COVID-19 Diagnosis based on CT Scan Images

Group 6
Het Jagani (015261415)
Akash Rupapara (015266511)
Keya Patel (015280876)

San Jose State University

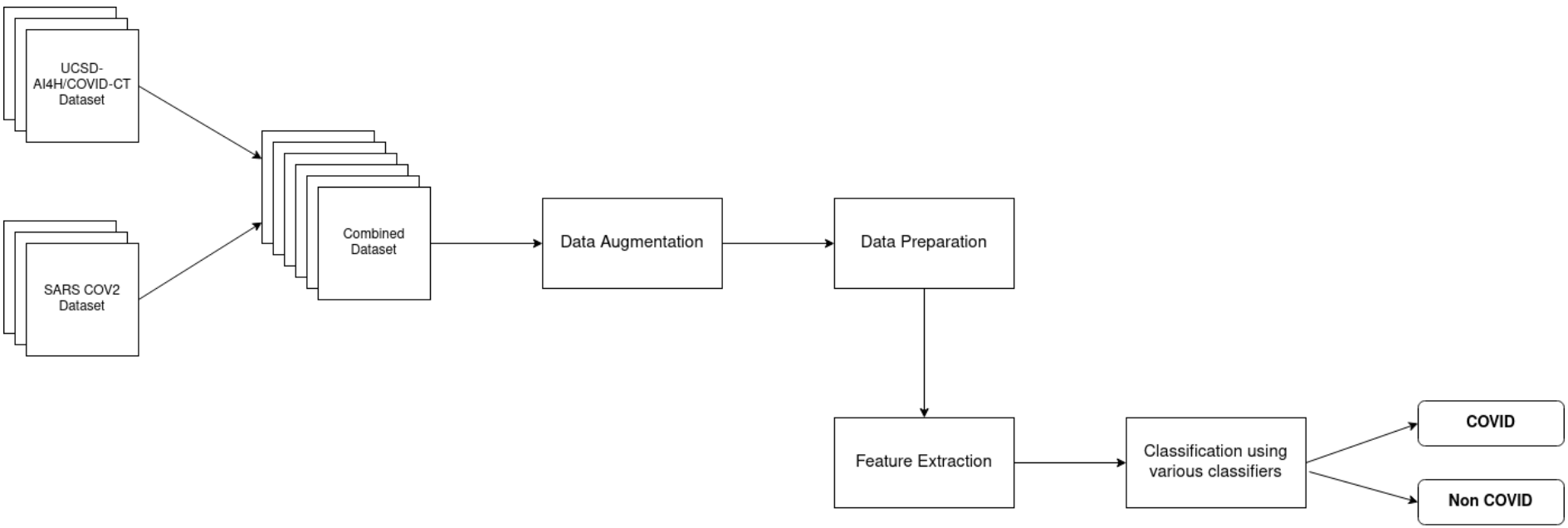
August 2, 2021

Table of Content

Introduction

- Coronavirus 2019 (COVID-19) has spread all over the world and caused deaths in thousands of number.
- Major reason of covid-19 becoming a pandemic is less accurate and delayed diagnosis of the disease.
- Most diagnosis methods have good chance of giving false negative test results.
- In this project, We intended to automate the diagnosis with reliable results using Data Mining techniques.

