A

PROJECT REPORT

ON

"Identification of Higgs Boson particle by Machine Learning Approach"

SUBMITTED TO

SHIVAJI UNIVERSITY, KOLHAPUR.

IN THE PARTIAL FULFILLMENT OF REQUIREMENT FOR THE AWARD OF DEGREE BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING.

SUBMITTED BY

Mr. Dhruv L. Patel	16ITTRCMPN71
Mr. Amit S. Mastake	14CMPN34
Mr. Sammed V. Neje	14CMPN40
Mr. Kedar V. Karpe	13CMPN27
Mr. Akshay B. Sanap	13CMPN48

UNDER THE GUIDANCE OF

MR. A. B. MAJGAVE

DKTE
Promoting Excellence in
Teaching, Learning & Research

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

D.K.T.E. SOCIETY'S
TEXTILE AND ENGINEERING INSTITUTE, ICHALKARANJI
(AN AUTONOUMOUS INSTITUTE)
(A+ Grade Accreditation by NAAC)

(ISO 9001:2015 CERTIFIED)

2018-2019

D.K.T.E.SOCIETY'S
TEXTILE AND ENGINEERING INSTITUTE, ICHALKARANJI
(AN AUTONOUMOUS INSTITUTE)
(A+ Grade Accreditation by NAAC)

(ISO 9001:2015 CERTIFIED)

2018-2019

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DKTE
Promoting Excellence in
Teaching, Learning & Research

CERTIFICATE

This is to certify that, project work entitled

"Identification of Higgs Boson particle by Machine Learning Approach"

Is a bonafide record of project work carried out in this college by

Mr. Dhruv L. Patel	16ITTRCMPN71
Mr. Amit S. Mastake	14CMPN34
Mr. Sammed V. Neje	14CMPN40
Mr. Kedar V. Karpe	13CMPN27
Mr. Akshay B. Sanap	13CMPN48

In the partial fulfillment of award of degree Bachelor of Engineering in Computer Science & Engineering prescribed by Shivaji University, Kolhapur for the academic year 2018-2019.

MR. A. B. MAJGAVE
(PROJECT GUIDE)

PROF. (DR.) D.V.KODAVADE
(HOD CSE DEPT.)

EXAMINER: ________________

PROF. (DR.) P.V.KADOLE
(DIRECTOR)

DECLARATION

We hereby declare that, the project work report entitled "Identification of Higgs Boson particle by Machine Learning Approach" which is being submitted to D.K.T.E. Society's Textile and Engineering Institute Ichalkaranji, affiliated to Shivaji University, Kolhapur is in partial fulfillment of degree B.E. (CSE). It is a bonafide report of the work carried out by us. The material contained in this report has not been submitted to any university or institution for the award of any degree. Further, we declare that we have not violated any of the provisions under Copyright and Piracy / Cyber / IPR Act amended from time to time.

Name	Roll No.	Signature
Mr. Dhruv L. Patel	16ITTRCMPN71	
Mr. Amit S. Mastake	14CMPN34	
Mr. Sammed V. Neje	14CMPN40	
Mr. Kedar V. Karpe	13CMPN27	
Mr. Akshay B. Sanap	13CMPN48	

ACKNOWLEDGEMENT

With great pleasure we wish to express our deep sense of gratitude to Mr. A. B. Majgave for his valuable guidance, support and encouragement in completion of this project report.

Also, we would like to take opportunity to thank our head of department Dr. D. V. Kodavade for his co-operation in preparing this project report.

We feel gratified to record our cordial thanks to other staff members of Computer Science and Engineering Department for their support, help and assistance which they extended as and when required.

Thank you,

Mr. Dhruv L. Patel	16ITTRCMPN71
Mr. Amit S. Mastake	14CMPN34
Mr. Sammed V. Neje	14CMPN40
Mr. Kedar V. Karpe	13CMPN27
Mr. Akshay B. Sanap	13CMPN48

ABSTRACT

In particle physics, Higgs Boson to tau-tau decay signals are notoriously difficult to identify due to the presence of severe background noise generated by other decaying particles. Our approach uses the Machine Learning Algorithms to classify the events as signals and background noise.

To capture and study the properties of Elementary Particles after the Proton-Proton collision in LHC is quite difficult. This is due to highly unstable nature of these sub-atomic particles, so eventually it results in coupling of these lighter particles to other heavier particles.

The proposed project is based on the identification of the Higgs Boson sub-atomic particle from jet pull energy dataset of the particles' decay, as modeled by the ATLAS Experiment in the Large Hadron Collider (LHC) at CERN.

For the performance our system uses the Google Colab tool, viz. free online environment. It provides GPU for processing the data and to give accurate results quickly. It runs entirely on the cloud and no pre-setup is required. Colab enables increase in computing performance by harnessing the power of GPU.

INDEX

INTRODUCTION

a. Problem Definition

To identify the Higgs Boson sub-atomic particle from the jet pull energy of ATLAS Experiment conducted by CERN in Large Hadron Collider (LHC), by performing the Data Analysis on the relevant dataset along with Machine Learning Approach.

b. Aim and Objective of the Project

The overall aim of the project is to capture the tau-tau decay for identification of Higgs Boson sub-atomic particle.

The system is proposed to have the following objectives.

1. To classify the events of the dataset into the Signal and background Noise.
2. To determine the Noise as background disturbance that contradicts the tau-tau decay.
3. To determine the Signal as 'tau-tau decay' to identify Higgs Boson (large particle) made up of tau leptons (smaller particles).
4. To use ensemble Machine Learning Algorithm to enhance Accuracy of the Signal and classifying it from the Noise.

c. Scope and Limitation of the Project

Scope:

1. The proposed system will run on the online GPU.
2. The system is the part of the research in Particle Physics.
3. In future, the system can be used in the discovery of other sub-atomic particles.

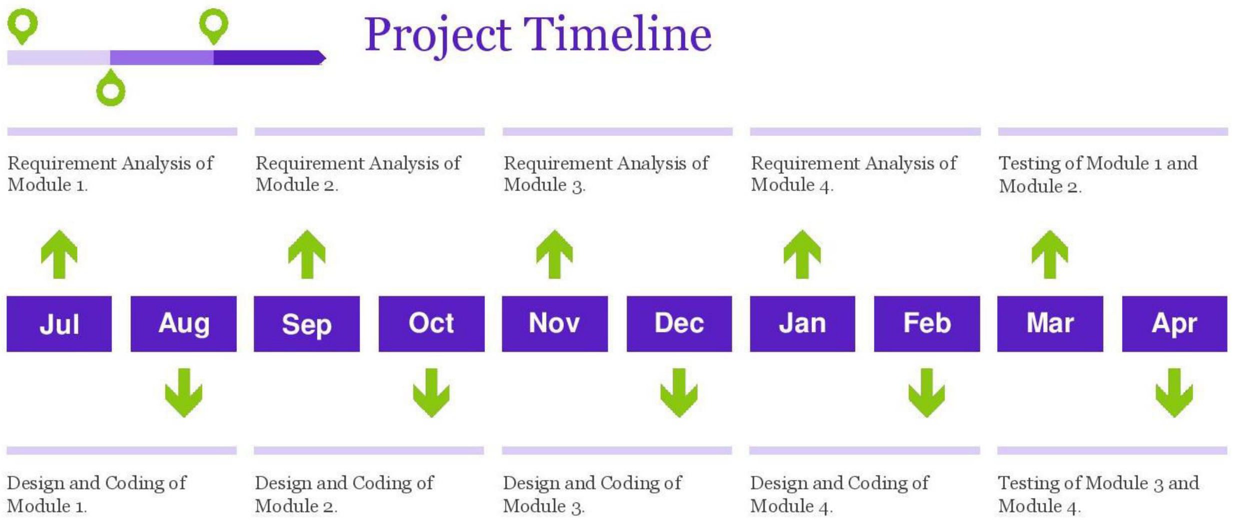
Limitation:

1. The system must use high performance computing, i.e. GPUs.
2. The system shall use third-party, Infrastructure as a Service (IAAS).
3. The system must have an Internet connection to execute the code.

d. Project Timeline

Here, the classic life cycle paradigm is used, which is also called “Water Fall Model”. For the software engineering it is a sequential approach for software development that begins at the system level and progress through analysis, design, coding testing and maintenance.

