

DEEP LEARNING

(Gender Detection and Age Prediction)

Summer Internship Report Submitted in partial fulfillment

of the requirement for undergraduate degree of

Bachelor of Technology

In

Computer Science Engineering

By

Joshi Aniruddha

221710308021

Under the Guidance of

Mr.

Assistant Professor



Department Of Computer Science and Engineering

GITAM School of Technology
GITAM (Deemed to be University)
Hyderabad-502329

July 2020

DECLARATION

I submit this industrial training work entitled **“PREDICTING THE AGE AND GENDER USING FACE IMAGES”** to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of **“Bachelor of Technology”** in **“Computer Science Engineering”**. I declare that it was carried out independently by me under the guidance of **Mr.** Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD

Joshi Aniruddha

Date:

221710308021



GITAM (DEEMED TO BE UNIVERSITY)

Hyderabad-502329, India

Dated:

CERTIFICATE

This is to certify that the Industrial Training Report entitled **“PREDICTING THE AGE AND GENDER USING FACE IMAGES”** is being submitted by Joshi Aniruddha (221710308021) in partial fulfillment of the requirement for the award of **Bachelor of Technology in & Computer Science Engineering** at GITAM (Deemed To Be University), Hyderabad during the academic year 2019-20.

It is faithful record work carried out by him at the **Computer Science and Engineering Department**, GITAM University Hyderabad Campus under my guidance and supervision.

Mr.

Assistant Professor
Department of CSE

Dr.Phani Kumar

Professor and HOD
Department of CSE

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful completion of this internship.

I would like to thank respected **Dr. N. Siva Prasad**, Pro Vice Chancellor, GITAM Hyderabad and **Dr. CH. Sanjay**, Principal, GITAM Hyderabad

I would like to thank respected **Dr. Phani Kumar**, Head of the Department of Computer Science and Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a internship report. It helped me a lot to realize of what we study for.

I would like to thank the respected faculties **Mr.** who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Joshi Aniruddha

221710308021

ABSTRACT

Machine learning algorithms are used to predict the values from the data set by splitting the data set in to train and test and building Machine learning algorithms models of higher accuracy to predict the age and gender is the primary task to be performed on face images set My perception of understanding the given data set has been in the view of undertaking a client's requirement of overcoming the prediction analysis of age and gender.

Over the last decade, the rate of image uploads to the Internet has grown at a nearly exponential rate. This newfound wealth of data has empowered computer scientists to tackle problems in computer vision that were previously either irrelevant or intractable. Consequently, we have witnessed the dawn of highly accurate and efficient facial detection frameworks that leverage convolutional neural networks under the hood. Applications for these systems include everything from suggesting who to “tag” in Facebook photos to pedestrian detection in self-driving cars. However the next major step to take building off of this work is to ask not only how many faces are in a picture and where they are, but also what characteristics do those faces have. The goal of this project do exactly that by attempting to classify the age and gender of the faces in an image.

TABLE OF CONTENTS

CHAPTER 1

1. MACHINE LEARNING	01
1.1 INTRODUCTION TO MACHINE LEARNING	01
1.2 IMPORTANCE OF MACHINE LEARNING	01
1.3 USES OF MACHINE LEARNING	03
1.4 TYPES OF LEARNING ALGORITHMS	04
1.4.1 SUPERVISED LEARNING	04
1.4.2 UNSUPERVISED LEARNING	05
1.4.3 SEMI SUPERVISED LEARNING	06
1.5 RELATION BETWEEN DATA MINING, ML AND DEEP LEARNING	07

CHAPTER 2

2. DEEP LEARNING	08
2.1 KEY TAKEAWAYS	08
2.2 HOW DEEP LEARNING WORKS	08
2.3 DEEP LEARNING VS MACHINE LEARNING	08
2.4 USES	09
2.5 DIFFERENCE BETWEEN ARTIFICIAL INTELLIGENCE AND MACHINELEARNING	10
2.6 ALGORITHMS IN DEEP LEARNING	11

CHAPTER 3

3. PYTHON	13
3.1 INTRODUCTION TO PYTHON	13
3.2 HISTORY OF PYTHON	14
3.3 FEATURES OF PYTHON	15
3.4 HOW TO SETUP PYTHON	16
3.4.1 INSTALLATION(USING PYTHON IDLE)	16
3.4.2 INSTALLATION(USING ANACONDA)	16
3.5 PYTHON VARIABLE TYPES	18

3.5.1 PYTHON NUMBERS	19
3.5.2 PYTHON STRINGS	19
3.5.3 PYTHON LISTS	19
3.5.4 PYTHON TUPLES	20
3.6 PYTHON FUNCTION	21
3.6.1 DEFINING A FUNCTION	21
3.6.2 CALLING A FUNCTION	21
3.7 PYTHON USING OOP'S CONCEPTS	22
3.7.1 CLASS	22
3.7.2 INIT METHOD IN CLASS	23

CHAPTER 4

4.GENDER DETECTION AND AGE PREDICTION	24
4.1 PROBLEM STATEMENT	24
4.2 OBJECTIVE OF THE CASE STUDY	24
4.3 PRE-PREPROCESSING THE DATA	24
4.3.1 GETTING THE FACE IMAGES	24
4.3.2 IMPORTING THE REQUIRED PACKAGES	24
4.3.3 WHAT IS DNN?	25
4.3.4 DEFINING A FUNCTION TO DETECT THE FACES	25
4.3.5 SETTING THE INPUT VALUE FOR NETWORK	26
4.3.6 DETECTION OF FACES AND THE CONFIDENCE	26
4.3.7 DRAWING THE RECTANGLE AROUND THE DETECTED FACE	27
4.3.7.1 SAMPLE OUTPUT FOR CV.RECTANGLE() METHOD()	27
4.3.8 WHAT IS ARGPARSE MODULE	28

4.3.9 TAKING THE INPUT FROM USER BY ARGPARSE	28
4.4 WHAT IS CAFEEMODEL,PROTOTXT,PB AND PBTXT	28
4.4.1 THE CNN ARCHITECTURE	29
4.5 MODEL BUILDING AND NETWORK ARCHITECTURE	30
4.5.1 HOW TO FIND THE MEAN VALUES FOR AN IMAGE IN OPENCV	31
4.5.2 WHY AGE LIST AND GENDER LIST	31
4.5.3 MODEL MEAN VALUES,AGE LIST AND GENDER LIST	31
4.5.4 LOADING AND READING THE NETWORK	32
4.6 READING THE INPUT THROUGH CAMERA	32
4.6.1 SAMPLE OUTPUT FOR VIDEOCAPTURE() METHOD	33
4.7 EXTRACTING THE REGION OF INTEREST(ROI) AND CONSTRUCTING A BLOB FOR ROI	33
4.7.1 VISUALIZATION FOR SELECTING A ROI FROM AN IMAGE	34
4.8 PREDICTING THE GENDER	35
4.8.1 SAMPLE OUTPUT FOR PREDICTION THE GENDER	35
4.9 PREDICTING THE AGE	35
4.9.1 SAMPLE OUTPUT FOR PREDICTION OF AGE	36
4.10 PRINTING THE IMAGE,GENDER,AGE AND TIME	36
4.10.1 OUTPUT FOR A SAMPLE FACE IMAGE & FROM CAMERA FEED	37
5. LIMITATIONS OF THE MODEL	39
6. CONCLUSION	40
7. REFERENCES	40

LIST OF FIGURES

Figure 1.2 : The Process flow	02
Figure 1.4.2 : Unsupervised Learning	03
Figure 1.4.3 : Semi Supervised Learning	06
Figure 1.5 : Relation between Deep Learning, Machine Learning and Artificial Intelligence	07
Figure 2.5 : Difference between Artificial Intelligence and Machine Learning	10
Figure 3.2 : Python Programming History	14
Figure 3.3 : Features of Python	15
Figure 3.4.1: Python download	16
Figure 3.4.2: Anaconda download	17
Figure 3.4.2.1 : Jupyter notebook	18
Figure 3.7.1: Defining a Class	22
Figure 4.3.2: Importing the packages	24
Figure 4.3.4 : Function which detects the faces	25
Figure 4.3.5 : Providing the input	26
Figure 4.3.6 : Loop for detection of faces and confidence	26
Figure 4.3.7 : Drawing the rectangle on the faces	27
Figure 4.3.7.1 : Sample output for an image	27
Figure 4.3.9 : Parsing the input from user	28
Figure 4.4 : Workflow of Deep Learning Algorithm	29
Figure 4.5 : Loading the pre-trained models	30
Figure 4.5.1 : Example to find mean values for an image	31
Figure 4.5.3 : Defining the mean values of model,agelist and genderlist	31
Figure 4.5.4 : Reading the models using pre-defined functions of opencv	32
Figure 4.6 : Reading the input through camera feed	32
Figure 4.6.1 : How to provide current working directory	33
Figure 4.6.2: Providing the python file to run the camera stream	33

Figure 4.7 : Loop for detecting the Region of Interest (ROI)	33
Figure 4.7.1 : Visualization of ROI	34
Figure 4.8 : Gender Prediction	35
Figure 4.8.1 : Sample output for gender prediction	35
Figure 4.9 : Prediction of age	35
Figure 4.9.1 : Sample output for age prediction	36
Figure 4.10 : Printing the image,gender,age and time using opencv methods	36
Figure 4.10.1 : Sample output for an image	37
Figure 4.10.2 : Output image with gender and age prediction	37
Figure 4.10.2.1 : Output for an image frame through camera	38
Figure 5.1 : Sample image for checking the accuracy	41
Figure 5.2 : Output of the sample image after prediction	41

CHAPTER 1

MACHINE LEARNING

1.1 INTRODUCTION TO MACHINE LEARNING

Machine Learning(ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence(AI).