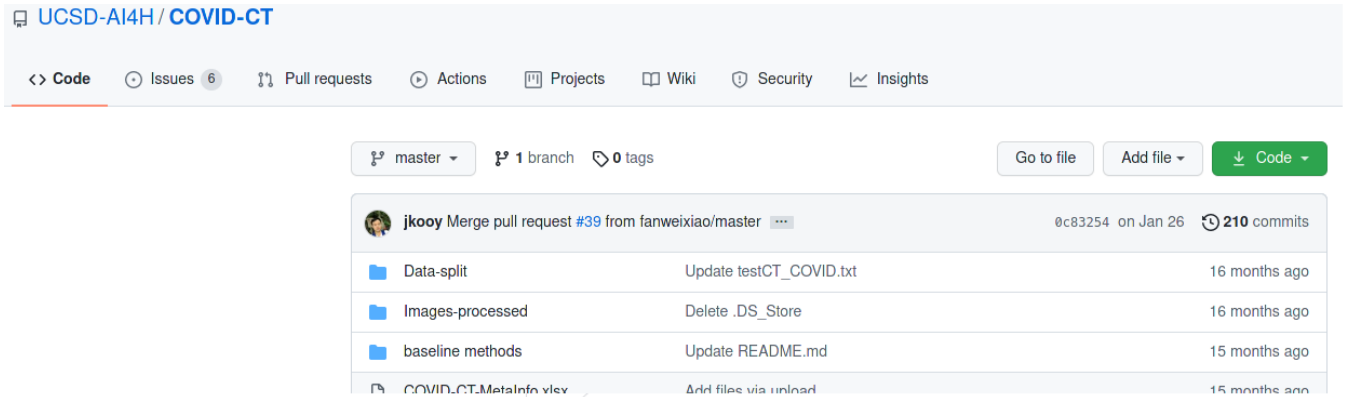
Approach

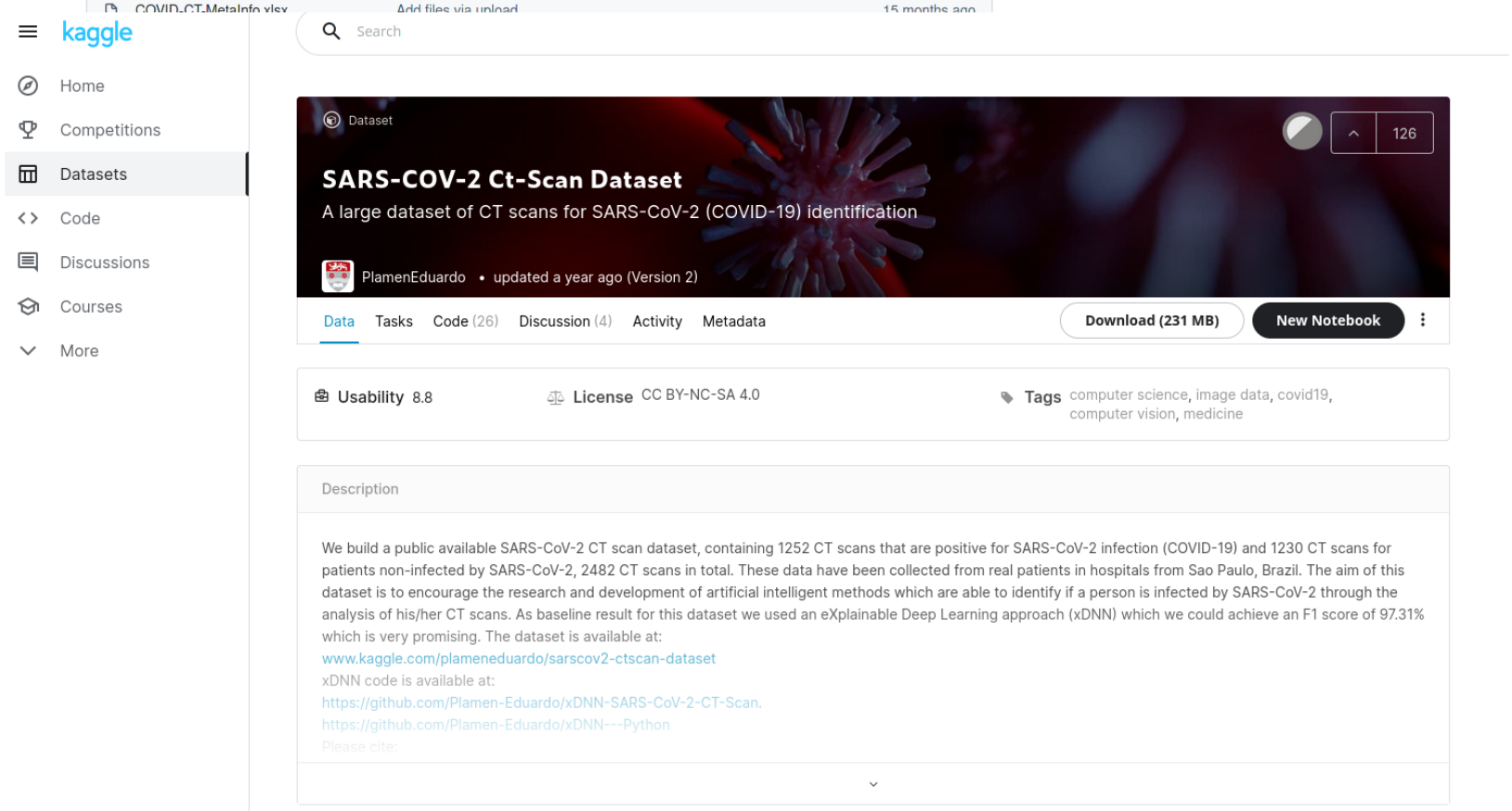

UCSD-AI4H/COVID-CT Dataset
SARS COV2 Dataset
Combined Dataset
Data Augmentation
Data Preparation
Feature Extraction
Classification using various classifiers
COVID
Non COVID

Dataset

- Due to privacy issues, very less medical data images are available but to get better accuracy and performance we need more amount of data.
- Two datasets are combined from two sources in order to get better results.
- Data Augmentation is performed by applying different transformations to further enhance the dataset and increase number of images.

Combining Datasets





Data Augmentation

Images are resized to 256x256 before applying augmentation to make consistent image size.
Six different types of augmentation techniques are applied to enhance the dataset:

- Flip Left Right
- Flip Up Down
- Rotate 20°Right
- Rotate 20°Left
- Adding Gaussian random noise
- Adding Salt and Pepper Noise

Number of Images after augmentation are:

- COVID - 9434
- NON-COVID - 11382

