SUMMER INTERNSHIP REPORT

PROJECT REPORT

On

"***BIOFUEL DETECTION USING MACHINE LEARNING***"

at

Indian Institute of Technology (BHU) Varanasi

BIOFUEL DETECTION USING MACHINE LEARNING

BIOFUEL CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK

SUBMITTED BY:-

Ritu Parna Banerjee

ASHOKA INSTITUTE OF TECHNOLOGY AND MANAGEMENT

Roll no: 2006410100042

SUBMITTED TO:-

Dr. JP Chakraborty

Indian Institute of Technology (BHU) Varanasi

JULY 10, 2023

CANDIDATE'S DECLARATION

I "RITU PARNA BANERJEE" hereby declare that I have undertaken 4 weeks/months Summer internship at "Indian Institute of Technology (BHU) Varanasi" during a period from 16th june2023 to 14th july 2023in partial fulfillment of requirements for the award of degree of B.Tech (Computer Science and Engineering) at ASHOKA INSTITUTE OF TECHNOLOGY AND MANAGEMENT, VARANASI.

The work which is being presented in the Project report submitted to Dr. JP Chakraborty (IIT BHU) & Department of Computer Science and Engineering at **ASHOKA INSTITUTE OF TECHNOLOGY AND MANAGEMENT**, VARANASI is an authentic record of internship.

Student sign:-

Ritu Parna Banerjee

ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would have been possible without the kind support and help of many individuals. It is indeed a great pleasure to work on this project and presenting this report to my department. This project had provided me a good exposure to the real world problem and also a solution to that.

I would like to pay my heartiest thanks to Indian Institute of Technology (BHU) TPO who provided me such a wonderful opportunity to pursue my project on such an interesting topics. My heartfelt thanks goes to all other faculties who provided valuable suggestions and kind co-operation. I would like to thanks our project guide Dr. JP Chakraborty for importing his valuable guidance and support. He has not only provided suggestions but also rectified my problems whenever I have faced any problems.

I would like to express my special gratitude and thanks to persons who rendered their assistance directly or indirectly. I would like to express my gratitude towards my parents & friends for their kind co-operation and encouragement which help me in this project.

