MRNet for Knee Diagnosis

Introduction

Magnetic resonance imaging (MRI) is the most prefered used method by clinicians to diagnose knee injuries but it takes long time to diagnose them and is subject to error so Deep learning methods are developed to help in the interpretation of the medical images . one of these methods is the MRNet model which cares about detecting general abnormalities and special abnormalities like anterior cruciate ligament (ACL) tears and meniscal tears.

Dataset

We have 1250 MRI exams in which 1130 exams for training and 120 for validation so we classified them as following:

- ➔ **Training**
  90% of training data (1017 exams)
- ➔ **Validation**
  10% of training data (113 exams)
- ➔ **Test**
  whole Validation data (120 exams)

Exams setup

Each MRI exam consists of S slices each one is of size (256 X 256 X 3) . Exams of patients are passed through three different series (Axial , Sagittal and Coronal) to check the probability of detecting the anomaly (abnormal , ACL and Meniscal) from each series.

Model

It mainly consists of three parts :

- ➔ **Feature extractor**

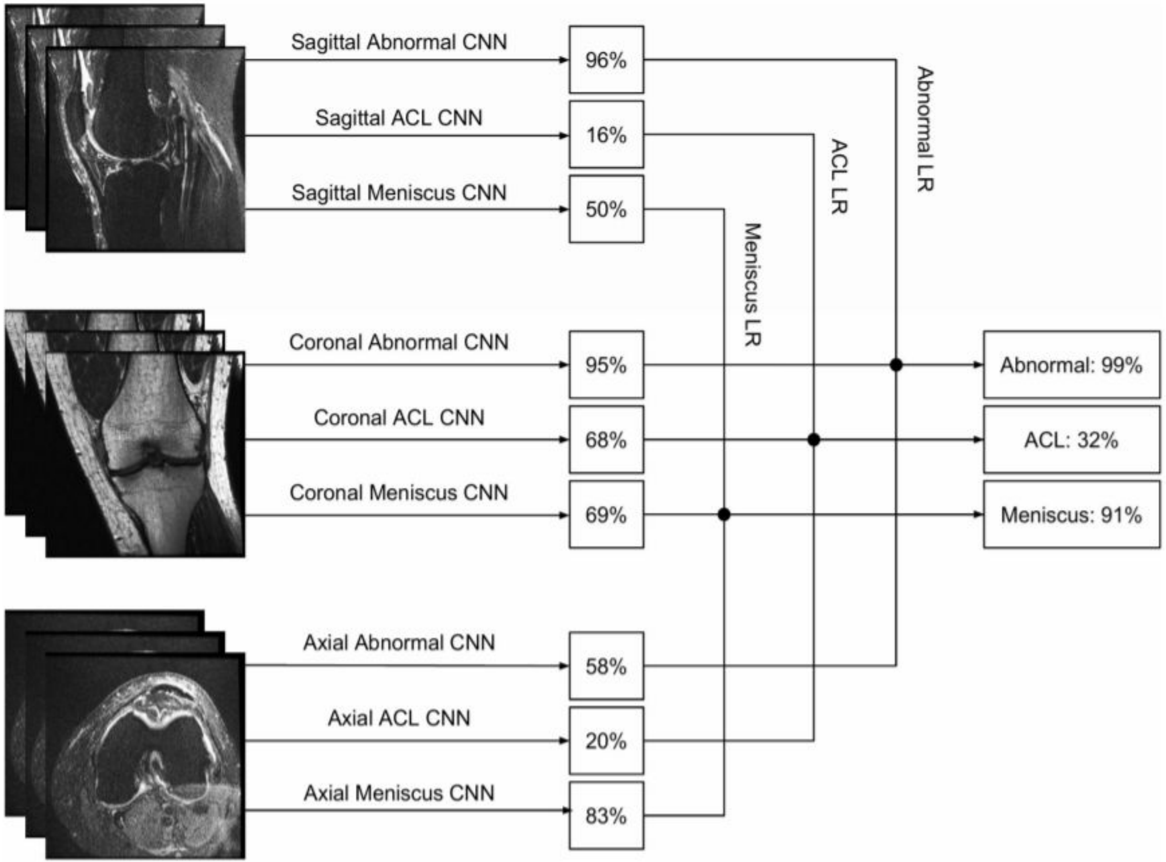
  For training, it takes the first slice from a certain series to specify the features of each anomaly according to a specific series. So, we have 9 total.

- ➔ **Classifier**

  To detect the probability of each anomaly according to its series. Also, 9 total.

- ➔ **Regressor**

  It finds the probability of detecting the anomaly by taking the predictions of three classifiers for the same anomaly from the three different series. 3 total.



Sagittal Abnormal CNN
96%
Sagittal ACL CNN
16%
Sagittal Meniscus CNN
50%
Coronal Abnormal CNN
95%
Coronal ACL CNN
68%
Coronal Meniscus CNN
69%
Axial Abnormal CNN
58%
Axial ACL CNN
20%
Axial Meniscus CNN
83%
Meniscus LR
ACL LR
Abnormal LR
Abnormal: 99%
ACL: 32%
Meniscus: 91%

Feature extractors models

- We used three different CNN models for training the feature extractors :