Data Preparation for further analysis

- Total 20614 Images are available of size 256x256 after data pre-processing.
- Loading huge dataset is memory intensive.
- In order to minimize it, We have considered grey scale image of CT Scan and prepared numpy array to save and load model(.npy) using np.save() and np.load() functions.
- This step includes loading the image dataset, preparing input samples and output labels, train-test split, converting images to gray scale, resizing the image and save those arrays such that they can be loaded easily.
- Dataset is divided into 80% train and 20% test.

Feature Extraction

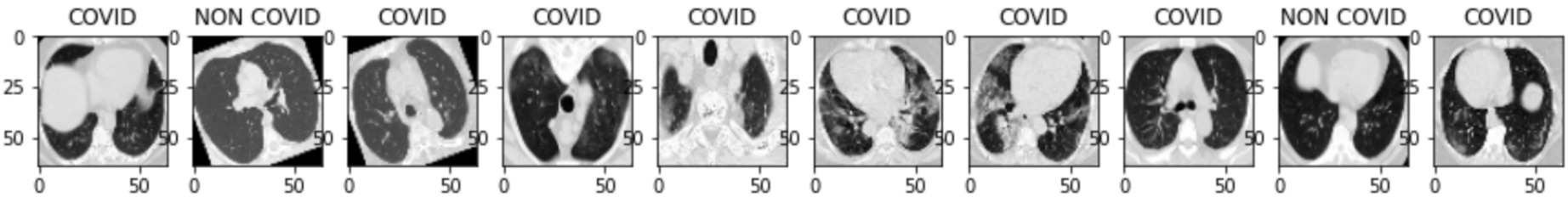
- Feature extraction is done to reduce number of features by creating new features from existing ones which can summarize information in original data.
- RESNET-50 from keras library with imagenet weights (pretrained on imagenet dataset) is used for feature extraction.
- Weights of RESNET-50 are freezed.
  - # of Trainable parameters: 4,392,193
- Output of feature extractor for each image is array of 8192 elements.

