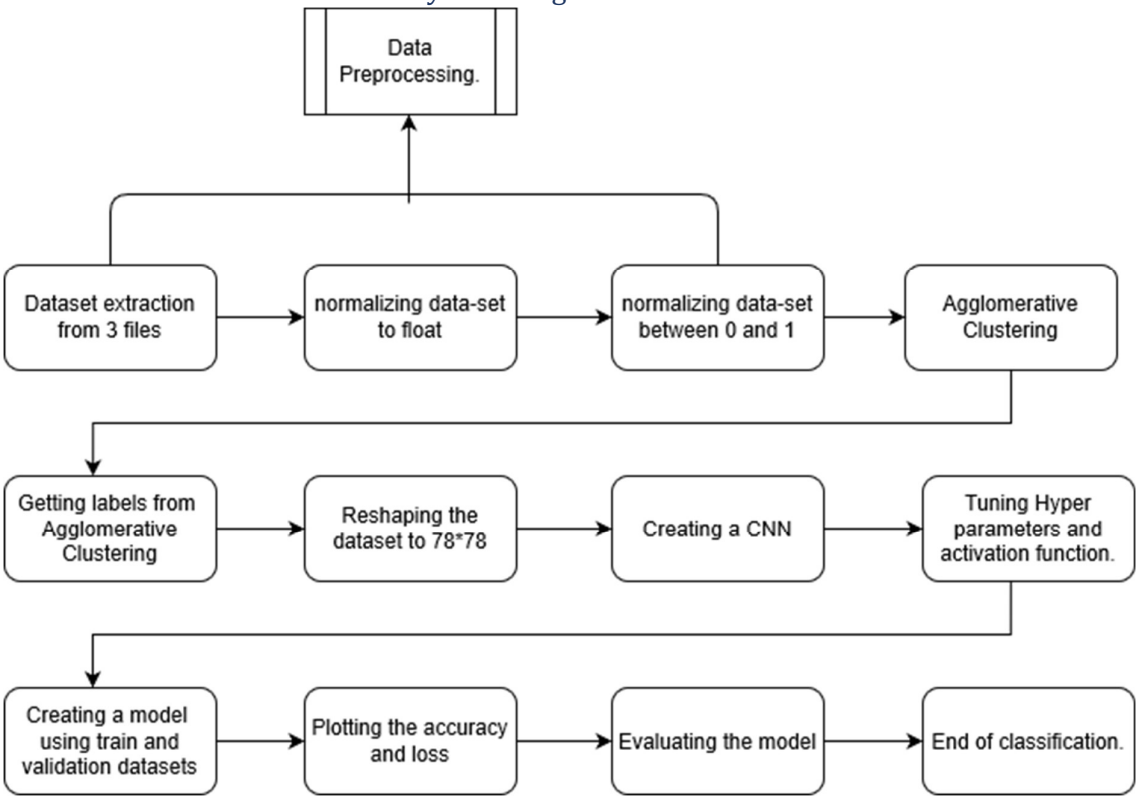
Step-4

A. classification technique

For the Hep-2 data I think the CNN image classification method is the best suit for the provided dataset. I used a sequential model from keras library, which consists of functions like Dense, Conv2D, Dropout, Flatten, MaxPooling2D. To optimize the model, I used SGD optimizer to make a custom learning rate and momentum. For activating the layers, to add some non-linearity in the input data, I've used leaky relu because it blocks the problem of having the values under zero. In leaky Rectified Linear Units, 1 denotes a positive part and small fraction denotes the negative part. In addition, for activating the dense layers, I used SoftMax activation as it highlights the largest input and suppresses all the significantly smaller ones in certain conditions.