Module 1 and Module 2 are implemented up to October, 2018. After completion of Module 1 and Module 2, further Requirement Analysis, Design and Coding of Module 3 are completed up to December. Then the Requirement Analysis of Module 4 is completed up to January, 2019. Further, the part of Coding of Module 4 is completed up to February. Finally, the implementation and the testing on all the four modules are completed upto April, 2019.


Project Timeline
Requirement Analysis of Module 1.
Requirement Analysis of Module 2.
Requirement Analysis of Module 3.
Requirement Analysis of Module 4.
Testing of Module 1 and Module 2.
Jul
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
Design and Coding of Module 1.
Design and Coding of Module 2.
Design and Coding of Module 3.
Design and Coding of Module 4.
Testing of Module 3 and Module 4.

e. Project Cost

Hardware Cost:

Sr.No	Component	Quantity*price	Price
1.	Computer System	1*40000	40000
Total			**40000**

Software Cost:

Cost estimation based on COCOMO model:

In this project the Cost Estimation based on COCOMO (Constructive Cost Model) the formula for this Model is follows:

Effort = Constant * (Size) scale factor * Effort Multiplier

- Effort in terms of person-months.
- Constant: 2.45 in 1998 based on Organic Mode.
- Size: Estimated size in KLOC.
- Scale Factor: Combined process factors.
- Effort Multiplier (EM): Combined effort factors.

The basic COCOMO equations takes the form:

- Effort Applied (E) = a_b $(KLOC)^{b}{}_{b}$ [man-months]
- Development Time (D) = c_b $(\text{Effort Applied})^{d}{}_{b}$ [months]
- People required (P) = Effort Applied / Development Time [count]
- Productivity = KLOC / Effort Applied [KLOC/man-months]

Where, KLOC is the estimated number of delivered lines (expressed in thousands) of code for project.

The coefficients a_b, b_b, c_b and d_b are given in the following table:

Software Project	**a_b**	**b_b**	**c_b**	**d_b**
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Organic Mode:

Effort Applied (E) = $2.4*(0.255)^{1.05}$ = 0.57 Man-months.
Development Time (D) = $2.5*(0.57)^{0.38}$ = 2.02 Months.
People required (P) = 0.57/2.02 = 0.28 ~ 1 People.
Productivity: $\frac{KLOC}{E}$ = 0.4473.

Literature Overview

a. Literature Overview

Investigation of current Project and Related Work:

1. Dataset from the ATLAS Higgs Boson Machine Learning Challenge:

The dataset has been built from official ATLAS full-detector simulation, with "Higgs to tautau" events mixed with different backgrounds. The simulator has two parts. In the first, random proton-proton collisions are simulated based on the knowledge that have been accumulated on particle physics. It reproduces the random microscopic explosions resulting from the proton-proton collisions. In the second part, the resulting particles are tracked through a virtual model of the detector. The process yields simulated events with properties that mimic the statistical properties of the real events with additional information on what has happened during the collision, before particles are measured in the detector.

2. Quark-gluon tagging in the forward region of ATLAS at the LHC:

This paper presents an idea of the Particle collisions in the center of the ATLAS detector that cause the creation of observable high-energy particles, of which quarks and gluons comprise one of the largest physically “interesting” components. They deposit large signals, called “jets”, within the surrounding detector subsystems.

3. Neural Networks for calibrating ATLAS jets:

The Large Hadron Collider (LHC) collides bunches of 109 protons at energies of 14 TeV1 every 25 ns. These parameters lead to an average of 40 collisions per bunch crossing, but most of them won't correspond to head-on collisions, but rather a glancing collision, labeled as a pile-up (PU) event. Physicists at the LHC are interested in the high-energy or hard-scatter collision in a brunch crossing. ATLAS is one of the multi-purpose detectors at the LHC, and looks at the highly collimated streams of particles, or jets, from the collision. To reconstruct a jet's transverse momentum, pT , it must therefore correctly disentangle the true energy depositions and pile-up contributions that can cause extra energy to leak into a jet. This challenge will intensify over the next few years when the LHC plans to increase the average PU from 40 to 140.
The goal for this project was to use supervised ML techniques to improve the reconstruction of the jet's transverse momentum, pT . The input to our algorithm was the pTs cluster deposits and locations in the detector, and linear regression is used, a single layer perceptron, and a convolutional neural net to output the pT of the jet.

4. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC:

The paper presents a particular set of sensitive decay modes of the SM Higgs boson depends strongly on mH. The results presented in this paper are based on the five most sensitive decay modes in the low-mass region: H → γγ; H → ZZ followed by ZZ decays to 4`; H → WW followed by decays to 2`2ν; H → ττ followed by at least one leptonic τ decay; and H → bb followed by b-quark fragmentation into jets. This list is presented in Table 1 and comprises the full set of decay modes and subchannels, or categories, for which both the 7 and 8 TeV data sets have been analysed. Other lower sensitivity subchannels (ttH, H → bb; W/ZH, H → ττ; W/ZH, H → WW → 2`2ν; H → ZZ → 2`2q) have also been studied, so far only in the 7 TeV data, and are not included here. Adding these analyses in the combination results in an improvement of 0.1 σ in the overall expected local significance at mH = 125 GeV.

Decay Mode	Production tagging	No. of subchannels	mH range (GeV)	Int. Lum 7 TeV	(fb−1) 8 TeV
γγ	untagged dijet (VBF)	4 1 or 2	110-150	5.1	5.3
ZZ	untagged	3	110-160	5.1	5.3
WW	untagged dijet (VBF)	4 1 or 2	110-160	4.9	5.1
ττ	untagged dijet (VBF)	16 4	110-145	4.9	5.1
bb	lepton, E^{miss}	10	110-135	5.0	5.1

5. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC:

A search for the Standard Model Higgs boson in proton-proton collisions with the ATLAS detector at the LHC is presented. The datasets used correspond to integrated luminosities of approximately 4.8 fb−1 collected at √ s = 7 TeV in 2011 and 5.8 fb−1 at √ s = 8 TeV in 2012. Individual searches in the channels H→ ZZ(∗)→ 4ℓ, H→ γγ and H→ WW(∗)→ eνμν in the 8 TeV data are combined with previously published results of searches for H→ ZZ(∗) , WW(∗) , bb¯ and τ + τ − in the 7 TeV data and results from improved analyses of the H→ ZZ(∗)→ 4ℓ and H→ γγ channels in the 7 TeV data. Clear evidence for the production of a neutral boson with a measured mass of 126.0 ± 0.4 (stat) ± 0.4 (sys) GeV is presented. This observation, which has a significance of 5.9 standard deviations, corresponding to a background fluctuation probability of 1.7×10−9, is compatible with the production and decay of the Standard Model Higgs boson.

Requirement Analysis

Input: The .csv (comma-separated value) file.

1. Train set.
2. Test set.

Output: Classification and Prediction.

1. Binary Classification – Signal or Background.
2. Predicted values.
3. Pie chart.

Function Requirement:

i. **Data Cleansing and Data Wrangling:**

To clean the data by removing redundant features, if any and also create the derived features as per the existing features. Then, check for the null values in the dataset and replace the by the median strategy using Imputer() function. Further, perform the scaling operation on the dataset by using StandardScaler() function.

ii. **Data Visualization:**

To visualize the data, by using Count Plot from the Seaborn library. Also, displaying the percentage of Signals, i.e. label ‘s’ in the train set.

iii. **Data Validation:**

To shuffle and split the dataset into training and testing set. This is done by train_test_split() function from sklearn.model_selection library.

iv. **Machine Learning Model Building and Prediction:**

To initialize the Artificial Neural Network (ANN) by using Sequential() function from the Keras library. Now, add the 3 hidden layer and 1 output layer by ReLU and Sigmoid activation, respectively. Then compile the classifier by nadam optimizer and the loss function as binary_crossentropy. Further, fit the model on the classifier with the specified parameters and predict the results. Finally, display the F1 score and Accuracy for the training and testing set. Similarly, fit the model on XGBoost classifier from the xgboost library and then fit the model on the classifier, followed by the prediction and displaying the F1 score and Accuracy. Also, the distribution of the Signal and Background in prediction is depicted using Pie chart.

Software and Hardware Requirement:

Hardware Requirement:

- Tesla K80 GPU, provided by Google on the Colab tool.