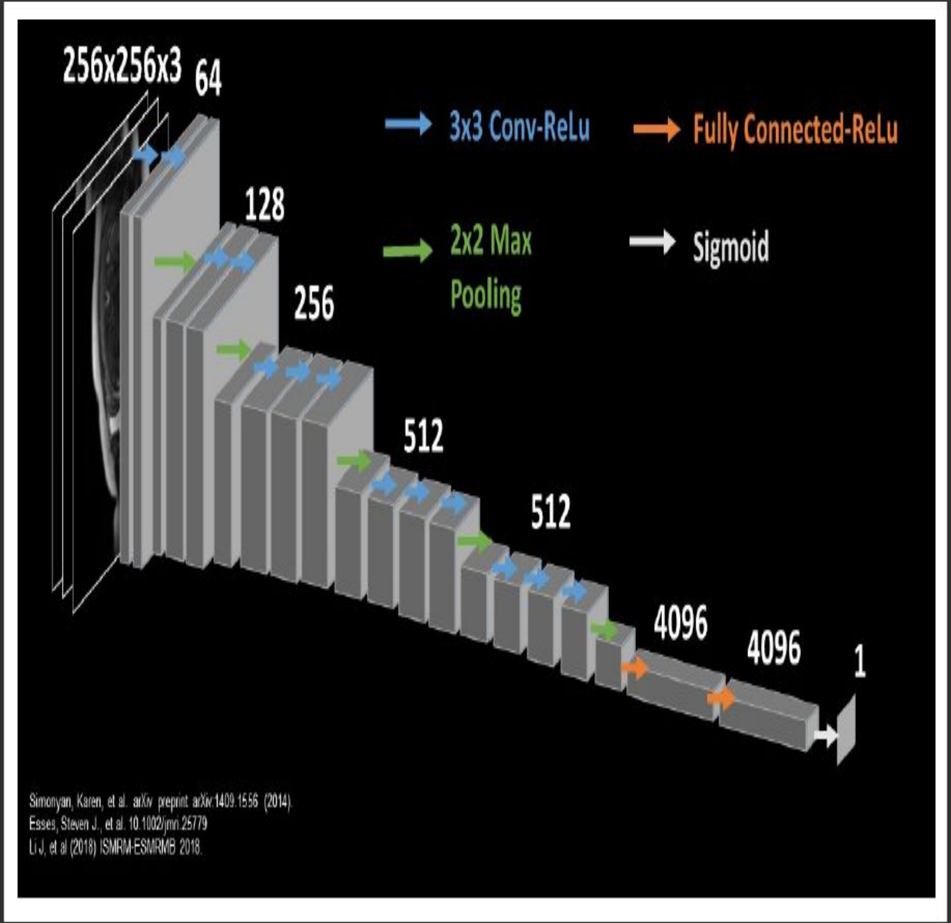
VGG

RESNet

Inception V3

VGG

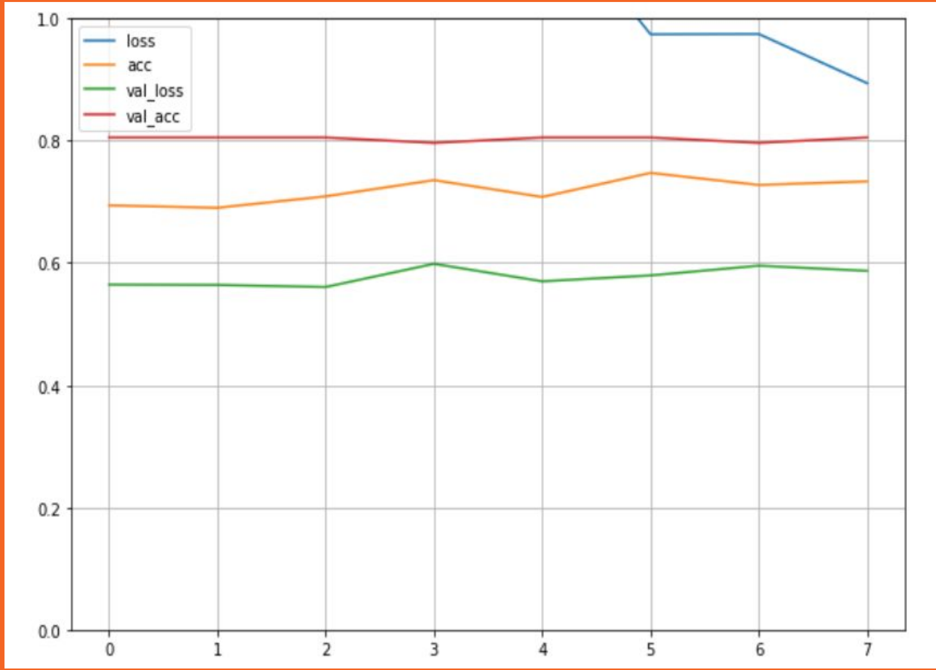
- It is a simple neural network architecture of 16 layers with simple hyperparameters. All conv layers are 3X3 filters and same padding . for all max pooling layers we used 2 X 2 filters with stride of 2



Training Extractors

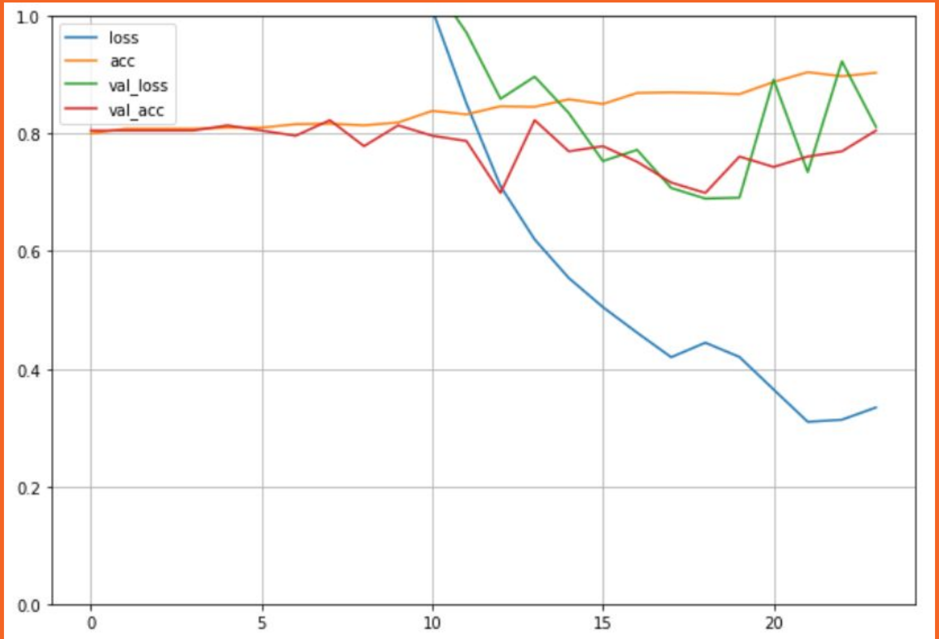
Extractor 1

By using low learning rate (10^-6) but the loss decayed slowly.



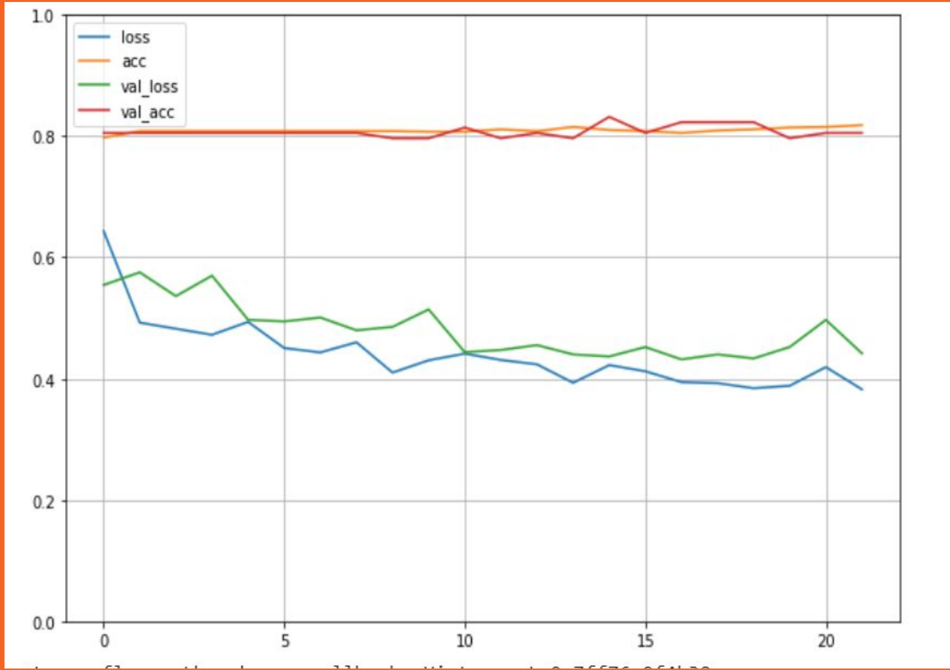
Extractor 2

Using regularization term = 0.01 but it increases the validation loss



Extractor 3

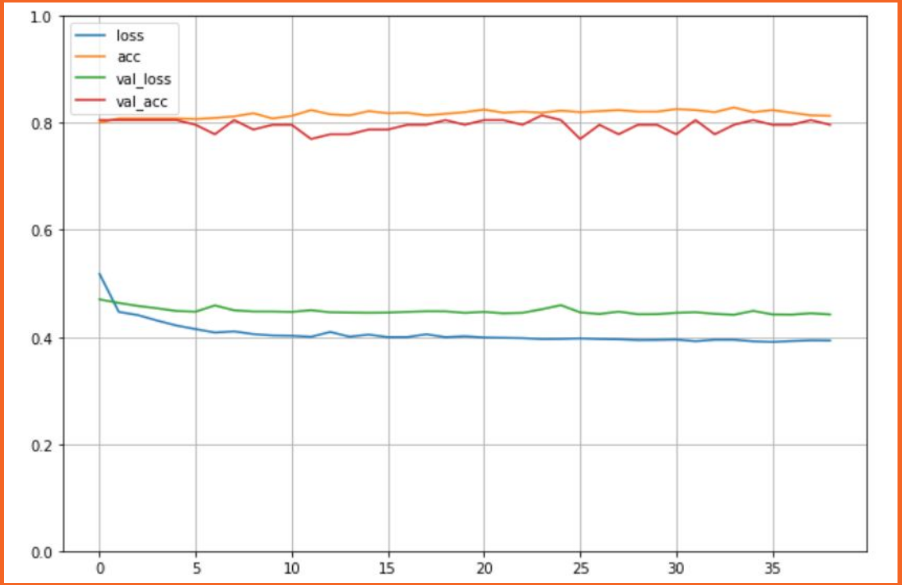
It is the final model for extractor with learning rate (10^-4) and use some dropout layers