Principal Component Analysis

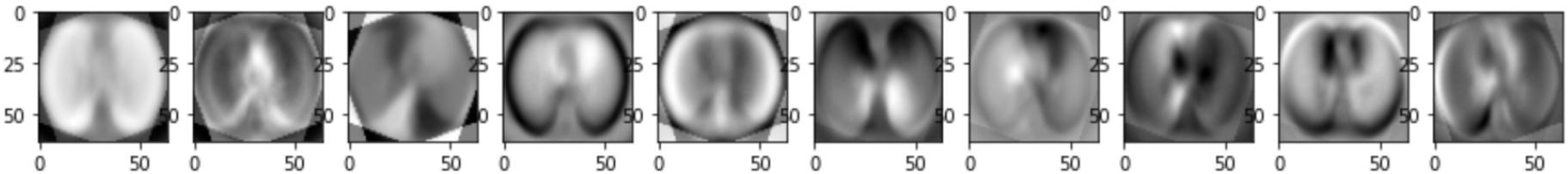
- To avoid Curse of dimensionality due to higher dimensions, we have performed PCA on the input dataset.
- PCA helps to represent a multivariate data as smaller set of variables (summary indices) by observing pattern in data based on correlation between features.
- PCA is performed preserving 80% of components. In total, 86 features are preserved from 8192 features (original).

