Eye Disease Classification

[MINI PROJECT - I]

V Semester B. Tech. Department of Information Technology

Group Members

Dhiraj Damani(IIT2020014)

Nikhil Dubey(IIT2020016)

Devesh Parte(IIT2020089)

Jambhule Sahas Devidas(IIT2020105)

Instructor

Dr. Anjali Gautam

Table Content

- ➢ Problem statement
- ➢ Objectives
- ➢ Literature review
- ➢ Methodology and Dataset
- ➢ VGGNet
- ➢ DenseNet
- ➢ Result (DenseNet)
- ➢ Confusion Matrix (DenseNet)
- ➢ Result (VGGNet)

Problem Statement

- Classification of Diabetic Retinopathy Disease in [0-4] Stage.

- Dataset : APTOS (2019) BLINDNESS DATASET

- Brief : Here, we will classify and grade the stage of severity that occurs in the retinal region of the eye caused due to complications of diabetes.

Objectives

- To perceive the severity stage of Diabetic Retinopathy [Disease causing in Eye] in an individual in order to provide the good treatment at their early stage.

- For assisting a humanitarian cause.

S.No.	Title	Author	Methodology	Dataset	Advantages	Disadvantages	Accuracy
1	Classification and Localisation of Diabetic-Related Eye Disease (2002)	Alireza Osareh And others.	The proposed method is implemented on color image prepossessing Using color-based mathematical morphology.	60 color retinal images obtain from non-hydriatic retinal camera	The trade-off between sensitivity and specificity was appropriately balanced for this particular problem.	The largest correlation coefficient value does not necessarily correspond to the true optic disk center..	Overall diagnostics with 90.1% accuracy, 93.4% sensitivity, and 82.7% specificity,
2	Detection and classification of retinal lesions for grading of diabetic retinopathy (2014)	M.Usman Akram And others	This paper describe all NPDR lesions.The paper uses SVM .It have three stage model compromising , retinal analysis, and classification.	6360 lesions from 1410 retinal images with 3544 red lesions and 2816 bright lesions	The proposed system detects all types of NPDR lesions and correctly grades the retinal images with high accuracy.	Computation and complexity is the issue..	82.6% and 88.3% for HMA and EXs, resp.
3	Diabetic Retinopathy classification Using a Modified Xception Architecture (2019)	Sara Hosseinzadeh Kassani And others.	Based on the deep layer aggregation which combines multilevel features from various convolutional layers of Xception Architecture.	APTOS 2019 consist of 3662 retinal images	The modified Xception deep extractor achieved a much better performance on the APTOS dataset in comparison to the original Xception architecture.	Requires high volumes of training data & reaching the network architecture is time-consuming.	83.09%

4	A lightweight CNN for Diabetic Retinopathy classification from fundus images (2020)	Gayathri S. And others.	Uses CNN for Feature Extraction. Extracted features are then provided to the classifier for binary and multi-class classifications.	IDRiD, Messidor KAGGLE	By using CNN as a feature extractor,it reduces the computational time and complexity. Existing models used pre-trained networks and transfer learning methods for classification which inturn may increase the computational complexity.	This approach might not extract the very tiny lesions at the primary stage of DR.	99.89% for binary classification & 99.59% for multi-class classification
5	Fundus Disease Image classification based on Improved Transformer (2021)	Honggang Yang And others.	This paper proposes a Transformer Eye (TransEye) fine-grained fundus disease image classification method based on the self-attention mechanism.	OIA-ODIR, consists of 5000 retinal fundus images.	OIA-ODIR, consists of 5000 retinal fundus images.	The optimal prediction accuracy using this method is much higher as compared to CNN.	Accuracy given by this model is 84 %. Better for large datasets.