Training Classifiers

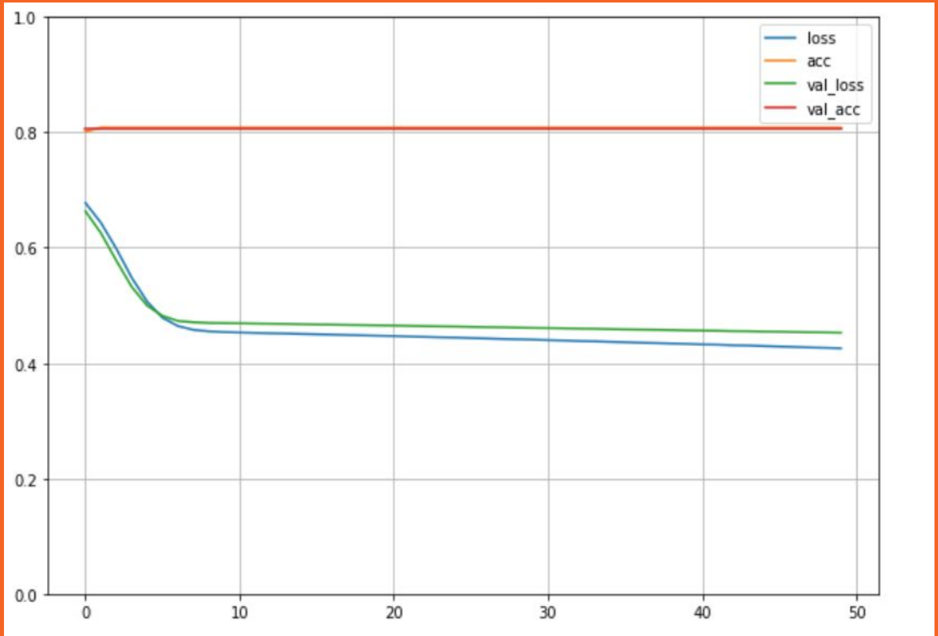
Classifier 1

Using many neurons in dense layers (4096)



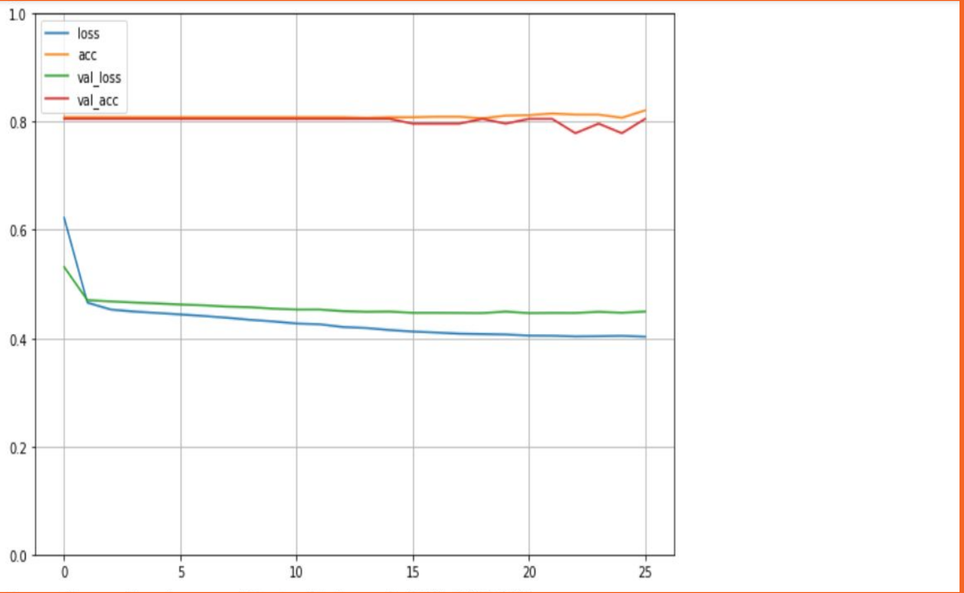
Classifier 2

Using less neurons in dense layers with high learning rate (0.01)



Classifier 3

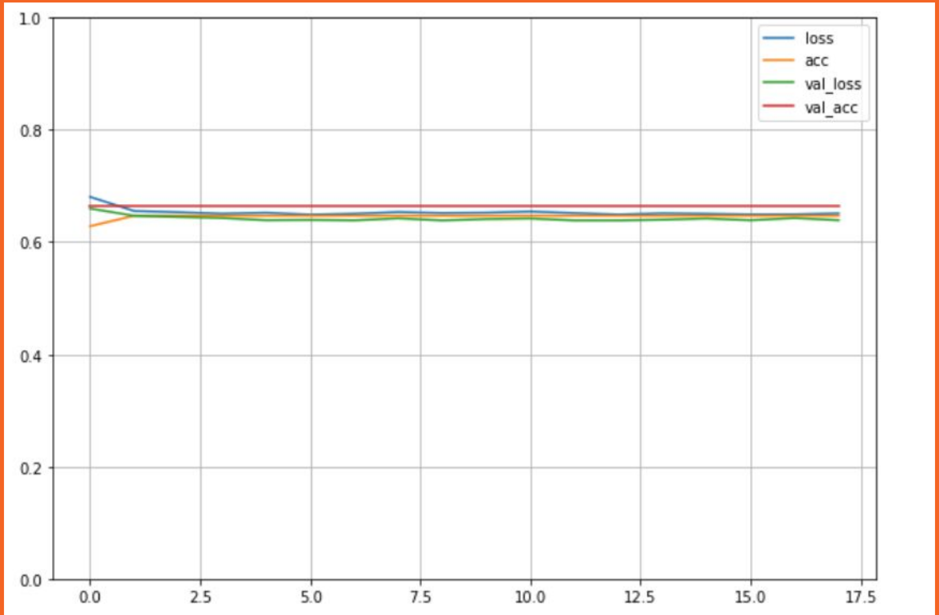
It is the final model for classifier with learning rate (10^-4) and use (1024 neurons in first dense layer and 512 in the second one).



Training Regressors

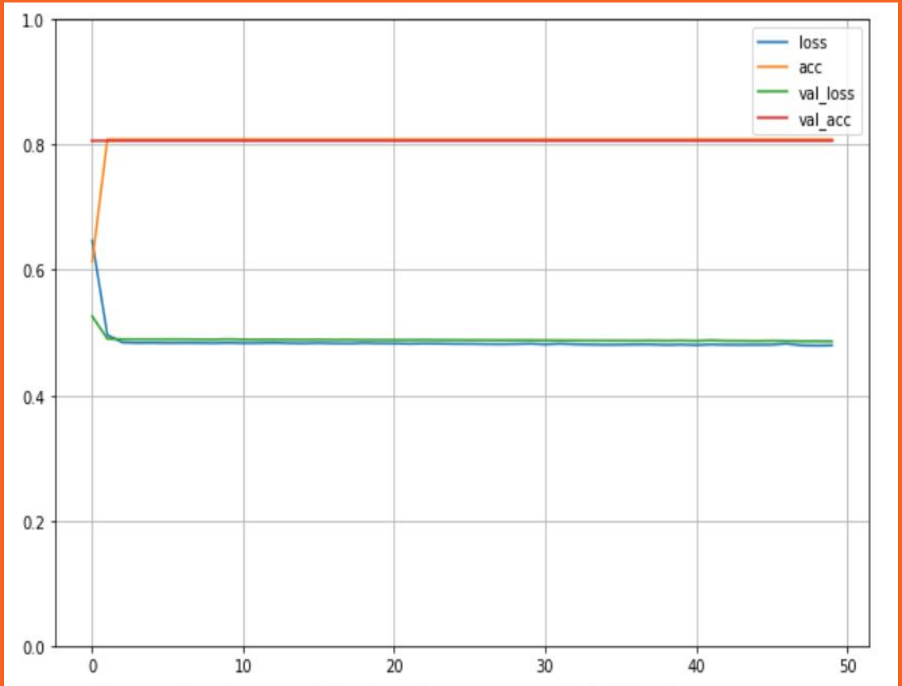
Regressor 1

Using low learning rate (10^-4)



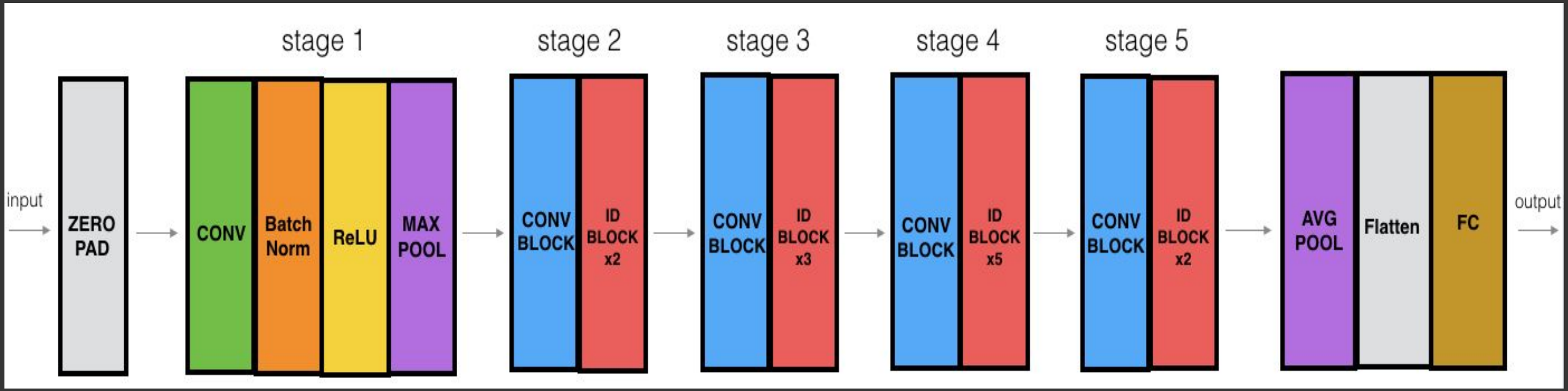
Regressor 2

It is the executed model for regressor by
Using higher learning rate (10^-2)