Software Requirement:

- Windows 7 / 8 / 10.
- Google Colab tool.
- Anaconda Navigator.
- Jupyter Notebook.
- Numpy.
- Pandas.
- Sklearn.
- Seaborn.
- Matplotlib.
- Tensorflow.

Performance Requirements:

1. The environment being used is the Google Colab tool, viz. free online environment. It runs entirely on the cloud and no pre-setup is required. It provides the GPU, precisely a Tesla K80 GPU for enhancing the computing and subsequently reducing the total computation time.

2. The number of simultaneous users to be supported: single user.

3. Amount and type of information to be handled: the dataset taken has been modeled by the ATLAS Experiment in the Large Hadron Collider(LHC) at CERN as an input to our system. Dimensionality - It has approx. 800000 events with 33 features.

Software System Attributes:

- **Reliability**

The proposed software system is reliable and performs failure-free operation in a specified environment.

- **Availability**

The proposed software system will work as required during the period of a mission. The software system is available most of the time.

- **Security**

This should specify the factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure**.**

- **Maintainability**

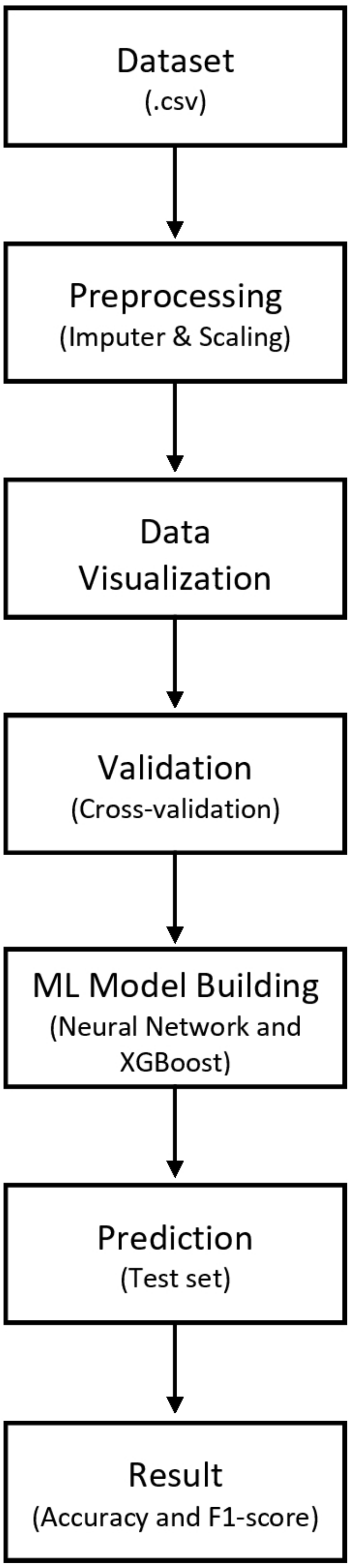
The proposed project is maintainable and capable of performing successful repair action in stipulated time. The system can be restored to operational status after a failure occurs.

- **Portability**

The proposed software system is portable and can be used in other environment other than the one in which it is created without requiring major network.

System Design

a. Architectural Design:



b. <u>Algorithmic</u> <u>Description</u>:

1] Import packages - numpy, pandas, seaborn, matplotlib and sklearn.

2] Read data from the CSV into a data frame by using read_csv() function of the pandas library.

3] Split the entire dataset into train set and test set.

4] Check the Missing Values in the train set and test set by using is_null() function of the pandas library.

5] Now, the missing values marked as -999.00 are replaced and filled using median strategy, by using Imputer() function of the sklearn library.

6] Here, some of the features possess high deviation. Hence, all the features are scaled, by using StandardScaler() function of the sklearn library.

7] The distribution of the target variable in the training set is visualized, depicting 's' as signal and 'b' as background is plotted, by using countplot() function of the seaborn library.

8] Now, the dataset is shuffled and split into training and testing set by using train_test_split() function of the sklearn library.

9] Building the Machine Learning model:

I] The Feed-Forward Artificial Neural Network (ANN) is implemented as follows:

1. Import the TensorFlow and the Keras library.

2. The first input layer of the ANN is initialized and constructed using the Sequential() function of the Keras library.

3. Now, add a hidden layer by using Dense() function of the Keras library, with the units = 16, activation = ReLU (Rectified Linear Unit) and input_dim = 30.

4. Add another hidden layer by using Dense() function of the Keras library, with the units = 8 and activation = ReLU (Rectified Linear Unit).

5. Add another hidden layer by using Dense() function of the Keras library, with the units = 6 and activation = ReLU (Rectified Linear Unit).

6. Finally, add the output layer by using Dense() function of the Keras library, with the units = 1 and activation = Sigmoid (for probability).

7. The above constructed ANN is compiled using compile() function of the Keras library, with the optimer = nadam and the loss = binary_crossentropy.

8. Now, the user defined function is used to convert the labels as:

i.) If label = 'b' (for background), then return 0.

ii.) Else, return 1.

9. Now, apply the conversion from above mentioned, user defined function to both the train set and test set, respectively.

10. Further, fit the classifier by using the fit() function of the Keras library with the batch size = 256 and the epochs = 20.

II] The Extreme Gradient Boosting (XGBoost) algorithm is implemented as follows:

1. Import the XGBoost library.

2. Now, initialize the classifier by using the XGBClassifier() function of the XGBoost library.

3. Further, fit the classifier by using the fit() function.

10] Now, predict the results by using the predict() function for the respective Machine Learning Algorithms used by the fit() function in training these two classifiers. Also, calculate the time taken by these classifiers in estimating the predictions.

11] To calculate the Accuracy and F1 score, as follows:

I] Import sklearn.metrics.

II] For the training set:

1. Calculate F1 score by using the f1_score() function of the sklearn library.

2. Calculate Accuracy by using the accuracy_score() function of the sklearn library.

III] For the test set:

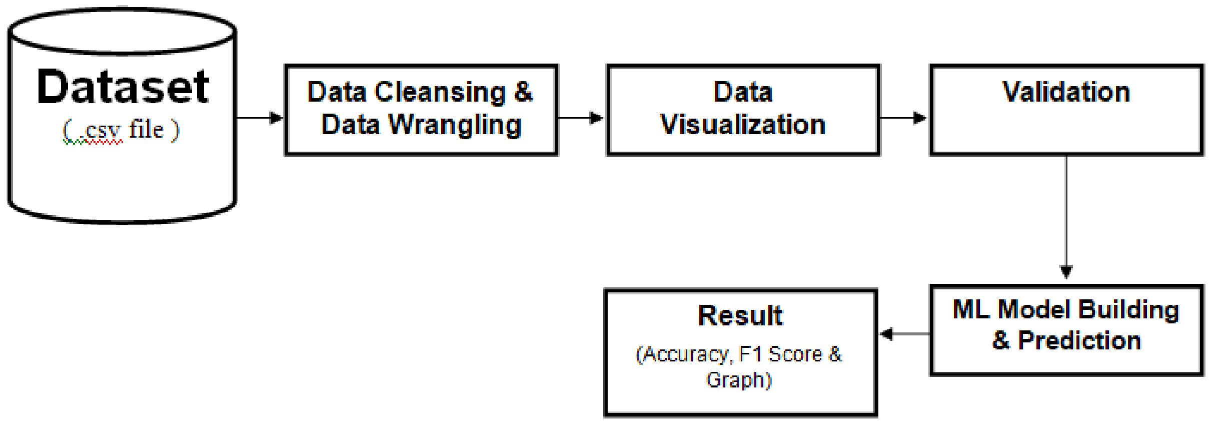
1. Calculate F1 score by using the f1_score() function of the sklearn library.

