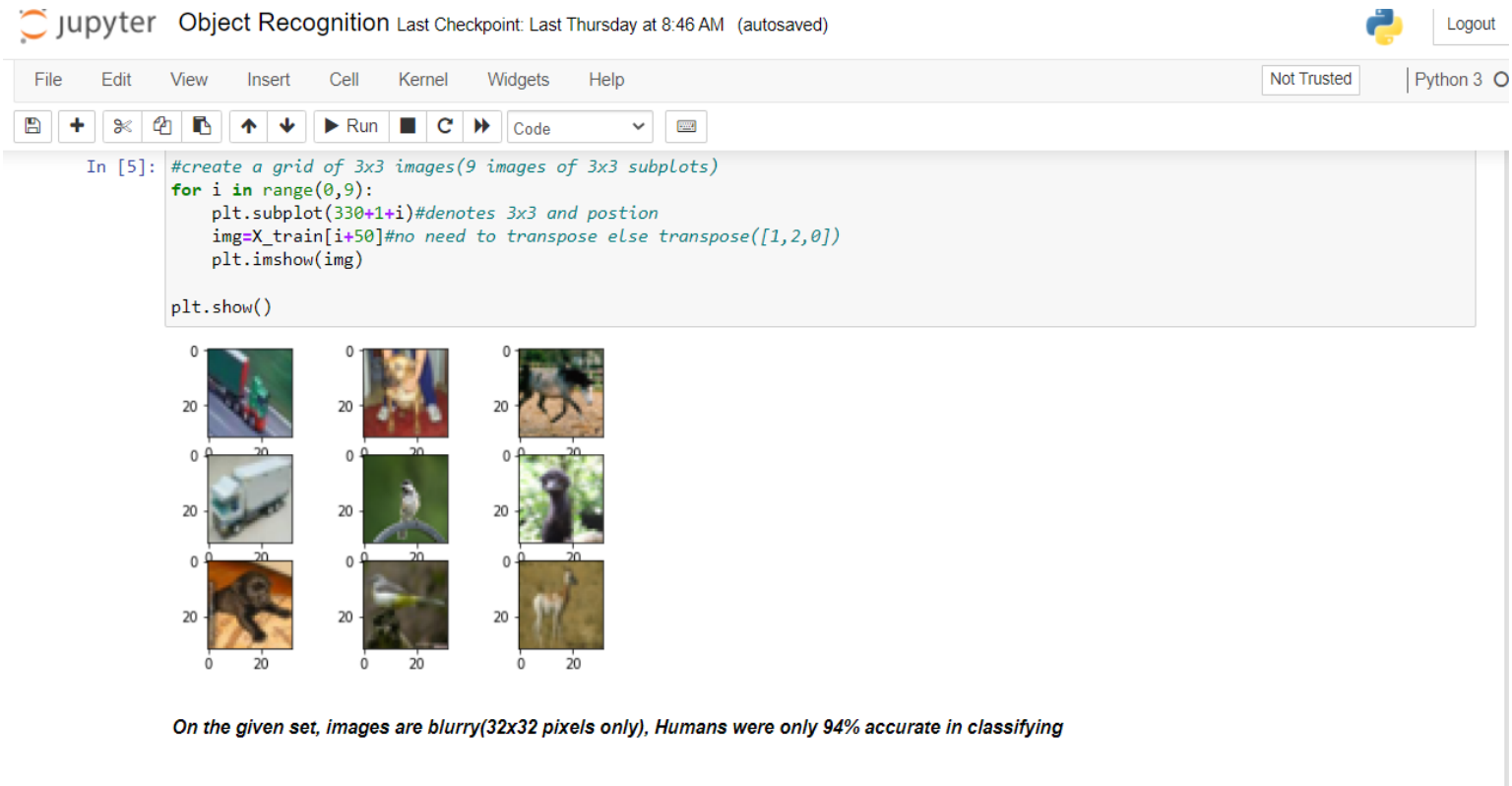


FIG 5.2

CHAPTER 6

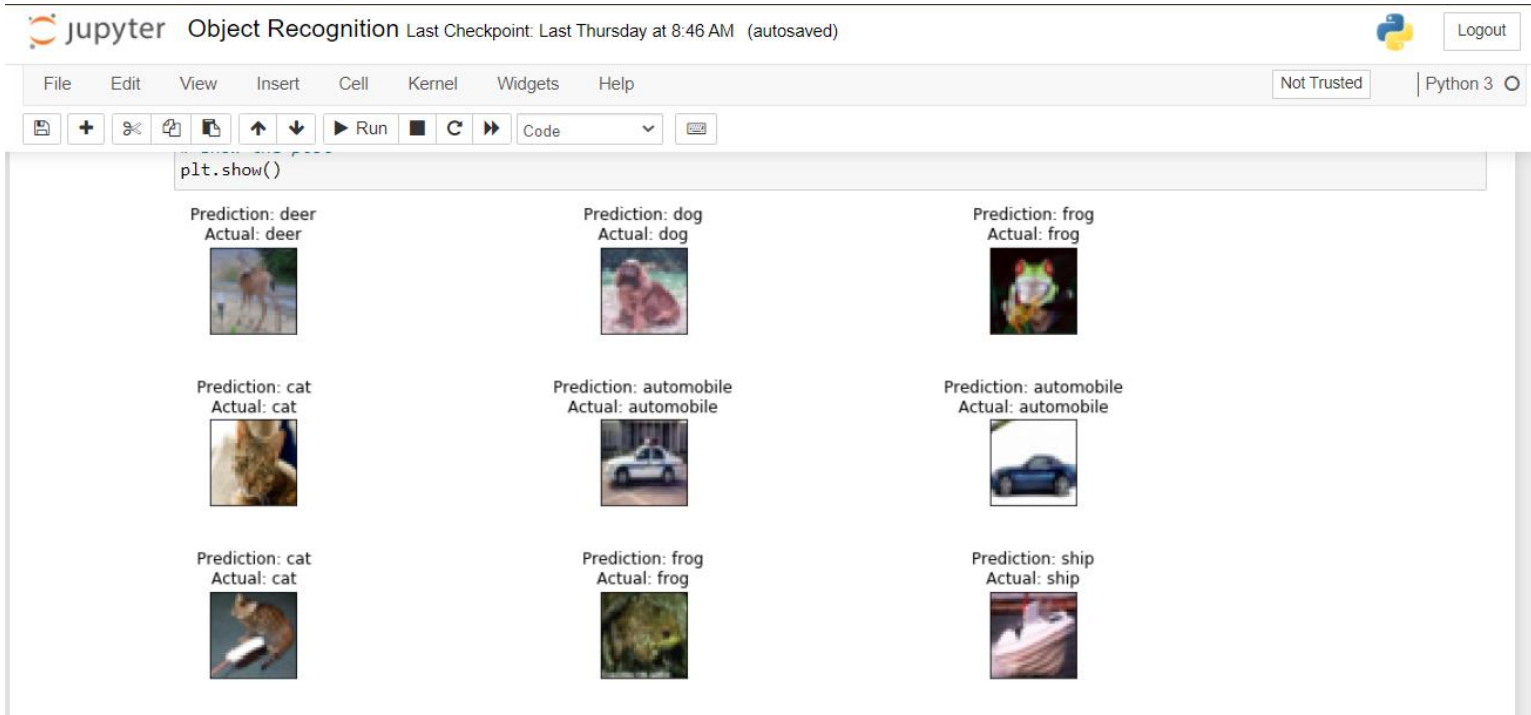
RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

6.1 RESULTS AND DISCUSSION :



BEFORE PREDICTION

On the given set, images are blurry(32x32 pixels only), Humans were only 94% accurate in classifying



AFTER PREDICTION

To use object recognition for your dataset, you will take the pretrained weights and then do fine tuning, ie, additional training on your dataset to convert some of these cnn's into a network that can produce result for your dataset. Fine tuning will require a GPU.

However, we can use a lot of the info that has already been learned by this network. So in the pretrained weights, we have like feature extractors, in some of those first layers we have group of neurons that does edge detection, shape detection, etc AND those things transfer really well.

So if you are using this on your own dataset, all you have to do is fine tuning; you don't have to that initial traing from random weights. SAVES TIME!!!

6.2 PERFORMANCE ANALYSIS:

PREDICTING-BATCH, VERBOSE

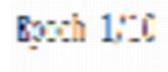
BATCHES : Keras can predict multiple values at the same time, like if you input a vector of 100 elements, Keras can compute one prediction for each element giving 100 outputs. This computation can also be done in batches

VERBOSE:

By setting verbose 0,1 or 2 you just say how do you want to 'see' the training progress for each epoch.