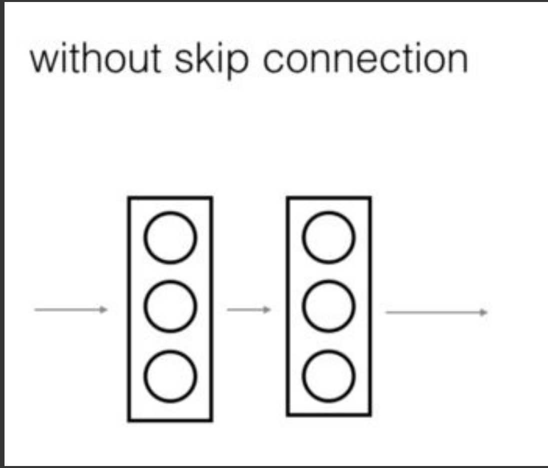
RESNet

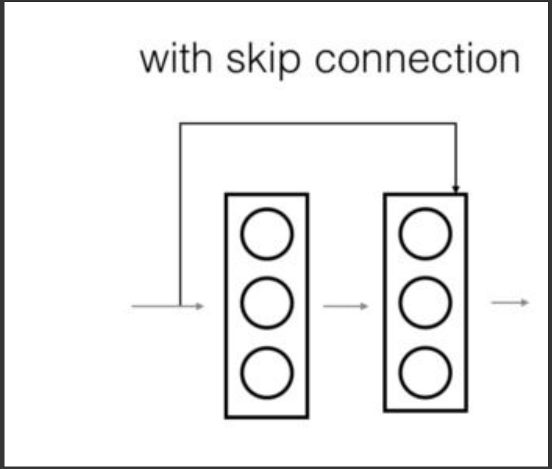
- It introduces the idea of training a very deep neural networks using the residual blocks that allows us to take the activation from a layer and feed it to a further layer in the network.



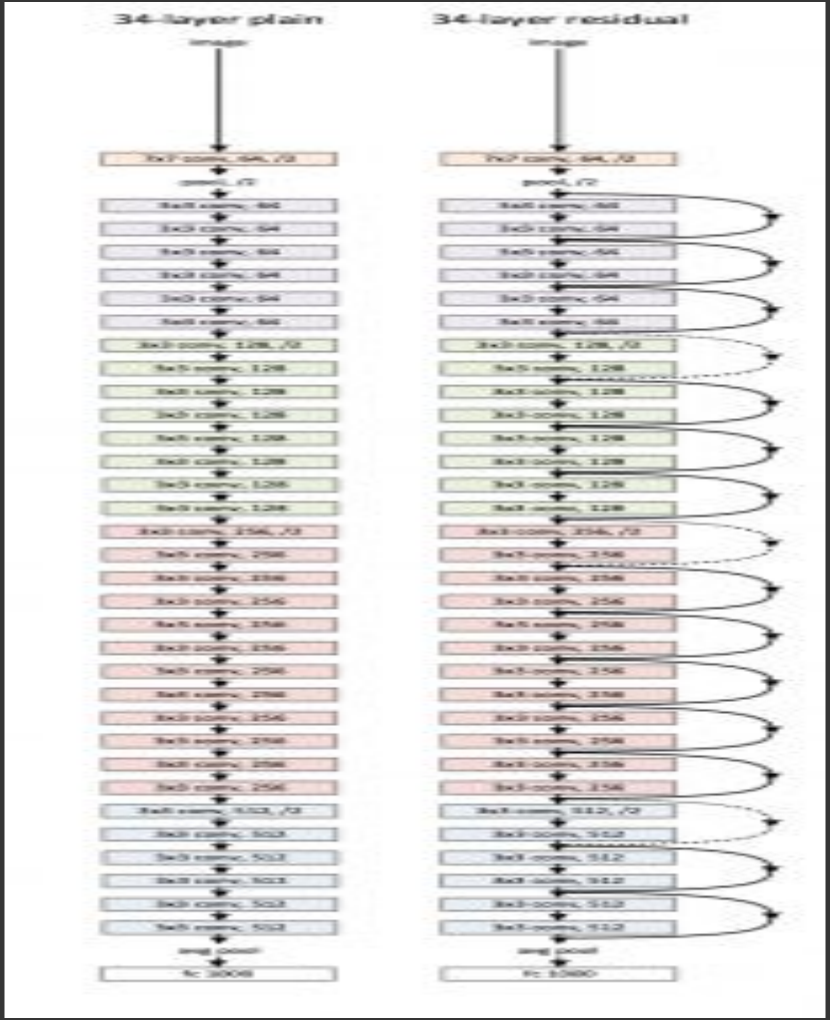
Idea

- When the problem is more complex we need more complex network to solve
- deep networks suffer from overfitting because it may get harder to learn anything in some layers
- In this situation the residual blocks allows the layer to be skipped by feeding the output of the previous layer instead of overfitting



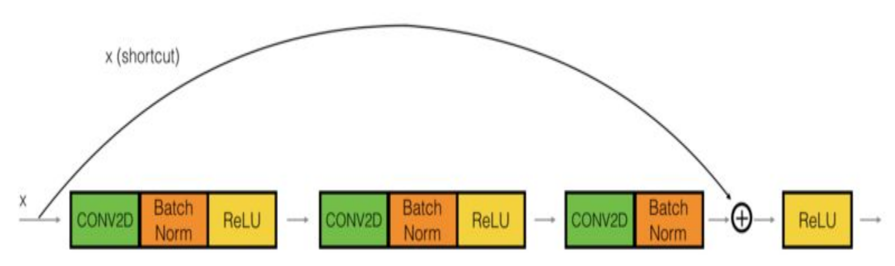


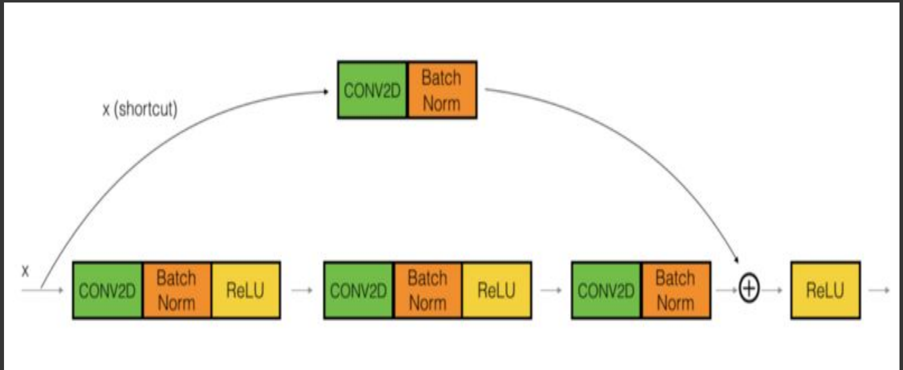
This transforms the regular network to this new shape with the skip connections



Residual blocks

- The skipping operation is implemented via adding the i/p of the block to the o/p of the same block before performing the activation
- If the dimensions are compatible we will add them , if not we will need to pass the i/p via conv layer to make them add compatible
- That's why we have 2 types of block identical,conv blocks