Ritu parna Banerjee

AITM

Contents

I.Definition.....6

1.1. Project Overview.....6

1.2. Problem Statement.....7

1.3. Metrics.....8

II.Analysis.....8

2.1. Dataoverview.....8

2.2. Algorithms and Techniques.....12

2.3. Benchmark ..... ……14

III. Methodology ..... ….15

3.1. Data Pre-processing..... ……..15

3.2. Implementation..... 15

3.3. Refinement.....,,,,,,,, 15

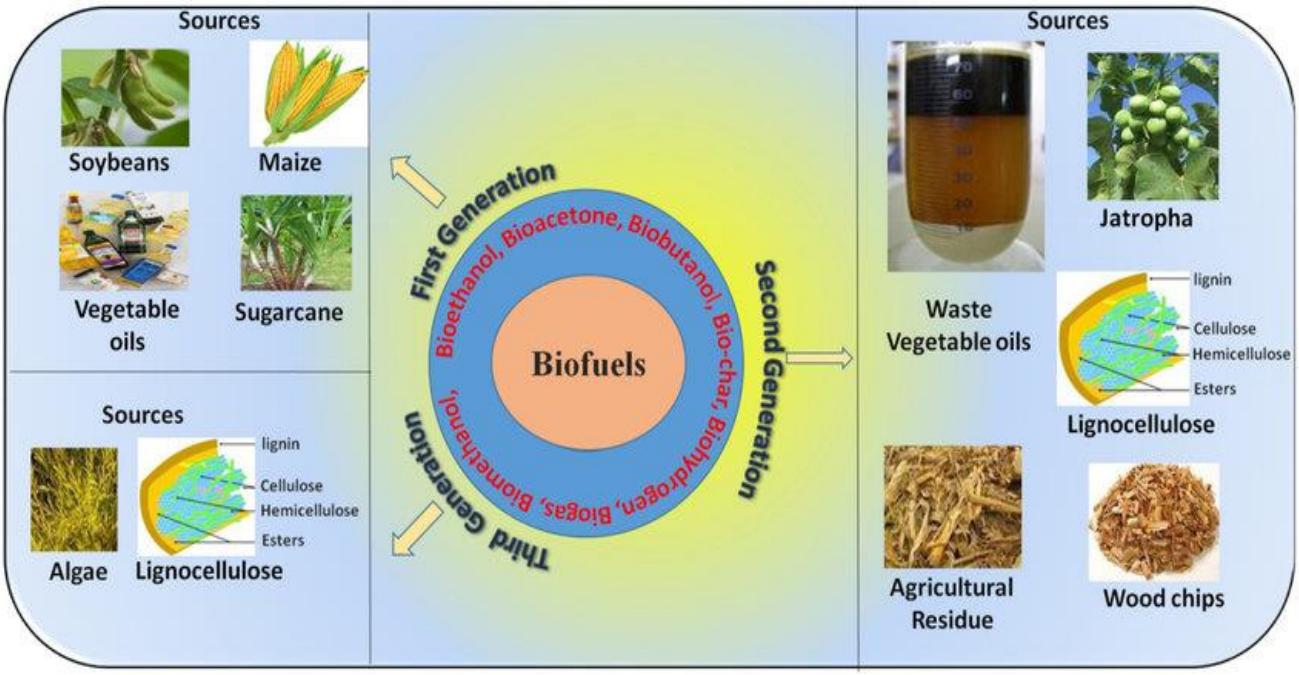
IV.Conclusion.....16

I. Definition

1.1. Project Overview

Biofuels are liquid fuels produced from renewable biological sources, including plants and algae. Biofuels offer a solution to one of the challenges of solar, wind, and other alternative energy sources. Unlike other renewable energy sources, biomass can be converted directly into liquid fuels, called "biofuels," to help meet transportation fuel needs. The two most common types of biofuels in use today are ethanol and biodiesel, both of which represent the first generation of biofuel technology. Biofuels are classified into primary and secondary fuels. Primary fuels include fuel woods used for cooking. Biofuels are also classified into different generations based on the source of feedstock used for biofuel production. First-generation biofuels are produced from food crops like corn, sugarcane, rapeseed, etc.

The aim of this project is to build a convolutional neural network that classifies different sources of biofuel while working reasonably well under constraints of computation. The work described in this report translates to contributions to the field of biofuel detection. It is exploring the use of data augmentation technique, considering different Convolutional Neural Network (CNN) architectures for the feature extraction process when classifying image in a multi-class setting using general-purpose images



1.2. Problem Statement

The dataset was saved in local disk in respective folder labelled by identifying the image such as biofuel or non biofuel. The goal is to predict the source of biofuel from a certain class from the provided classes, thus making it a multi-class classification problem in machine learning terms. Eleven target classes are provided in this dataset: Sugarcane, Rice, Beet, Wood chippings, Corn ,Palm ,Sunflower ,Soya bean, Coconut, Potato, Non biofuel .

. The goal is to train a CNN that would be able to classify the weather condition into these eleven classes. Deep-learning based techniques (CNNs) has been very popular in the last few years where they consistently outperformed traditional approaches for feature extraction to the point of winning ImageNet challenges. In this project, transfer learning along with data augmentation will be used to train a convolutional neural network to classify images of weather to their respective classes. Transfer learning is referred as a machine learning method where a model developed for a task is reused as the starting point for a model on a second task or in other words, Transfer learning refers to the process of using the weights from pre-trained networks on large dataset. As the pre-trained networks have already learnt how to identify lower level features such as edges, lines, curves etc with the convolutional layers which is often the most computationally time-consuming parts of the process, using those weights help the network to converge to a good score faster than training from scratch. To train a CNN model from scratch successfully, the dataset needs to be huge (which is definitely not the case here, the available dataset is very small, only 655 images for training and validation) and machines with higher computational power is needed, preferably with GPU, which I don't have access to at this point. Fortunately, many such networks such as ResNet, Inception, VGG pretrained on ImageNet challenge is available for use publicly. I'll use inception v3for best accuracy