Verbose = 0 will show you nothing (silent)

Verbose = 1 will show you an animated progress bar like this



Verbose = 2 will just mention the number of epoch like this

CHAPTER 7

CONCLUSION

Object detection is a key ability for most computer and robot vision system.

Although great progress has been observed in the last years, and some existing techniques are now part of many consumer electronics (e.g., face detection for auto-focus in smartphones) or have been integrated in assistant driving technologies, we are still far from achieving human-level performance, in particular in terms of open-world learning. It should be noted that object detection has not been used much in many areas where it could be of great help. As mobile robots, and in general autonomous machines, are starting to be more widely deployed (e.g., quad-copters, drones and soon service robots), the need of object detection systems is gaining more importance. Finally, we need to consider that we will need object detection systems for nano-robots or for robots that will explore areas that have not been seen by humans, such as depth parts of the sea or other planets, and the detection systems will have to learn to new object classes as they are encountered. In such cases, a real-time open-world learning ability will be critical.

This chapter presented an overview of the machine learning techniques using convolutional neural networks for image object detection. The main algorithms for solving this type of problem were presented: Faster RCNN, YOLO, and SSD. To exemplify the functioning of the algorithms, datasets recognized by the scientific literature and in the field of computer vision were selected, tests were performed, and the results were presented, showing the advantages and differences of each of the techniques. This content is expected to serve as a reference for researchers and those interested in this broadly developing area of knowledge.

At the moment, we are experiencing the era of machine learning applications, and much should be developed in the coming years from the use and improvement of these techniques. Further improvements in the development of even more specific hardware and fundamental changes in related mathematical theory are expected shortly, making artificial intelligence increasingly present and important to the contemporary world.