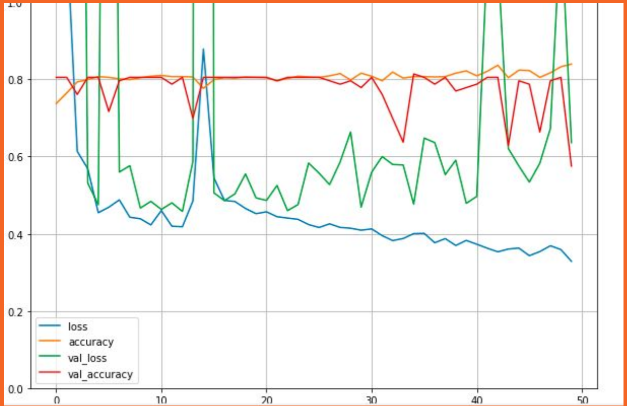


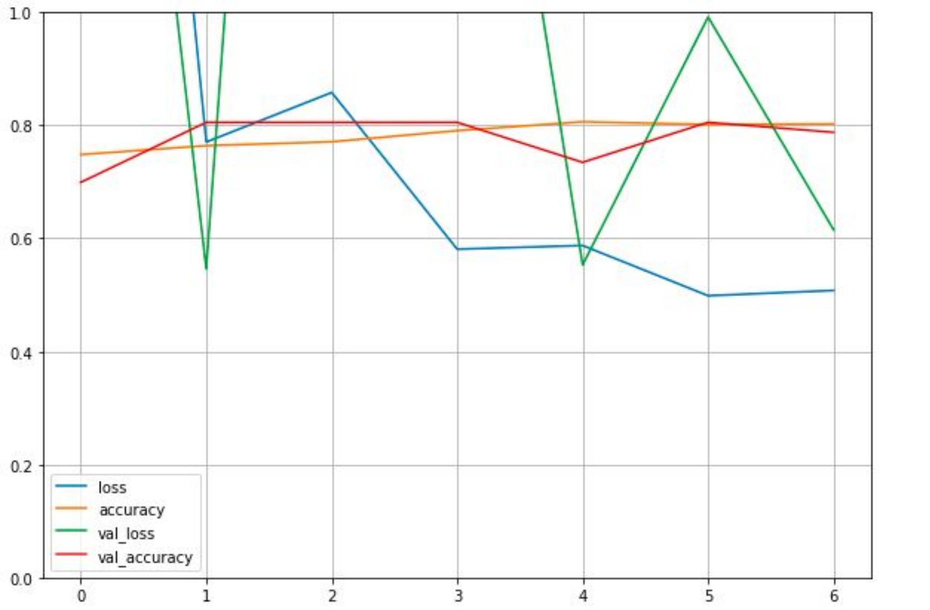


Training Extractors

Extractor 1

First attempts were too bad without any modifications

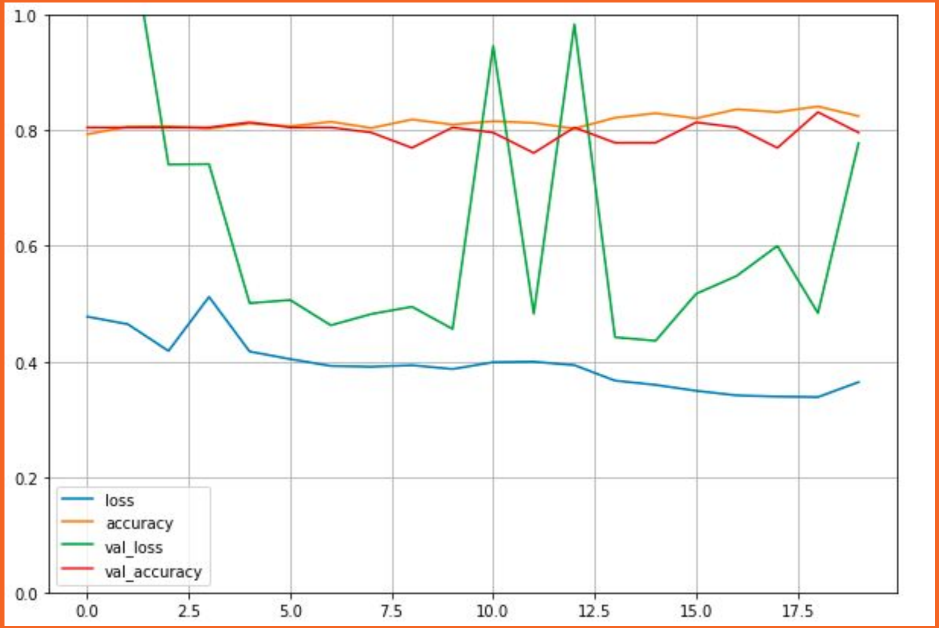




Extractor 2

Adjusting the params and the dense layer

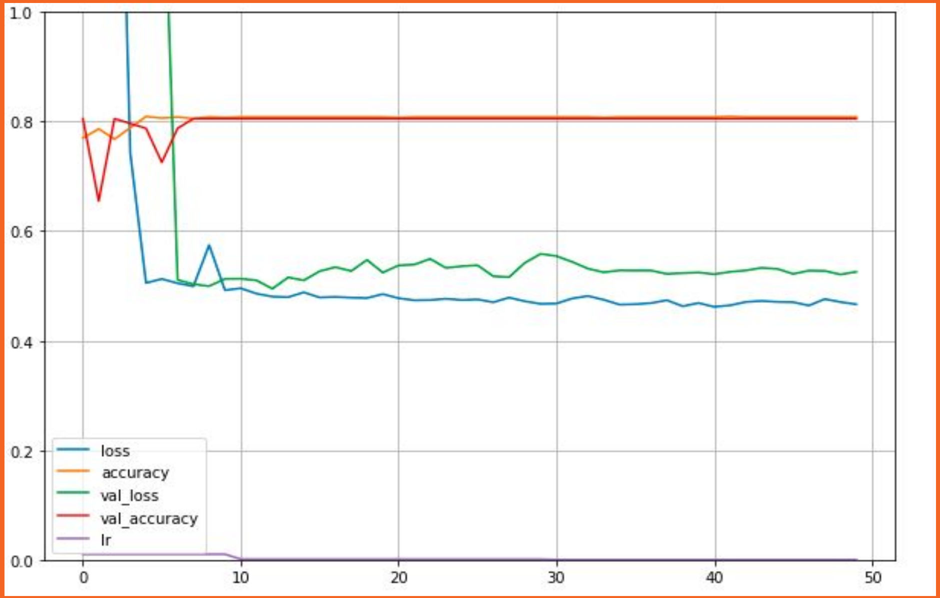
it was better but seems to over fit



Extractor 3

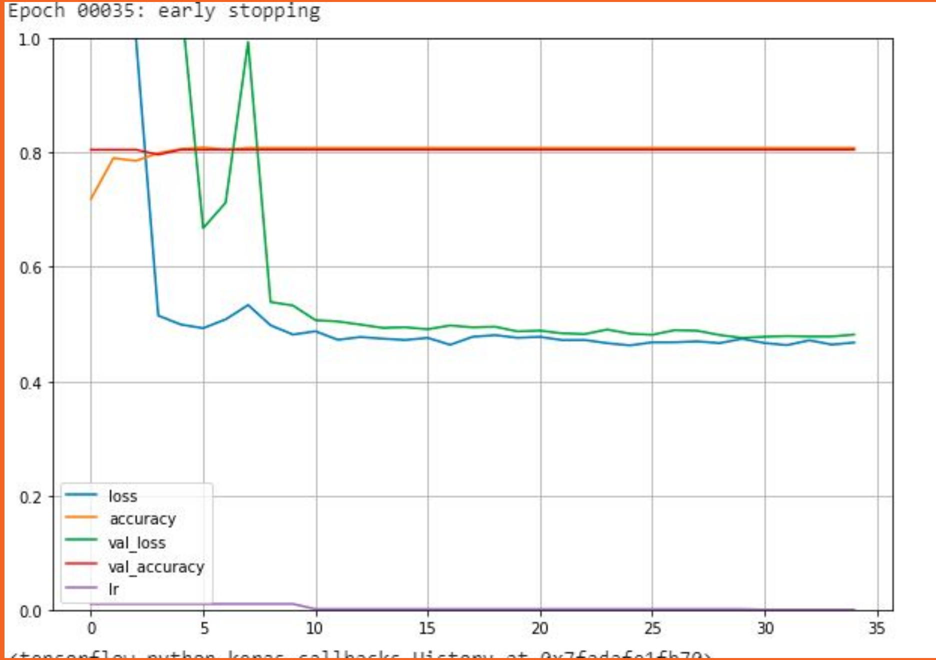
Adding learning rate scheduler call-back to start with 0.01 then 0.001 then if we continue it will be 0.0001