Principal Component Analysis

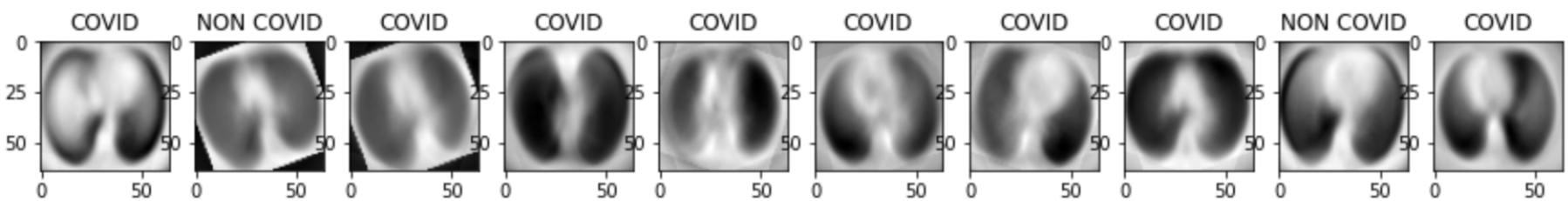
Sample of original images



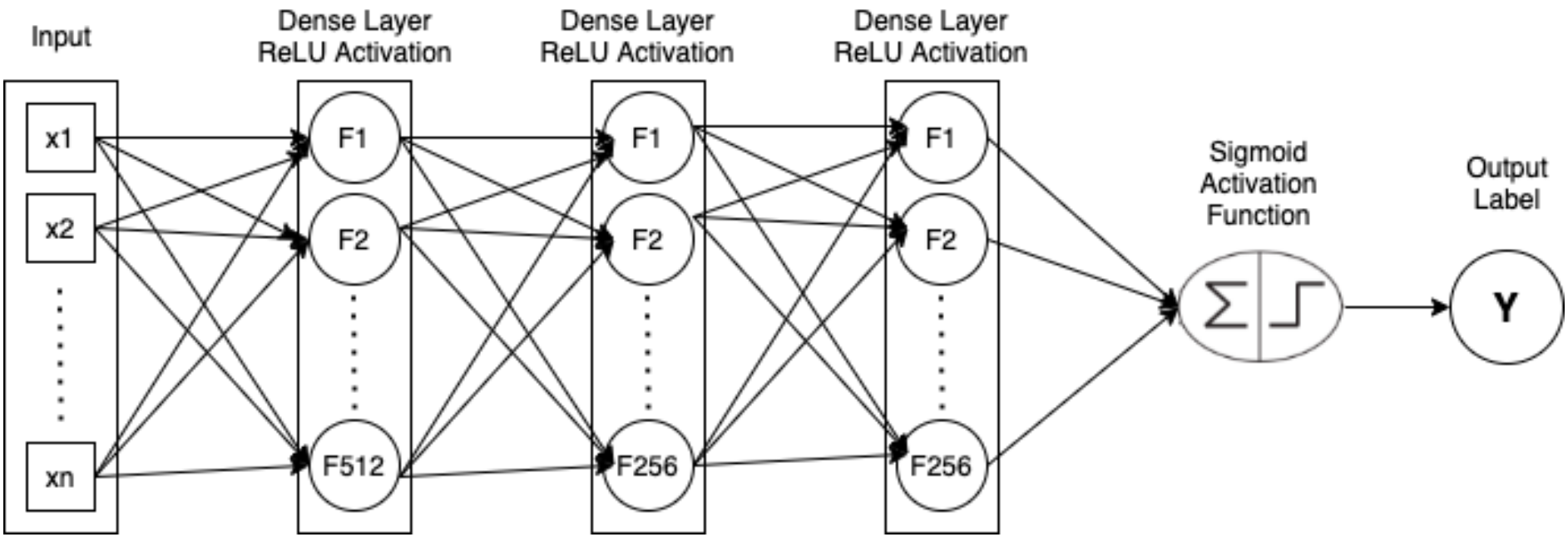
Eigen faces



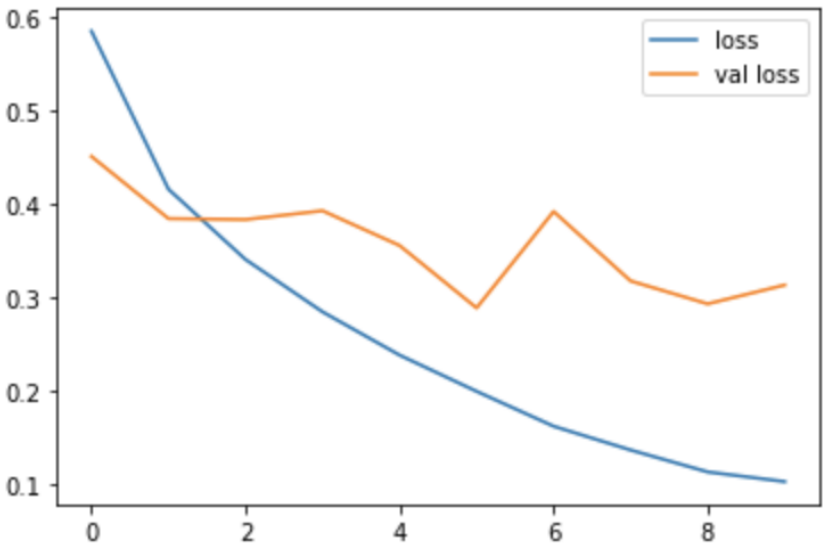
Transfomed images after performing PCA



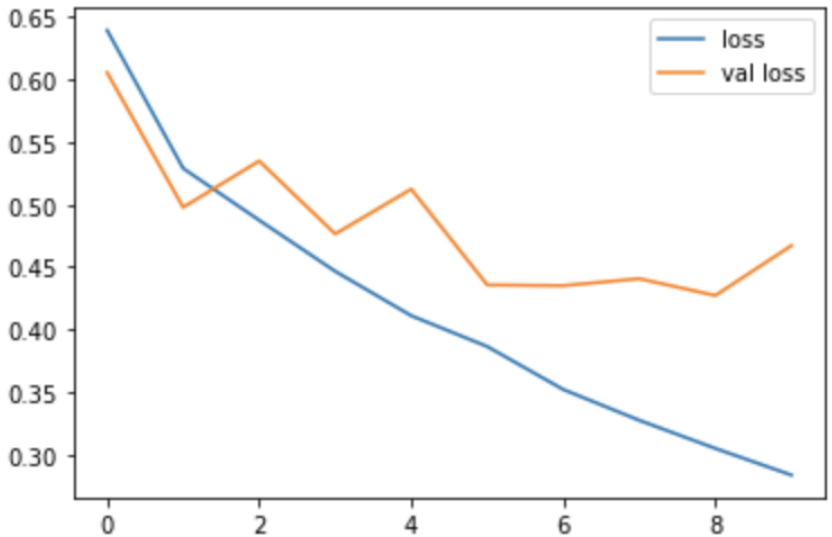
Deep Neural Network



Deep Neural Network



(a) Loss Function without PCA



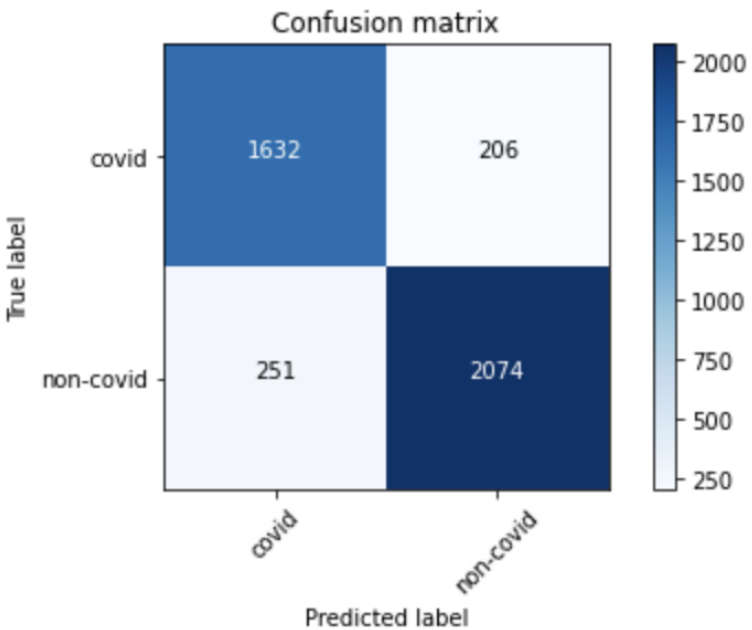
(b) Loss Function with PCA

K Nearest Neighbour

- Observing that Deep Neural Network takes more time in training, KNN was implemented.
- KNN is simple, intuitive algorithm which do not require any training, It just tags the new data on the basis of previous observations (Seen data). Hence, It is executed faster.
- Features extracted from CNN are used to fit KNN classifier.
- Being major cons of KNN, to avoid Curse of dimensionality we implemented PCA by preserving 80% components on features extracted ($8192 \rightarrow 86$ features) and than fitted to KNN Classifier.