Methodology And Dataset

- APTOS (2019) Blindness Dataset

- Methodology : Convolution Neural Network Architectures

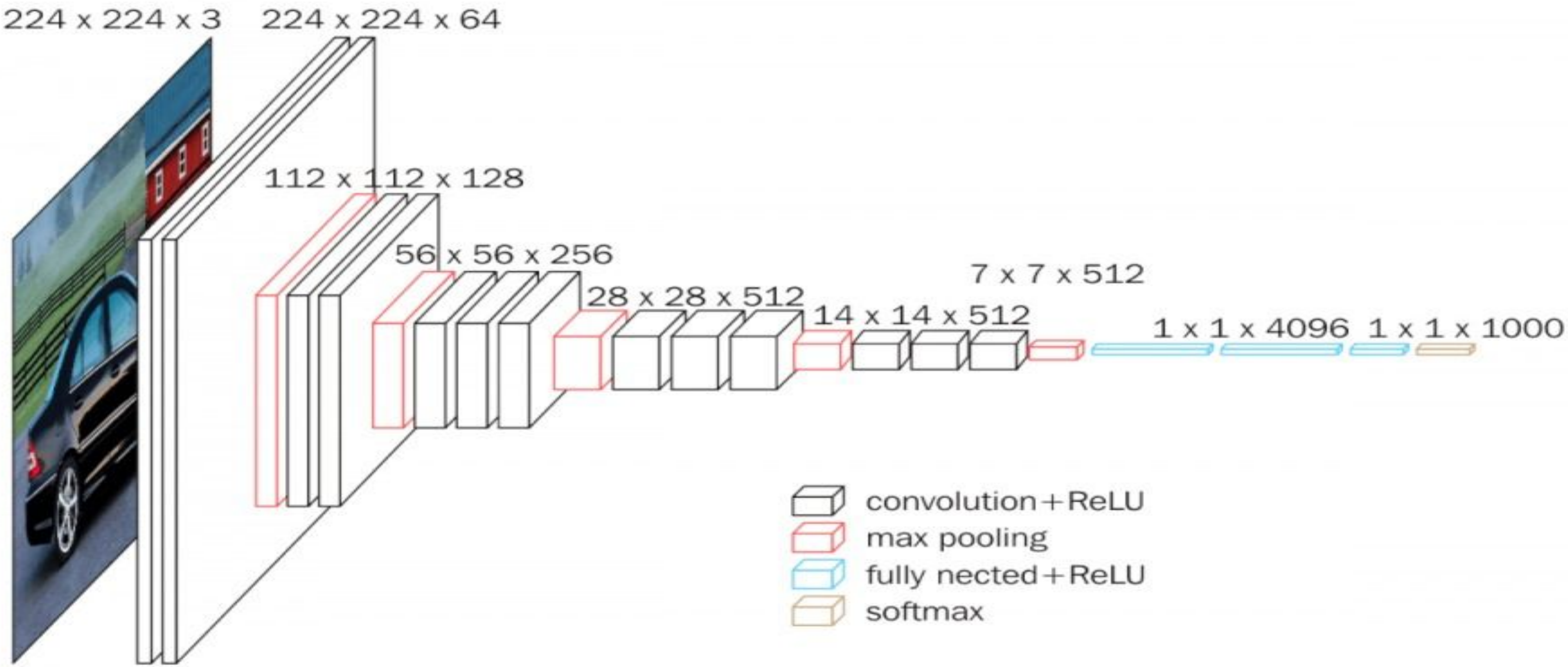
  → VGGNet

  → DenseNet

VGGNet

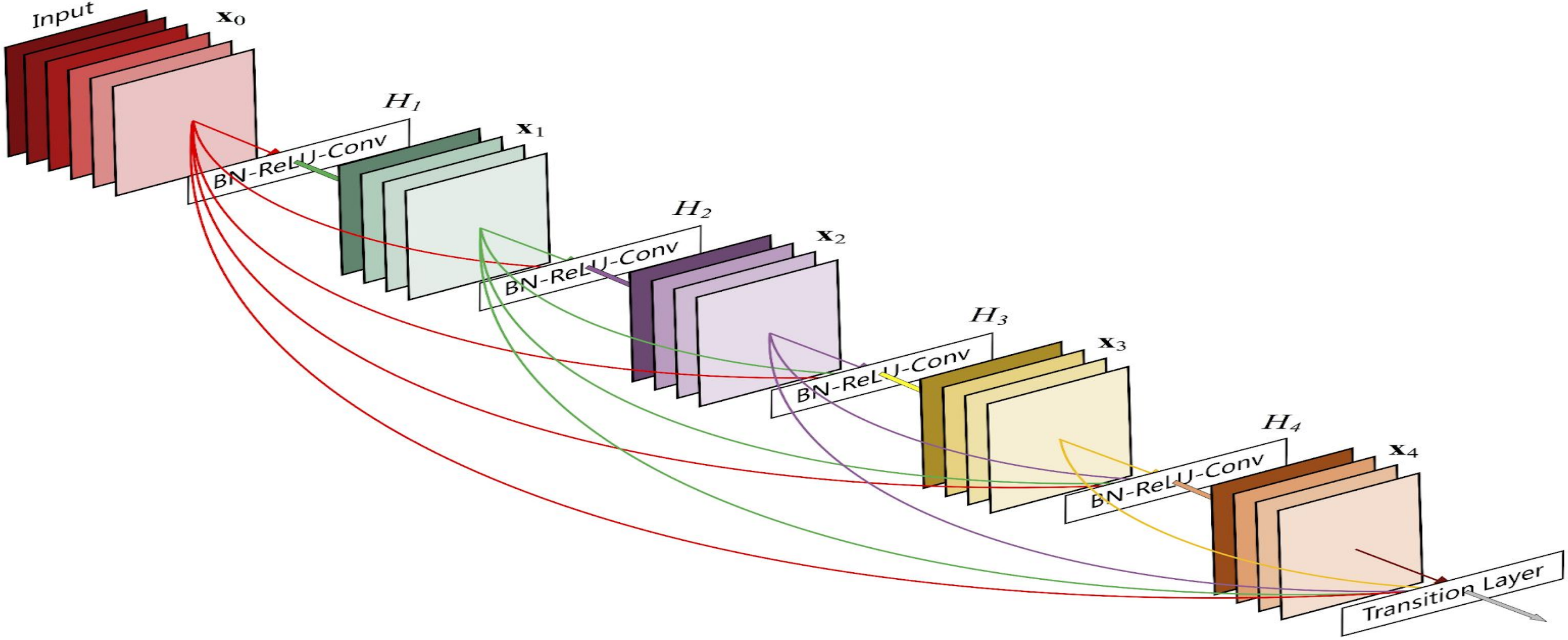
- The VGG-16 consists of 13 convolutional layers and 3 fully connected layers.
- Convolutional layer - It extracts out the features from the input image.