And adding a dropout layer



Extractor 4

Now adding the early stop call-back



Training Classifiers

Classifier 1

With the classifier the same params from the extractor wew just fine

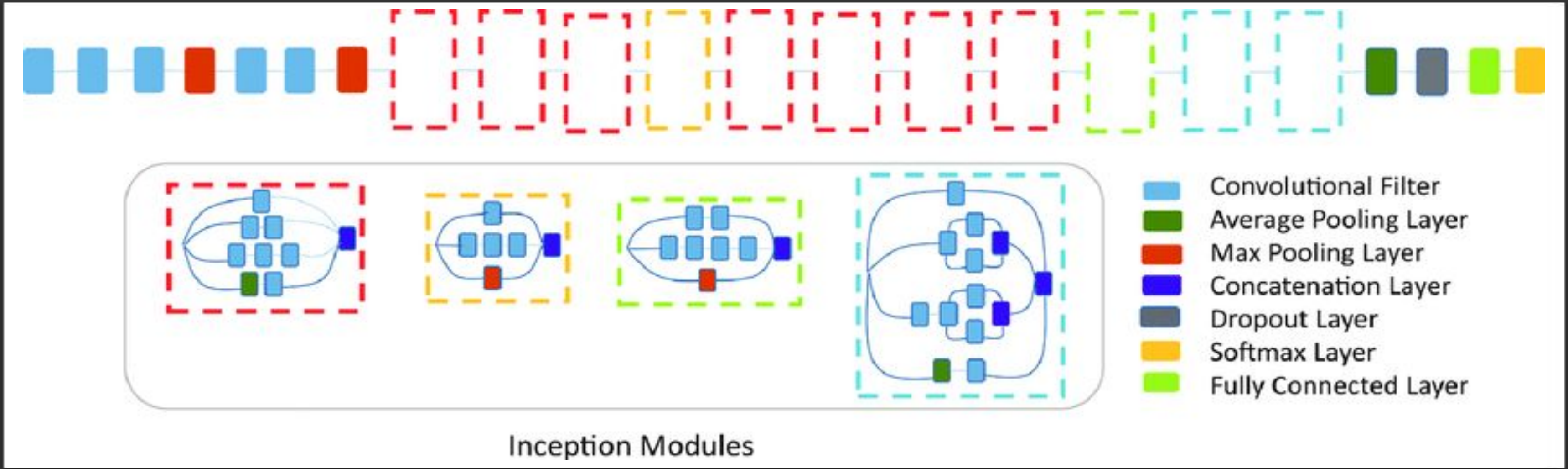
Training Regressors

Regressor 1

Using the same previous params

Inception V3

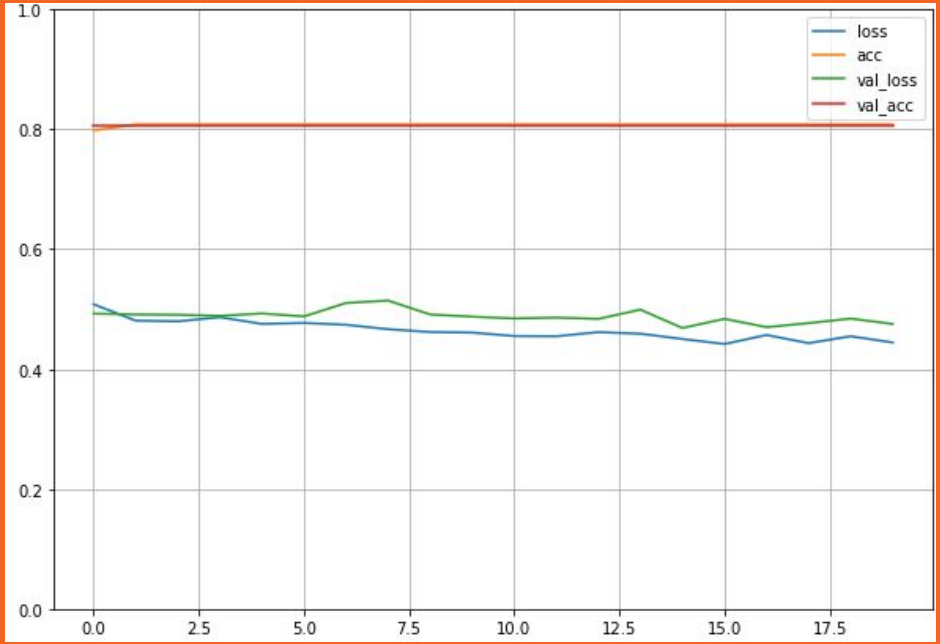
- Instead of using all filters of the same size for a specific layer , We use different sizes(eg: 3 X 3 , 1 X 1 , ...) of filters for a layer . It also helps in reducing the computational cost by using 1 X 1 filters.



Training Extractors

Extractor

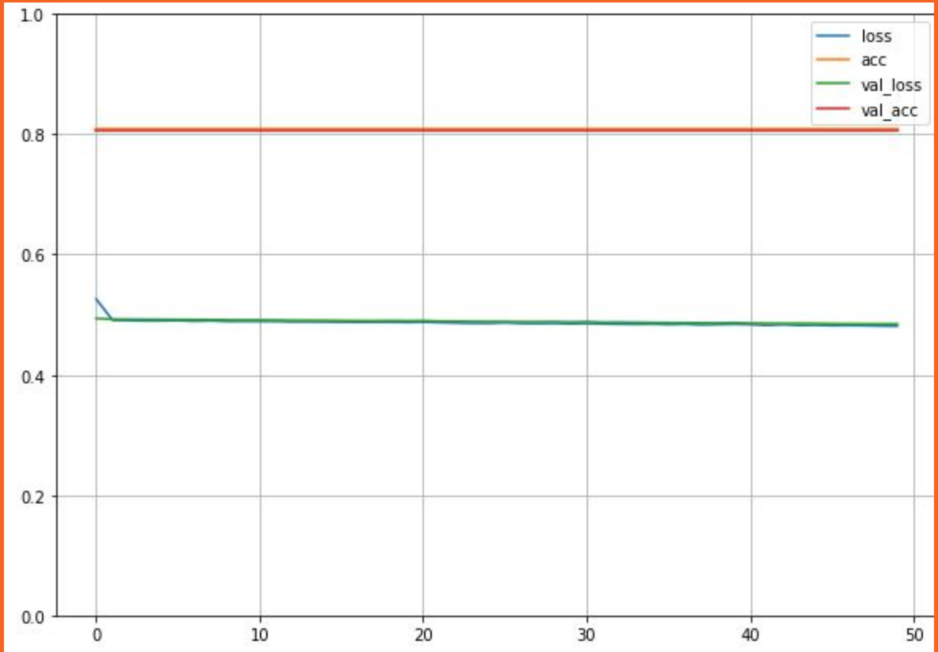
Decreasing the learning rate from 10^-3 to 10^-5 we now can see the changing in loss as it's decreasing but before it was approaching straight line.



Training Classifiers

Classifier

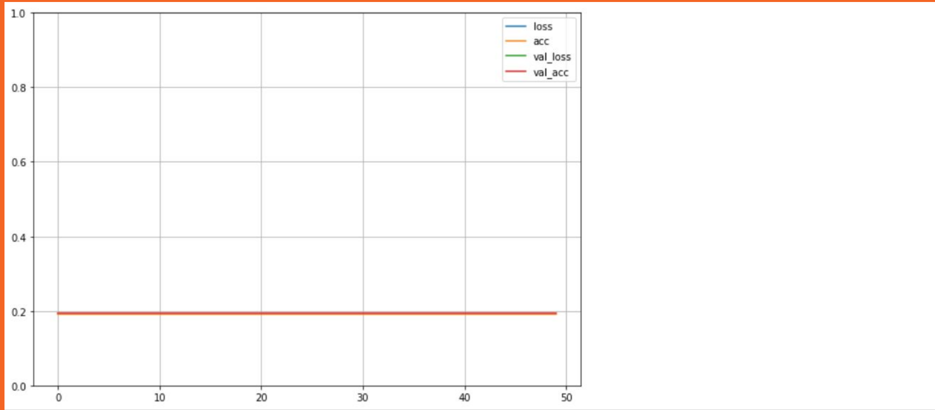
Using learning rate 10^-5 there is a little decreasing in loss but this can be changed by using different learning rate



