

OBJECT RECOGNITION USING CONVOLUTIONAL NEURAL NETWORKS

Submitted in partial fulfillment of the requirements for
the award of Bachelor of Engineering degree in
Computer Science and Engineering

by

N Maheswara Reddy: 37110492

P Lohith: 37110557



**DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**Accredited with Grade "A" by NAAC I 12B
Status by UGC I Approved by AICTE**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

March - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC I 12B Status by UGC

I Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600

119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING **BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of Mr. N Maheswara Reddy(37110557) Mr. P Lohith(37110557) who carried out the project entitled **Object Recognition Using Convolutional Neural Networks** under my supervision from November 2020 to March 2021.

Internal Guide

(Name in Capital letters with signature)

Head of the Department

Submitted for viva voce Examination Held on

Internal Examiner

External Examiner

DECLARATION

I N Maheswara Reddy and P Lohith here by declare that the Project Report entitled Object Recognition Using Convolutional Neural Network done by me under the guidance of Dr. /Prof./ Mr./Ms _(Internal) and ___(External)

at (Company nameand address) is submitted in partial fulfillment of the requirements

for the award of Bachelor ofEngineering / Technology degree in Sathyabama Institute Of Science and Technology.

DATE:

PLACE:

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 and Dr.L.Lakshmanan M.E., Ph.D., and Dr.S.Vigneshwari M.E., Ph.D., Heads of the Department of Computer Science and Engineering 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. K Ashok Kumar for his valuable guidance, suggestions and constant support 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 Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The Convolutional neural network (CNN), or simply neural network, is a machine learning method evolved from the idea of simulating the human brain. Compared to a traditional regression approach, the CNN is capable of modeling complex nonlinear relationships. The CNN also has excellent fault tolerance and is fast and highly scalable with parallel processing. The base idea of the project is to train the neural network using desired set of objects using several sets of images of the selected objects. Then we give a random image of the selected objects and the neural network gives the name of the given object.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	iv
	LIST OF FIGURES	vii
1	INTRODUCTION	1
	1.1 Intro	1
2	LITETATURE SURVEY	3
3	AIM AND SCOPE	8
	3.1 Aim	8
	3.2 Scope	8
	3.3 Existing System	8
	3.4 Proposed System	9
	3.5 Use case diag	10
	3.6 Architecture	11
4	EXPERIMENTAL OR MATERIALS AND METHODS USED	12
	4.1 Libraries and Algs	12
	4.2 System Req	16
	4.3 Soft Req	16
	4.4 Why neural nets'	19

	4.5 Neural networks Vs Conventional systems	19
	4.6 Network Layers	20
	4.7 The Learning Process	21
5	RESULTS AND DISCUSSION	23
	5.1 Programming of CNN	23
	5.2 Result	26
6	SUMMARY AND CONCLUSION	27
	6.1 Summary	27
	6.2 Conclusion	27
	REFERENCES	28
	APPENDIX	29
	RAW CODE	

TABLE OF IMAGES

S.no	Name	Page no.
1.	Fig 1	23
2.	Fig 2	24
3.	Fig 3	24
4.	Fig 4	25
5.	Fig 5	26

CHAPTER 1

INTRODUCTION

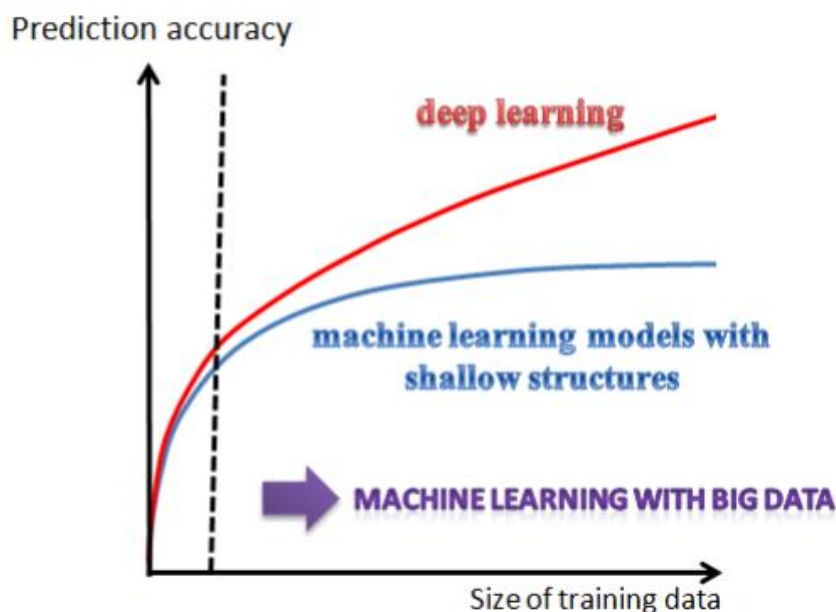
1.1 INTRO

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.² 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).



CHAPTER 2

LITERATURE SURVEY

Project title: Application of furniture images selection based on neural network

Author / s: Yong Wang, Wenwen Gao, and Ying Wang

Year of Publication: 2018

In the construction of 2 million furniture image databases, aiming at the problem of low quality of database, a combination of CNN and Metric learning algorithm is proposed, which makes it possible to quickly and accurately remove duplicate and irrelevant samples in the furniture image database. Solve problems that images screening method is complex, the accuracy is not high, time-consuming is long. Deep learning algorithm achieve excellent image matching ability in actual furniture retrieval applications after improving data quality.

Project title: Object Recognition in Very Low-Resolution Images Using Deep Collaborative Learning

Author / s: JEONGIN SEO AND HYEYOUNG PARK

Year of Publication: 2019

Although recent studies on object recognition using deep neural networks have reported remarkable performance, they have usually assumed that adequate object size and image resolution are available, which may not be guaranteed in real applications. This paper proposes a framework for recognizing objects in very low resolution images through the collaborative learning of two deep neural networks: image enhancement network and object recognition network. The proposed image enhancement network attempts to enhance extremely low resolution images into sharper and more informative images with the use of collaborative learning signals from the object recognition network. The object recognition network with trained weights for high resolution images actively participates in the learning of the image enhancement network. It also utilizes the output from the image enhancement network as augmented learning data to boost its recognition performance on very low resolution objects. Through experiments on various low resolution image benchmark datasets, we verified that the proposed method can improve the image reconstruction and classification performance.

Project title: A Deep Learning Framework Using Convolutional Neural Network for Multi-class Object Recognition

Author / s: Shaukat Hayat, She Kun, Zuo Tengtao, Yue Yu, Tianyi Tu, Yantong Du

Year of Publication: 2018

Object recognition is classic technique used to effectively recognize an object in the image. Technologies specifically in field of computer vision are expected to detect and recognize more complex tasks with help of local features detection methods. Over the last decade, there has been sustained increase in the number of researchers from various kind of disciplines i.e. academia, industry, security agencies and even from general public has caught an attention to explore the covered aspects of object detection and recognition concerned problems. It is further significantly amended by adopting deep learning model. In this paper, we applied deep learning to multi-class object recognition and explore convolutional neural network (CNN). The convolutional neural network is created with normalized standard initialization and trained with training set of sample images from 9 different object categories plus sample test images using widely varied dataset. All results are implemented in python tensorflow framework. We examine and compared CNN results with final feature vectors extracted from variant approaches of BOW based on linear L2-SVM classifier. Based on it, sufficient experiments verify our CNN model effectiveness and robustness with rate of 90.12% accuracy.

Project Title: Image Classification Using Deep Learning

Author / s: M Manoj krishna, M Neelima, M Harshali, M Venu Gopala Rao

Year of Publication: 2018

The image classification is a classical problem of image processing, computer vision and machine learning fields. In this paper we study the image classification using deep learning. We use AlexNet architecture with convolutional neural networks for this purpose. Four test images are selected from the ImageNet database for the classification purpose. We cropped the images for various portion areas and conducted experiments. The results show the effectiveness of deep learning based image classification using AlexNet.