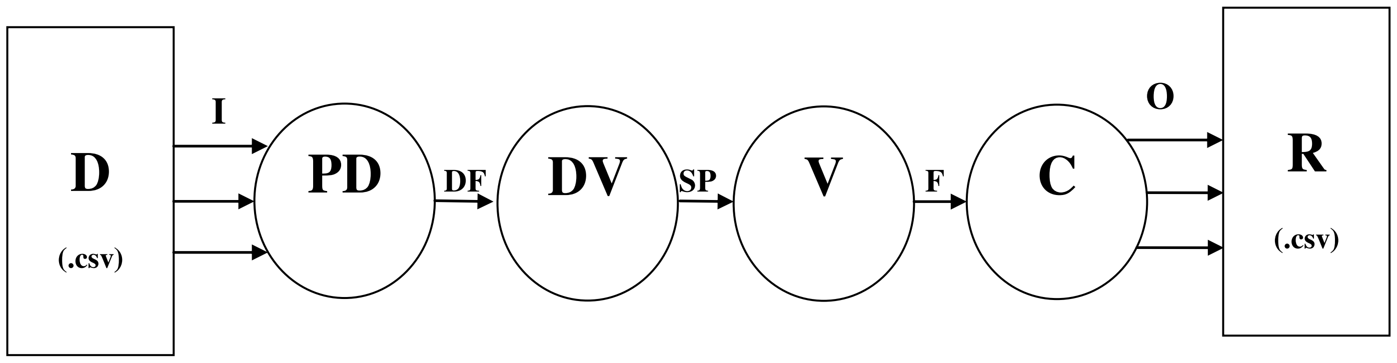
2. Calculate Accuracy by using the accuracy_score() function of the sklearn library.

12] Finally, plot the Pie-charts of the predicted results. These plots shall clearly depict the distribution of the Signal and Background.

c. System Modeling:

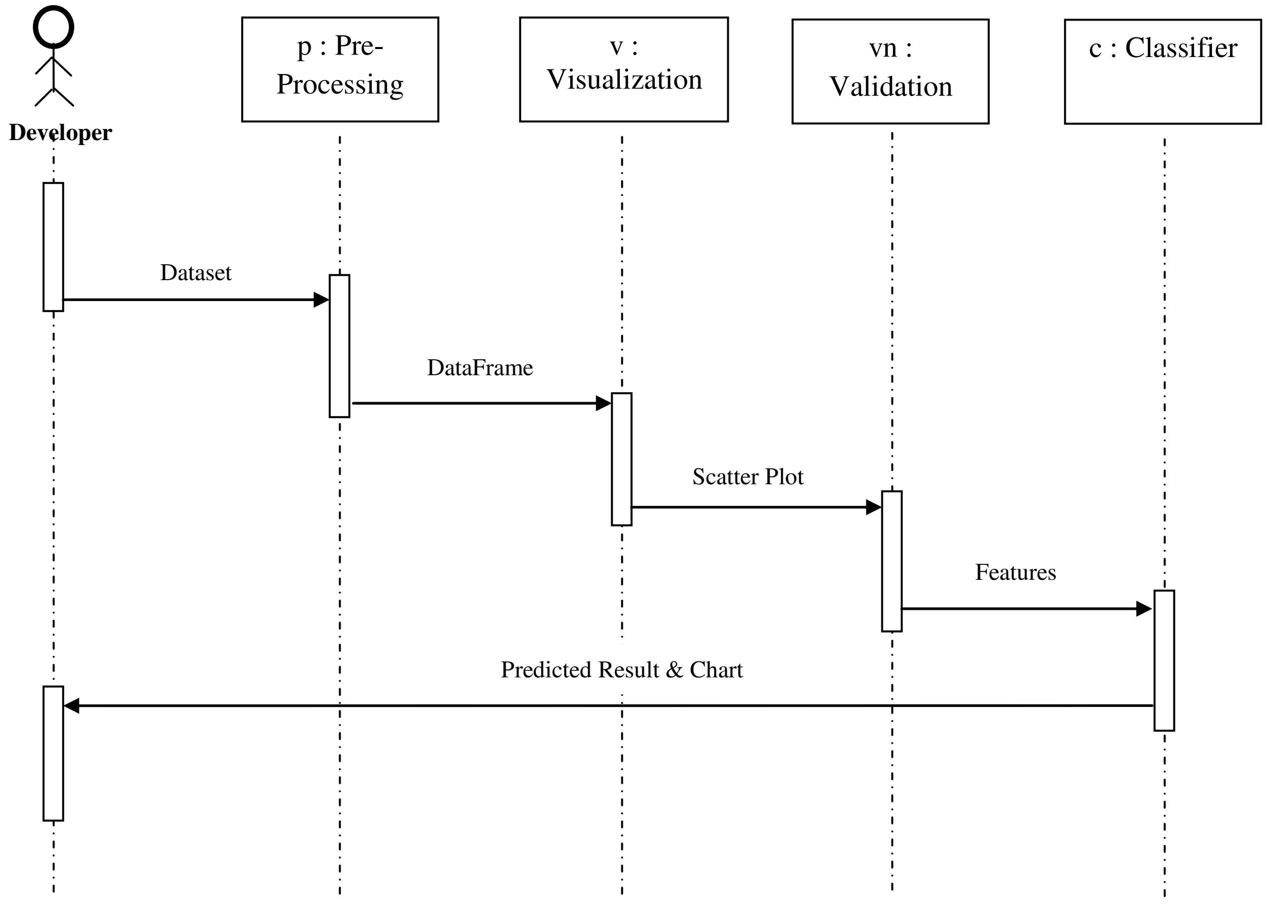
1. Block Diagram:


Dataset
(.csv file)
Data Cleansing & Data Wrangling
Data Visualization
Validation
ML Model Building & Prediction
Result
(Accuracy, F1 Score & Graph)

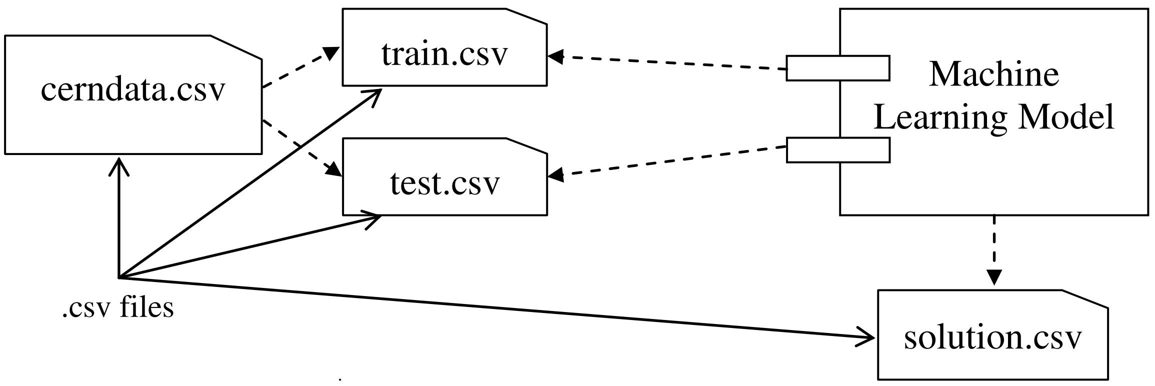


D :	Developer.
PD :	Preprocessing Dataset.
DV :	Data Visualization.
V :	Validation.
C :	Machine Learning Classifier.
R :	Result.
DF :	DataFrame.
SP :	Scatter Plot.
F :	Features.
I & O :	Input and Output.