1.2 IMPORTANCE OF MACHINE LEARNING

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and “more items to consider” and “get yourself a little something” on Amazon are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today’s data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that’s in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works.

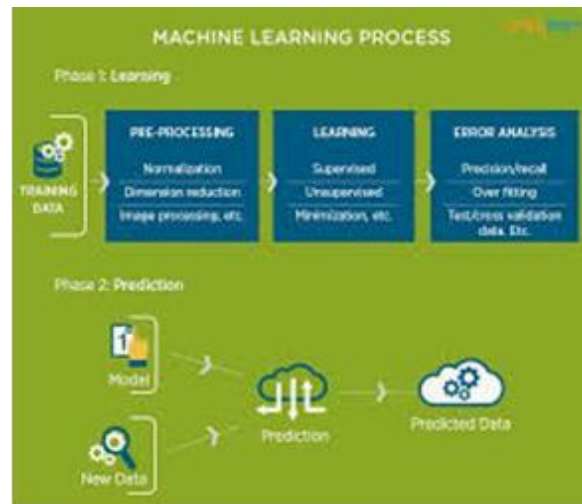


Figure 1.2 : The process flow

1.3 USES OF MACHINE LEARNING

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let's consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data.

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results.

The image recognition is one of the most common uses of machine learning applications. The face recognition is also one of the great features that have been developed by machine learning only. It helps to recognize the face and send the notifications related to that to people.

By using machine learning we can build the applications that predict the price of cab or travel for a particular duration congestion of traffic where can be found.

1.4 TYPES OF LEARNING ALGORITHMS

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

1.4.1 SUPERVISED LEARNING

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to “learn” how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

1.4.2 UNSUPERVISED LEARNING

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

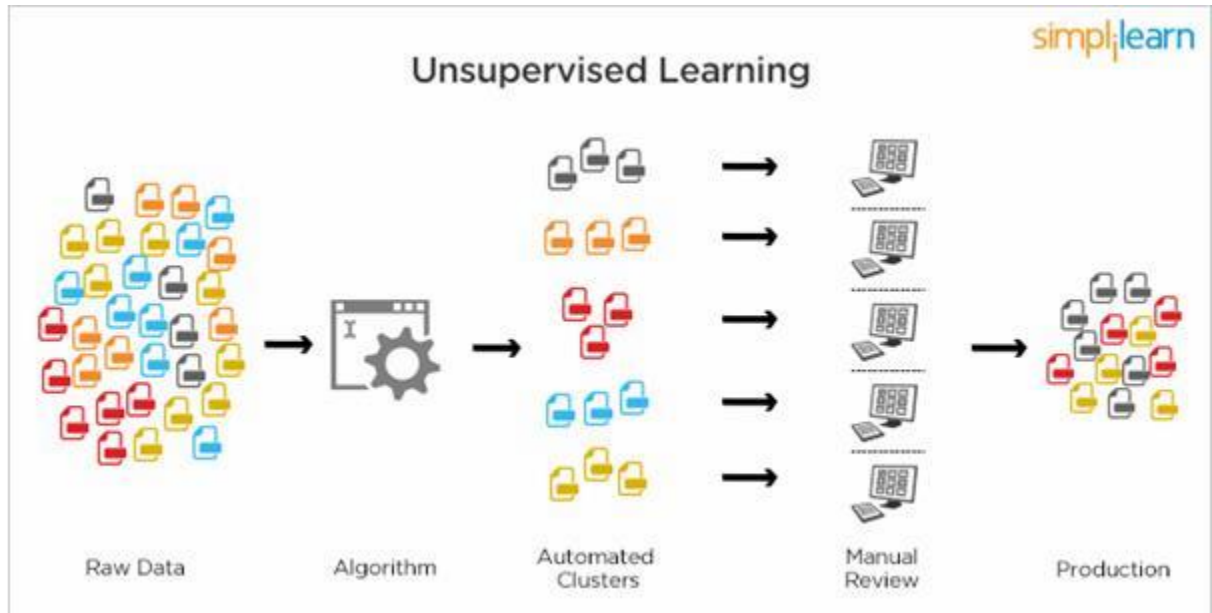


Figure 1.4.2 : Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

1.4.3 SEMI SUPERVISED LEARNING

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

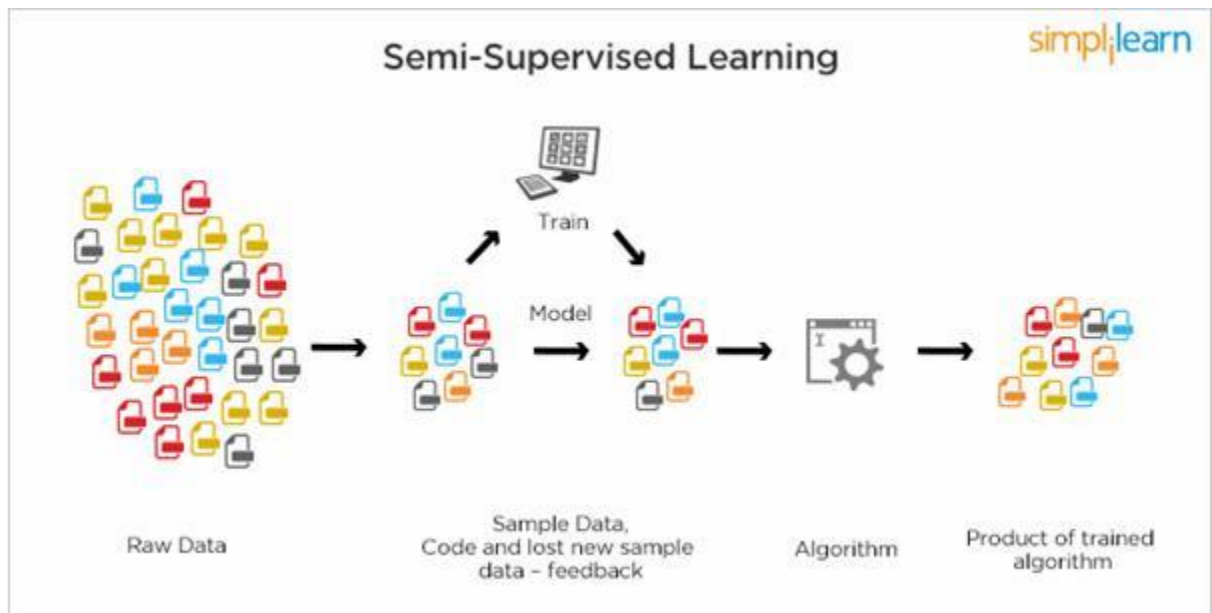


Figure 1.4.3: Semi Supervised Learning

1.5 RELATION BETWEEN DATA MINING, MACHINE LEARNING AND DEEP LEARNING

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns.