Project title: Object Recognition in Remote Sensing Images Using Combined Deep Features

Author / s: Bitao Jiang , Xiaobin Li , Lu Yin , Wenzhen Yue , Shengjin Wang

Year of Publication: 2018

Object recognition, which is also referred as object classification or object type recognition, aims at discriminating object types in remote sensing images. With the availability of high resolution remote sensing images, object recognition attracts more and more attention. Different from traditional methods mainly using hand-crafted features, we propose an object recognition method that combines deep features extracted from a convolutional neural network (CNN) to recognize aircrafts and ships in remote sensing images. The proposed method consists of two stages. In the training stage, images of objects with different types and corresponding labels are exploited to fine-tune a pretrained CNN. Convolutional features are extracted from a convolutional layer of the fine-tuned CNN and pooled by Fisher Vector, and fully-connected features are extracted from a fully connected layer of the CNN. These features are combined by concatenation and used to train a support vector machine (SVM). In the test stage, the type of each object is determined by the trained SVM using its combined features. Experiments on two data sets collected from Google Earth demonstrate the effectiveness of our method.

Project title: Image Recognition Based on Deep Learning

Author / s: Meiyin Wu and Li Chen

Year of Publication: 2019

Deep learning is a multilayer neural network learning algorithm which emerged in recent years. It has brought a new wave to machine learning, and making artificial intelligence and human-computer interaction advance with big strides. We applied deep learning to handwritten character recognition, and explored the two ainstream algorithm of deep learning: the Convolutional Neural Network (CNN) and the Deep Belief NetWork (DBN). We conduct the performance evaluation for CNN and DBN on the MNIST database and the real-world handwritten character database. The classification accuracy rate of CNN and DBN on the MNIST database is 99.28% and 98.12% respectively, and on the real-world handwritten character database is 92.91% and 91.66% respectively. The experiment results show that deep learning does have an excellent feature learning ability. It don't need to extract features manually. Deep learning can learn more nature features of the data.

Project title: A Review of Deep Learning in Image Recognition

Author / s: Myeongsuk Pak, Sanghoon Kim

Year of Publication: 2018

Deep learning has been the core topic in machine learning and convolutional neural network is one of the most prominent approaches. Convolutional neural network has won numerous competitions in recent years. It has outstanding results in image recognition. We review the different deep learning approaches which have been used in the field of image classification and localization.

Project title: Contour Tracking Based Knowledge Extraction And Object Recognition Using Deep Learning Neural Networks

Author / s: Annapareddy V. N. Reddy

Year of Publication: 2016

Object recognition in digital images are carried out using syntactic or spectral domain pattern recognition techniques. Due to ever increasing size of data collected by digital image acquisition systems there is a need to go in for developing faster, reliable and intelligent pattern recognition methods which would mostly supplement human intelligence in recognizing objects which otherwise remain latent and unnoticed. One such effort is use of deep learning neural networks for object recognition. The input to this system is knowledge extracted from the contours of various objects pre valent in a digital image. This paper advocates a novel method for extracting knowledge about the contours of various objects and components in a digital image and for recognizing objects using a neural network

Project title: Object Recognition in Very Low Resolution Images using Deep Collaborative Learning

Author / s: Jeongin Seo and Hyeyoung Park

Year of Publication: 2018

Although recent studies on object recognition using deep neural networks have reported remarkable performance, they have usually assumed that adequate object size and image resolution are available, which may not be guaranteed in real applications. This paper proposes a framework for recognizing objects in very low resolution images through the collaborative learning of two deep neural networks: image enhancement network and object recognition network. The proposed image enhancement network attempts to enhance extremely low resolution images into sharper and more informative images with the use of collaborative learning signals

from the object recognition network. The object recognition network with trained weights for high resolution images actively participates in the learning of the image enhancement network. It also utilizes the output from the image enhancement network as augmented learning data to boost its recognition performance on very low resolution objects. Through experiments on various low resolution image benchmark datasets, we verified that the proposed method can improve the image reconstruction and classification performance.

Project title: An Incremental Intelligent Object Recognition System Based on Deep Learning

Author / s: Long Yan, Yongxiong Wang, Tianzhong Song

Year of Publication: 2019

The accuracy of object recognition has been greatly improved due to the rapid development of deep learning, but the deep learning generally requires a lot of training data and the training process is very slow and complex. We propose an incremental object recognition system based on deep learning techniques and speech recognition technology with high learning speed and wide applicability. The system can learn from scratch through the way of human-computer interaction. Through the interaction of user, the system continues to improve its identification ability by updating or adding object's feature templates gradually. The types of objects that it can be identified become more and more, and recognition rate is also getting high increasingly. The GoogLeNet inception v4 network is used to extract the object features. Then the object is classified based on the extracted features by measuring the similarity between the object and its template. Experiments show that our system can identify the object accurately after the system learns about ten samples of this object. The self-learning system has a wide range of applicability and flexibility because of the incremental frame based on deep learning.

Project title: Scene Recognition by Manifold Regularized Deep Learning Architecture

Author / s: Yuan Yuan

Year of Publication: 2017

Scene recognition is an important problem in the field of computer vision, because it helps to narrow the gap between the computer and the human beings on scene understanding. Semantic modeling is a popular technique used to fill the semantic gap in scene recognition. However, most of the semantic modeling approaches learn shallow, one-layer representations for scene recognition, while ignoring the structural information related between images, often resulting in poor performance. Modeled after our own human visual system, as it is intended to inherit humanlike judgment, a manifold regularized deep architecture is proposed for scene recognition. The proposed deep architecture exploits the structural information of the data, making for a mapping between visible layer and hidden layer. By the proposed approach, a deep architecture could be designed to learn the high-level features for scene recognition in an unsupervised fashion. Experiments on standard data sets show that our method outperforms the state-of-the-art used for scene recognition

CHAPTER 3

AIM AND SCOPE OF PRESENT INVESTIGATION

3.1 AIM: Detection of desired object from images having various backgrounds.

3.2 SCOPE: Object detection is breaking into a wide scope of enterprises, with use cases extending from individual security to efficiency in the working environment. Object detection is applied in numerous territories of image processing, including picture retrieval, security, observation, computerized vehicle systems and machine investigation. Critical difficulties remain in the field of object detection. The potential outcomes are inestimable with regards to future use cases for object detection.

3.3 EXISTING SYSTEM:

The most popular work in this category is the boosted cascade classifier of Viola and Jones (2004). It works by efficiently rejecting, in a cascade of test/filters, image patches that do not correspond to the object. Cascade methods are commonly used with boosted classifiers due to two main reasons: (i) boosting generates an additive classifier, thus it is easy to control the complexity of each stage of the cascade and (ii) during training, boosting can be also used for feature selection, allowing the use of large (parametrized) families of features. A coarse-to-fine cascade classifier is usually the first kind of classifier to consider when efficiency is a key requirement.

The best example in this category is the Bag of Word method [e.g., Serre et al. (2005) and Mutch and Lowe (2008)]. This approach is basically designed to detect a single object per image, but after removing a detected object, the remaining objects can be detected [e.g., Lampert et al. (2009)]. Two problems with this approach are that it cannot robustly handle well the case of two instances of the object appearing near each other, and that the localization of the object may not be accurate.

One of the first successful methods in this family is based on convolutional neural networks (Delakis and Garcia, 2004). The key difference between this and the above approaches is that in this approach the feature representation is learned instead of being designed by the user, but with the drawback that a large number of training samples is required for training the classifier.

3.4 PROPOSED SYSTEM:

We develop a machine learning algorithm using python programming language.

Python programming language has many built in library files that play a major role in developing machine learning algorithms.

For this objective we use Keras and TensorFlow library files that help in training our neural networks and help achieving our objective.