Training Regressors

Regressor 1

Low Learning rate 10^-5 for abnormal regresor
- failed trial

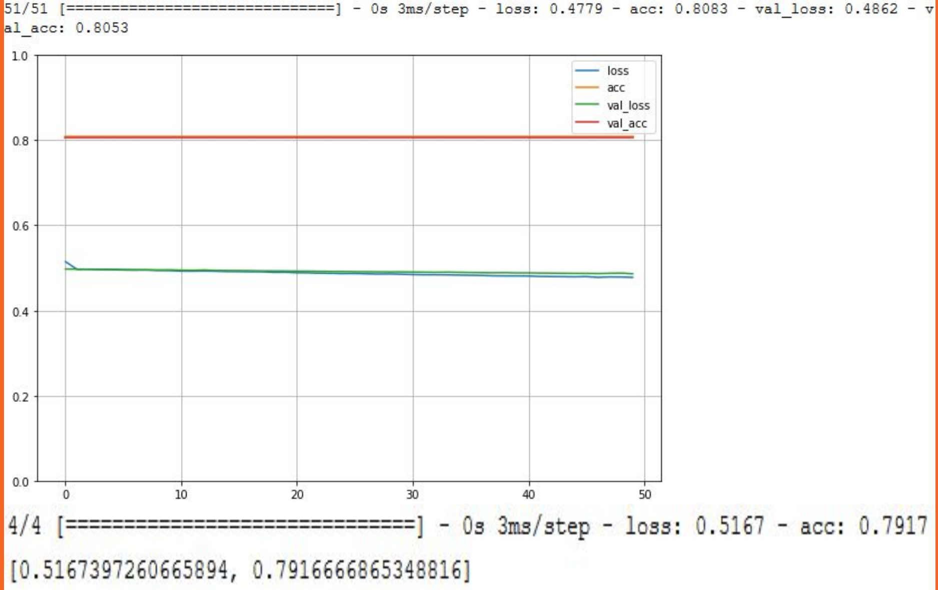



```
processing.test_regressor(axial_extractor,sagittal_extractor,coronal_extractor, processing.inception, processing.abnormal)

4/4 [==============================] - 0s 2ms/step - loss: 1.0370 - acc: 0.2083
[1.0369518995285034, 0.2083333283662796]
```