B. structure of the classification system diagram



C. process and prepare the image data

The provided dataset should have same shape, mean and the scalability. In order to achieve this, we need to convert the data into same shape first, which is 78*78. And also, as the images are colour images which RGB 3 channel, it will be hard to extract the features of the data. So, we convert the whole data into a single channel 1 which is grey scale. In addition, to keep the images in the same scale using the mean of each and every image, so that, we can reduce the effort of training our model. However, we need to normalize the data between 0 and 1 so, first we convert the data-set into float32 and divide it with 255 to make all the values to stay been 0 and 1. Moreover, the provided data set should be achieved by unsupervised learning, which mean it's a non-labelled dataset. To train our model efficiently we generate the labels using the clustering method, AgglomerativeClustering which plots the dataset according to the provided no of clusters and generate the labels for us.

D. parameter of this classification system

The parameters I used to train the model is, lr = 0.01, momentum = 0.9, weight decay = 0.00005,dropout = 0, epochs = 35, Mini-batch size = 77, verbose = 1, layer activation = leaky relu and final activation = softmax.

The hyper-parameters should be tuned properly in order to train the model properly without any fluctuations. For instance, assigning more values to learning rate and dropout will fluctuate the model and effect the accuracy of the training process.

Model summary:

Model: "sequential_13"