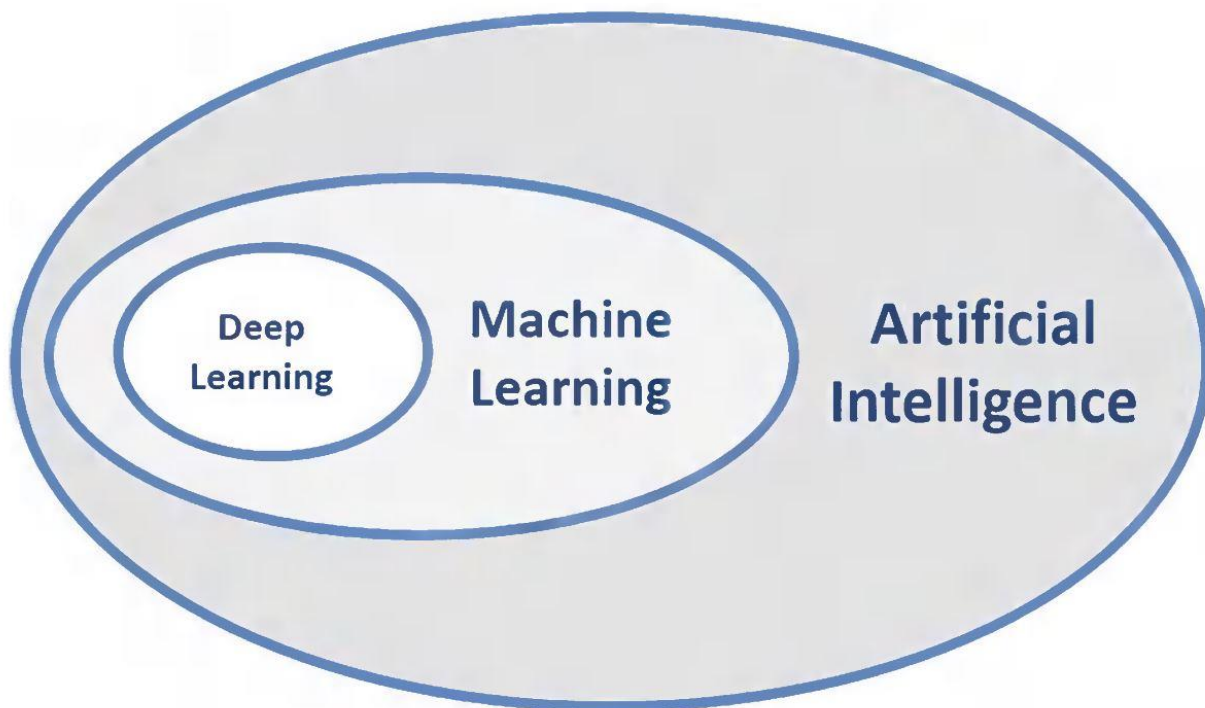


Figure 1.5 : Relation between Deep Learning, Machine Learning and Artificial Intelligence

CHAPTER 2

DEEP LEARNING

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

2.1 KEY TAKEAWAYS

Deep learning is an AI function that mimics the workings of the human brain in processing data for use in detecting objects, recognizing speech, translating languages, and making decisions.

Deep learning AI is able to learn without human supervision, drawing from data that is both unstructured and unlabeled.

Deep learning, a form of machine learning, can be used to help detect fraud or money laundering, among other functions.

2.2 HOW DEEP LEARNING WORKS

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.

2.3 DEEP LEARNING VS MACHINE LEARNING

One of the most common AI techniques used for processing big data is machine learning, a self-adaptive algorithm that gets increasingly better analysis and patterns with experience or with newly added data.

If a digital payments company wanted to detect the occurrence or potential for fraud in its system, it could employ machine learning tools for this purpose. The computational algorithm built into a computer model will process all transactions happening on the digital platform, find patterns in the data set, and point out any anomaly detected by the pattern.

Deep learning, a subset of machine learning, utilizes a hierarchical level of artificial

neural networks to carry out the process of machine learning. The artificial neural networks are built like the human brain, with neuron nodes connected together like a web. While traditional programs build analysis with data in a linear way, the hierarchical function of deep learning systems enables machines to process data with a nonlinear approach.

2.4 USES

Deep learning models are widely used in extracting high-level abstract features, providing improved performance over the traditional models, increasing interpretability and also for understanding and processing biological data. To predict splicing action of exons, a fully connected feedforward neural network was designed by Xiong et al. In recent years, CNNs were applied on the DNA dataset directly without the requirement of defining features a priority. Compared to a fully connected network, CNNs use less parameters by applying a convolution operation on the input data space and also parameters are shared between the regions. Hence, large DNA sequence data can be trained using these models and also improved pattern detection accuracy can be obtained.

Deepbind, a deep architecture based on CNNs, was proposed by Alipanathi et al. which predicts specificities of DNA and RNA binding proteins. CNNs were also used for predicting chromatin marks from a DNA sequence. Angermueller et al. have incorporated CNNs for predicting DNA methylation states. Like CNNs, other deep architectures were also applied for extracting features from raw DNA sequence data and for processing the data.

2.5 DIFFERENCE BETWEEN ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Artificial intelligence (AI) is the overarching discipline that covers anything related to making machines smart. Whether it's a robot, a refrigerator, a car, or a software application, if you are making them smart, then it's AI. Machine Learning (ML) is commonly used alongside AI but they are not the same thing. ML is a subset of AI. ML refers to systems that can learn by themselves. Systems that get smarter and smarter over time without human intervention. Deep Learning (DL) is ML but applied to large data sets. Most AI work now involves ML because intelligent behavior requires considerable knowledge, and learning is the easiest way to get that knowledge. The image below captures the relationship between AI, ML, and DL.

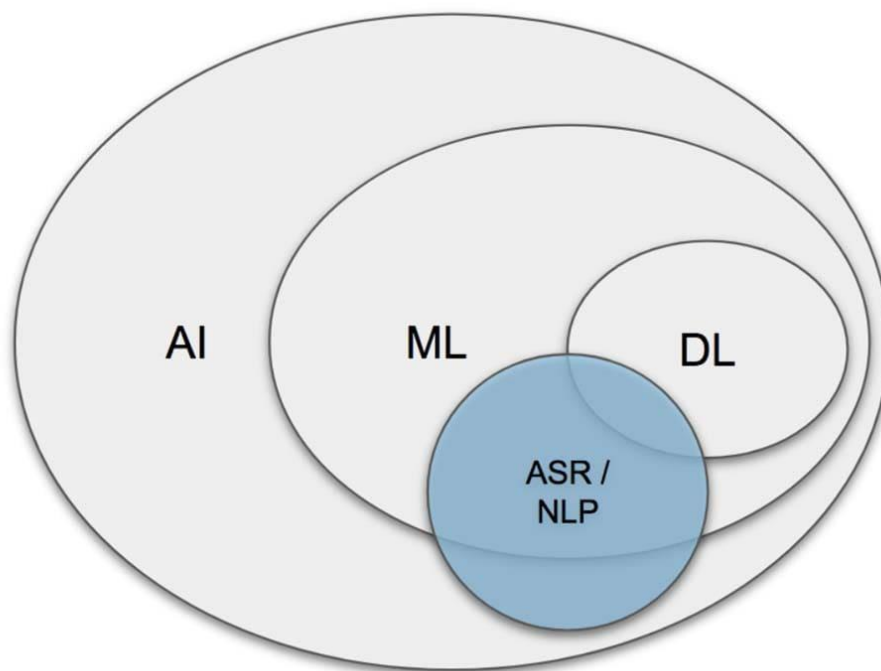


Figure 2.5 : Difference between Artificial Intelligence and Machine Learning

2.6 ALGORITHMS IN DEEP LEARNING

Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals. Artificial Intelligence. See these course notes for a brief intro of ml of ai and an introduction to deep learning algorithms

Deep Learning is about learning multiple levels of representation and abstraction that help to make sense of data such as images, sound, and text. For more about deep learning algorithms, see for example:

Things presented here will introduce you to some of the most important deep learning algorithms and will also show you how to run them using theano. Theano is a python library that makes writing deep learning models easy, and gives the option of training them on a GPU.

The algorithm tutorials have some prerequisites. You should know some python, and be familiar with numpy. Since this tutorial is about using Theano.

The purely supervised learning algorithms are meant to be read in order:

1. Logistic Regression - using Theano for something simple
2. Multilayer perceptron - introduction to layers
3. Deep Convolution Network- a simplified version of LeNet5

The unsupervised and semi-supervised learning algorithms can be read in any order (the auto-encoders can be read independently of the RBM/DBN thread):

- Auto Encoders, Denoising Autoencoders - description of autoencoders
- Stacked Denoising Auto-Encoders - easy steps into unsupervised pre-training for deep nets
- Restricted Boltzmann Machines - single layer generative RBM model
- Deep Belief Networks - unsupervised generative pre-training of stacked RBMs followed by supervised fine-tuning

Building towards including the mcRBM model, we have a new tutorial on sampling from energy models:

- HMC Sampling - hybrid (aka Hamiltonian) Monte-Carlo sampling with scan()

Building towards including the Contractive auto-encoders tutorial.

- Contractive auto-encoders code - There is some basic doc in the code.

Recurrent neural networks with word embeddings and context window:

- Semantic Parsing of Speech using Recurrent Net

LSTM network for sentiment analysis:

- LSTM network

Energy-based recurrent neural network (RNN-RBM):

- Modeling and generating sequences of polyphonic music

Segmentation for medical imagery (meant to be read in order):

- Fully Convolutional Networks (FCN) for 2D segmentation
- U-Net
- 1D segmentation

CHAPTER 3

PYTHON

3.1 INTRODUCTION TO PYTHON

Python is a high-level, interpreted, interactive and object-oriented system scripting language. Python is a programming language that is often applied in scripting roles. Basic programming language used for machine learning is Python.

Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.

Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects. Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc). It has a simple syntax similar to the English language. Python has syntax that allows developers to write programs with fewer lines than some other programming languages.

Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick. Python can be treated in a procedural way, an object-orientated way or a functional way.

Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses. Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

3.2 HISTORY OF PYTHON

Python was developed by Guido Van Rossum in early 1990's. It's latest version is 3.7, it is generally called as Python 3.

Python reached version 1.0 in January 1994. The major new features included in this release were the functional programming tools `lambda`, `map`, `filter` and `reduce`. Van Rossum stated that "Python acquired `lambda()`, `reduce()`, `filter()` and `map()`, courtesy of a Lisp hacker who missed them and submitted working patches.

The last version released while Van Rossum was at CWI was Python 1.2. In 1995, Van Rossum continued his work on Python at the Corporation for National Research Initiatives (CNRI) in Reston, Virginia from where he released several versions.

By version 1.4, Python had acquired several new features. Notable among these are the Modula-3 inspired keyword arguments (which are also similar to Common Lisp's keyword arguments) and built-in support for complex numbers. Also included is a basic form of data hiding by name mangling, though this is easily bypassed.

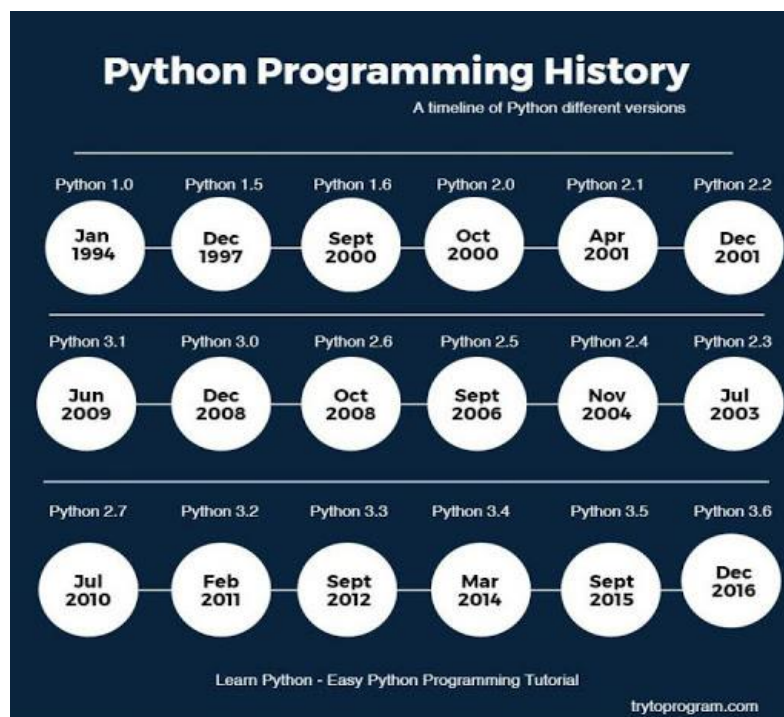


Figure 3.2 : Python Programming History

3.3 FEATURES OF PYTHON

- Easy-to-read: Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain: Python's source code is fairly easy-to-maintaining.
- A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases: Python provides interfaces to all major commercial databases.
- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

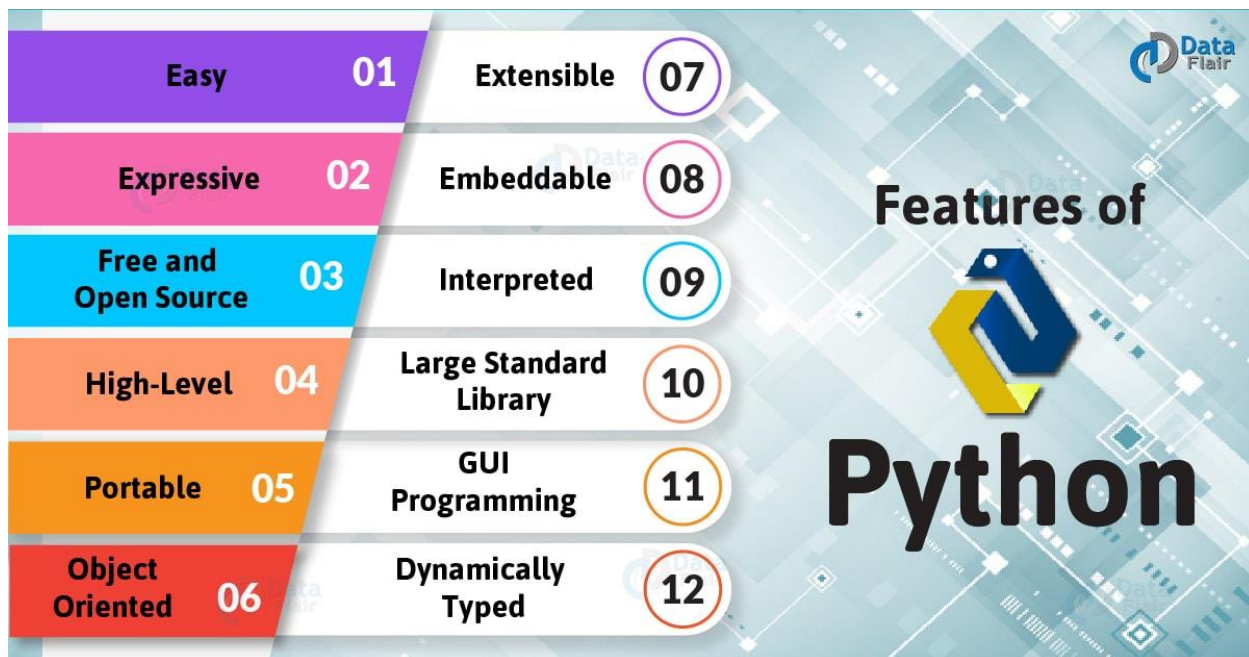


Figure 3.3 : Features of Python

3.4 HOW TO SETUP PYTHON

- If you need to install Python, you may as well download the most stable version. This is the one with the highest number that isn't marked as an alpha or beta release.

3.4.1 INSTALLATION(USING PYTHON IDLE)

- Download python from www.python.org
- When the download is completed, double click the file and follow the given instructions to install it.
- When python is successfully installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.

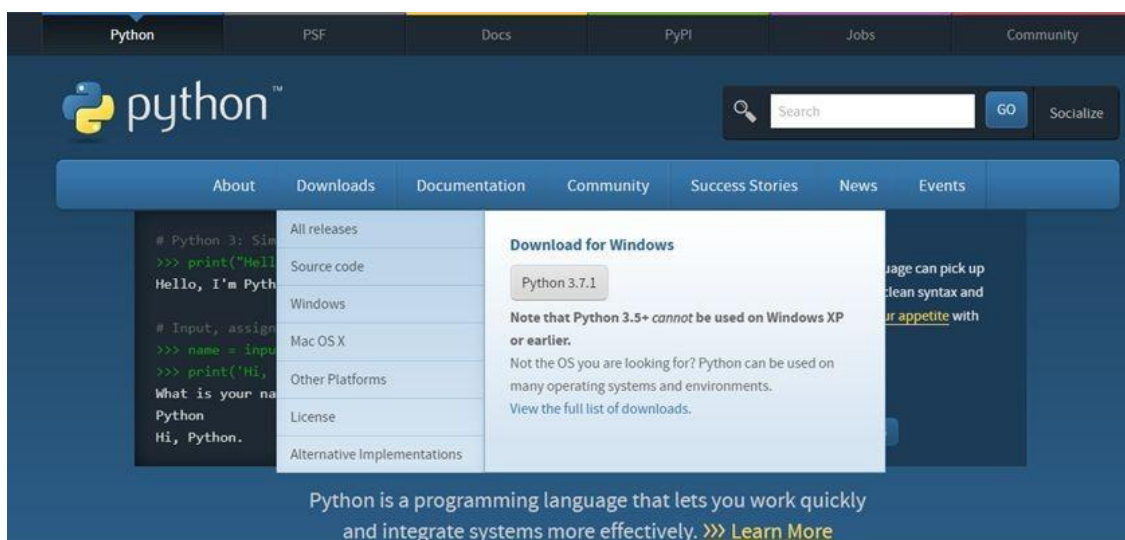


Figure 3.4.1 : Python download

3.4.2 INSTALLATION(USING ANACONDA)

- Python programs are also executed using Anaconda.
- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
- Conda is a package manager quickly installs and manages packages.

- In Windows:
- Step 1: Open Anaconda.com/downloads in web browser.
- Step 2: Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)
- Step 3: select installation type(all users).
- Step 4: Select path(i.e. add anaconda to path & register anaconda as default python 3.4) next click install and next click finish.
- Step 5: Open jupyter notebook (it opens in default browser).