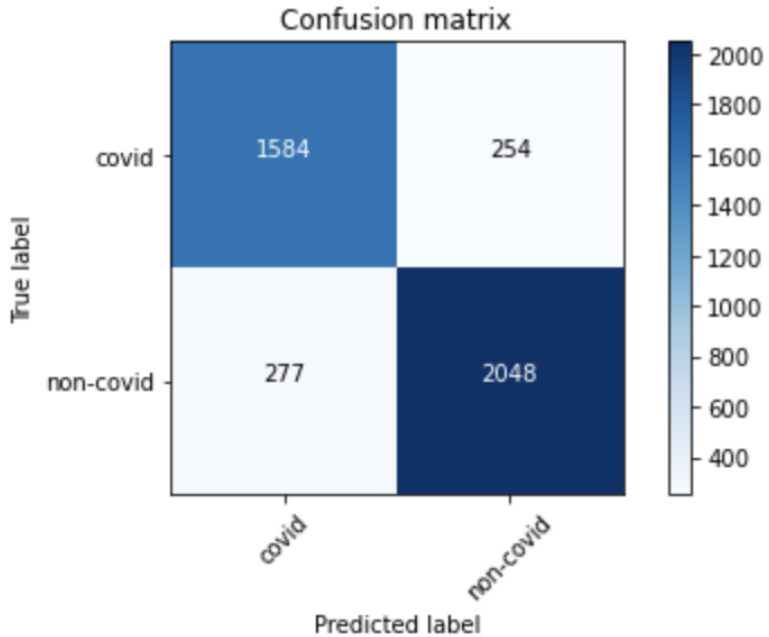
K Nearest Neighbour

KNN



Accuracy: 89%
F1 Score: 87.72%

KNN With PCA



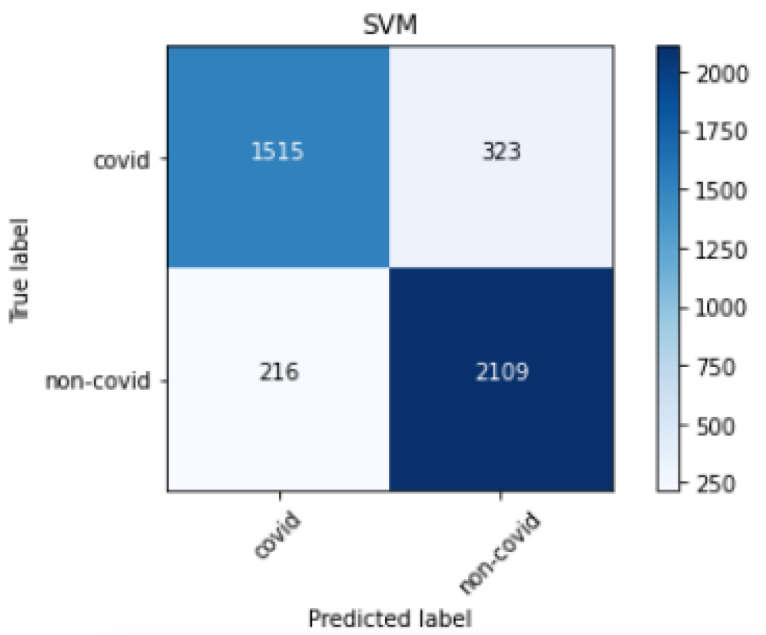
Accuracy: 87.24%
F1 Score: 85.64%

Support Vector Machine

- Researches have shown that SVM performs better in medical diagnosis classification tasks.
- Initially, SVM took too long time to execute. Time of execution improved when features are reduced by performing PCA.
- For SVM two models are trained, one with PCA and without PCA. RBF Kernel was selected for training SVM Kernel.