- Pooling layers are used to reduce the dimensions of the feature maps.
- Fully-Connected Layer - Neurons in the preceding and succeeding layers are fully connected with each other.

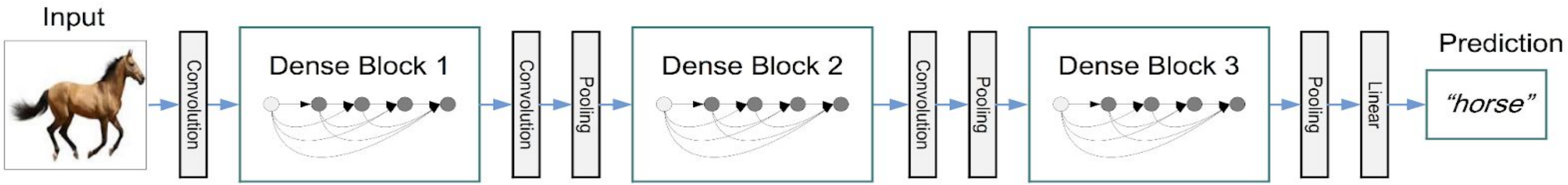
VGG16 Architecture



DenseNet

- DenseNet type CNN
- Dense Blocks connection
  - Feature maps
  - Feed-Forward
- Pooling layers
- Convolutional Layers
- Transition layer
- Global Average Pooling
- Classification layer