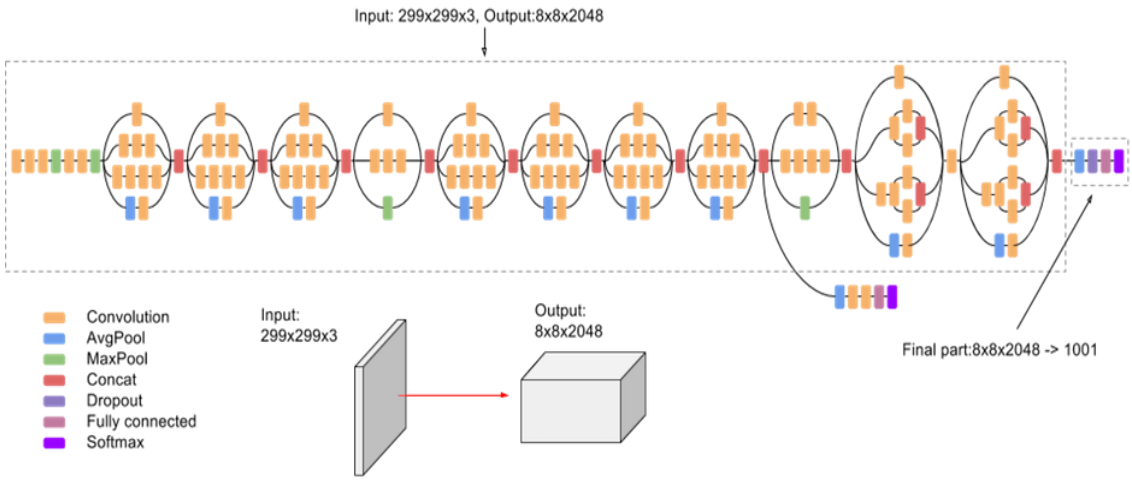


Fig2- The architecture of Inception-V3 model.

1.3. Metrics

```
[12]: model.compile(
          optimizer='adam',
          loss='categorical_crossentropy',
          metrics=['accuracy']
      )
```

The most simple way to calculate the accuracy of any classification machine learning model is to calculate the simple accuracy using number of correctly classified point in test data and total number of points in the test data.
accuracy of the model = (# points correctly classified)/(total points)
This accuracy is represented by the matrix called **Confusion Matrix.**
Here in matrix TN (True Negative) means, total number of points those were of class 0 and correctly predicted by the model as class 0. FP (False Positive) means, total points that were of class 0 but predicted as 1 by my model. TP (True Positive) means, total points those were of class 1 and predicted correctly of class 1 by the model. FN (False Negative) total points those were of class 1 but predicted of class 0 by the model.

II. Analysis

2.1. Data overview

The dataset was created, the various images are downloaded and saved in the local disk which contains weather depicting images. The dataset was labelled by identifying objects in the image such as : Sugarcane, Rice, Beetroot, Wood chippings, Corn ,Palm ,Sunflower ,Soya bean, Coconut, Potato, Non biofuel. The dataset features 11 different classes biofuel collected from the above said different sources. Training set includes about 650 labelled images including the validation images. Images are not of fixed dimensions and the photos are of different sizes. Images do not contain any border. Each image has only one weather category and are saved in separate folder as of the labelled class. The images are saved in folders as shown in following figure:

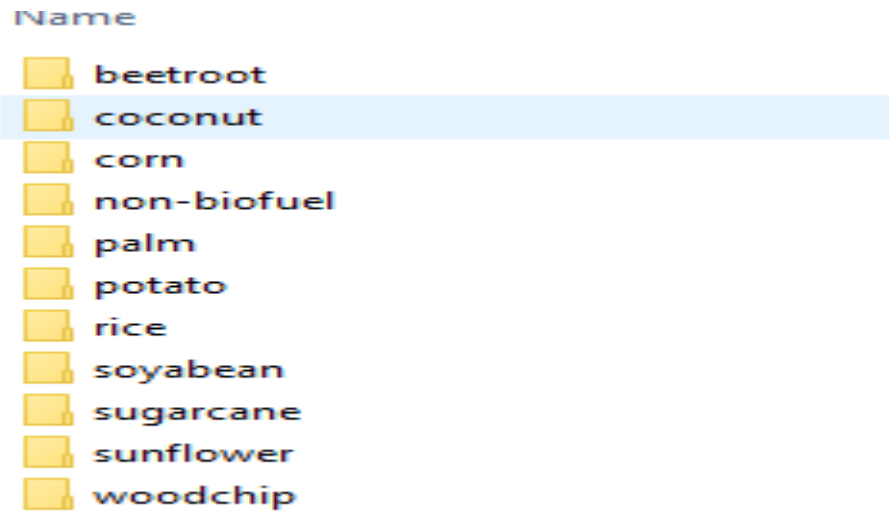


Fig. 3.

Dataset containing respective class of folders

As the input is just raw images (3-dimensional arrays with height x width x channels for computers) it'd be important to pre-process them for classifying them into provided labels. However, the exact details of the pre-processing depend on our choice of the architecture to apply transfer learning.