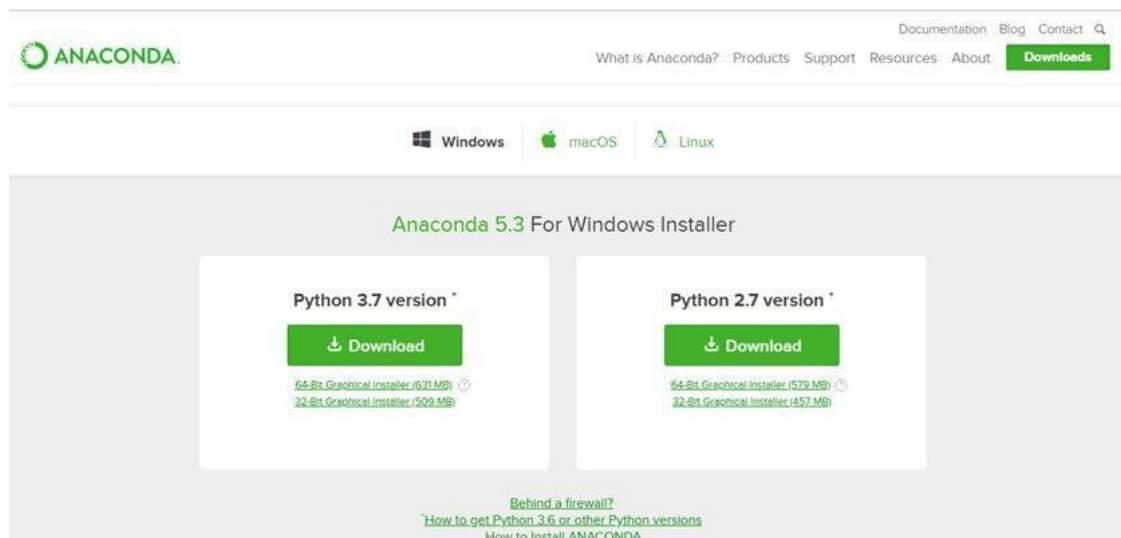


Figure 3.4.2 : Anaconda download



Figure 3.4.2.1 : Jupyter notebook

3.5 PYTHON VARIABLE TYPES

- Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
- Variables are nothing but reserved memory locations to store values.
- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.
- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.
- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.
- Python has five standard data types – Numbers, Strings, Tuples, Dictionary and Lists.

3.5.1 PYTHON NUMBERS

- Number data types store numeric values. Number objects are created when you assign a value to them.
- Python supports four different numerical types – int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

3.5.2 PYTHON STRINGS

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

3.5.3 PYTHON LISTS

- A list contains items separated by commas and enclosed within square brackets([]).
- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

- The values stored in a list can be accessed using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

3.5.4 PYTHON TUPLES

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([]) and their elements and size can be changed, while tuples are enclosed in parentheses (()) and cannot be updated.
- Tuples can be thought of as read-only lists.
- For example – Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple.
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).
- You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.
- What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

3.6 PYTHON FUNCTION

3.6.1 DEFINING A FUNCTION

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword `def` followed by the function name and parentheses (i.e.()).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses. The code block within every function starts with a colon (:) and is indented. The statement `return [expression]` exits a function, optionally passing back an expression to the caller.

3.6.2 CALLING A FUNCTION

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

3.7 PYTHON USING OOP's CONCEPTS

3.7.1 CLASS

- **Class:** A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- **Class variable:** A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- **Data member:** A class variable or instance variable that holds data associated with a class and its objects.
- **Instance variable:** A variable that is defined inside a method and belongs only to the current instance of a class.
- **Defining a Class:**
 - We define a class in a very similar way how we define a function.
 - Just like a function ,we use parentheses and a colon after the class name(i.e. ():) when we define a class. Similarly, the body of our class is
 - indented like a functions body is.

```
def my_function():  
    # the details of the  
    # function go here
```

```
class MyClass():  
    # the details of the  
    # class go here
```

Figure 3.7.1 : Defining a Class

3.7.2 INIT(__) METHOD IN CLASS

- The init method also called a constructor is a special method that runs when an instance is created so we can perform any tasks to set up the instance.
- The init method has a special name that starts and ends with two underscores: `init ()`.

CHAPTER 4

GENDER DETECTION AND AGE PREDICTION

4.1 PROBLEM STATEMENT:

To predict the Gender and Age of face images using Open CV and Convolutional Neural Networks.

4.2 OBJECTIVE OF THE CASE STUDY

In the modern world the age and gender prediction is most needed for computers in various aspects. Such as in validation of the clients working in a company. It is very difficult for an individual to check the gender and age for a mass group of people. In those situations a system which consists of prediction of age and gender is necessary to save the time and work. Google, Instagram and other AI detectors of face and gender feature use the machine learning algorithms and build the model. Then they deploy the machine learning algorithms to process the images.

MODEL BUILDING

4.3 PRE – PROCESSING THE DATA

Preprocessing of the data actually involves the following steps:

4.3.1: GETTING THE FACE IMAGES

We can get the face images from a user at real time or we can pass a set of images to the model.

4.3.2: IMPORTING THE REQUIRED PACKAGES

```
1 import cv2
2 import argparse
3 import time
4 import math
5 print(cv2.__version__)
6 print(argparse.__version__)
```

4.3.0
1.1

Figure 4.3.2 : Importing the packages

- OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel.
- Math module provides access to the mathematical functions defined by the C standard.
- Time(time) module which allows us to handle various operations regarding time.
- Argparse is the recommended command-line parsing module in Python programming standard library.

4.3.3 WHAT IS DNN?

- DNN stands for DotNetNuke which is a web content management system and web application framework. It is a Deep Learning framework.
- DNNs are typically feedforward networks in which data flows from the input layer to the output layer without looping back.
- DNNs can also store pre-trained model which can be deployed on a problem.
- For gender detection and age prediction problem a pre-trained model is used to solve the problem.
- From OpenCV 3.4 version we can use DNN models in which we can use pre-trained models. For face detections we have `getFaceBox` function.

4.3.4 DEFINING A FUNCTION TO DETECT THE FACES

```
def getFaceBox(net, frame, conf_threshold=0.7): #threshold is for min limit for detected face
    frameOpencvDnn = frame.copy() #dnn is for using the pretrained model
    frameHeight = frameOpencvDnn.shape[0]
    frameWidth = frameOpencvDnn.shape[1]
    blob = cv.dnn.blobFromImage(frameOpencvDnn, 1.0, (300, 300), [104, 117, 123], True, False)
```

Figure 4.3.4 : Function which detects the faces

- `conf_threshold` is for validation, whether the detected face is minimum of 70%. The default value of threshold in OpenCV is 0.5.
- Blob is a library for computer vision to detect connected regions in binary digital images.
- Blob stands for Binary large object. It is a collection of binary data stored as a single entity.
- `blobFromImage` creates 4-Dimensional blob from image. It performs the mean subtraction, scaling and optionally channel swapping.
- `frameHeight` and `frameWidth` stands for rows and columns pixel counts for further calculations in the problem.
- The syntax and parameters for `blobFromImage` are

`blobFromImage(image, scalefactor=1.0, size, mean, swapRB).`

4.3.5 SETTING THE INPUT VALUE FOR NETWORK

```
net.setInput(blob)
detections = net.forward() #for detecting the face
bboxes = [] #boundary boxes
```

Figure 4.3.5 : Providing the input

- net.forward() method gives Numpy ndarray as output which can be used for plot box on the input image.
- bbox is a Python library that is intended to ease the use of 2D and 3D bounding boxes in areas such as Object Detection.

4.3.6 DETECTION OF FACES AND THE CONFIDENCE

```
bboxes = [] #boundary boxes
for i in range(detections.shape[2]): #for loop for detection the face and to know confidence
    confidence = detections[0, 0, i, 2] #Extracting the confidence
    if confidence > conf_threshold:  #(if conf >0.7 then execute if block)
        # for object location
        x1 = int(detections[0, 0, i, 3] * frameWidth) # (x min)
        y1 = int(detections[0, 0, i, 4] * frameHeight) # (y min)
        x2 = int(detections[0, 0, i, 5] * frameWidth) # (x max)
        y2 = int(detections[0, 0, i, 6] * frameHeight) # (y max)
        bboxes.append([x1, y1, x2, y2])
        cv.rectangle(frameOpencvDnn, (x1, y1), (x2, y2), (0, 255, 0), int(round(frameHeight/150)), 8) #for
drawing rectangle box
    return frameOpencvDnn, bboxes
```

Figure 4.3.6 : Loop for detection of faces and confidence

- Extracting the confidence (i.e. probability) associated with the prediction.
- Filtering out the weak detections by ensuring that the confidence is greater than the minimum confidence.
- Computing the (x,y) coordinates of the bounding box for the object.
- In the above code, the image is converted to a blob and passed through the network using forward() function. The output detections is a 4-D matrix, where
 - i. The 3rd dimension iterates over the detected faces. (i is the iterator over the number of faces)
 - ii. The 4th dimension contains information about the bounding box and score for each face. For example, detections[0,0,0,2] gives the confidence score for the first face, and detections[0,0,0,3:6] gives the bounding box.