Input

Convolution

Dense Block 1

Convolution

Pooling

Dense Block 2

Convolution

Pooling

Dense Block 3

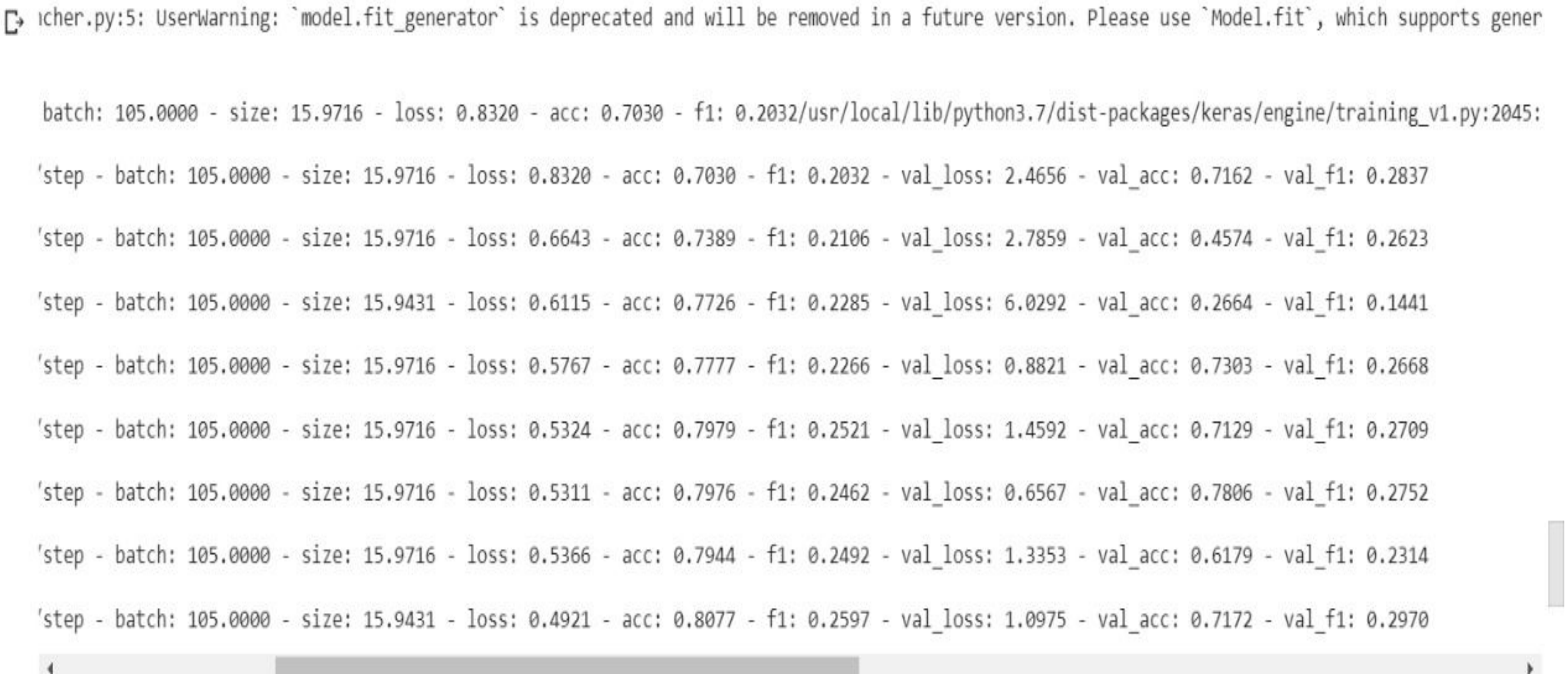
Pooling

Linear

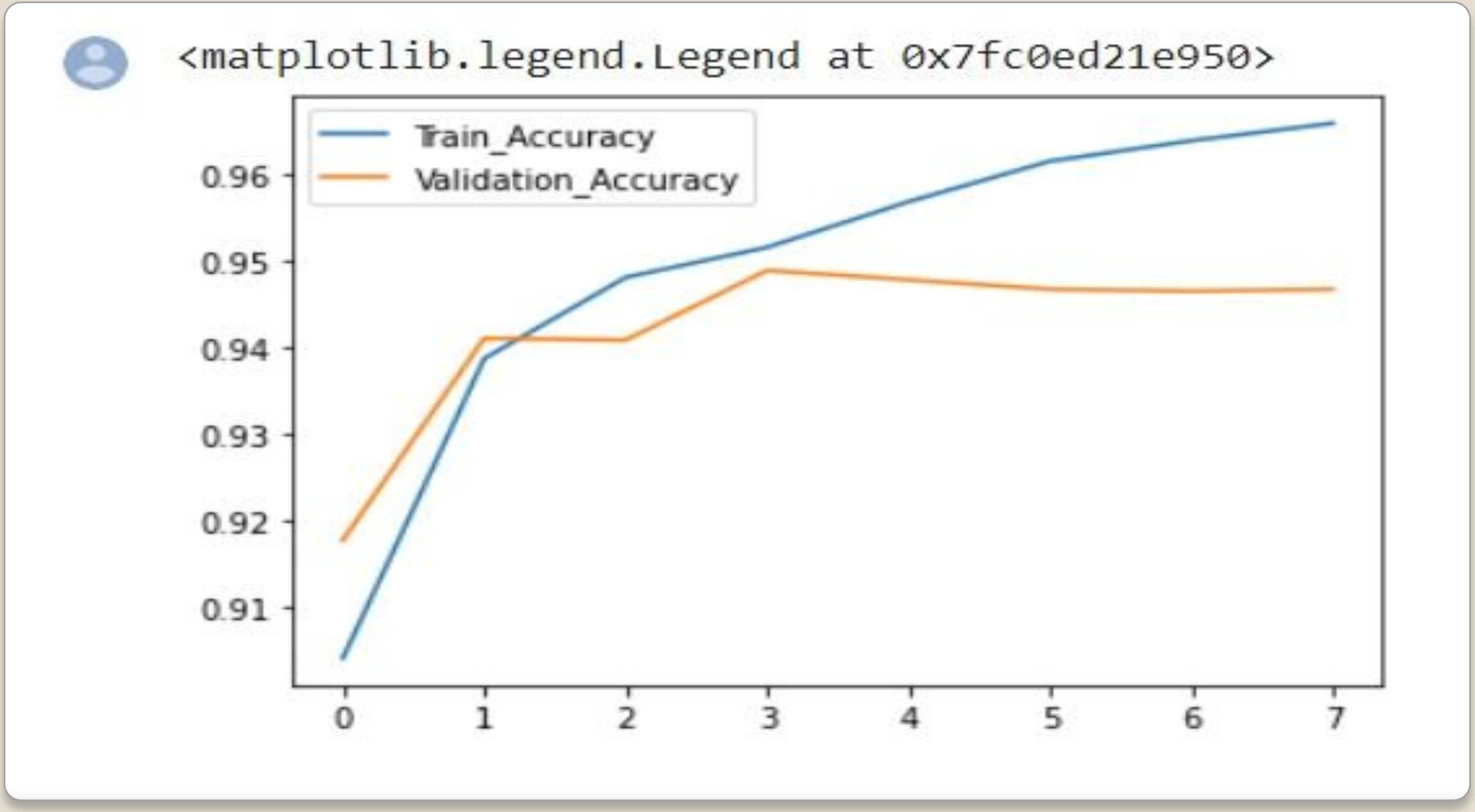
Prediction

"horse"