In order to get the best accuracy, we have to select several different images of the required object. We then give a random image as input and let the program tell us if there is a presence of the said object or not.

3.5 USE CASE DIAGRAM

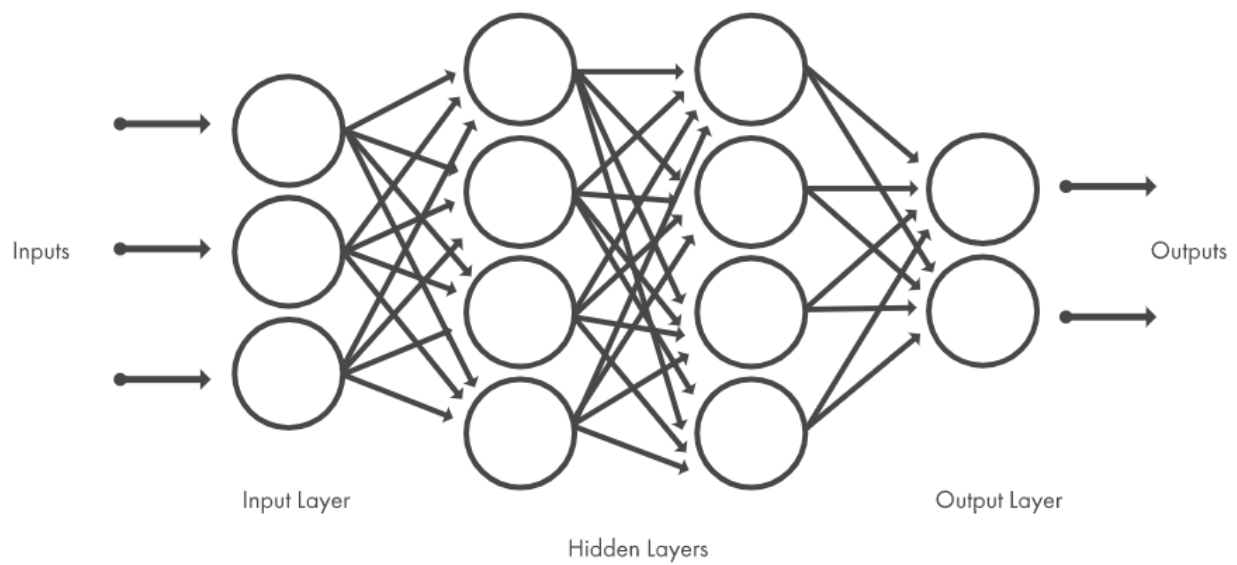
Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development. s simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases. The purpose of the use case diagrams is simply to provide the high level view of the system and convey the requirements in laypeople's terms for the stakeholders. Additional diagrams and documentation can be used to provide a complete functional and technical view of the system.

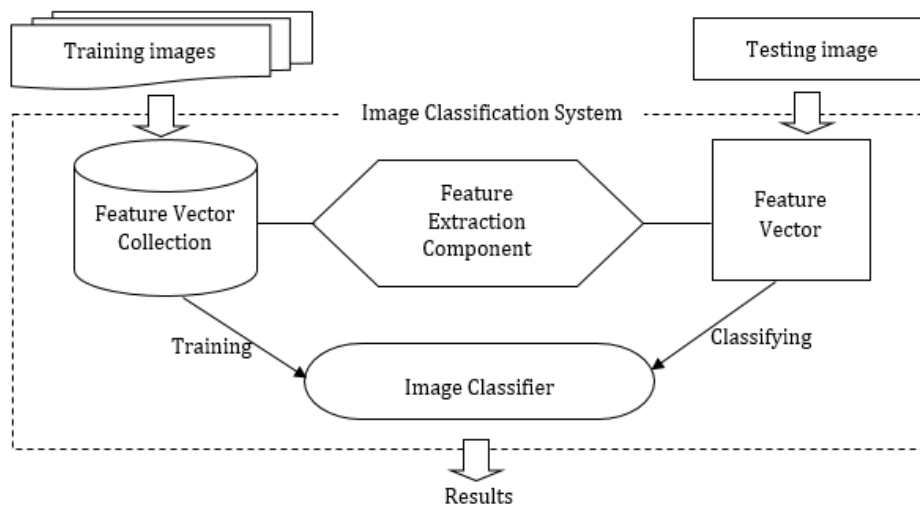
Use case diagram consists of two parts:

Use case: A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

Actor: An actor is a person, organization or external system that plays a role in one or more interaction with the system.



3.6 SYSTEM ARCHITECTURE



We train the machine learning algorithm using the training images that are available before hand and we test the algorithm using different set of images called test images. We then observe the result by inserting a random image.

CHAPTER 4

EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED

4.1 LIBRARIES AND ALGORITHMS

Neural networks:

The performance of machine learning changes with the scale of training data. As the training data becomes very large, the performance of machine learning models with shallow structures gets saturated because their limited learning capacity, while the performance of deep learning keeps increasing. In the past decades, machine learning research focused on solving the overfitting problem because only small training data was available. With large-scale training data, people need to solve the underfitting problem, which is the focus of deep learning. The research focus of deep learning has been shifted from solving the overfitting problem to these aspects, which have not been well explored in the past decades.

Convolutional Neural Networks with TensorFlow

In this tutorial, you'll learn how to construct and implement Convolutional Neural Networks (CNNs) in Python with the TensorFlow framework.

TensorFlow is a popular deep learning framework. In this tutorial, you will learn the basics of this Python library and understand how to implement these deep, feed-forward artificial neural networks with it.

To be precise, you'll be introduced to the following topics in today's tutorial:

- You'll be first introduced to tensors and how they differ from matrices; Once you understand what tensors are then, you'll be introduced to the Tensorflow Framework, within this you will also see that how even a single line of code is implemented via a computational graph in TensorFlow, then you will learn about some of the package's concepts that play a major role in you to do deep learning like constants, variables, and placeholders.
- Then, you'll be headed to the most interesting part of this tutorial. That is the implementation of the Convolutional Neural Network: first, you will try to understand the data. You'll use Python and its libraries to load, explore, and analyze your data.

You'll also preprocess your data: you'll learn how to visualize your images as a matrix, reshape your data and rescale the images between 0 and 1 if required.

- With all of this done, you are ready to construct the deep neural network model. You'll start by defining the network parameters, then learn how to create wrappers to increase the simplicity of your code, define weights and biases, model the network, define loss and optimizer nodes. Once you have all this in place, you are ready for training and testing your model.

- Finally, you will learn to work with your own dataset. In this section, you would download the CIFAR-10 dataset from Kaggle, load the images and labels using Python modules like glob & pandas. You will read the images using OpenCV, one-hot the class labels, visualize the images with labels, normalize the images, and finally split the dataset into train and test set.

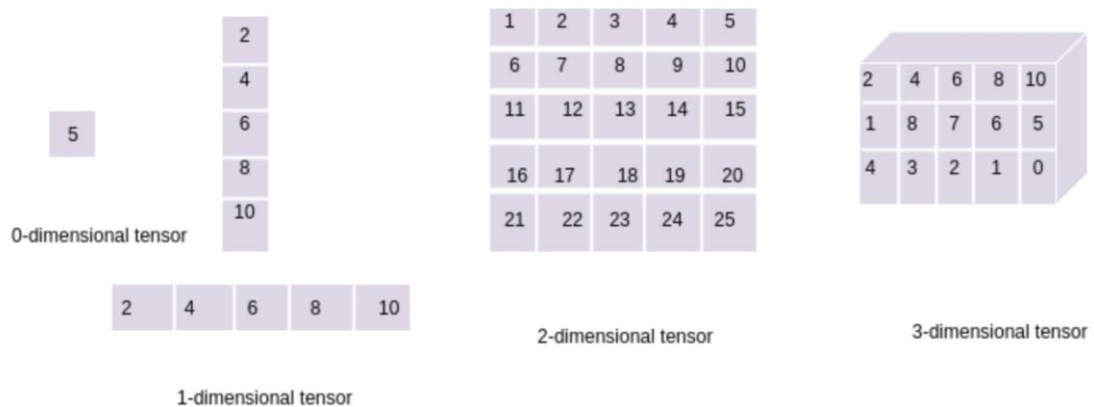
Tensors

In layman's terms, a tensor is a way of representing the data in deep learning. A tensor can be a 1-dimensional, a 2-dimensional, a 3-dimensional array, etc. You can think of a tensor as a multidimensional array. In machine learning and deep learning, you have datasets that are high dimensional, in which each dimension represents a different feature of that dataset.