A. Images of Sugarcane
Dataset contains 30 images

B. Images of Rice

Dataset contains 30 images.

C. Images of Beetroot

However, 30 quality images with landscapes and urban scenes .

D. Images of Wood chippings

25images in dataset

E. Images of Corn

Collecting images of sunrise was relatively easy, as the different image hosting web sites have a decent collection of this type of pictures.

F. Images of Palm

G. Images of Sunflower

Dataset-30 images

H. Images of Soya bean

.

I. Images of Coconut

J. Images of potato

K. Images of Non biofuel

All the elements which are not source of biofuel comes under this class

Dataset -40 images

2.2. Algorithms and Techniques Transfer Learning:

Transfer learning refers to the process of using the weights of a pretrained network trained on a large dataset applied to a different dataset (either as a feature extractor or by finetuning the network). Finetuning refers to the process of training the last few or more layers of the pretrained network on the new dataset to adjust the weight. Transfer learning is very popular in practice as collecting data is often costly and training a large network is computationally expensive. Here, in this case the dataset is very small so weights from a convolutional neural network pretrained on ImageNet dataset is finetuned to classify weather condition.

Benchmark method:

- CNN: A convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analysing visual imagery. CNNs are regularized versions of multilayer perceptron (fully connected networks). The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers. A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a RELU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution. The final convolution, in turn, often involves backpropagation in order to more accurately weight the end product.

Convolutional:

When programming a CNN, the input is a tensor with shape (number of images) x (image width) x (image height) x (image depth). Then after passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map width) x (feature map height) x (feature map channels). A convolutional layer within a neural network should have the following attributes:

• Convolutional kernels defined by a width and height (hyper-parameters).

• The number of input channels and output channels (hyper-parameter).

• The depth of the Convolution filter (the input channels) must be equal to the number channels (depth) of the input feature map.

Pooling:

Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. In addition, pooling may compute a max or an average. Max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Average pooling uses the average value from each of a cluster of neurons at the prior layer.

Fully Connected: Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

Receptive Field: In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from every element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer.

Typically, the subarea is of a square shape (e.g., size 5 by 5). The input area of a neuron is called its receptive field. So, in a fully connected layer, the receptive field is the entire previous layer. In a convolutional layer, the receptive area is smaller than the entire previous layer.

Weights: Each neuron in a neural network computes an output value by applying a specific function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers). Learning, in a neural network, progresses by making iterative adjustments to these biases and weights. The vector of weights and the bias are called filters and represent particular features of the input (e.g., a particular shape).

Architecture of CNN: A 13-layered network is used to predict the weather condition of each class. The following layers are used in the ConvNet.

Layers:

• **Convolution:** Convolutional layers convolve around the image to detect edges, lines, blobs of colours and other visual elements. Convolutional layers hyperparameters are the number of filters, filter size, stride, padding and activation functions for introducing non-linearity.

• **MaxPooling:** Pooling layers reduces the dimensionality of the images by removing some of the pixels from the image. MaxPooling replaces a n x n area of an image with the maximum pixel value from that area to down sample the image.

• **Dropout**: Dropout is a simple and effective technique to prevent the neural network from overfitting during the training. Dropout is implemented by only keeping a neuron active with some probability p and setting it to 0 otherwise. This forces the network to not learn redundant information.

• **Flatten:** Flattens the output of the convolution layers to feed into the Dense layers.

• **Dense**: Dense layers are the traditional fully connected networks that maps the scores of the convolutional

Inception v3 :

We propose a deep convolutional neural network architecture codenamed "Inception", which was responsible for setting the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC 2014). The main hallmark of this architecture is the improved utilization of the computing resources inside the network. This was achieved by a carefully crafted design that allows for increasing the depth and width of the network while keeping the computational budget constant. To optimize quality, the architectural decisions were based on the Hebbian principle and the intuition of multi-scale processing. One particular incarnation used in our submission for ILSVRC 2014 is called GoogLeNet, a 22 layers deep network, the quality of which is assessed in the context of classification and detection.