_____

Layer (type) Output Shape Param #

==========

conv2d_33 (Conv2D) (None, 72, 72, 6) 300

_____

max_pooling2d_31 (MaxPooling (None, 36, 36, 6) 0

_____

conv2d_34 (Conv2D) (None, 33, 33, 16) 1552

_____

max_pooling2d_32 (MaxPooling (None, 11, 11, 16) 0

_____

conv2d_35 (Conv2D) (None, 9, 9, 32) 4640

_____

max_pooling2d_33 (MaxPooling (None, 3, 3, 32) 0

_____

flatten_11 (Flatten) (None, 288) 0

_____

dense_21 (Dense) (None, 150) 43350

_____

dropout_11 (Dropout) (None, 150) 0

_____

dense_22 (Dense) (None, 6) 906

==========

Total params: 50,748

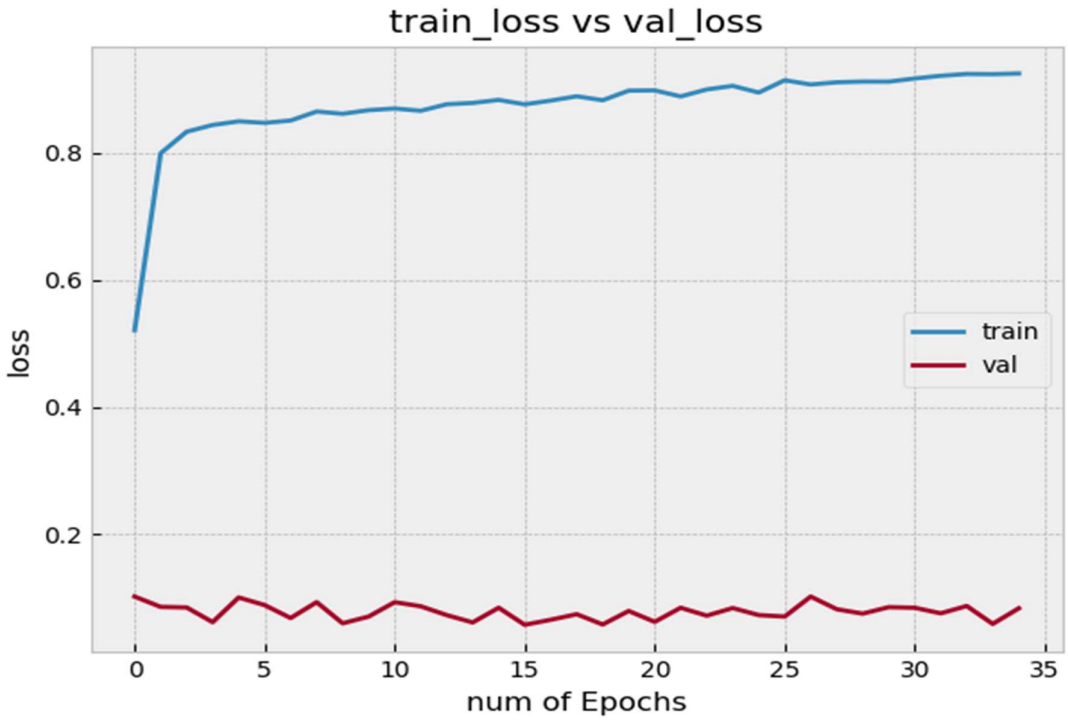
Trainable params: 50,748

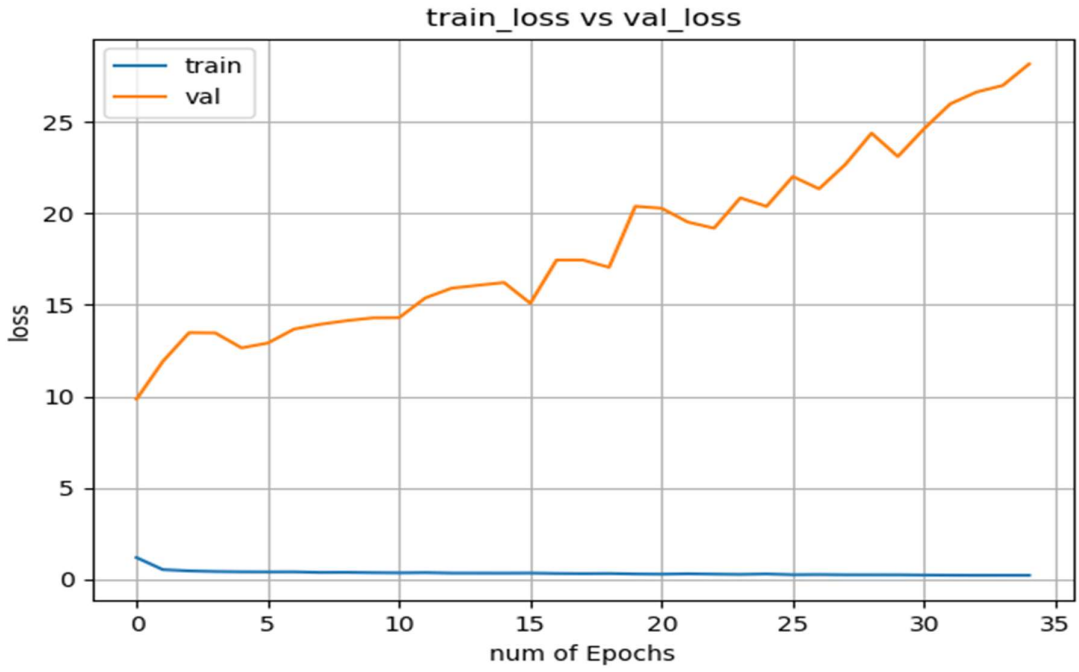
Non-trainable params: 0

_____

E. training and validation accuracy

Using the CNN model along with its hyper parameters, I acquired the training accuracy of **92.43%** which is good, but I'm getting the validation accuracy of only **0.08%** which means that the model is not fully efficient. Below figures will describe the training accuracy, validation accuracy and training loss and validation loss.