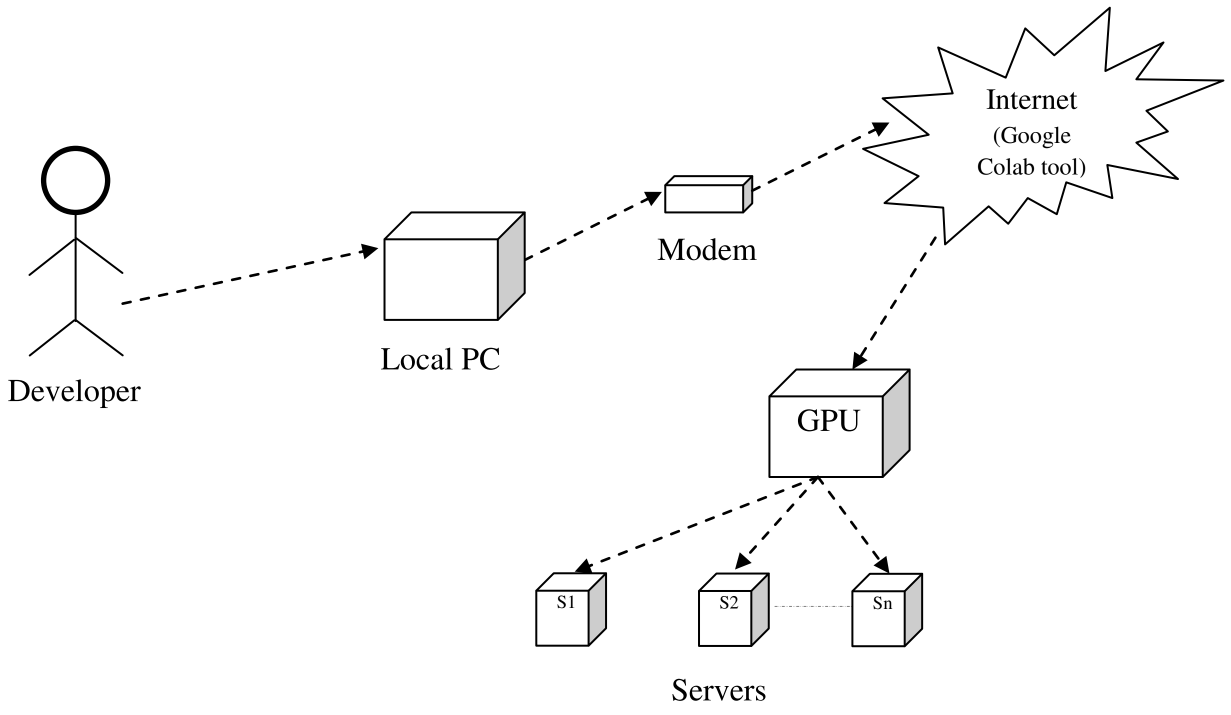
3. Sequence Diagram:



4. Component Diagram:



5. Deployment Diagram:



Implementation

```
from google.colab import drive
drive.mount('/content/gdrive')
```

Here, the Google Drive is mounted on the respective Google Colaboratory code file.

```
import pandas as pd
df = pd.read_csv('gdrive/My Drive/ATLAS Collaboration CERN Dataset.csv')
```

Here, the dataset is read from the Google Drive and stored in df variable, by using the Pandas library.

```
cnt = df.isnull().sum().sum()
```

Here, the null values in the dataset are checked by using isnull() function of the Pandas library.

```
imp = Imputer(missing_values = -999.00, strategy = 'median')
```

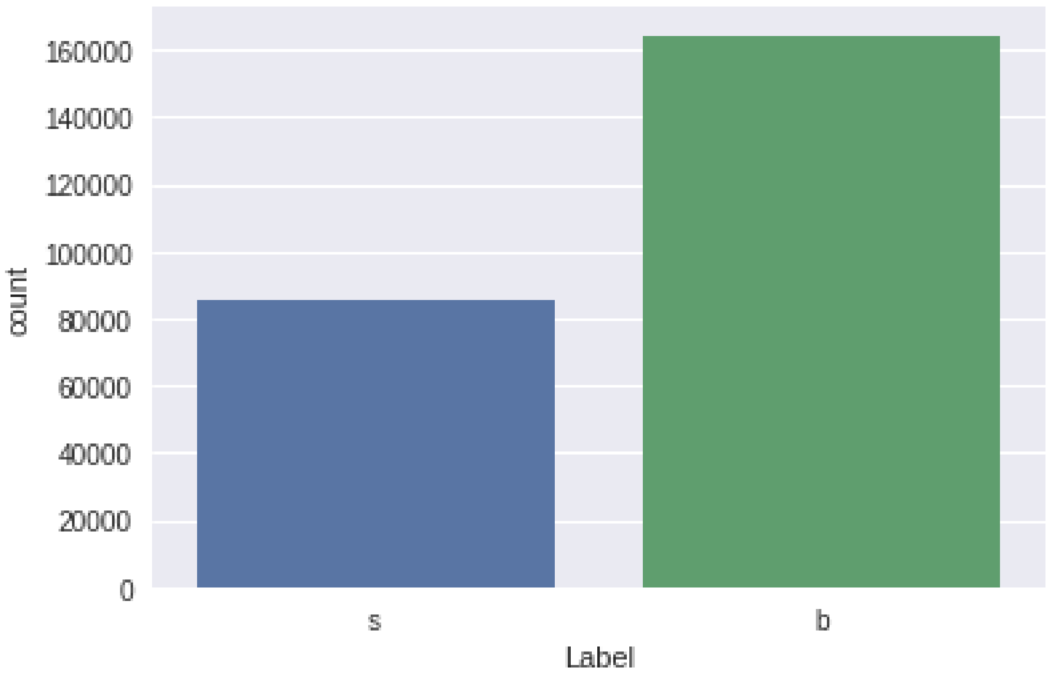
Here, the missing values marked as -999.00 are filled using median strategy, by using Imputer() function of the sklearn library.

```
scaler = StandardScaler()
```

Here, some of the features possess high deviation. Hence, all the features are scaled, by using StandardScaler() function of the sklearn library.

```
sns.countplot(y)
```

Here, the graph of the target variable in the training set, depicting ‘s’ as signal and ‘b’ as background is plotted, by using countplot() function of the Seaborn library.



```
train_test_split(X, y, test_size = 0.20, stratify = y)
```

Here, the dataset is shuffled and split into training and testing set by using train_test_split() function of the sklearn library.

```
classifier = Sequential()           #Initialising the ANN
classifier.add(Dense(units = 16, activation = 'relu', input_dim = 30))
classifier.add(Dense(units = 8, activation = 'relu'))
classifier.add(Dense(units = 6, activation = 'relu'))
classifier.add(Dense(units = 1, activation = 'sigmoid'))
```

Here, the the Artificial Neural Network (ANN) is initialized using Sequential() function of the Keras library, followed by Dense() function to add the hidden layers in ANN.

```
classifier.compile(optimizer = 'nadam', loss = 'binary_crossentropy')
```

Here, the classifier is compiled with nadam optimizer and binary_crossentropy loss, by using compile() function of the Keras library.

```
def categ(string):
    if string == 'b':
        return 0
    else:
        return 1
```

Here, the categ is the user defined function used to convert the labels ‘b’ and ‘s’ as 0 and 1, respectively. This is done to convert the string to integer, before fitting it to ANN.

```
classifier.fit(X_train, y_train, batch_size = 256, epochs = 20)
```

Here, the classifier is fitted with batch size as 256 and epochs as 20 on the ANN using fit() function of the Keras library.

```
y_pred = classifier.predict(X_train)
y_pred_2 = classifier.predict(X_test)
```

Here, the X_train and X_test are predicted on the classifier using the predict() function by the Keras library.

```
train_f1_scr = f1_score(y_train , y_pred, pos_label = 1)
```

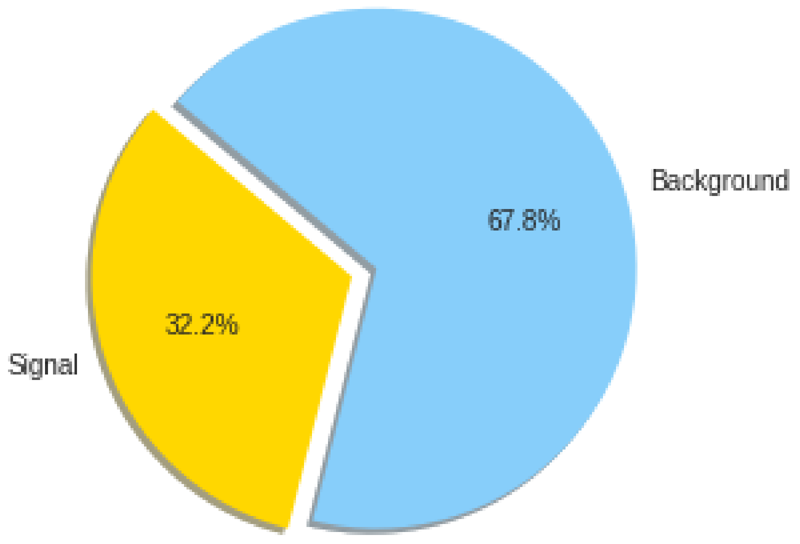
Here, the F1 score is calculated using f1_score() function by the sklearn library.

```
train_acc = accuracy_score(y_train , y_pred)
```

Here, the accuracy is calculated using accuracy_score() function by the sklearn library.

```
def gplot(c1,c2):
    # Data to plot
    labels = 'Signal', 'Background'
    sizes = [c1, c2]
    colors = ['gold', 'lightskyblue']
    explode = (0.1, 0)    # explode 1st slice

    # Plot
    plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%',
    shadow=True, startangle=140)
    plt.axis('equal')
    plt.show()
```

Here, the gplot is the user defined function used to plot the predicted results. This plot depicts the distribution of the Signal and Background.