Consider the following example of a dog versus cat classification problem, where the dataset you're working with has multiple varieties of both cats and dogs images.

Now, in order to correctly classify a dog or a cat when given an image, the network has to learn discriminative features like color, face structure, ears, eyes, the shape of the tail, etc.

These features are incorporated by the tensors.



But how are tensors then any different from matrices? You'll find out in the next section!

Tensors versus Matrices: Differences

A matrix is a two-dimensional grid of size $n \times m$

that contains numbers: you can add and subtract matrices of the same size, multiply one matrix with another as long as the sizes are compatible ($(n \times m) \times (m \times p) = n \times p$)

, and multiply an entire matrix by a constant.

A vector is a matrix with just one row or column (but see below).

The dimension of the tensor is called its rank.

Any rank-2 tensor can be represented as a matrix, but not every matrix is a rank-2 tensor. The numerical values of a tensor's matrix representation depend on what transformation rules have been applied to the entire system.

TensorFlow: Constants, Variables, and Placeholders

TensorFlow is a framework developed by Google on 9th November 2015. It is written in Python, C++, and Cuda. It supports platforms like Linux, Microsoft Windows, macOS, and Android. TensorFlow provides multiple APIs in Python, C++,

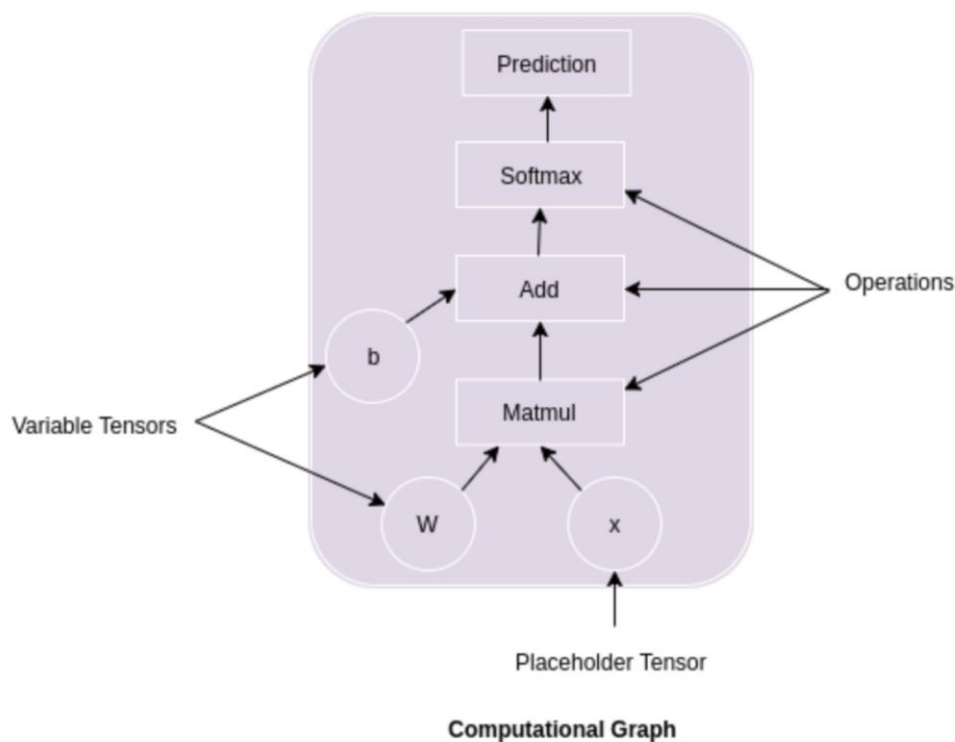
Java, etc. It is the most widely used API in Python, and you will implement a convolutional neural network using Python API in this tutorial.

The name TensorFlow is derived from the operations, such as adding or multiplying, that artificial neural networks perform on multidimensional data arrays. These arrays are called tensors in this framework, which is slightly different from what you saw earlier.

So why is there a mention of a flow when you're talking about operations?

Let's consider a simple equation and its diagram, represented as a computational graph. Note: don't worry if you don't get this equation straight away, this is just to help you to understand how the flow takes place while using the TensorFlow framework.

```
prediction = tf.nn.softmax(tf.matmul(W,x) + b)
```



In TensorFlow, every line of code that you write has to go through a computational graph. As in the above figure, you can see that first W

and x get multiplied. Then comes b , which is added to the output of W and x .

After adding the output of W and x with b

, a softmax function is applied, and the final output is generated.

You'll find that when you're working with TensorFlow, constants, variables, and placeholders come handy to define the input data, class labels, weights, and biases.

Constant takes no input, you use them to store constant values. They produce a constant output that it stores.

4.2 SYSTEM REQUIREMENTS

Processor: intel i5 or above

Amd ryzen 5 or above

Ram: 8gb or better

Graphics: Nvidia gtx 960 or better

Display : 30hz screen or better

4.3 SOFTWARE REQUIREMENTS

1. Visual Studio

Download the Visual Studio version required for the CUDA version you selected. For Mask RCNN I needed Tensorflow 1.14. The CUDA version compatible with Tensorflow 1.14 is CUDA 10. Find this information related to TensorFlow and CUDA versions in this link.

Now the CUDA version requires a particular version of Visual Studio with an appropriate Visual C++ compiler. In my case, it was Visual Studio 2017 having Visual C++ 15.0 compiler. Use this link for the appropriate version.

2. Anaconda

Install Anaconda for Python 3.x version from here. It will help you create a virtual environment that will be enabled with tensor flow-gpu. In this environment, you will launch jupyter notebook.

3. CUDA

Download and install CUDA from [here](#). I had to download CUDA 10 (an older version). Follow the exact installation procedure for CUDA from [here](#) (except the CUDA version). CUDA version should be compatible with your TensorFlow version.

4. cuDNN

Your cuDNN version has to be according to your CUDA version. Follow this link to find the appropriate version. I had to download cuDNN 7.4 (an older version). The detailed steps for cuDNN installation and setting up environment variables are given [here](#).

5. TensorFlow

You need to create an environment using Anaconda prompt to install TensorFlow. Select the GPU enabled TensorFlow compatible with your model. Follow the steps from [here](#).

After you installed TensorFlow, you should check if you can access the GPU. The code snippet mentioned in the above link is not working because of version change. Use the following script to check for the same. It will show the available computing resources.

```
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

6. Jupyter Notebook

It's time to install jupyter notebook in your environment. Follow the instructions given [here](#). Now you are ready for the exploration of the object detection model.

7. The Model

Download the model you want to use for the object detection task. You need to clone it from the git repository. I used Mask RCNN. So I downloaded and installed it using the following command. You can find an implementation of the Mask RCNN model [here](#).# To clone the Model

```
!git clone https://github.com/matterport/Mask_RCNN.git
```

```
# To change directory to the installation folder
```

```
cd Mask_RCNN
```

```
# To install the model
```

```
!python setup.py install
# To confirm the installation
pip show mask-rcnn
```

8. Python Packages

Now you can download other python packages required for building the model. There is a list of requirements in the Mask RCNN repository. It contains all the necessary packages that are required for this implementation. It does not mention the version of all the packages, though.

I installed all the necessary packages with the required versions to avoid further version conflict.