4.3.7 DRAWING THE RECTANGLE AROUND THE DETECTED FACE

```
cv.rectangle(frameOpencvDnn, (x1, y1), (x2, y2), (0, 255, 0), int(round(frameHeight/150)), 8) #for  
drawing rectangle box  
return frameOpencvDnn, bboxes
```

Figure 4.3.7 : Drawing the rectangle on the faces

- cv.rectangle() method is used to draw a rectangle on any image.
- Here the cv.rectangle() method draws the rectangle around the detected output face.
- The parameters for this method are cv2.rectangle(image , top left coordinate.

4.3.7.1 SAMPLE OUTPUT FOR CV.RECTANGLE METHOD:

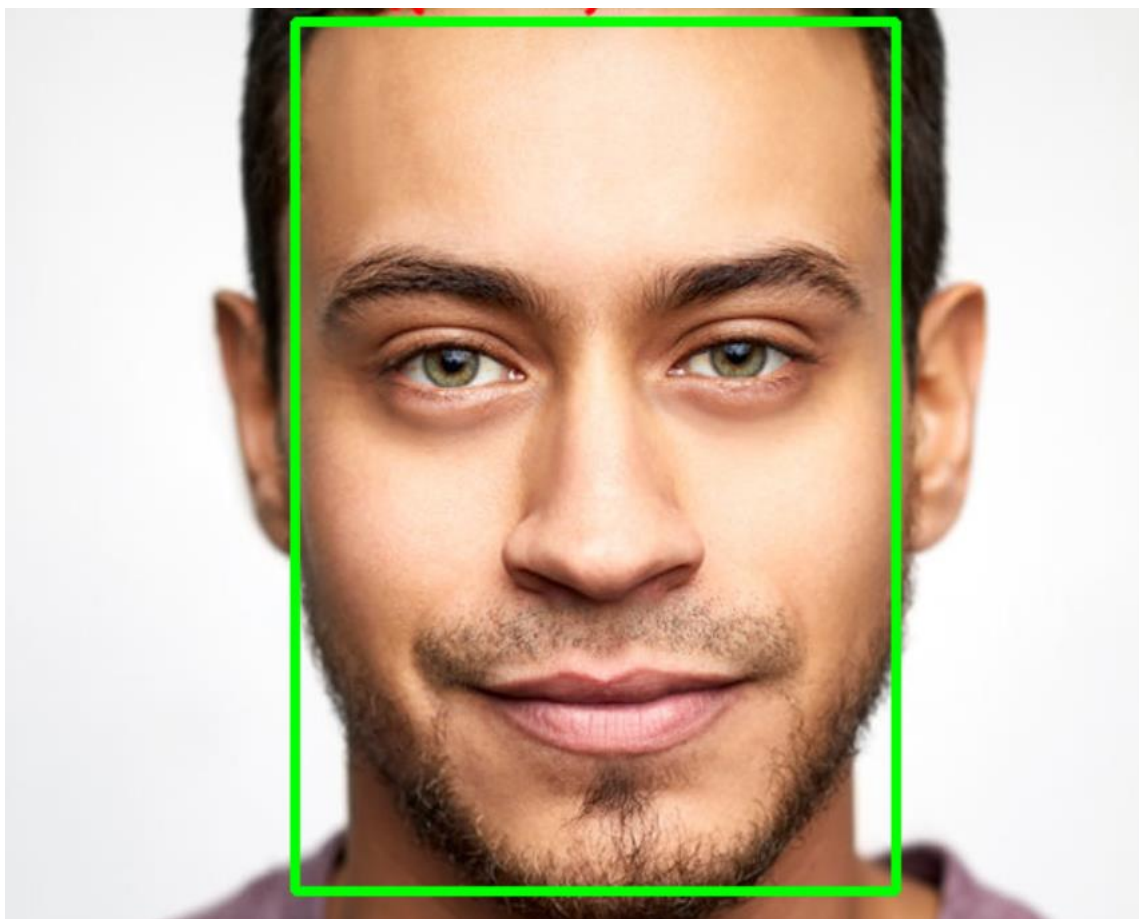


Figure 4.3.7.1 : Sample output for an image

4.3.8 WHAT IS ARGPARSE MODULE?

- Argparse is the recommended command-line parsing in the python standard library.
- The argparse module was added to python 2.7 as a replacement for optparse.
- The implementation of argparse supports features that would not have been easy to add to optparse.

4.3.9 TAKING THE INPUT FROM USER BY ARGPARSE

```
parser = argparse.ArgumentParser(description='To get input from user')
parser.add_argument("-i", help='Path to input image or video file.We can skip this argument to capture frames from a camera live feed.')

args = parser.parse_args()
```

Figure 4.3.9 : Parsing the input from user

- The first step in using the argparse is creating an ArgumentParser object.
- The ArgumentParser object will hold all the information necessary to parse the command line into Python data types.
- Filling an ArgumentParser with information about program arguments is done by making calls to the add_argument method.
- Generally, these calls tell the ArgumentParser how to take the strings on the command line and turn them into objects.

4.4 WHAT IS CAFEEMODEL , PROTOTXT , PB AND PBTXT ?

- Caffe is a deep learning framework made with expression, speed, and modularity in mind.
- It stands for Convolutional Architecture for Fast Feature Embedding.
- Speed makes Caffe perfect for research experiments and industry deployment.
- A prototxt file is the network architecture. In this file all the layers will be present, including data layer, convolutional layer, pooling, ReLU etc.
- These layers are used for the network architecture.
- In python pb stands for protobuf. In Tensorflow, the protobuf file contains the graph definition as well as the weights of the models. Thus, a pb file is all you need to be able to run a given trained model.
- ptxt file holds a network of nodes, each representing one operation, connected to each other as inputs and outputs.
- A cafeemodel file holds the weights of the pre-trained model. It is present in the binary format.

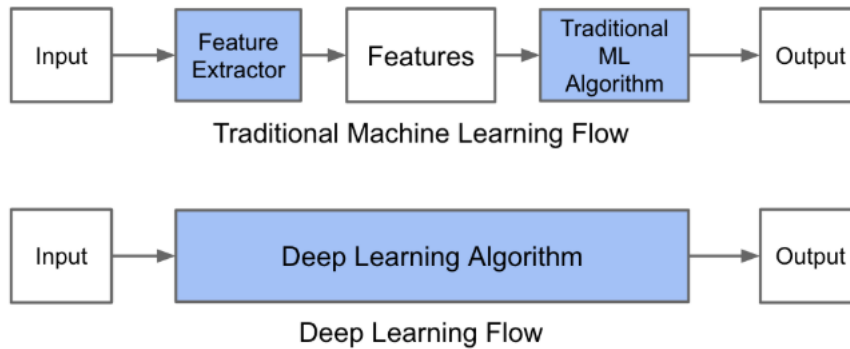


Figure 4.4 : Workflow of Deep Learning Algorithm

- The pre-trained models are taken from the opencv github repository. The links are placed below.

https://github.com/opencv/opencv_extra/blob/master/testdata/dnn/opencv_face_detector.pbtxt
https://github.com/opencv/opencv_extra/blob/master/testdata/dnn/opencv_face_detector.prototxt
https://github.com/opencv/opencv_extra/tree/master/testdata/dnn

4.4.1 THE CNN ARCHITECTURE

- The architecture of CNN is present in agedeploy.prototxt file.
- The convolutional neural network for this python project has 3 convolutional layers:
- Conv1- 96 filters of size 3x7x7 are convolved with stride 4 and padding 0, resulting in an output volume size of 96x56x56. This is followed by a ReLU, maxpooling pooling which reduces the size to 96x28x28, and a local-response normalization (LRN).
- Conv2- 256 filters of size 96x5x5 are convolved with stride 1 and padding 2, resulting in an output volume size of 256x28x28. This is also followed by a ReLU, max-pool, and LRN, reducing the output size to 256x14x14.
- Conv3- 256 filters of size 256x3x3 are convolved with stride 1 and padding 1, followed by a ReLU and maxpool, resulting in an output volume of 256x7x7.
- The fully connected layers are
- Fully Connected layer 6(fc6) - 512 neurons fully connected to the 256x7x7 output of Conv3, followed by a ReLU layer and dropout layer.
- Fully Connected layer 7(fc7) - 512 neurons fully connected to the 1x512 output of FC6 followed by a ReLU layer and dropout layer.
- FC8- 2 or 8 neurons fully connected to the 1x512 output of FC7, yielding the un-normalized class scores for either gender or age, respectively.

- Finally there is a softmax layer which gives the loss and final class probabilities.