Integration and Testing

Testing Performed:

Test Plan:

a. Time:

This project should train the Machine Learning model as fast as possible, considering the high dimension of the dataset. Also, it should predict the precise and accurate results. There are different methods implemented in this project, time taken by these methods are as follows:

Methods	**Time Required**
Fitting Model (per Epoch)	5 seconds per epoch
Fitting Model (total)	20*5 = 100 seconds
Prediction (Train Set)	5.3585 seconds
Prediction (Test Set)	1.3338 seconds

b. Output:

1. Optimizer:

   Here, the different optimizers are used that are provided by the Keras for compiling the Feed-forward Neural Network. It is observed that the **nadam** optimizer gives more accurate results.

2. Loss Function:

   As the target variable in the given dataset has the binary classification, it has been observed that using **binary_crossentropy** as our loss function in compiling the Feed-forward Neural Network, yields more accurate results.

3. Batch Size:

   After several trials, it has been observed that batch size of **256** is accurate for fitting the classifier. It is due to the parallel processing done on GPU, i.e. Google Colab tool.

4. Epoch:

   It has been observed that the epoch size of **20**, results in gradual decrease of loss computed after each epoch. Eventually, it yields to more precise and accurate results.

Performance Analysis

Performance Requirement:

The following table shows the performance of each method:

Methods	**Time Required**
Fitting Model (per Epoch)	5 seconds per epoch
Fitting Model (total)	20*5 = 100 seconds
Prediction (Train Set)	5.3585 seconds
Prediction (Test Set)	1.3338 seconds

The above methods are executed on the Google Colab tool, viz. free online environment. It runs entirely on the cloud and no pre-setup is required. It provides the GPU, precisely a Tesla K80 GPU for enhancing the computing and subsequently reducing the total computation time. Also, it gives the accurate results quickly, on such a high dimension dataset.

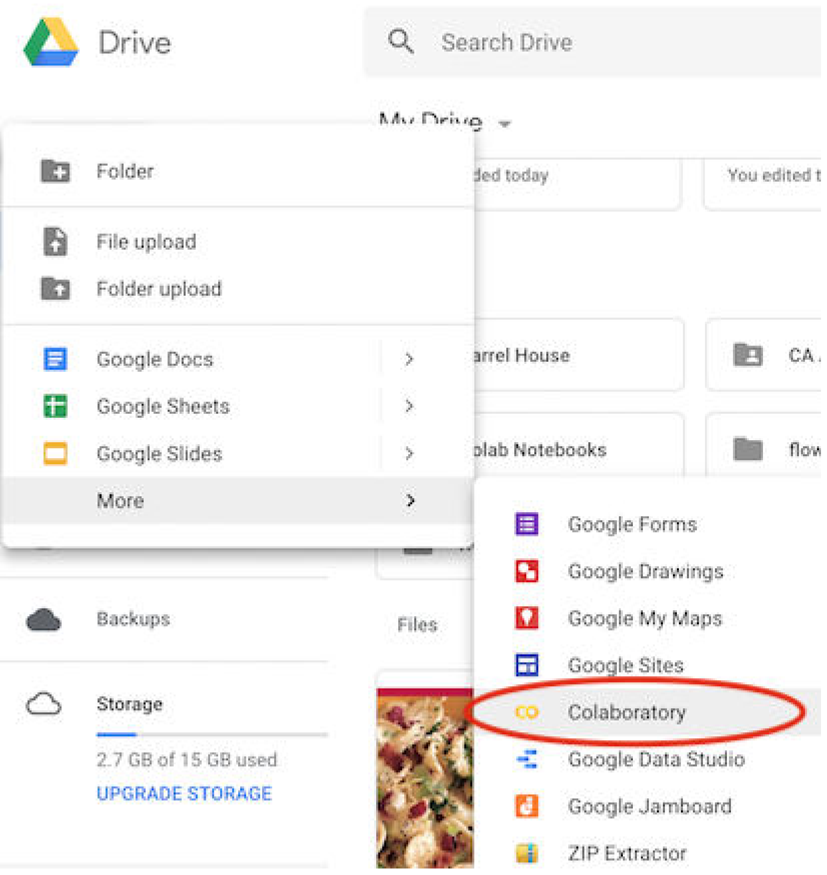
Application

- **Research.**
- **Manufacturing.**
- **Transportation.**
- **Heat Transfer.**
- **Defence.**
- **Miscellaneous.**

Installation Guide and User Manual

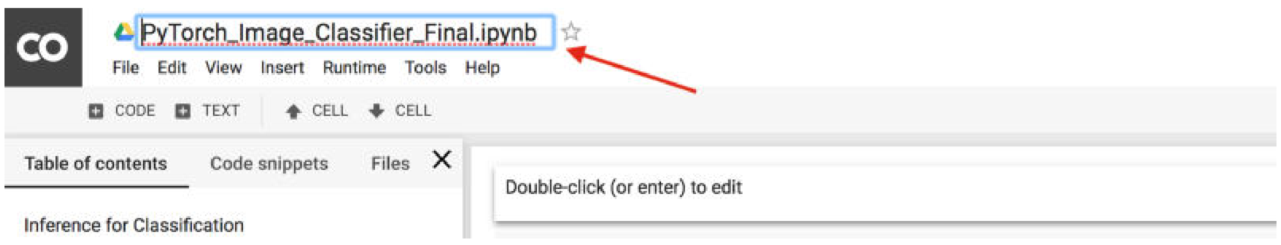
1. To get started with Google Colab tool:

If you want, while you're already in your Google Drive you can create a new Colab notebook. Just click "New" and drop the menu down to "More" and then select "Colaboratory."



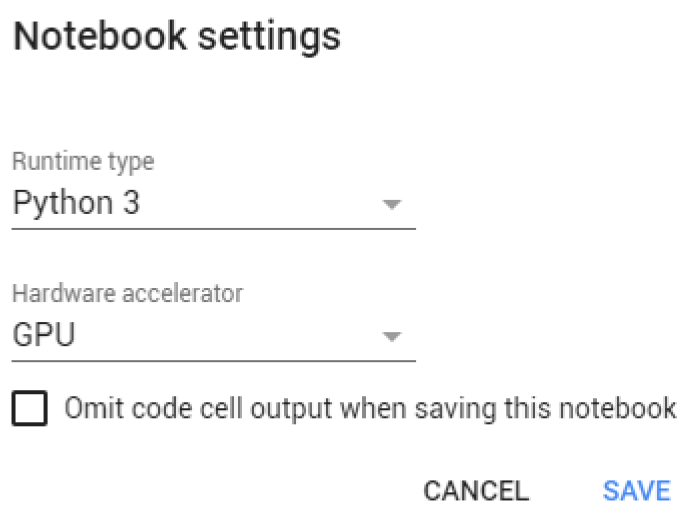
2. To rename the notebook:

You can rename your notebook by clicking on the name of the notebook and changing it or by dropping the "File" menu down to "Rename."


PyTorch_Image_Classifier_Final.ipynb
File Edit View Insert Runtime Tools Help
CODE TEXT CELL CELL
Table of contents Code snippets Files
Inference for Classification
Double-click (or enter) to edit

3. To Set up your free GPU:

Want to use GPU? It's as simple as going to the "runtime" dropdown menu, selecting "change runtime type" and selecting GPU in the hardware accelerator drop-down menu!