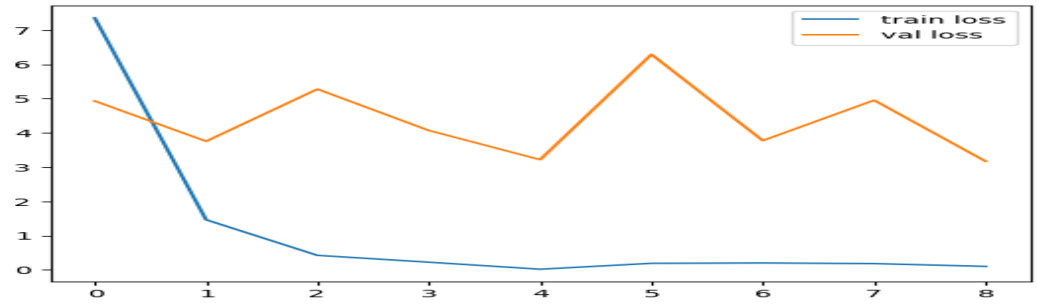
Inception v3 is a convolutional neural network for assisting in image analysis and object detection, and got its start as a module for GoogLeNet. It is the third edition of Google's Inception Convolutional Neural Network, originally introduced during the ImageNet Recognition Challenge.

2.3. Benchmark

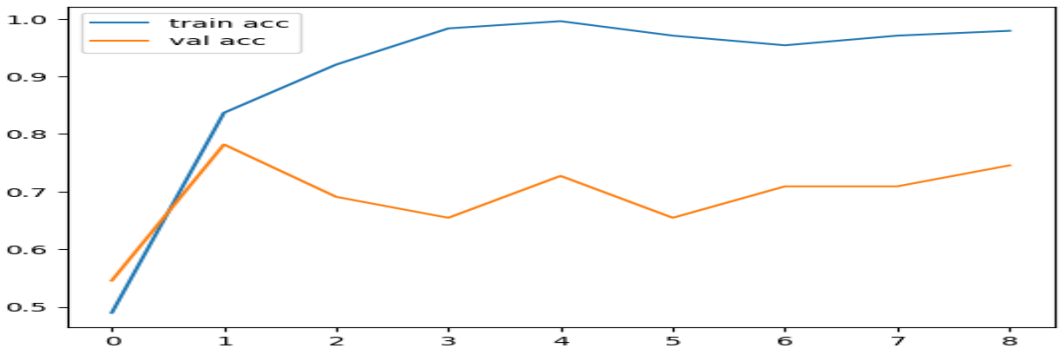
CNN: This model predicts the probabilities for a class to belong to any class of the eleven classes for the naive benchmark. This model yields the categorical cross entropy of 0.1042
showing the best validation accuracy of 78.1%

```
Epoch 1/9
16/16 [==============================] - 136s 8s/step - loss: 7.3537 - accuracy: 0.4895 - val_loss: 4.9297 - val_accurac
y: 0.5455
Epoch 2/9
16/16 [==============================] - 45s 3s/step - loss: 1.4645 - accuracy: 0.8368 - val_loss: 3.7540 - val_accuracy:
0.7818
Epoch 3/9
16/16 [==============================] - 48s 3s/step - loss: 0.4255 - accuracy: 0.9205 - val_loss: 5.2760 - val_accuracy:
0.6909
Epoch 4/9
16/16 [==============================] - 47s 3s/step - loss: 0.2262 - accuracy: 0.9833 - val_loss: 4.0702 - val_accuracy:
0.6545
Epoch 5/9
16/16 [==============================] - 48s 3s/step - loss: 0.0215 - accuracy: 0.9958 - val_loss: 3.2175 - val_accuracy:
0.7273
Epoch 6/9
16/16 [==============================] - 47s 3s/step - loss: 0.1929 - accuracy: 0.9707 - val_loss: 6.2888 - val_accuracy:
0.6545
Epoch 7/9
16/16 [==============================] - 46s 3s/step - loss: 0.2034 - accuracy: 0.9540 - val_loss: 3.7743 - val_accuracy:
0.7091
Epoch 8/9
16/16 [==============================] - 46s 3s/step - loss: 0.1849 - accuracy: 0.9707 - val_loss: 4.9521 - val_accuracy:
0.7091
Epoch 9/9
16/16 [==============================] - 45s 3s/step - loss: 0.1042 - accuracy: 0.9791 - val_loss: 3.1716 - val_accuracy:
0.7455
```

```
Inception_v3 Model
------------------
Minimum  validation loss: 3.17
Maximum validation accuracy: 0.78
```



.