```
pip install scikit-image==0.14.2
pip install Keras==2.2.4
pip install scipy==1.2.1
pip install Pillow==7.2.0
pip install Cython==0.29.6
pip install scikit-image==0.14.2
pip install opencv-python==3.4.5.20
pip install imgaug==0.2.8
pip install h5py==2.10.0
```

Now your jupyter notebook is ready with all installations. You don't have to think about compatibility issues anymore. You can employ your full concentration on training and inference of the object detection model you desire to build.

Computer vision is a specialized field of artificial intelligence. The approach towards solving a computer vision problem is a bit different from usual machine learning approaches..I wrote this article to help the newbies in data science to dive into the object detection models. It will also be helpful for those professionals, like me, who never have tried object detection tasks before. I used Mask RCNN as an example to show the procedure. My intension was to propose a structured approach to configuring the system before starting any object detection assignment.

4.4 WHY NEURAL NETWORKS?

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze this expert can then be used to provide projections given new situations of interest and answer "what if" questions. Other advantages include:

Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience. Self-Organization: An ANN can create its own organization or representation of the information it receives during learning time. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

4.5 NEURAL NETWORKS VERSUS CONVENTIONAL COMPUTERS:

Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers use an algorithmic approach i.e. the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That restricts the problem solving capability of conventional computers to problems that we already understand and know how to solve. But computers would be so much more useful if they could do things that we don't exactly know how to do.

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. The examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable.

On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is to be solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault.

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

4.6 NETWORK LAYERS

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of "input" units is connected to a layer of "hidden" units, which is connected to a layer of "output" units.

The activity of the input units represents the raw information that is fed into the network. The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden

units. The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents. We also distinguish single-layer and multi-layer architectures. The single-layer organization, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organizations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

4.7 THE LEARNING PROCESS

- 1) Associative mapping: in which the network learns to produce a particular pattern on the set of input units whenever another particular pattern is applied on the set of input units. The associative mapping can generally be broken down into two mechanisms:
 - 2) Auto-association: an input pattern is associated with itself and the states of input and output units coincide. This is used to provide pattern competition, i.e., to produce a pattern whenever a portion of it or a distorted pattern is presented. In the second case, the network actually stores pairs of patterns building an association between two sets of patterns.
 - 3) Hetero-association: is related to two recall mechanisms:
 - 4) Nearest-neighbor recall, where the output pattern produced corresponds to the input pattern stored, which is closest to the pattern presented, and
 - 5) Interpolative recall, where the output pattern is a similarity dependent interpolation of the patterns stored corresponding to the pattern presented. Yet another paradigm, which is a variant associative mapping, is classification, i.e. when there is a fixed set of categories into which the input patterns are to be classified.
 - 6) Regularity detection in which units learn to respond to particular properties of the input patterns. Whereas in associative mapping the network stores the relationships among patterns, in regularity detection the response of each unit has a particular 'meaning'. This type of learning mechanism is essential for feature discovery and knowledge representation.

Every neural network possesses knowledge which is contained in the values of the connections weights. Modifying the knowledge stored in the network as a function of experience implies a learning rule for changing the values of the weights.

Information is stored in the weight matrix W of a neural network. Learning is the determination of the weights. Following the way learning is performed, we can distinguish two major categories of neural networks:

Fixed networks in which the weights cannot be changed, i.e. $dW/dt=0$. In such networks, the weights are fixed a priori according to the problem to solve.

Adaptive networks which are able to change their weights, i.e. $dW/dt \neq 0$.

All learning methods used for adaptive neural networks can be classified into two major categories:

Supervised learning which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning. An important issue concerning supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values. The aim is to determine a set of weights which minimizes the error. One well-known method, which is common to many learning paradigms is the least mean square (LMS) convergence.

Unsupervised learning uses no external teacher and is based upon only local information. It is also referred to as self-organization, in the sense that it self-organizes data presented to the network and detects their emergent collective properties. Paradigms of unsupervised learning are Hebbian learning and competitive learning. Another aspect of learning concerns the distinction or not of a separate phase, during which the network is trained, and a subsequent operation phase. We say that a neural network learns off-line if the learning phase and the operation phase are distinct.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 PROGRAMMING OF CNN

This experiment was conducted on jupyter notebook under the usage of various machine learning libraries that are available via python programming language and python packages.

The following were observed during the experiment:

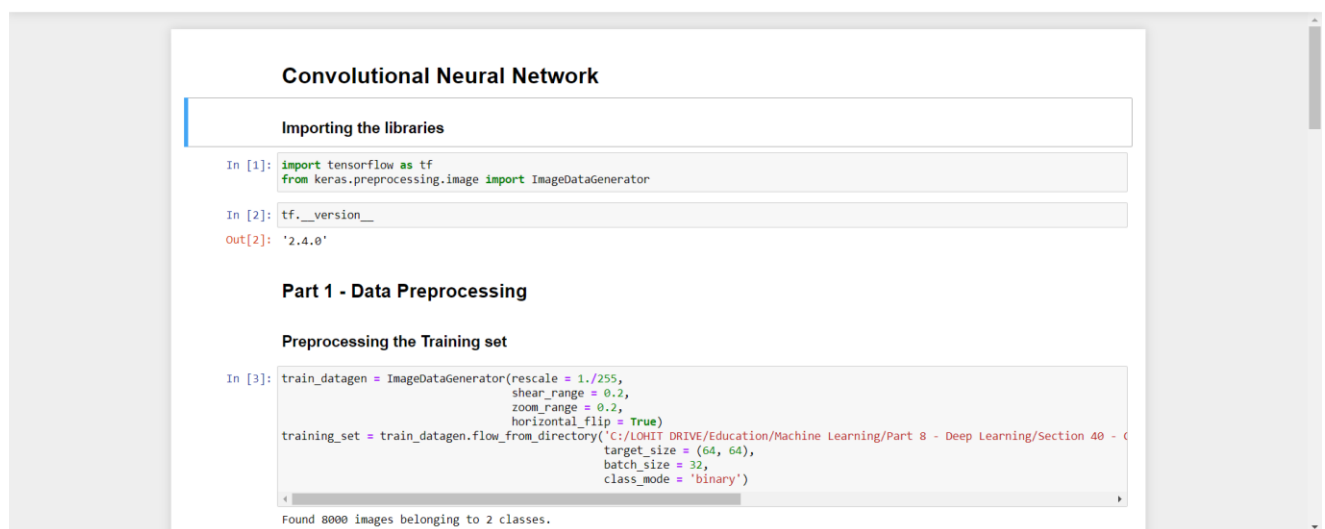


Fig1.img

The above image indicates the importing the necessary libraries to our program.

And the second half of the image depicts the importing and training of training set by using the set of images.

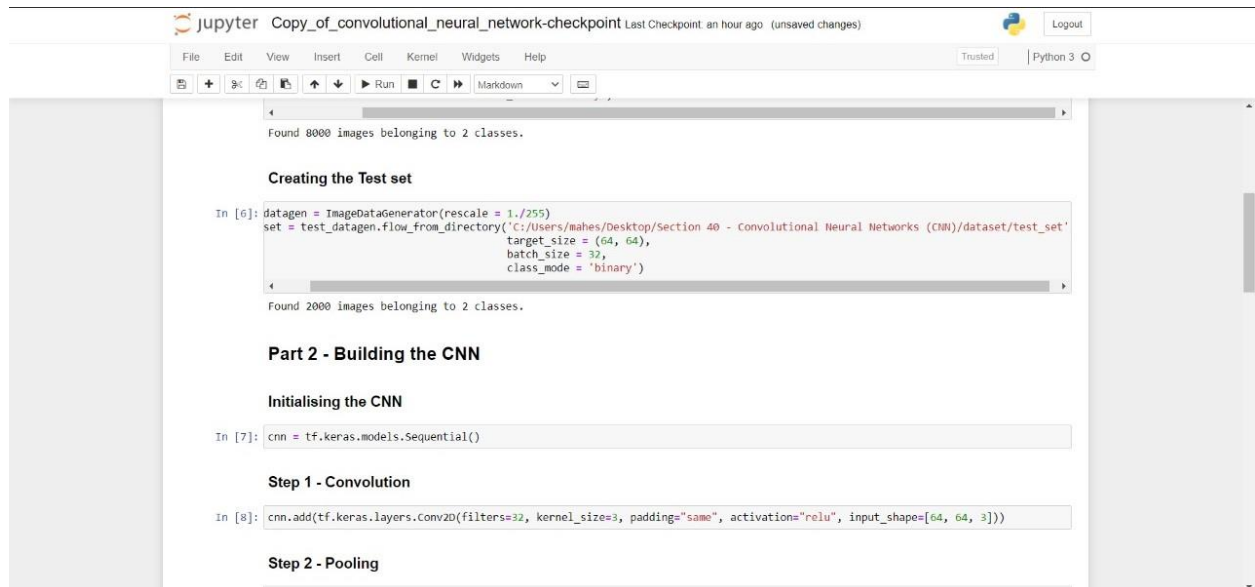


Fig 2.img

The above image depicts the training of the test set using the specified set of test images.

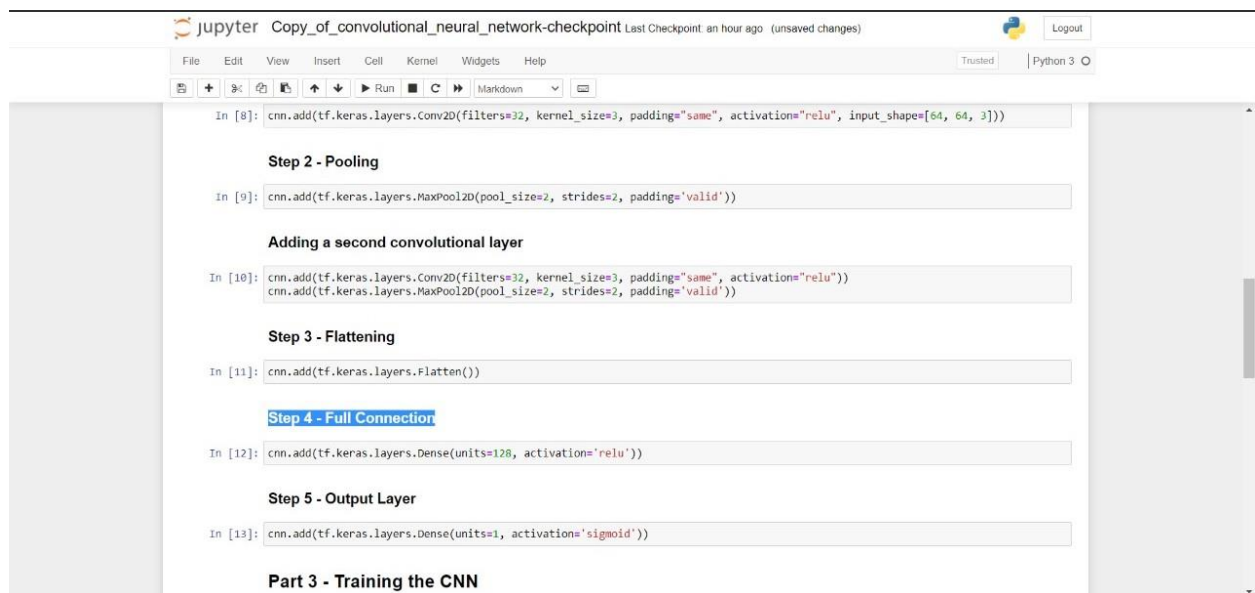
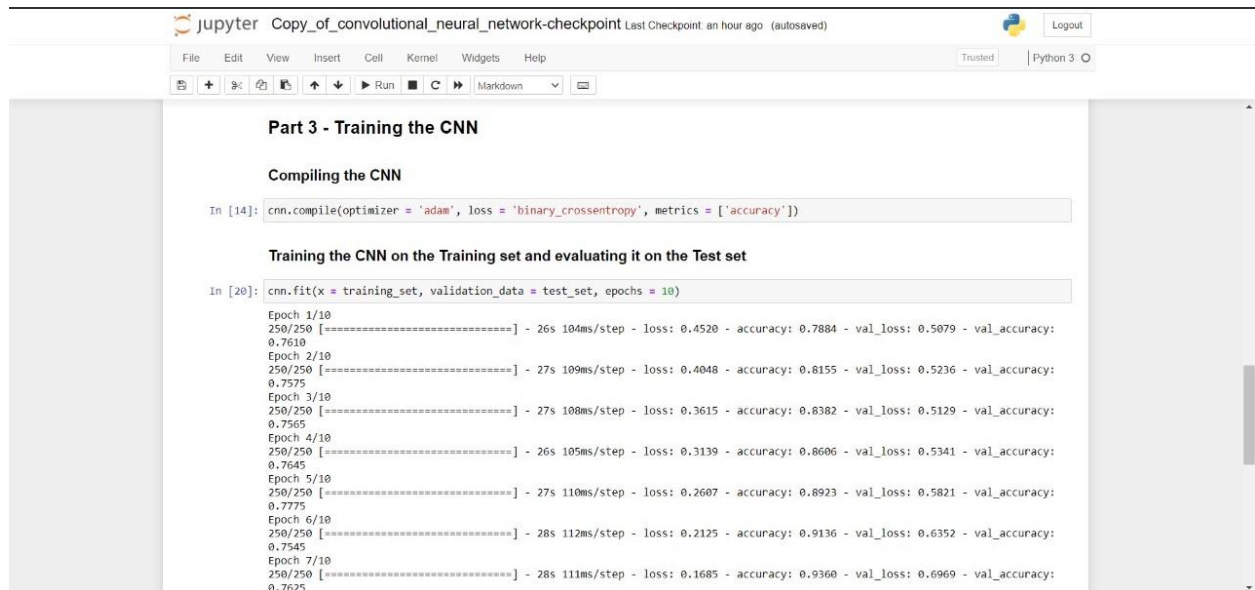


Fig 3.img

The above image depicts the creation of CNN