REFERENCES

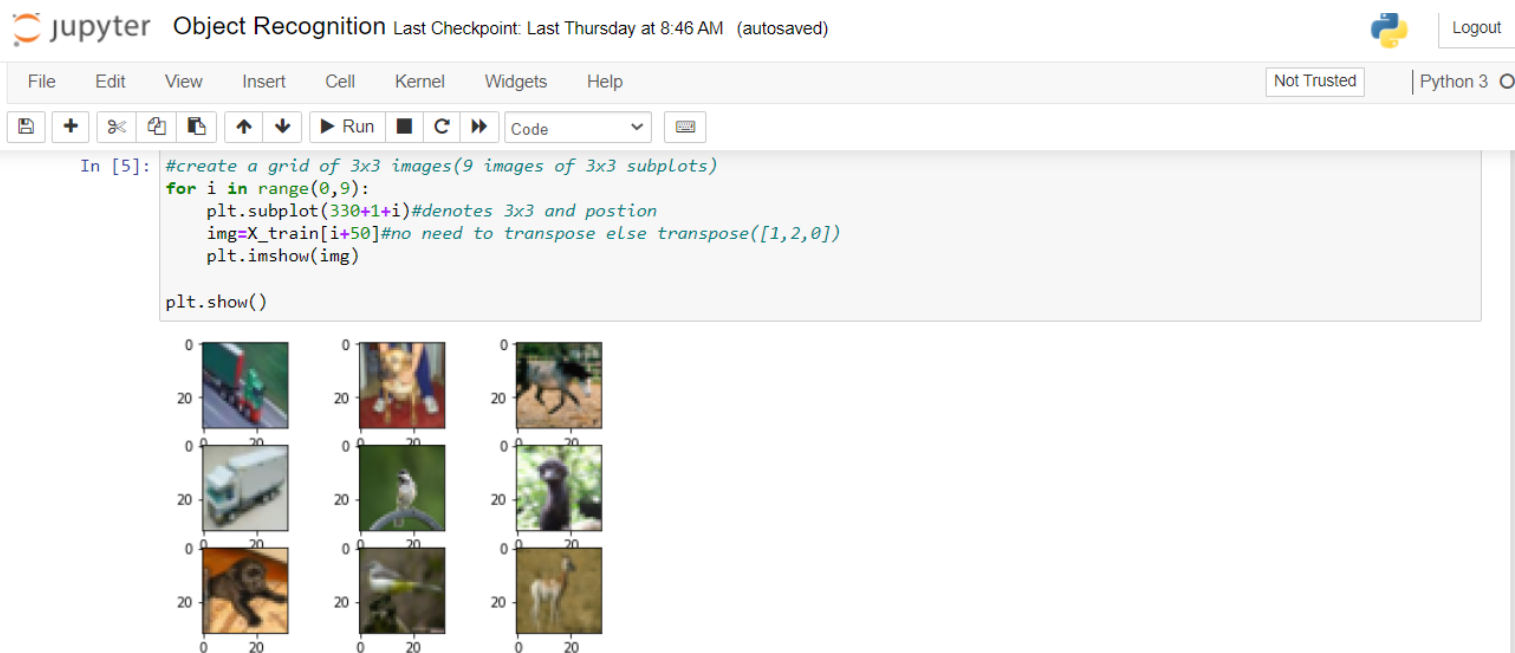
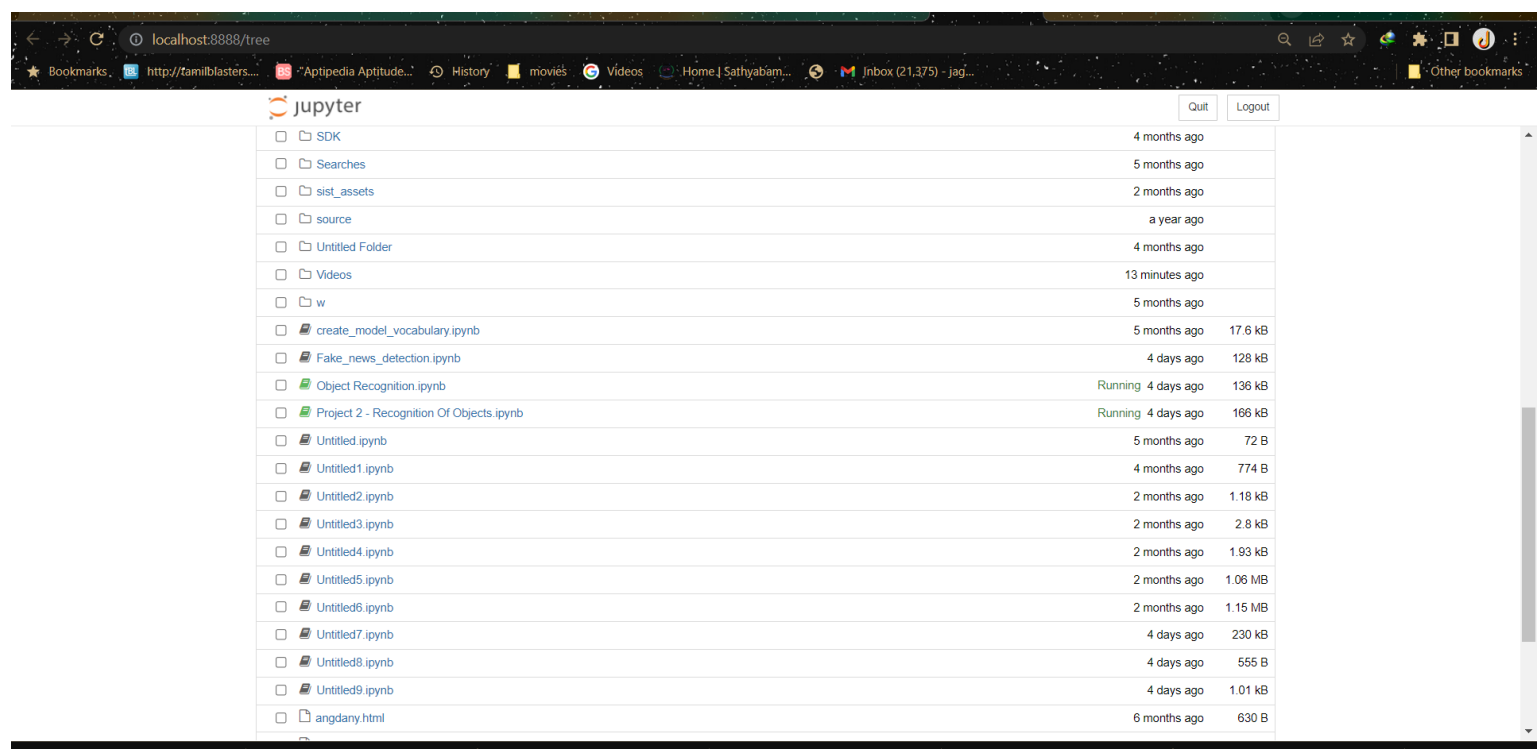
- Esteva A et al. Dermatologist-level classification of skin cancer with deep neural networks. Nature. 2017;542(7639):115
- Pan WD, Dong Y, Wu D. Classification of malaria-infected cells using deep convolutional neural networks. In: Machine Learning: Advanced Techniques and Emerging Applications. 2018. p. 159
- <https://ieeexplore.ieee.org/abstract/document/8398893>
- <https://www.1stop.ai/>
- <https://ieeexplore.ieee.org/abstract/document/8683987>
- https://books.google.co.in/books?hl=en&lr=&id=6UP8DwAAQBAJ&oi=fnd&pg=PA81&dq=references+for+object+recognition+using+cnn&ots=NNm_l0YmA&sig=6Mojb6UmHBECJayGOowpgEAzpo#v=onepage&q=references%20for%20object%20recognition%20using%20cnn&f=false
- https://scholar.google.co.in/scholar_url?url=https://citeseerx.ist.psu.edu/viewdoc/download%3Fdoi%3D10.1.1.446.4723%26rep%3Drep1%26type%3Dpdf&hl=en&sa=X&ei=B6BIYrKqC4ySyASZk6HgCA&scisig=AAGBfm1_L31e1jFPU1QE0b6HZB0Tq3_wig&oi=scholar