III. Methodology

3.1. Data Pre-processing

```
trainDataGenerator = ImageDataGenerator(validation_split = 0.2,
                                        rescale = 1./255,
                                        shear_range = 0.2,
                                        zoom_range = 0.2,
                                        horizontal_flip = True,
                                        fill_mode = 'reflect')
```

```
# training & validation data generators

trainingData = trainDataGenerator.flow_from_directory(TrainDataset,
                                                      subset = 'training',
                                                      seed = 132,
                                                      target_size= (224,224),
                                                      batch_size = batchSize,
                                                      class_mode= 'categorical')
```

```
Found 239 images belonging to 11 classes.
```

```
validationData = trainDataGenerator.flow_from_directory(TrainDataset,
                                                        subset = 'validation',
                                                        seed = 132,
                                                        target_size = (224,224),
                                                        batch_size = batchSize,
                                                        class_mode= 'categorical')
```

```
Found 55 images belonging to 11 classes.
```

3.2. Implementation

Initially the baselines with a deep ConvNet with and without augmentation were implemented for comparison. After that the models with transfer learning were used. So, incepton v3 architecture without the last fully connected layers were used to extract the convolutional features from the pre-processed images.

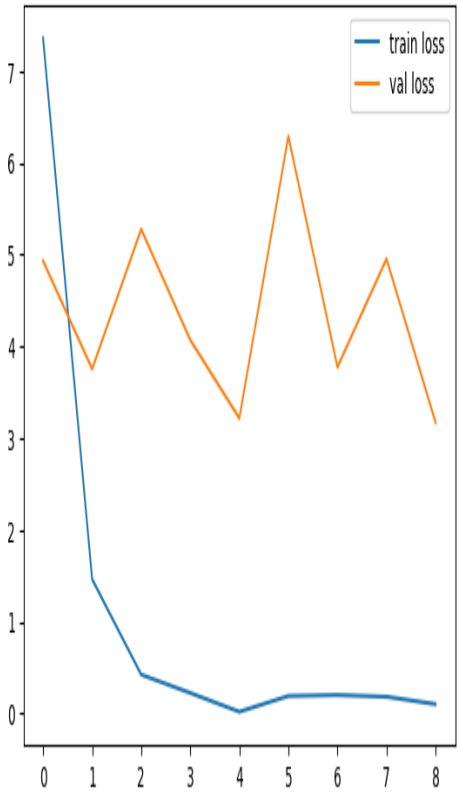
3.3. Refinement

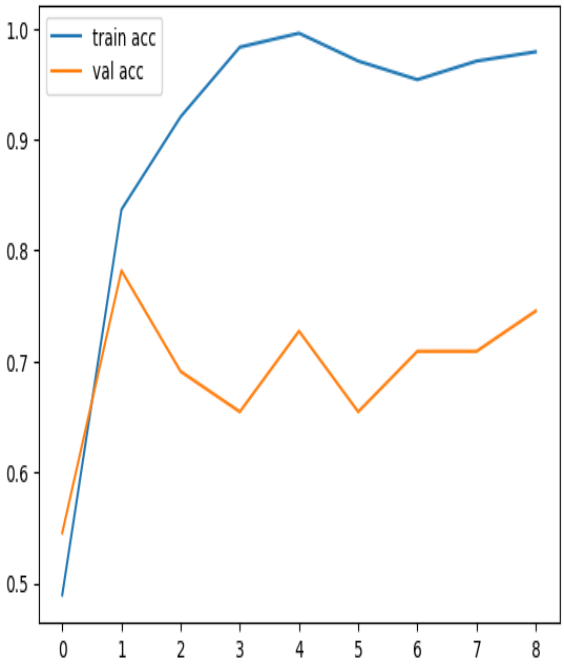
I applied batch normalization in the model to prevent arbitrary large weights in the intermediate layers as the batch normalization normalizes the intermediate layers thus helping to converge well. Even in the model with batch normalization enabled during some epochs training accuracy was much higher than validation accuracy, often going near about 98-99%.

IV Conclusion

As I've recorded the accuracy and loss curve of the models per epoch, the final model's graph can be shown below. Here's the accuracy/loss graph of the model with data augmentation, dropout and batch normalization

```
Inception_v3 Model
------------------
Minimum  validation loss: 3.17
Maximum validation accuracy: 0.78
```





References :

• https://unsplash.com/

• https://www.pexels.com/

• https://www.mendeley.com/?interaction_required=true

• https://blog.keras.io/building-powerful-image-classification-models-using-very-littledata.html

• https://github.com/keras-team/keras-applications/releases