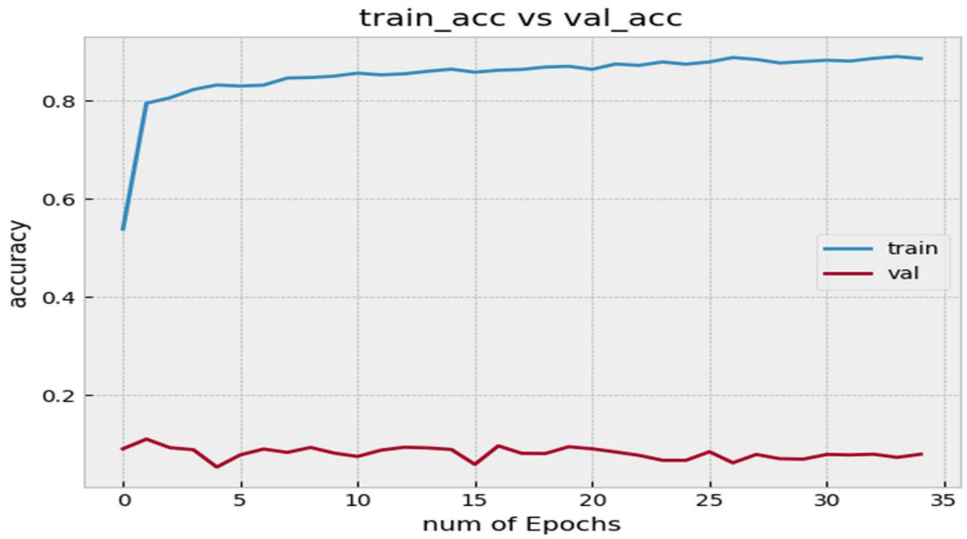
F. final classification accuracy on the test set

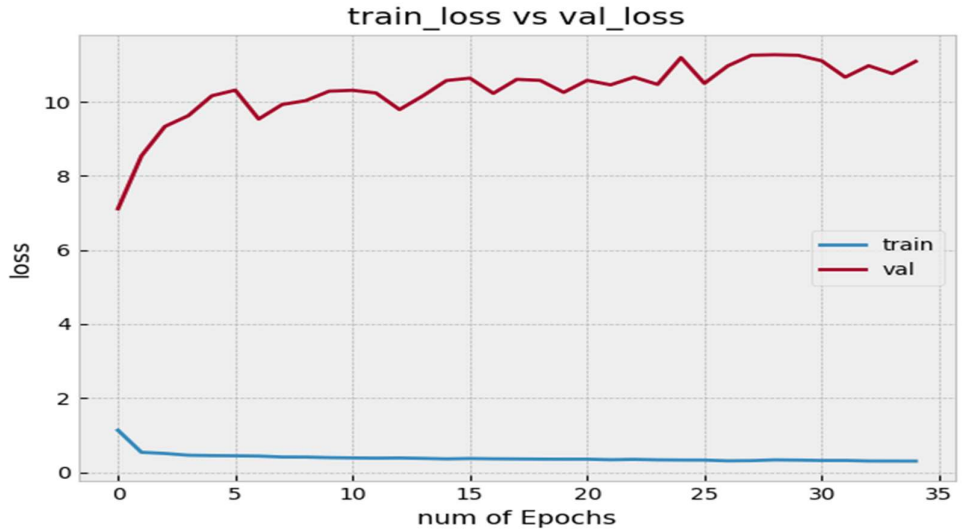
I used the method evaluate to get the accuracy of the test dataset and I got accuracy of **12.99%** with loss of **21.0.**