IMPLEMENTATION VISUAL

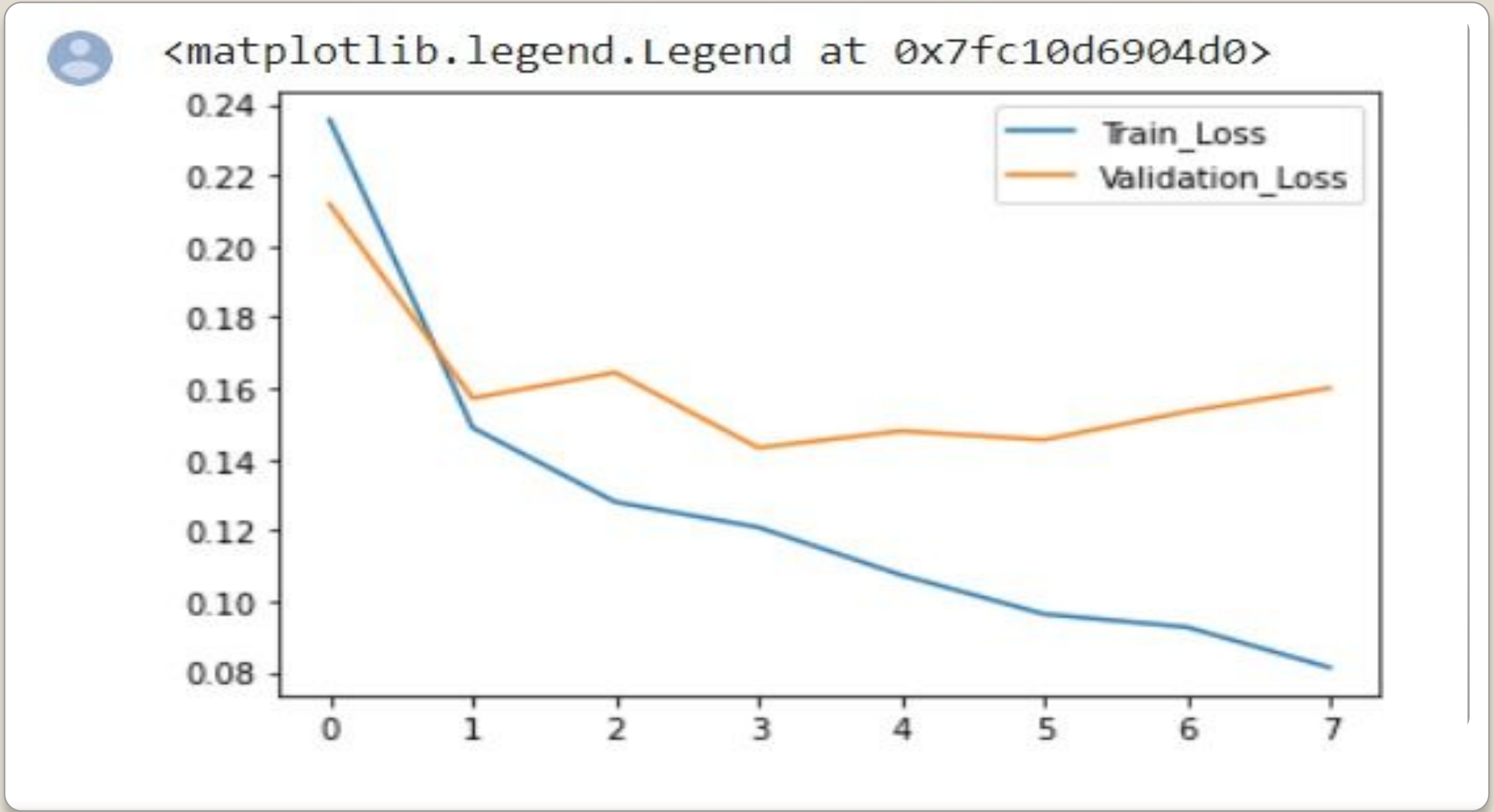


```
icher.py:5: UserWarning: `model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports gener


batch: 105.0000 - size: 15.9716 - loss: 0.8320 - acc: 0.7030 - f1: 0.2032/usr/local/lib/python3.7/dist-packages/keras/engine/training_v1.py:2045:

'step - batch: 105.0000 - size: 15.9716 - loss: 0.8320 - acc: 0.7030 - f1: 0.2032 - val_loss: 2.4656 - val_acc: 0.7162 - val_f1: 0.2837

'step - batch: 105.0000 - size: 15.9716 - loss: 0.6643 - acc: 0.7389 - f1: 0.2106 - val_loss: 2.7859 - val_acc: 0.4574 - val_f1: 0.2623

'step - batch: 105.0000 - size: 15.9431 - loss: 0.6115 - acc: 0.7726 - f1: 0.2285 - val_loss: 6.0292 - val_acc: 0.2664 - val_f1: 0.1441

'step - batch: 105.0000 - size: 15.9716 - loss: 0.5767 - acc: 0.7777 - f1: 0.2266 - val_loss: 0.8821 - val_acc: 0.7303 - val_f1: 0.2668

'step - batch: 105.0000 - size: 15.9716 - loss: 0.5324 - acc: 0.7979 - f1: 0.2521 - val_loss: 1.4592 - val_acc: 0.7129 - val_f1: 0.2709

'step - batch: 105.0000 - size: 15.9716 - loss: 0.5311 - acc: 0.7976 - f1: 0.2462 - val_loss: 0.6567 - val_acc: 0.7806 - val_f1: 0.2752

'step - batch: 105.0000 - size: 15.9716 - loss: 0.5366 - acc: 0.7944 - f1: 0.2492 - val_loss: 1.3353 - val_acc: 0.6179 - val_f1: 0.2314

'step - batch: 105.0000 - size: 15.9431 - loss: 0.4921 - acc: 0.8077 - f1: 0.2597 - val_loss: 1.0975 - val_acc: 0.7172 - val_f1: 0.2970
```



```
<matplotlib.legend.Legend at 0x7fc0ed21e950>
```