APPENDIX A

SNAP SHOTS

```
Jupyter Notebook (Anaconda)
[I 23:47:01.355 NotebookApp] JupyterLab extension loaded from D:\Anaconda\lib\site-packages\jupyterlab
[I 23:47:01.355 NotebookApp] JupyterLab application directory is D:\Anaconda\share\jupyter\lab
[I 23:47:01.361 NotebookApp] Serving notebooks from local directory: C:\Users\jagadees
[I 23:47:01.361 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 23:47:01.361 NotebookApp] http://localhost:8888/?token=13d546b7963760fdf3a766a95f7b57112c4fbd9e4903f897
[I 23:47:01.361 NotebookApp] or http://127.0.0.1:8888/?token=13d546b7963760fdf3a766a95f7b57112c4fbd9e4903f897
[I 23:47:01.361 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 23:47:01.627 NotebookApp]

To access the notebook, open this file in a browser:
    file:///C:/Users/jagadees/AppData/Roaming/jupyter/runtime/nbserver-25368-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=13d546b7963760fdf3a766a95f7b57112c4fbd9e4903f897
    or http://127.0.0.1:8888/?token=13d546b7963760fdf3a766a95f7b57112c4fbd9e4903f897
[I 23:52:23.685 NotebookApp] Kernel started: 9b610212-3654-4db7-ba5f-6a3ad1cf7b72, name: python3
[W 23:52:59.709 NotebookApp] Notebook Object Recognition.ipynb is not trusted
[I 23:53:00.547 NotebookApp] Kernel started: 7f927650-4cda-45c1-ac5c-94dbc386096f, name: python3
```