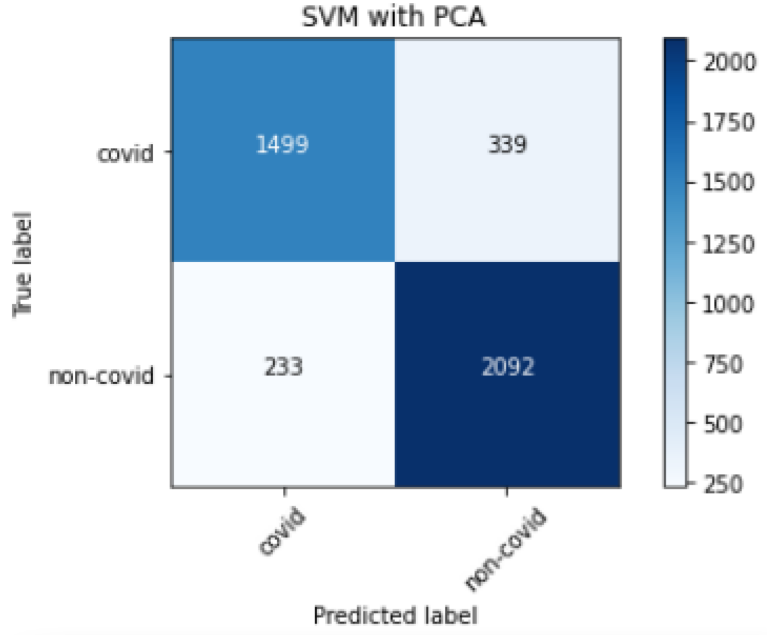
Support Vector Machine

SVM



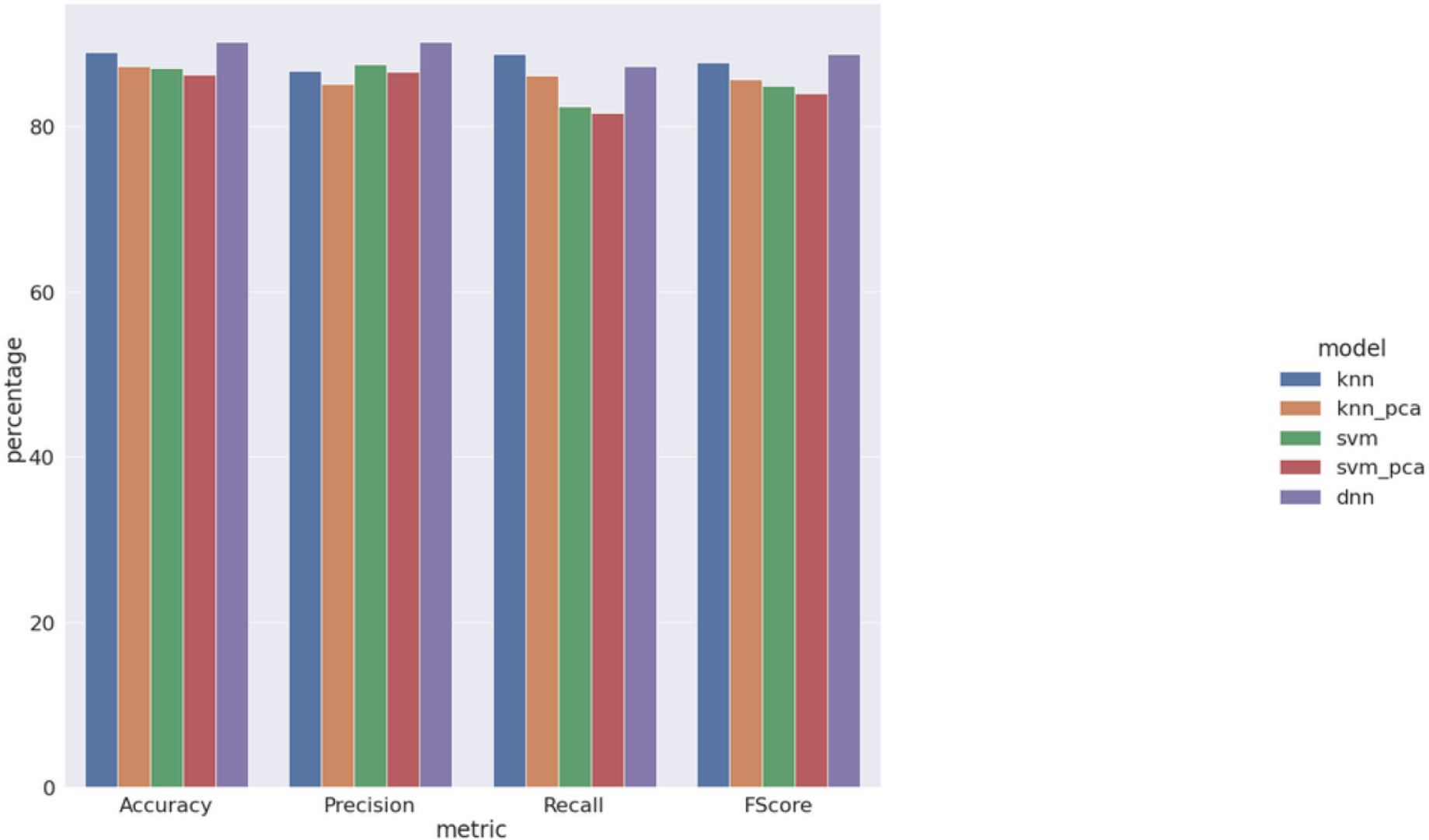
Accuracy: 87.05%
F1 Score: 84.9%

SVM With PCA



Accuracy: 86.26%
F1 Score: 83.98%

Results


percentage
0
20
40
60
80
Accuracy
Precision
Recall
FScore
metric
model
knn
knn_pca
svm
svm_pca
dnn

Results

Time taken to execute

Model	Time in seconds
PCA	144.64
DNN	466.14
KNN	9.03
KNN with PCA	1.59
SVM	526.2
SVM with PCA	12.59

Comparison of Results

	Accuracy	F1 Score	Precision	Recall
DNN	90.2	88.72	90.17	87.32
KNN	89.02	87.72	86.67	88.79
KNN with PCA	87.24	85.64	85.12	86.18
SVM	87.05	84.9	87.52	82.42
SVM with PCA	86.26	83.98	86.55	81.55

Conclusion

- Dataset used for project is not skewed, hence accuracy can be considered good measure.
- DNN performed best in terms of accuracy (around 88%) and time taken to train is 466s.
- DNN model gave slight better performance than baseline methods used in dataset research paper (Accuracy: 86%)[1].
- By performing various experiments, We have observed that larger dataset will yield better performance.

[1] https://www.medrxiv.org/content/10.1101/2020.04.13.20063941v1.full

THANK YOU