4.5 MODEL BUILDING AND NETWORK ARCHITECTURE

```
#Network Architecture
faceProto = "opencv_face_detector.pbtxt"#Holds the network of nodes each represents one operation,
#connected to each other as inputs and outputs.
faceModel = "opencv_face_detector_uint8.pb"#Stores actual tensorflow program

ageProto = "age_deploy.prototxt"# used to describe the structure of the data to be serialized.
ageModel = "age_net.caffemodel"# This contains the information of the trained neural network.
# define the internal states of the parameters/gradients of the layers.

genderProto = "gender_deploy.prototxt"
genderModel = "gender_net.caffemodel"
```

Figure 4.5 : Loading the pre-trained models

- faceProto is assigned with “face_detector.pbtxt” pre-trained file which holds the network of nodes each represents one operation.
- faceModel is assigned with “face_detector_uint8.pb” pre-trained tensorflow program
- ageProto is assigned with “agedeploy.prototxt” pre-trained model which is used for serialization of data.
- ageModel is assigned with “age_net.caffemodel” pre-trained model which contains the information of trained model.
 - i. A caffemodel is used to integrate trained models into data pipelines.
 - ii. It is used to deploy against the new data from user through command line arguments.
- Similarly genderProto and genderModel are assigned with respective pre-trained models.

4.5.1 HOW TO FIND THE MEAN VALUES FOR AN IMAGE IN OPENCV

```
1 import cv2
2 import numpy as np

1 image_bgr = cv2.imread('10.jpg', cv2.IMREAD_COLOR)

1 channels = cv2.mean(image_bgr)
2 # Swap blue and red values (making it RGB, not BGR)
3 mean_values = np.array([(channels[2], channels[1], channels[0])])

1 mean_values

array([[136.9485384 , 109.91775168, 109.60328113]])
```

Figure 4.5.1 : Example to find mean values for an image

- The mean values for an image can be calculated using numpy and open cv module.
- For the pre-trained model used in this problem the mean values for RGB ordering are 78.42 , 87.76 , 114.89.

4.5.2 WHY AGE LIST AND GENDER LIST?

- The age list is used to predict the age in a particular range approximately.
- Here in age list variable 8 list items of age ranges are stored for predicting the age.
- Same as age list the gender list provides the gender of the individual present in the image.
- In gender list variable 2 list items are stored for prediction of gender.

4.5.3 MODEL MEAN VALUES, AGE LIST AND GENDER LIST

```
MODEL_MEAN_VALUES = (78.4263377603, 87.7689143744, 114.895847746)# Mean "RGB" values
ageList = ['(0-2)', '(4-6)', '(8-12)', '(15-20)', '(25-32)', '(38-43)', '(48-53)', '(60-100)']
genderList = ['Male', 'Female']
```

Figure 4.5.3 : Defining the mean values of model, agelist and gender list

- Model mean values are used for mean subtraction of RGB channels.
- Mean subtraction is used to help combat illumination changes in the input images in our data.

4.5.4 LOADING AND READING THE NETWORK

- The readNetFromCaffe method is used to read a network model stored in Caffe framework's format.
- The parameters are prototxt file and cafeemodel file.
- readNet is a method used to read deep learning network represented in one of the supported formats.

```
# Loading and reading the network  
  
ageNet = cv.dnn.readNetFromCaffe(ageProto,ageModel)#Reads a network model stored in Caffe framework's format.  
genderNet = cv.dnn.readNetFromCaffe(genderProto,genderModel)  
faceNet = cv.dnn.readNet(faceModel,faceProto)#Reading deep Learning network
```

Figure 4.5.4 : Reading the models using pre-defined functions of opencv

- Here cv.dnn.readNetFromCaffe is used for reading the ageProto and ageModel from the stored caffemodel framework.
- cv.dnn.readNet is used for reading faceModel and faceProto networks.

4.6 READING THE INPUT THROUGH CAMERA

```
# Open a video file or an image file or a camera stream  
cap = cv.VideoCapture(args.i if args.i else 0)  
padding = 20  
while cv.waitKey(1) < 0:  
    # Read frame  
    t = time.time()  
    hasFrame, frame = cap.read()  
    if not hasFrame:  
        cv.waitKey()  
        break  
    frameFace, bboxes = getFaceBox(faceNet, frame)  
    if not bboxes:  
        print("No face Detected, Checking next frame")  
        continue
```

Figure 4.6 : Reading the input through camera feed

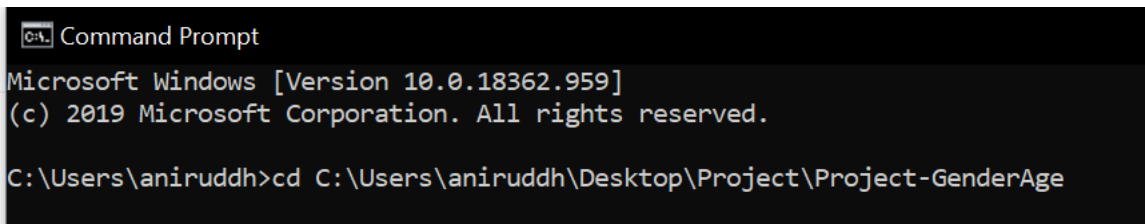
- OpenCV provides VideoCapture() method to get the image frame from the camera feed.
- Padding is used for adding borders for the frame.

- OpenCV also provides waitKey() method. The function waitKey() waits for a key event for a delay. It is calculated in milliseconds.

4.6.1 SAMPLE OUTPUT FOR VIDEOCAPTURE() METHOD

- To get the live camera feed frames as input to the program the following command should be provided during command-line arguments.

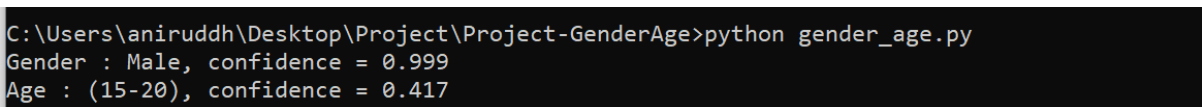
```
cd current working directory
python filename.py
```



```
Command Prompt
Microsoft Windows [Version 10.0.18362.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\aniruddh>cd C:\Users\aniruddh\Desktop\Project\Project-GenderAge
```

Figure 4.6.1: How to provide current working directory



```
C:\Users\aniruddh\Desktop\Project\Project-GenderAge>python gender_age.py
Gender : Male, confidence = 0.999
Age : (15-20), confidence = 0.417
```

Figure 4.6.2 : Providing the python file to run the camera stream

- By VideoCapture() method the prediction of age and gender is similar to my present age.

4.7 EXTRACTING THE REGION OF INTEREST(ROI) AND CONSTRUCTING A BLOB FOR ROI

```
for bbox in bboxes:
    face = frame[max(0, bbox[1]-padding):min(bbox[3]+padding, frame.shape[0]-1), max(0, bbox[0]-padding):min(bbox[2]+padding, frame.shape[1]-1)] # extract the ROI of the face and then construct a blob from ROI

    blob = cv.dnn.blobFromImage(face, 1.0, (227, 227), MODEL_MEAN_VALUES, swapRB=False)
    genderNet.setInput(blob)
    genderPreds = genderNet.forward()
    gender = genderList[genderPreds[0].argmax()]
```

Figure 4.7: Loop for detecting the Region of Interest(ROI)

- A region of interest (abbreviated as ROI), are samples within a data set identified for a particular purpose.

- `swapRB` is used for BGR ordering. Here `swapRB` is false because the mean values assumes the order as RGB.

4.7.1 Visualization for selecting a ROI from an image:

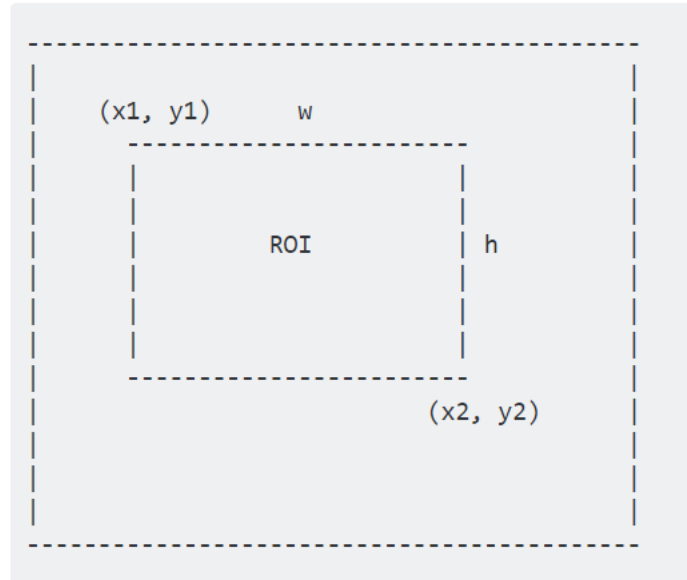


Figure 4.7.1 : Visualization of ROI

- Let us consider the outer boundary line as image image height and width.
- Consider $(0,0)$ as the top-left corner of the image with left-to-right as the x-direction and top-to-bottom as the y-direction. If we have $(x1,y1)$ as the top-left and $(x2,y2)$ as the bottom-right vertex of a ROI, we can use Numpy slicing to crop the image.
- Then the obtained cropped image will be the ROI for that image.