```
Part 3 - Training the CNN

Compiling the CNN

In [14]: cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

Training the CNN on the Training set and evaluating it on the Test set

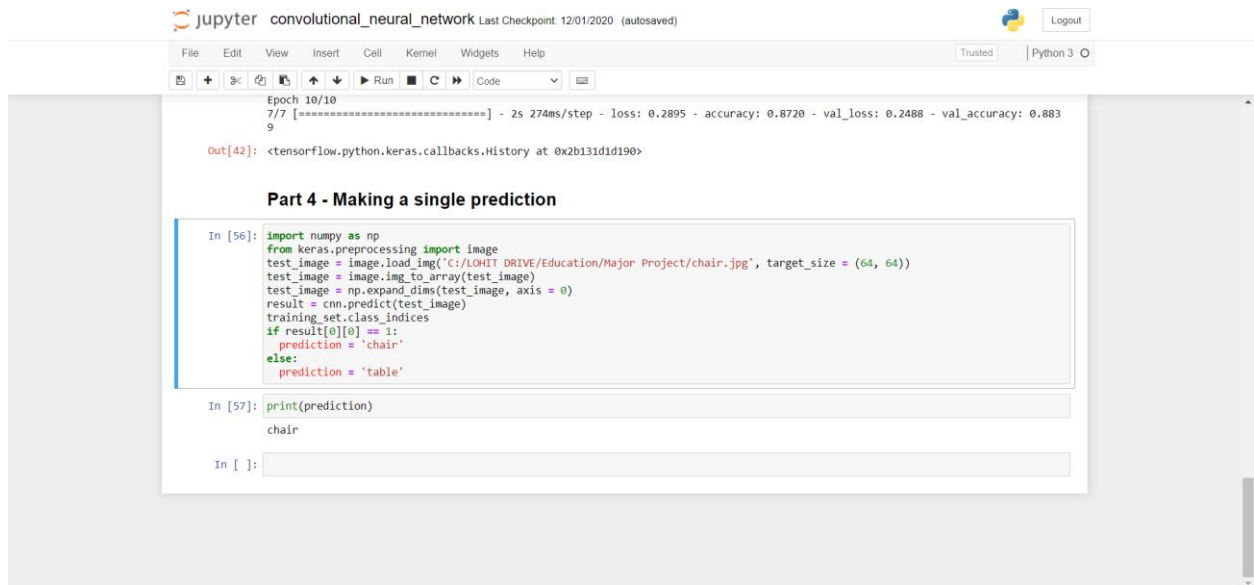
In [20]: cnn.fit(x = training_set, validation_data = test_set, epochs = 10)