On the given set, images are blurry(32x32 pixels only), Humans were only 94% accurate in classifying

```
localhost:8888/notebooks/Object%20Recognition.ipynb
Object Recognition Last Checkpoint: Last Thursday at 8:46 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

In [6]: print(X_train[0])

[[[ 59  62  63]
  [ 43  46  45]
  [ 50  48  43]
  ...
  [158 132 108]
  [152 125 102]
  [148 124 103]]

  [[ 16  20  20]
  [  0  0  0]
  [ 18  8  0]
  ...
  [123  88  55]
  [119  83  50]
  [122  87  57]]

  [[ 25  24  21]
  [ 16  7  0]
  [ 49  27  8]
  ...
  [118  84  50]
  [120  84  50]
  [109  73  42]]

  ...

  [[208 170  96]
  [201 153  34]
  [198 161  26]
  ...
  [160 133  70]
  [ 56  31  7]
```

```
localhost:8888/notebooks/Object%20Recognition.ipynb
Object Recognition Last Checkpoint: Last Thursday at 8:46 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Building the All-CNN

Using the paper as a reference, we can implement the All-CNN network in Keras. Keras models are built by simply adding layers, one after another.

We will wrap this model in a function, which will allow us to quickly and neatly generate the model later on in the project.

In [11]: from keras.models import Sequential
from keras.layers import Dropout, Activation, Conv2D, GlobalAveragePooling2D
#conv2d is the main convolutional layer
from keras.optimizers import SGD as stochastic_gradient_descent

In [18]: def allcnn(weights=None):
    #taking random weights by default else use passed pretrained weights

    model=Sequential()#we will be adding one layer after another

    #not the input layer but need to tell the conv. layer to accept input
    model.add(Conv2D(96,(3,3),padding='same',input_shape=(32,32,3)))#32x32x3 channels
    model.add(Activation('relu'))#required for each conv. layer
    model.add(Conv2D(96,(3,3),padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(96,(3,3),padding='same',strides=(2,2)))
    model.add(Dropout(0.5))#drop neurons randomly;helps the network generalize(prevent overfitting on training data) better so in
    #that are controlling specific classes/features, the features are spread out over the entire network

    model.add(Conv2D(192,(3,3),padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(192,(3,3),padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(192,(3,3),padding='same',strides=(2,2)))
    model.add(Dropout(0.5))
```

localhost:8888/notebooks/Object%20Recognition.ipynb

jupyter Object Recognition Last Checkpoint: Last Thursday at 8:46 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

The original paper mentions that it took approximately 10 hours to train the All-CNN network for 350 epochs using a modern GPU, which is considerably faster (several orders of magnitude) than it would take to train on CPU.

we'll save some time by loading pre-trained weights for the All-CNN network. Using these weights, we can evaluate the performance of the All-CNN network on the testing dataset.

```
In [19]: #define the hyper parameters(generic or do Grid Search)
learning_rate=0.01
weight_decay=1e-6
momentum=0.9

#define training parameters
epochs=350 #from research paper
batch_size=32#run 32 images times then update the parameters instead of updating them after every image

model=allcnn()

#define optimizer and compile model
sgd=SGD(lr=learning_rate,decay=weight_decay,momentum=momentum,nesterov=True)
model.compile(loss='categorical_crossentropy',optimizer=sgd,metrics=['accuracy'])

print(model.summary())#1.3m parameters and all are trainable

# #fit the model(update the parameters and loss)
# model.fit(X_train,Y_train,validation_data=(X_test,Y_test),epochs=epochs,batch_size=batch_size,verbose=1)
```

Layer (type)	Output Shape	Param #
conv2d_28 (Conv2D)	(None, 32, 32, 96)	2688
activation_22 (Activation)	(None, 32, 32, 96)	0

localhost:8888/notebooks/Object%20Recognition.ipynb

jupyter Object Recognition Last Checkpoint: Last Thursday at 8:46 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

```
#make predictions
predictions=model.predict(batch,verbose=1)

{0: 'airplane', 1: 'automobile', 2: 'bird', 3: 'cat', 4: 'deer', 5: 'dog', 6: 'frog', 7: 'horse', 8: 'ship', 9: 'truck'}
9/9 [=====] - 0s 24ms/step
```

```
In [36]: print(predictions)
print(predictions.shape)
```

```
[[3.34924293e-19 2.15254109e-19 1.36195855e-11 7.40426736e-14
 9.99999940e-01 7.98329847e-09 7.02509698e-16 6.67015883e-08
 2.10513932e-21 6.66427573e-18]
[6.29249311e-13 3.24949181e-14 3.37529343e-07 4.88842372e-04
 7.45294665e-06 9.98557985e-01 7.16788229e-09 9.45367385e-04
 4.90906751e-12 1.51182011e-10]
[3.73205361e-24 6.11098715e-24 7.35285344e-19 2.90920977e-17
 1.78007700e-22 2.12187410e-16 1.00000000e+00 1.14791076e-22
 8.27280588e-27 8.52564011e-22]
[2.25066187e-18 7.57587193e-20 4.78785811e-09 1.00000000e+00
 4.77508122e-09 2.97874295e-11 1.40485887e-12 1.34755287e-15
 1.82800220e-19 1.35613839e-16]
[3.47531121e-33 1.00000000e+00 1.36119330e-40 7.24741207e-37
 8.84121240e-41 1.55544130e-43 1.24009309e-40 1.69776626e-35
 1.02667689e-26 3.85468670e-21]
[6.84771727e-31 1.00000000e+00 3.20374304e-34 4.26633685e-31
 1.68705825e-39 2.88708484e-34 4.96821747e-31 5.68708229e-30
 1.12214496e-30 2.82542822e-19]
[8.00852953e-19 6.52500424e-21 2.11737697e-10 1.00000000e+00
 4.16917161e-11 2.01022310e-08 8.66802463e-12 6.64950679e-15
 1.52367522e-19 1.71009656e-20]
[5.12621774e-24 1.11924783e-20 7.95129690e-20 3.13645150e-17
 8.10548496e-22 7.86593325e-24 1.00000000e+00 7.24286872e-25
 1.43619243e-23 2.43633791e-21]
[8.16904048e-26 8.30323813e-23 1.08176581e-29 1.20483928e-26
 5.89444678e-28 4.31820913e-23 9.99157528e-23 7.07115194e-24]
```