Notebook settings

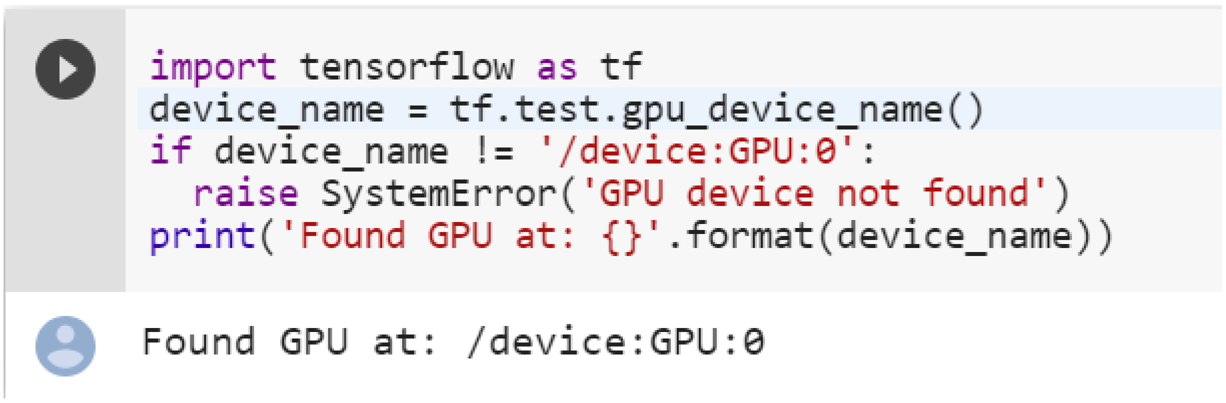
Runtime type: Python 3

Hardware accelerator: GPU

☐ Omit code cell output when saving this notebook

4. To Confirm TensorFlow can see the GPU:

Simply select "GPU" in the Accelerator drop-down in Notebook Settings (either through the Edit menu or the command palette at cmd/ctrl-shift-P).



```
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

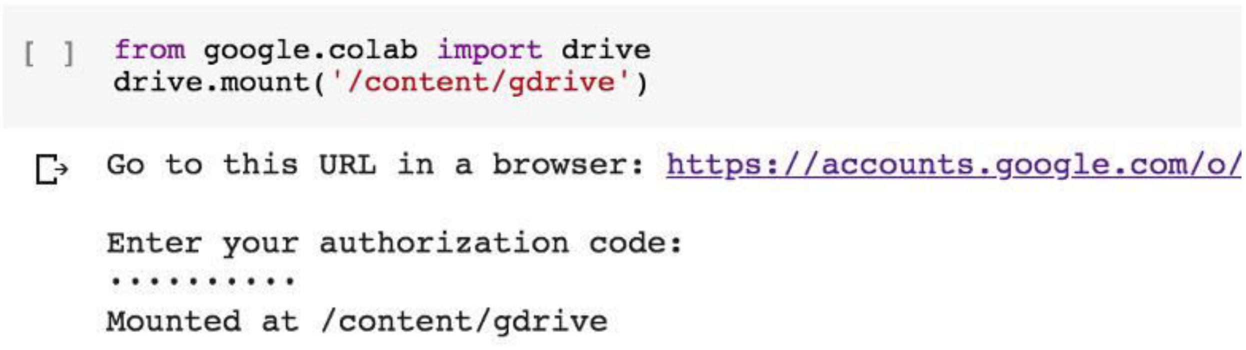
```
Found GPU at: /device:GPU:0
```

5. To mount the Google Drive:

Want to mount your Google Drive? Use:

from google.colab import drive
drive.mount('/content/gdrive')

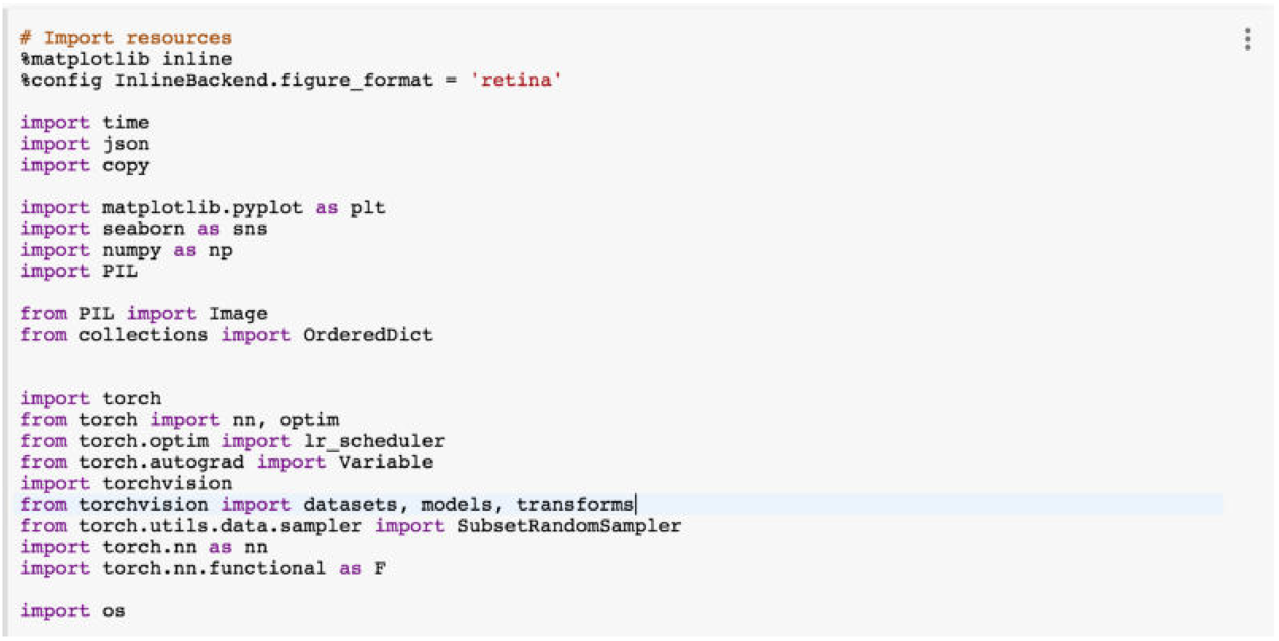
Then, you'll see a link, click on that, allow access, copy the code that pops up, paste it in the box, hit enter, and you're good to go! If you don't see your drive in the side box on the left, just hit "refresh" and it should show up.
Next, Run the cell, click the link, copy the code on the page, paste it in the box, hit enter, and you'll see this when you've successfully mounted your drive.



```
[ ]  from google.colab import drive
     drive.mount('/content/gdrive')

     Go to this URL in a browser: https://accounts.google.com/o/

     Enter your authorization code:
     ··········
     Mounted at /content/gdrive
```

6. To import libraries:

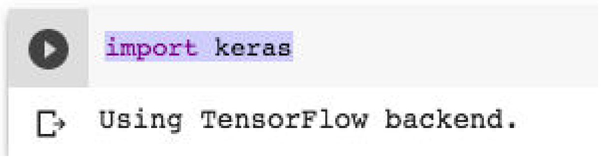
You can import your libraries by running **import** like you do in any other notebook.



```
# Import resources
%matplotlib inline
%config InlineBackend.figure_format = 'retina'

import time
import json
import copy

import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import PIL

from PIL import Image
from collections import OrderedDict


import torch
from torch import nn, optim
from torch.optim import lr_scheduler
from torch.autograd import Variable
import torchvision
from torchvision import datasets, models, transforms
from torch.utils.data.sampler import SubsetRandomSampler
import torch.nn as nn
import torch.nn.functional as F