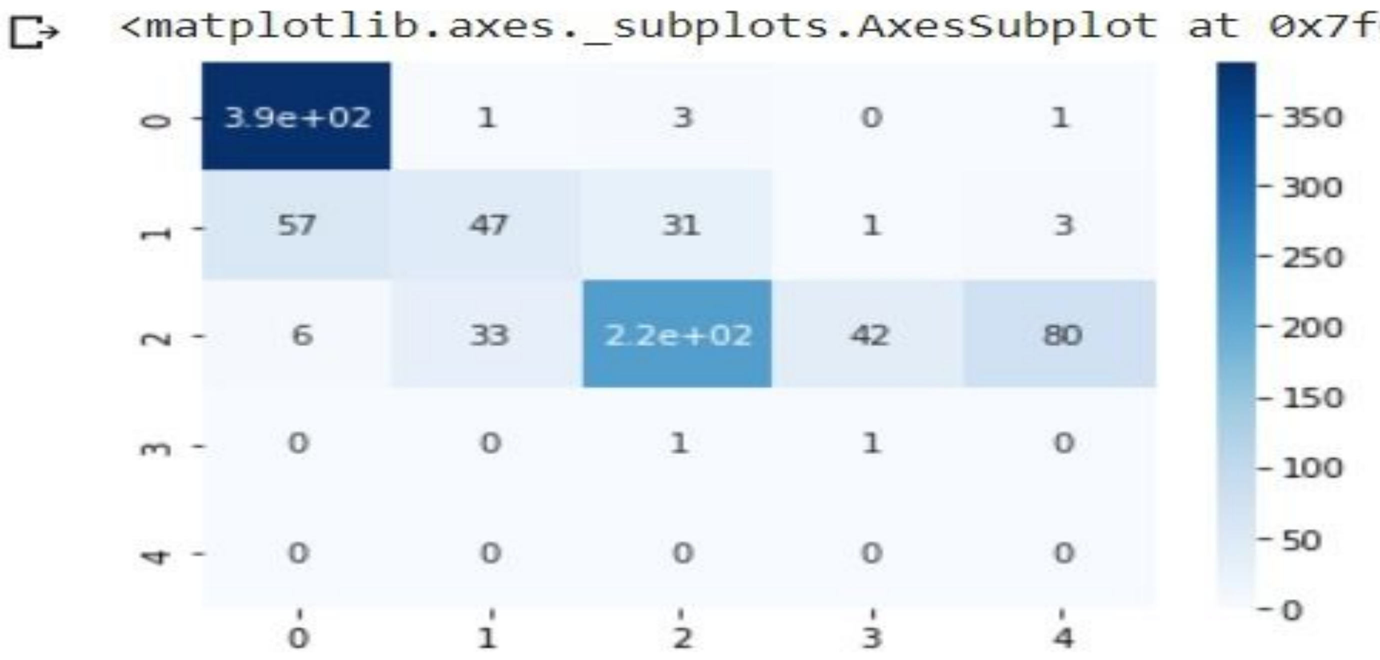
Train_Accuracy
Validation_Accuracy
0.96
0.95
0.94
0.93
0.92
0.91
0
1
2
3
4
5
6
7



```
<matplotlib.legend.Legend at 0x7fc10d6904d0>
```

0.24
0.22
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0
1
2
3
4
5
6
7
Train_Loss
Validation_Loss

Confusion Matrix



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f012f53b810>
```

	0	1	2	3	4
0	3.9e+02	1	3	0	1
1	57	47	31	1	3
2	6	33	2.2e+02	42	80
3	0	0	1	1	0
4	0	0	0	0	0

THANK YOU