localhost:8888/notebooks/Object%20Recognition.ipynb

jupyter Object Recognition Last Checkpoint: Last Thursday at 8:46 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

```
#make predictions
predictions=model.predict(batch,verbose=1)

{0: 'airplane', 1: 'automobile', 2: 'bird', 3: 'cat', 4: 'deer', 5: 'dog', 6: 'frog', 7: 'horse', 8: 'ship', 9: 'truck'}
9/9 [=====] - 0s 24ms/step
```

In [36]:

```
print(predictions)
print(predictions.shape)
```

```
[[3.34924293e-19 2.15254109e-19 1.36195855e-11 7.40426736e-14
 9.99999940e-01 7.98329847e-09 7.02589698e-16 6.67815883e-08
 2.10513932e-21 6.66427573e-18]
 [6.29249311e-13 3.24949181e-14 3.37529343e-07 4.88842372e-04
 7.45294665e-06 9.98557985e-01 7.16788229e-09 9.45367385e-04
 4.90906751e-12 1.51182011e-10]
 [3.73285361e-24 6.11098713e-24 7.35285344e-19 2.90920977e-17
 1.78007790e-22 2.12187418e-16 1.00000000e+00 1.14791076e-22
 8.27280588e-27 8.52564011e-22]
 [2.25066187e-18 7.57587193e-20 4.78785811e-09 1.00000000e+00
 4.77508122e-09 2.97874295e-11 1.40485887e-12 1.34755287e-15
 1.82800220e-19 1.35613839e-16]
 [3.47531121e-33 1.00000000e+00 1.36119330e-40 7.24741207e-37
 8.84121240e-41 1.55544130e-43 1.24009309e-40 1.69776626e-35
 1.02667689e-26 3.85468670e-21]
 [6.84771727e-31 1.00000000e+00 3.20374304e-34 4.26633685e-31
 1.68705825e-39 2.88708484e-34 4.96821747e-31 5.68708229e-30
 1.12214496e-30 2.82542822e-19]
 [8.00852953e-19 6.52500424e-21 2.11737697e-10 1.00000000e+00
 4.16917161e-11 2.01022310e-08 8.66802463e-12 6.64950679e-15
 1.52367522e-19 1.71009656e-20]
 [5.12621774e-24 1.11924783e-20 7.95129690e-20 3.13645150e-17
 8.10548496e-22 7.86593325e-24 1.00000000e+00 7.24286872e-25
 1.43619243e-23 2.43633791e-21]
 [8.16904048e-26 8.30323813e-23 1.08176581e-29 1.20483928e-26
 5.89444678e-28 4.31820913e-23 9.99157528e-23 7.07115194e-24]]
```

jupyter Object Recognition Last Checkpoint: Last Thursday at 8:46 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

```
plt.show()
```

Prediction: deer Actual: deer 	Prediction: dog Actual: dog 	Prediction: frog Actual: frog 
Prediction: cat Actual: cat 	Prediction: automobile Actual: automobile 	Prediction: automobile Actual: automobile 
Prediction: cat Actual: cat 	Prediction: frog Actual: frog 	Prediction: ship Actual: ship 