Epoch 1/10
250/250 [=====] - 26s 104ms/step - loss: 0.4520 - accuracy: 0.7884 - val_loss: 0.5079 - val_accuracy: 0.7610
Epoch 2/10
250/250 [=====] - 27s 109ms/step - loss: 0.4048 - accuracy: 0.8155 - val_loss: 0.5236 - val_accuracy: 0.7575
Epoch 3/10
250/250 [=====] - 27s 108ms/step - loss: 0.3615 - accuracy: 0.8382 - val_loss: 0.5129 - val_accuracy: 0.7565
Epoch 4/10
250/250 [=====] - 26s 105ms/step - loss: 0.3139 - accuracy: 0.8606 - val_loss: 0.5341 - val_accuracy: 0.7645
Epoch 5/10
250/250 [=====] - 27s 110ms/step - loss: 0.2607 - accuracy: 0.8923 - val_loss: 0.5821 - val_accuracy: 0.7775
Epoch 6/10
250/250 [=====] - 28s 112ms/step - loss: 0.2125 - accuracy: 0.9136 - val_loss: 0.6352 - val_accuracy: 0.7545
Epoch 7/10
250/250 [=====] - 28s 111ms/step - loss: 0.1685 - accuracy: 0.9360 - val_loss: 0.6969 - val_accuracy: 0.7625
```

Fig 4.img

The above image depicts the training of the CNN using specified number of epochs.

5.2 RESULT



The image shows a Jupyter Notebook interface with the title 'convolutional_neural_network'. The top bar indicates the last checkpoint was on 12/01/2020 and it is autosaved. The notebook has a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for running, saving, and other actions. The main area displays the output of a Keras model training process, showing 'Epoch 10/10' and various metrics: '7/7 [=====] - 2s 274ms/step - loss: 0.2895 - accuracy: 0.8720 - val_loss: 0.2488 - val_accuracy: 0.8839'. Below this, the output of a cell is shown: 'Out[42]: <tensorflow.python.keras.callbacks.History at 0x2b131d1d190>'. The notebook then presents 'Part 4 - Making a single prediction'. This section contains three code cells. The first cell (In [56]) imports numpy and keras.preprocessing.image, loads an image from 'C:/LOHIT_DRIVE/Education/Major Project/chair.jpg', preprocesses it, and uses the trained CNN model to predict the class. The second cell (In [57]) prints the prediction, which is 'chair'. The third cell (In []) is empty.

```
Epoch 10/10
7/7 [=====] - 2s 274ms/step - loss: 0.2895 - accuracy: 0.8720 - val_loss: 0.2488 - val_accuracy: 0.8839

Out[42]: <tensorflow.python.keras.callbacks.History at 0x2b131d1d190>

Part 4 - Making a single prediction

In [56]: import numpy as np
from keras.preprocessing import image
test_image = image.load_img('C:/LOHIT_DRIVE/Education/Major Project/chair.jpg', target_size = (64, 64))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = cnn.predict(test_image)
training_set.class_indices
if result[0][0] == 1:
    prediction = 'chair'
else:
    prediction = 'table'

In [57]: print(prediction)
chair

In [ ]:
```

Part 5.img

The above image shows the successful running of the program and the expected output.

CHAPTER 6

SUMMARY AND CONCLUSION

6.1 SUMMARY

Object recognition from an image was successful by the process of machine learning by python, since our project initiates by Convolutional Neural Network which is having various hidden layers that undergoes convolution process. By this method we can illustrate differences between objects from the different images

6.2 CONCLUSION

Object recognition and detection Deep learning started to have a huge impact on computer vision in 2012, when Hinton's group won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with deep learning [80]. Before that, there were attempts to apply deep learning to relatively small datasets and the obtained improvement was marginal compared with other computer vision methods. The computer vision community was not fully convinced that deep learning would bring revolutionary breakthrough without strong evidence on grand challenges until 2012. ILSVRC is one of the most important grand challenges in computer vision, and has drawn a lot of attention recently especially after the great success of deep learning in 2012. It was originally proposed in 2009 [36]. The challenge was to classify images collected from the web into 1,000 categories. Its training data includes more than one million images, much larger than other datasets previously used to evaluate deep learning, such as MNIST. This competition has been running for several years and many top computer vision groups participated in the competition. However, different computer vision systems for object recognition tended to converge and there was no real breakthrough until 2012. This section reviews the ILSVRC results from 2012 to 2014, so that readers can understand how fast deep learning has been developing in computer vision

REFERENCES

- [1] <https://in.mathworks.com/matlabcentral/fileexchange/59133-neural-network-toolbox-tm--model-for-alexnet-network>
- [2] H. Lee, R. Grosse, R. Ranganath, and A.Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In Proceedings of the 26th Annual International Conference on Machine Learning, pages 609–616. ACM, 2019
- [3] Deep Learning with MATLAB – matlab expo2018
- [4] Introducing Deep Learning with the MATLAB – Deep Learning E-Book provided by the mathworks.
- [5] <https://www.completegate.com/2017022864/blog/deep-machine-learning-images-lenet-alexnet-cnn/all-pages>
- [6] Berg, J. Deng, and L. Fei-Fei. Large scale visual recognition challenge 2017. www.image-net.org/challenges. 2017.
- [7] Fei-Fei Li, Justin Johnson and Serena Yueng, “Lecture 9: CNN Architectures” May 2017.
- [8] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. Computer Vision and Image Understanding, 106(1):59–70, 2017.

APPENDIX

RAW CODE:

```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "name": "convolutional_neural_network.ipynb",
      "provenance": [],
      "collapsed_sections": [],
      "toc_visible": true
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    }
  },
  "cells": [
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "3DR-eO17geWu",
        "colab_type": "text"
      },
      "source": [
        "# Convolutional Neural Network"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "EMefrVPCg-60",
        "colab_type": "text"
      },
    },
  ]
}
```

```

"source": [
    "### Importing the libraries"
]
},
{
    "cell_type": "code",
    "metadata": {
        "id": "sCV30xyVhFbE",
        "colab_type": "code",
        "colab": {}
    },
    "source": [
        "import tensorflow as tf\n",
        "from keras.preprocessing.image import ImageDataGenerator"
    ],
    "execution_count": 0,
    "outputs": []
},
{
    "cell_type": "code",
    "metadata": {
        "id": "FIleuCAjoFD8",
        "colab_type": "code",
        "colab": {}
    },
    "source": [
        "tf.__version__"
    ],
    "execution_count": 0,
    "outputs": []
},
{
    "cell_type": "markdown",
    "metadata": {
        "id": "oxQxCBWyogPE",

```