4.8 PREDICTING THE GENDER

```
genderNet.setInput(blob)
genderPreds = genderNet.forward()
gender = genderList[genderPreds[0].argmax()]

print("Gender : {}, confidence = {:.3f}".format(gender, genderPreds[0].max()))
```

Figure 4.8 : Gender Prediction

- From the above ROI blob, the blob is passed as input to the genderNet model, then from the genderList the gender is predicted.
- The print statement returns the gender with confidence in three digit format.

4.8.1 SAMPLE OUTPUT FOR PREDICTING THE GENDER

- For prediction of gender the following steps should be provided in command prompt.
 - i. Set the current working directory using “cd current working directory” command.
 - ii. Provide the input image or provide the image frame from live camera feed.

```
C:\Users\aniruddh\Desktop\Project\Project-GenderAge>cd C:\Users\aniruddh\Desktop\Project\Project-GenderAge
C:\Users\aniruddh\Desktop\Project\Project-GenderAge>python gender_age.py
Gender : Male, confidence = 0.996
```

Figure 4.8.1 : Sample output for gender prediction

4.9 PREDICTING THE AGE

```
ageNet.setInput(blob)
agePreds = ageNet.forward()
age = ageList[agePreds[0].argmax()]

print("Age : {}, confidence = {:.3f}".format(age, agePreds[0].max()))
```

Figure 4.9 : Prediction of age

- For age prediction the ROI blob is passed as input to the ageNet model, then from the ageList the model predicts the age.
- The print statement returns the age with confidence in three digit format.

4.9.1 SAMPLE OUTPUT FOR PREDICTION OF AGE

- For prediction of age the following steps should be provided in command prompt.
 - i. Set the current working directory using “cd current working directory” command.
 - ii. Provide the input image or provide the image frame from live camera feed.

```
C:\Users\aniruddh\Desktop\Project\Project-GenderAge>python gender_age.py
Gender : Male, confidence = 0.999
Age : (15-20), confidence = 0.417
```

Figure 4.9.1 : Sample output for age prediction

4.10 PRINTING THE IMAGE, GENDER, AGE AND TIME

```
label = "{},{}".format(gender, age)
cv.putText(frameFace, label, (bbox[0]-5, bbox[1]-10), cv.FONT_HERSHEY_SIMPLEX, 0.75, (0, 0, 255), 2,
cv.LINE_AA)
cv.imshow("Age Gender Demo", frameFace)
name = args.i
cv.imwrite('./detected/'+name, frameFace)

print("Time : {:.3f}".format(time.time() - t))
```

Figure 4.10 : Printing the image,gender,age and time using opencv methods

- Label contains the empty dictionary and the format as gender and then age.
- In OpenCV, cv.putText() method is used to draw a text string on any image.
- cv.imshow() method is used to display an image in a window.
- The cv.imwrite() method is used to save an image in local system. This method will save the image according to the specified format in current working directory.
- By using time module, the time taken for the execution of program is printed.

4.10.1 OUTPUT FOR A SAMPLE FACE IMAGE AND FROM CAMERA FEED

- Output for an image provided from local machine.

```
Command Prompt - python gender_age.py -i 1.jpg
Microsoft Windows [Version 10.0.18362.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\aniruddh>cd C:\Users\aniruddh\Desktop\Project\Project-GenderAge

C:\Users\aniruddh\Desktop\Project\Project-GenderAge>python gender_age.py -i 1.jpg
Gender : Male, confidence = 1.000
Age : (25-32), confidence = 0.933
Time : 0.225
```

Figure 4.10.1 : Sample output for an Image

- For executing a “.py” file first the current working directory should be set and then the following command should be executed.

`python filename.py -i imagename.jpg`

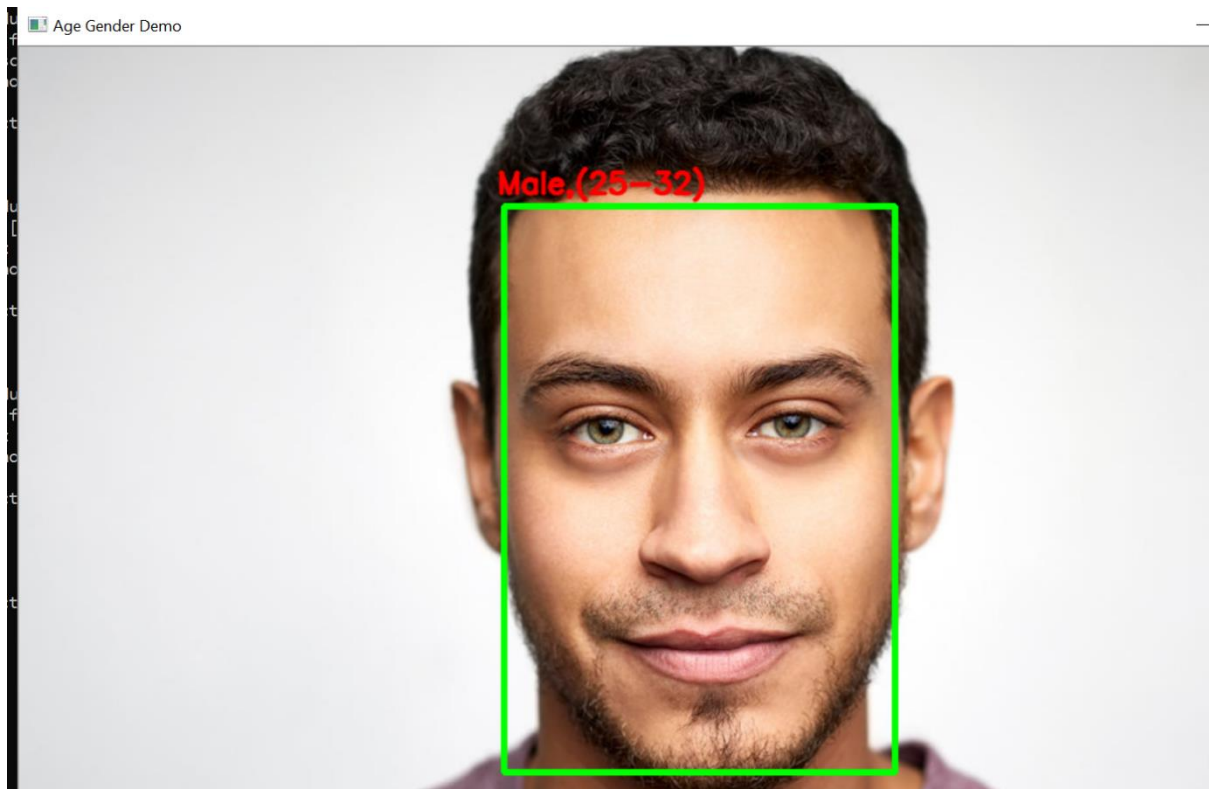


Figure 4.10.2 : Output image with gender and age prediction

Through camera feed:

- For taking image from camera stream use the following command.
 - i. Set the current working directory using “cd path to current working directory” command.
 - ii. Execute the “.py” file using “python filename.py” command.
 - iii. Then through VideoCapture() method the image frame is taken as input to the program.

The output will be as follows:

```
C:\Users\aniruddh\Desktop\Project\Project-GenderAge>python gender_age.py
Gender : Male, confidence = 0.999
Age : (15-20), confidence = 0.417
```

Figure 4.10.2.1 : Output for an image frame through camera

- The output will vary for every individual because of gender and age difference respectively.

5 LIMITATIONS OF THE MODEL

- The main drawback for this model is it can't detect the faces which may contain the blur background or any high contrast color background.
- The another limitation is the prediction of age and gender is not accurate with the images containing the uneasy background for the model.
- If there is a slight change in lighting conditions, it will make major impact on its results.
 - i. Below is an example with the output which shows the limitations of the opencv model.



Figure 5.1 : Sample image for checking the accuracy

- In the above image(Fig. 5.1) the face is slightly in blur condition which makes the model to predict either the age or gender inaccurately.
- When the image is provided to the model the following are the results:



Figure 5.2 : Output of the sample image after prediction

- As the output image shows the huge variation in the age due to slight changes in the lightning, this makes the model inaccurate in such conditions.

6. CONCLUSION

Two important conclusions can be made from our results. First, CNN can be used to provide improved age and gender prediction results, even considering the much smaller size of contemporary unconstrained image sets labeled for age and gender. Second, the simplicity of our model implies that more elaborate systems using more training data may well be capable of substantially improving results beyond those reported here.

7. REFERENCES

- [1] <https://en.wikipedia.org/wiki/OpenCV>
- [2] https://docs.opencv.org/master/d6/d00/tutorial_py_root.html
- [3] <https://www.pyimagesearch.com/category/deep-learning/>

Github Link [4] <https://github.com/Aniruddh2019/Project-DSPS->