APPENDIX B

SOURCE CODE

```
from keras.datasets import cifar10

from keras.utils import np_utils

from matplotlib import pyplot as plt

import numpy as np

from PIL import Image

(X_train, y_train), (X_test, y_test) = cifar10.load_data()

print("Training images: {}".format(X_train.shape))

print("Testing images: {}".format(X_test.shape))

print(X_train[0].shape)

for i in range(0,9):

    plt.subplot(330+1+i)#denotes 3x3 and postion

    img=X_train[i+50]#no need to transpose else transpose([1,2,0])

    plt.imshow(img)

plt.show()

print(X_train[0])

seed=6

np.random.seed(seed)

(X_train, y_train), (X_test, y_test) = cifar10.load_data()

X_train=X_train.astype('float32')

X_test=X_test.astype('float32')
```

```

X_train=X_train/255.0

X_test=X_test/255.0

print(X_train[0])

print(y_train.shape)

print(y_train[0])

print(y_train.min())

print(y_train.max())

Y_train=np_utils.to_categorical(y_train)

Y_test=np_utils.to_categorical(y_test)

num_classes=Y_test.shape[1]


print(Y_train.shape)

print(Y_train[0])

from keras.models import Sequential

from keras.layers import Dropout,Activation,Conv2D,GlobalAveragePooling2D

from keras.optimizers import SGD

def allcnn(weights=None):

model=Sequential()

model.add(Conv2D(96,(3,3),padding='same',input_shape=(32,32,3)))

model.add(Activation('relu'))

model.add(Conv2D(96,(3,3),padding='same'))

    model.add(Activation('relu'))

    model.add(Conv2D(96,(3,3),padding='same',strides=(2,2)))

model.add(Dropout(0.5))

```

```

model.add(Conv2D(192,(3,3),padding='same'))

model.add(Activation('relu'))

model.add(Conv2D(192,(3,3),padding='same'))

model.add(Activation('relu'))

model.add(Conv2D(192,(3,3),padding='same',strides=(2,2)))

model.add(Dropout(0.5))


model.add(Conv2D(192,(3,3),padding='same'))

model.add(Activation('relu'))

model.add(Conv2D(192,(1,1),padding='valid'))

model.add(Activation('relu'))

model.add(Conv2D(10,(1,1),padding='valid'))

model.add(GlobalAveragePooling2D())

model.add(Activation('softmax'))

if weights:

    model.load_weights(weights)

return model

learning_rate=0.01

weight_decay=1e-6

momentum=0.9

epochs=350

batch_size=32

model=allcnn()

```

```

sgd=SGD(lr=learning_rate,decay=weight_decay,momentum=momentum,nesterov=True)

model.compile(loss='categorical_crossentropy',optimizer=sgd,metrics=['accuracy'])

print(model.summary())

learning_rate = 0.01

weight_decay = 1e-6

momentum = 0.9

weights='all_cnn_weights_0.9088_0.4994.hdf5'

model=allcnn(weights)

sgd = SGD(lr=learning_rate, decay=weight_decay, momentum=momentum,
nesterov=True)

model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

scores=model.evaluate(X_test,Y_test,verbose=1)

print("Accuracy: %.2f%%" % (scores[1]*100))

classes=range(0,10)

names = ['airplane','automobile','bird','cat','deer','dog','frog','horse','ship','truck']

class_labels=dict(zip(classes,names))

print(class_labels)

batch=X_test[100:109]

labels=np.argmax(Y_test[100:109],axis=-1)

predictions=model.predict(batch,verbose=1)

print(predictions)

print(predictions.shape)

for image in predictions:

    print(np.sum(image))

```

```

class_result=np.argmax(predictions,axis=-1)

print(class_result)

fig,axs=plt.subplots(3,3,figsize=(15,6))

fig.subplots_adjust(hspace=1)

axs=axs.flatten()

for i,img in enumerate (batch):

for key,value in class_labels.items():

    if class_result[i]==key:

        title = 'Prediction: {} \nActual: {}'.format(class_labels[key],
class_labels[labels[i]])

        axs[i].set_title(title)

        axs[i].axes.get_xaxis().set_visible(False)

        axs[i].axes.get_yaxis().set_visible(False)

axs[i].imshow(img)

plt.show()

```