```

    "colab_type": "text"
  },
  "source": [
    "## Part 1 - Data Preprocessing"
  ]
},
{
  "cell_type": "markdown",
  "metadata": {
    "id": "MvE-heJNo3GG",
    "colab_type": "text"
  },
  "source": [
    "### Preprocessing the Training set"
  ]
},
{
  "cell_type": "code",
  "metadata": {
    "id": "0koUcJMJPtEBD",
    "colab_type": "code",
    "colab": {}
  },
  "source": [
    "train_datagen = ImageDataGenerator(rescale = 1./255,\n",
    "                                   shear_range = 0.2,\n",
    "                                   zoom_range = 0.2,\n",
    "                                   horizontal_flip = True)\n",
    "training_set = train_datagen.flow_from_directory('dataset/training_set',\n",
    "                                                  target_size = (64, 64),\n",
    "                                                  batch_size = 32,\n",
    "                                                  class_mode = 'binary')",
  ],
  "execution_count": 0,
  "outputs": []
}

```

```

},
{
  "cell_type": "markdown",
  "metadata": {
    "id": "mrCMmGw9pHys",
    "colab_type": "text"
  },
  "source": [
    "### Preprocessing the Test set"
  ]
},
{
  "cell_type": "code",
  "metadata": {
    "id": "SH4WzfOhpKc3",
    "colab_type": "code",
    "colab": {}
  },
  "source": [
    "test_datagen = ImageDataGenerator(rescale = 1./255)\n",
    "test_set = test_datagen.flow_from_directory('dataset/test_set',\n",
    "                                           target_size = (64, 64),\n",
    "                                           batch_size = 32,\n",
    "                                           class_mode = 'binary')\n",
    ],
  "execution_count": 0,
  "outputs": []
},
{
  "cell_type": "markdown",
  "metadata": {
    "id": "af8O4l90gk7B",
    "colab_type": "text"
  },
  "source": [

```

```

    "## Part 2 - Building the CNN"
  ]
},
{
  "cell_type": "markdown",
  "metadata": {
    "id": "ces1gXY2lmoX",
    "colab_type": "text"
  },
  "source": [
    "### Initialising the CNN"
  ]
},
{
  "cell_type": "code",
  "metadata": {
    "id": "SAUt4UMPIhLS",
    "colab_type": "code",
    "colab": {}
  },
  "source": [
    "cnn = tf.keras.models.Sequential()"
  ],
  "execution_count": 0,
  "outputs": []
},
{
  "cell_type": "markdown",
  "metadata": {
    "id": "u5YJj_XMI5LF",
    "colab_type": "text"
  },
  "source": [
    "### Step 1 - Convolution"
  ]
}

```

```

},
{
  "cell_type": "code",
  "metadata": {
    "id": "XPzPrMckl-hV",
    "colab_type": "code",
    "colab": {}
  },
  "source": [
    "cnn.add(tf.keras.layers.Conv2D(filters=32,      kernel_size=3,      activation='relu',
input_shape=[64, 64, 3]))"
  ],
  "execution_count": 0,
  "outputs": []
},
{
  "cell_type": "markdown",
  "metadata": {
    "id": "tf87FpvxmNOJ",
    "colab_type": "text"
  },
  "source": [
    "#### Step 2 - Pooling"
  ]
},
{
  "cell_type": "code",
  "metadata": {
    "id": "ncpqPl69mOac",
    "colab_type": "code",
    "colab": {}
  },
  "source": [
    "cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))"
  ],

```

```

    "execution_count": 0,
    "outputs": []
  },
  {
    "cell_type": "markdown",
    "metadata": {
      "id": "xaTOgD8rm4mU",
      "colab_type": "text"
    },
    "source": [
      "### Adding a second convolutional layer"
    ]
  },
  {
    "cell_type": "code",
    "metadata": {
      "id": "i_-FZjn_m8gk",
      "colab_type": "code",
      "colab": {}
    },
    "source": [
      "cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))\n",
      "cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))"
    ],
    "execution_count": 0,
    "outputs": []
  },
  {
    "cell_type": "markdown",
    "metadata": {
      "id": "tmiEuvTunKfk",
      "colab_type": "text"
    },
    "source": [
      "### Step 3 - Flattening"
    ]
  }

```

```

    ]
  },
  {
    "cell_type": "code",
    "metadata": {
      "id": "6AZeOGCvnNZn",
      "colab_type": "code",
      "colab": {}
    },
    "source": [
      "cnn.add(tf.keras.layers.Flatten())"
    ],
    "execution_count": 0,
    "outputs": []
  },
  {
    "cell_type": "markdown",
    "metadata": {
      "id": "dAoSECOM203v",
      "colab_type": "text"
    },
    "source": [
      "#### Step 4 - Full Connection"
    ]
  },
  {
    "cell_type": "code",
    "metadata": {
      "id": "8GtmUILd26Nq",
      "colab_type": "code",
      "colab": {}
    },
    "source": [
      "cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))"
    ],

```

```

    "execution_count": 0,
    "outputs": []
  },
  {
    "cell_type": "markdown",
    "metadata": {
      "id": "yTldFvbX28Na",
      "colab_type": "text"
    },
    "source": [
      "#### Step 5 - Output Layer"
    ]
  },
  {
    "cell_type": "code",
    "metadata": {
      "id": "1p_Zj1Mc3Ko_",
      "colab_type": "code",
      "colab": {}
    },
    "source": [
      "cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))"
    ],
    "execution_count": 0,
    "outputs": []
  },
  {
    "cell_type": "markdown",
    "metadata": {
      "id": "D6Xkl90snSDl",
      "colab_type": "text"
    },
    "source": [
      "## Part 3 - Training the CNN"
    ]
  }
]

```

```

},
{
  "cell_type": "markdown",
  "metadata": {
    "id": "vfrFQACEnc6i",
    "colab_type": "text"
  },
  "source": [
    "### Compiling the CNN"
  ]
},
{
  "cell_type": "code",
  "metadata": {
    "id": "NALksrNQpUIJ",
    "colab_type": "code",
    "colab": {}
  },
  "source": [
    "cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =
['accuracy'])"
  ],
  "execution_count": 0,
  "outputs": []
},
{
  "cell_type": "markdown",
  "metadata": {
    "id": "ehS-v3MIpX2h",
    "colab_type": "text"
  },
  "source": [
    "### Training the CNN on the Training set and evaluating it on the Test set"
  ]
},

```

```

{
  "cell_type": "code",
  "metadata": {
    "id": "XUj1W4PJptta",
    "colab_type": "code",
    "colab": {}
  },
  "source": [
    "cnn.fit(x = training_set, validation_data = test_set, epochs = 25)"
  ],
  "execution_count": 0,
  "outputs": []
},
{
  "cell_type": "markdown",
  "metadata": {
    "id": "U3PZasO0006Z",
    "colab_type": "text"
  },
  "source": [
    "## Part 4 - Making a single prediction"
  ]
},
{
  "cell_type": "code",
  "metadata": {
    "id": "gsSiWEJY1BPB",
    "colab_type": "code",
    "colab": {}
  },
  "source": [
    "import numpy as np\n",
    "from keras.preprocessing import image\n",
    "test_image = image.load_img('dataset/single_prediction/cat_or_dog_1.jpg',
target_size = (64, 64))\n",

```

```

"test_image = image.img_to_array(test_image)\n",
"test_image = np.expand_dims(test_image, axis = 0)\n",
"result = cnn.predict(test_image)\n",
"training_set.class_indices\n",
"if result[0][0] == 1:\n",
"    prediction = 'dog'\n",
"else:\n",
"    prediction = 'cat'"
],
"execution_count": 0,
"outputs": []
},
{
"cell_type": "code",
"metadata": {
"id": "ED9KB3I54c1i",
"colab_type": "code",
"colab": {}
},
"source": [
"print(prediction)"
],
"execution_count": 0,
"outputs": []
}
]
}

```