import os
```

7. To install libraries:

Not able to simply import something else that you want with an import statement? Try a pip install! Just be aware that Google Colab wants an exclamation point before most commands. Use:
!pip install -q keras
import keras

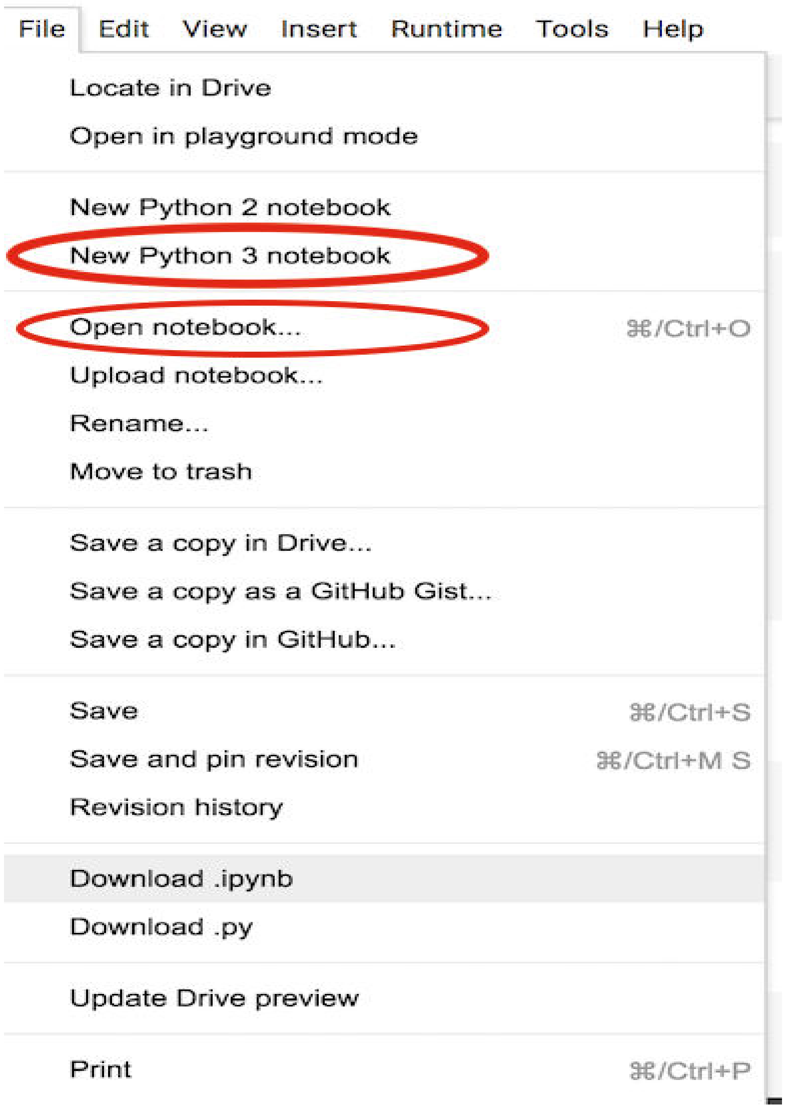


```
import keras
```

```
Using TensorFlow backend.
```

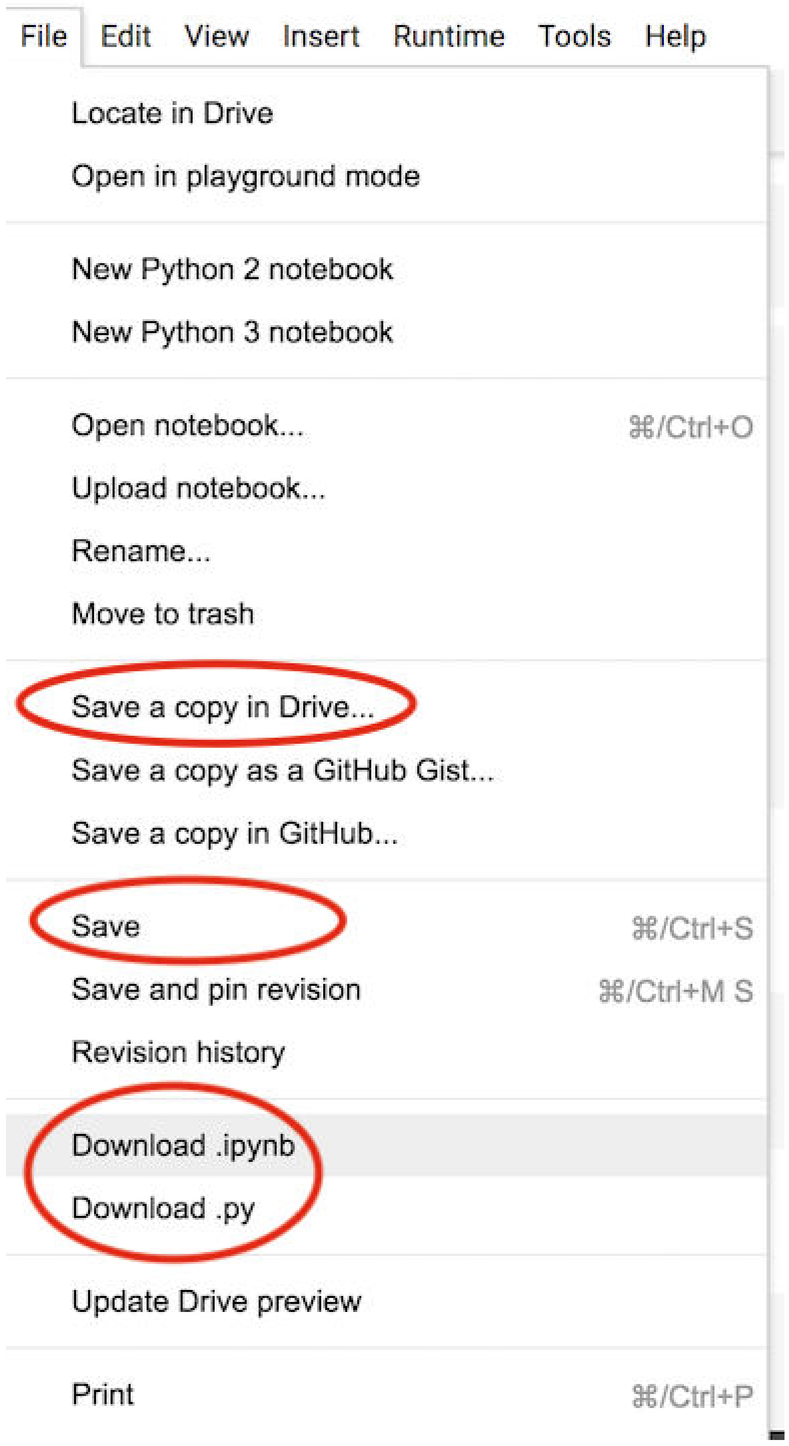
8. To create or open new notebook:

It’s easy to create a new notebook by dropping “File” down to “New Python 3 Notebook.” If you want to open something specific, drop the “File” menu down to “Open Notebook…”



9. To save or download the notebook:

Saving your work is simple! You can do a good ol' "command-s" or drop the "File" menu down to save. You can create a copy of your notebook by dropping "File" -> "Save a Copy in Drive." You can also download your workbook by "File"->"download.ipynb" or "download.py."



Ethics

<u>Declaration</u> <u>of</u> <u>Ethics</u>:

As a computer Science & Engineering Student, I believe it is Unethical To,

1. Surf the internet for personal interest and non-class related purposes during classes.
2. Make a copy of software for personal or commercial use.
3. Make a copy of software for friend.
4. Loan CDs of Software to friends.
5. Download pirated software from the internet.
6. Distribute pirated software from the internet.
7. Buy software with a single user license and then install it on multiple Computers.
8. Share a pirated copy of software.
9. Install a pirated copy of software.

References

[1] ATLAS collaboration (2014). “**Dataset from the ATLAS Higgs Boson Machine Learning Challenge 2014.**” CERN Open Data Portal. DOI:10.7483/OPENDATA.ATLAS.ZBP2.M5T8.

[2] Tom Cornelis. “**Quark-gluon Jet Discrimination at CMS.**” In Proceedings, 2nd Conference on Large Hadron Collider Physics Conference (LHCP 2014): New York, USA, June 2-7, 2014.

[3] C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, B. Kegl, and D. Rousseau. “**Learning to discover: the higgs machine learning challenge 2014 - documentation.**” CERN Open Data Portal, 2014. DOI:10.7483/OPENDATA.ATLAS.MQ5J.GHXA.

[4] S. Chatrchyan et al. “**Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC.**” Phys.Lett. DOI: 10.1016/j.physletb.2012.08.021.

[5] The ATLAS collaboration. “**Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC.**” Phys.Lett. B716 (2012) 1-29. arXiv:1207.7214. CERN-PH-EP-2012-218. DOI: 10.1016/j.physletb.2012.08.020.

[6] Safdari Hartman. “**Neural Networks for calibrating ATLAS jets.**” Phys.Lett. B659, pp. arXiv 1510.03823v1.pdf (2016).

[7] Matteo Cacciari, Gavin P. Salam. “**Pileup subtraction using 514 jet areas.**” Phys. Lett. B659, pp. 119126, (2008).

[8] Cukierman, Aviv and Benjamin Nachman. “**Mathematical Properties of Numerical Inversion for Jet Calibrations.**” arXiv:16.09.05195v1 [physics.data-an] 16 Sep 2016.

[9] Keras libraries, Deep Learning for Python. **https://keras.io** (2016).