G. Observations

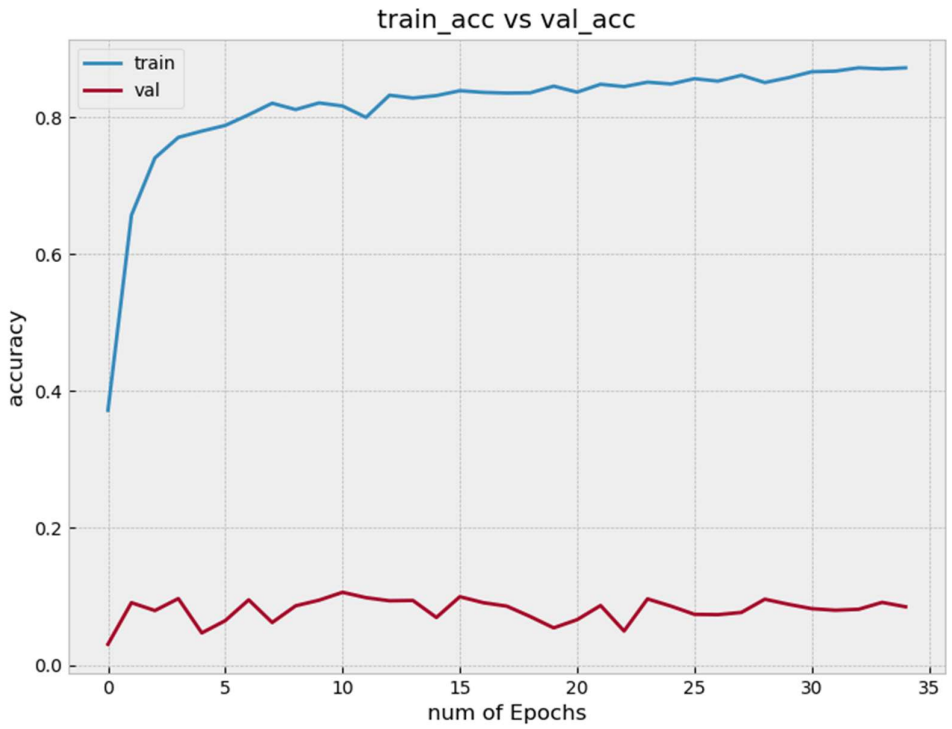
Firstly, to get the labels for the dataset from the provided .CSV file, I tried different ways to achieve this. But I got the labels by AgglomerativeClustering only which I am not sure of how accurately it categorized the data-set. Secondly, I think one of the main reasons why my models test accuracy is very low because of the imbalanced data-set. For instance, category class A has 3000 samples and , category class B has 1000 samples, and train the model with all the 4000 samples, it is more likely that the model would predict class A more accurately, when compared to class B. Finally, I've tried training the models with different activation functions and hyper parameters. If you see in below pictures, you can see that leaky_relu gave the best results possible when it is compared with relu and tanh.

Tanh:



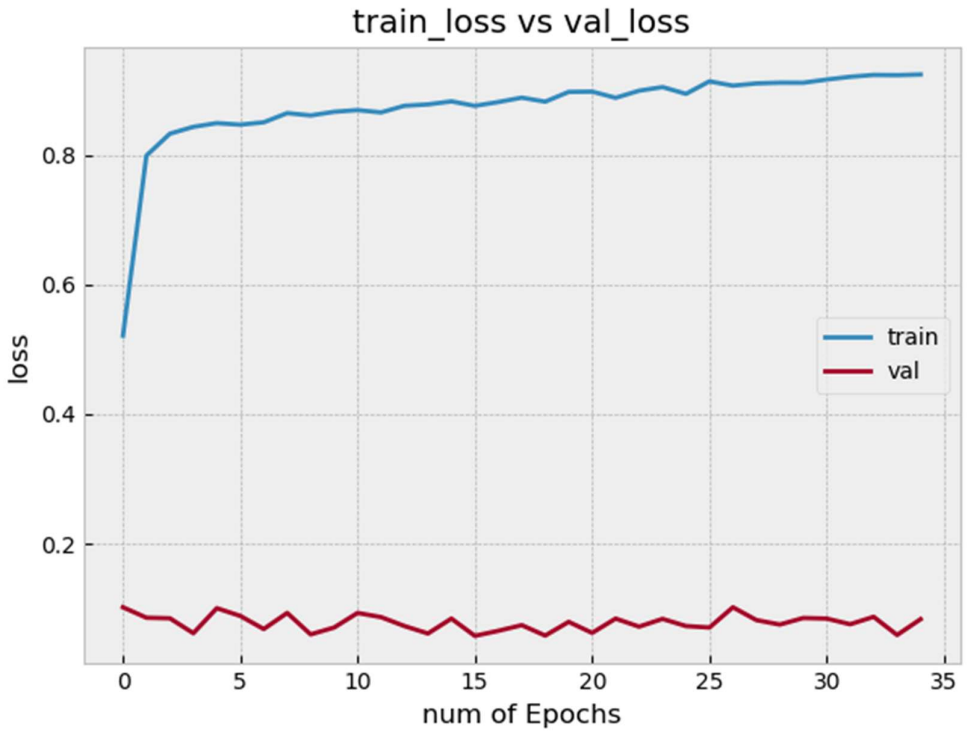


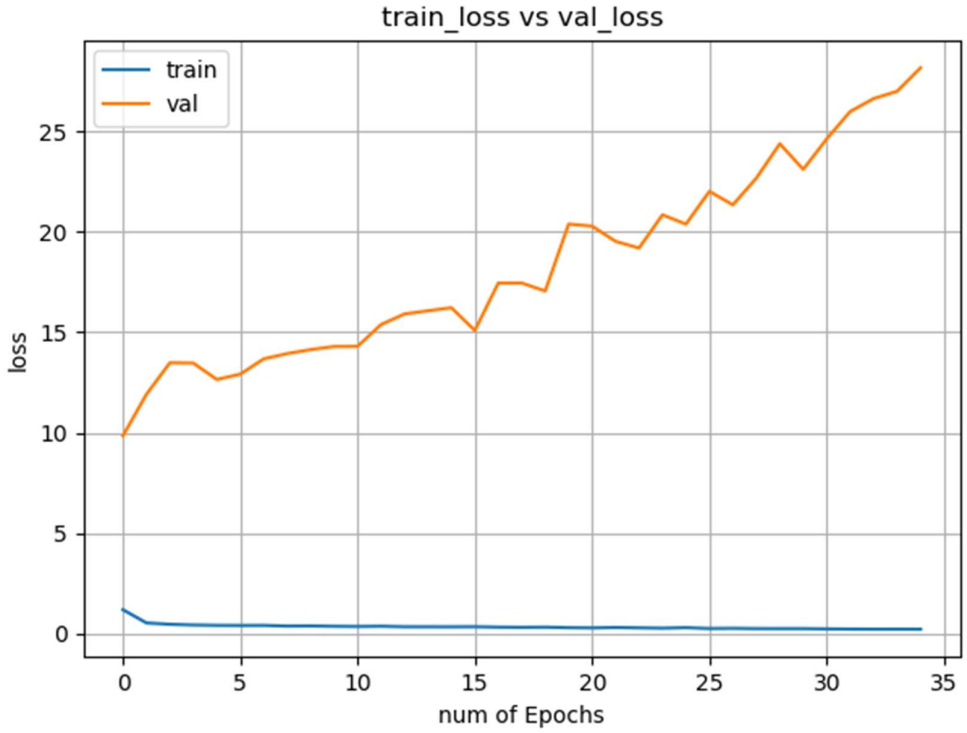
Relu:


train_acc vs val_acc
train
val
accuracy
num of Epochs
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
30
35


train_loss vs val_loss
train
val
loss
num of Epochs
0
2
4
6
8
10
12
14
16
0
5
10
15
20
25
30
35

Leaky Relu:


train_loss vs val_loss
loss
num of Epochs
train
val
0.2
0.4
0.6
0.8
0
5
10
15
20
25
30
35


train_loss vs val_loss
loss
num of Epochs
train
val
0
5
10
15
20
25
30
35

The table below showcases the observation of the 3 activation functions..

Activation / **Data-set**	**Tanh**		**Relu**		**LeakyRelu**	
	Epoch1	Epoch35	Epoch1	Epoch35	Epoch1	Epoch35
Training Accuracy	53.81	**88.58**	37.19	**87.21**	52.1	**92.43**
Validation accuracy	0.9	0.7	0.03	0.8	0.1	0.08
Validation loss	7.1	11.0	3.3	16.0	9.8	28.18