Regressor 2

Abnormal regressor after using learning rate 10^-2



Statistics

Regressors

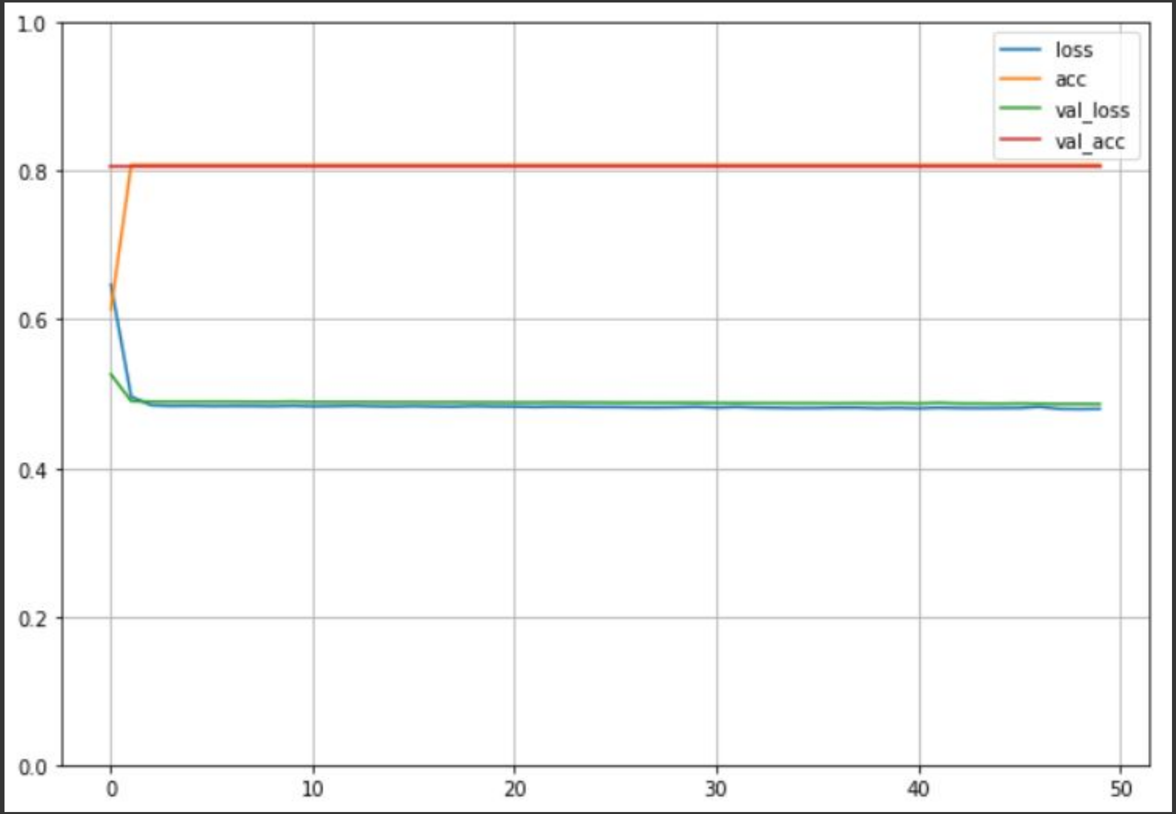
We judge the model by two methods

- ➔ Loss
- ➔ Accuracy

VGG

- **Abnormal**

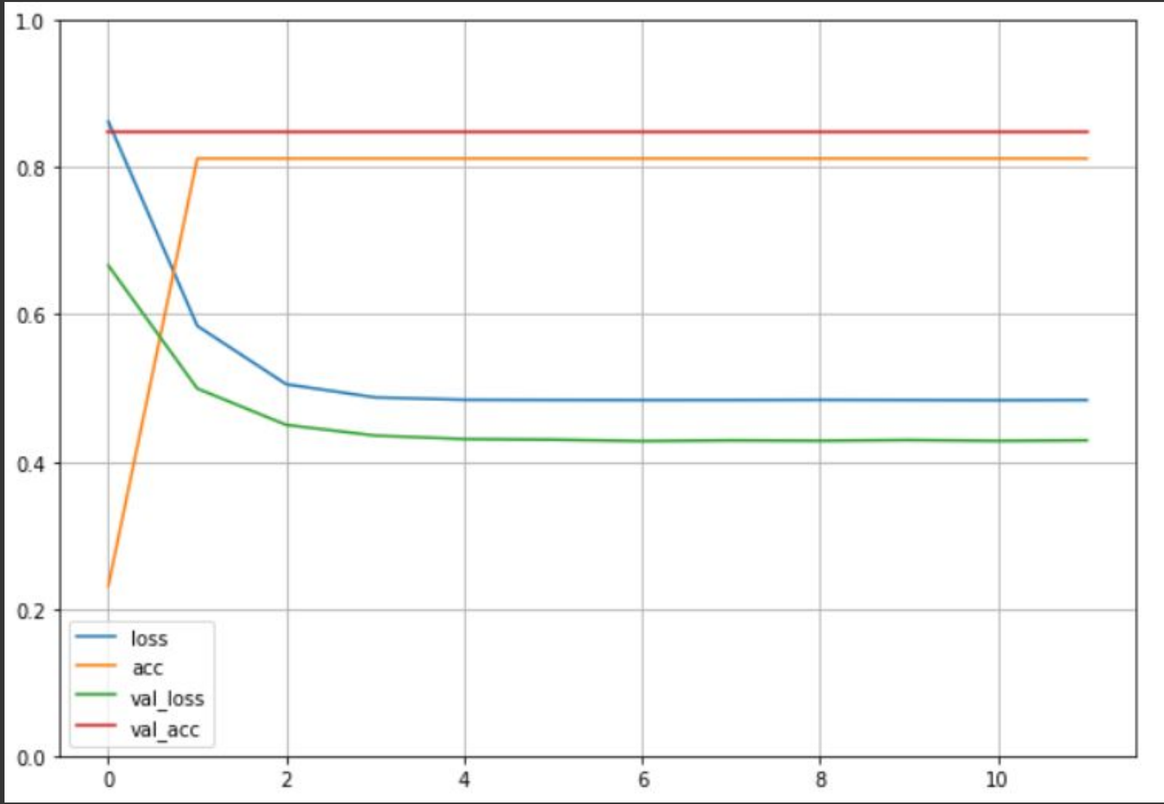
  Loss : 0.5356
  Accuracy : 0.7917



VGG

- **ACL**

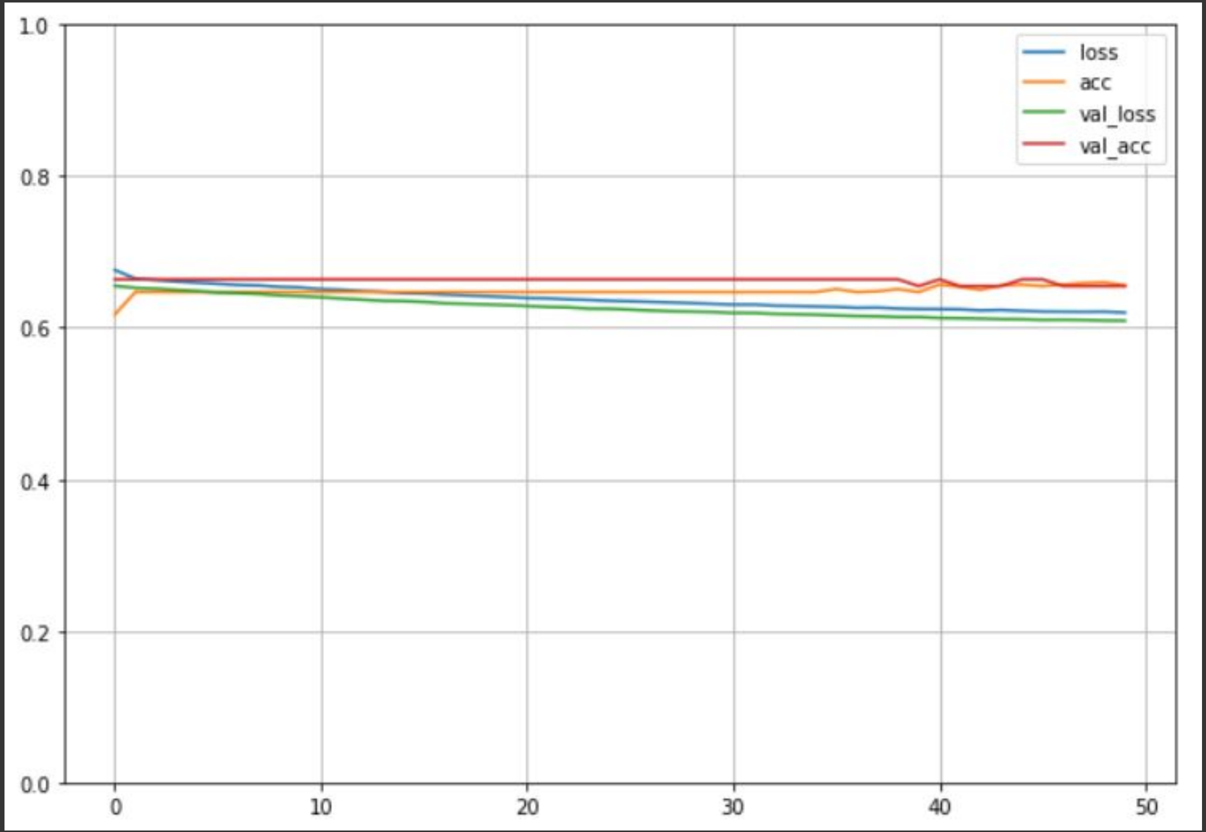
  Loss :   0.6903
  Accuracy : 0.5500



VGG

- **Meniscal**

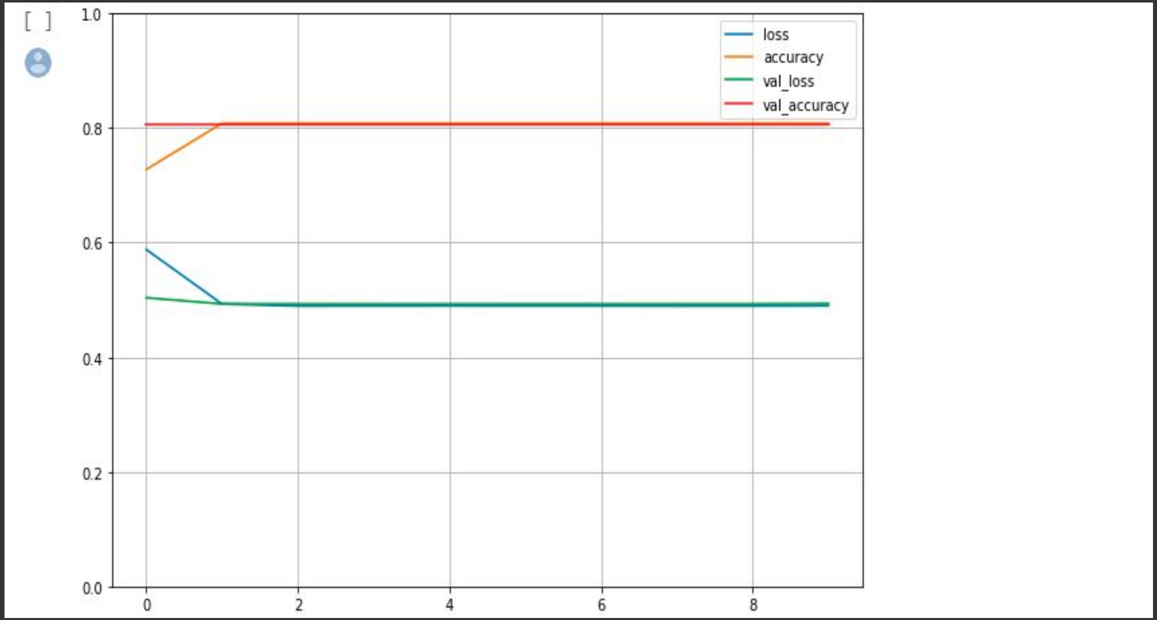
  Loss :   0.6982
  Accuracy : 0.5667



RESNet

- **Abnormal**

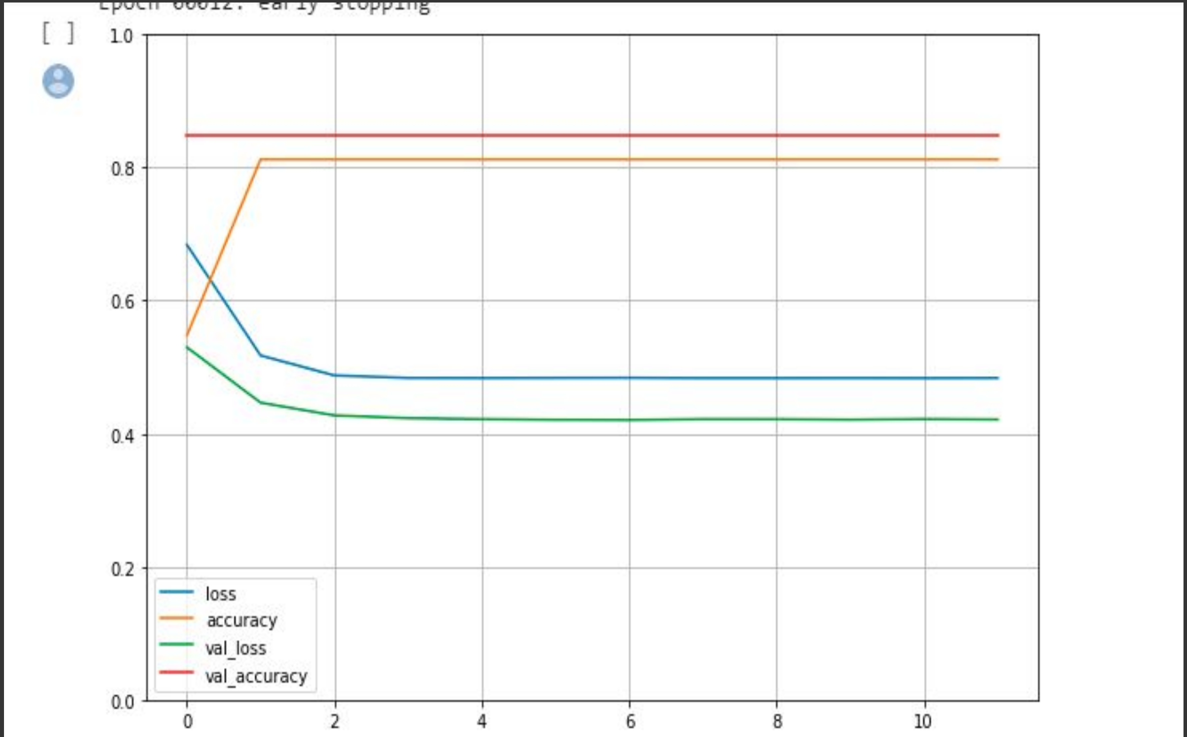
  Loss : 0.5182
  Accuracy : 0.7917



RESNet

- **ACL**

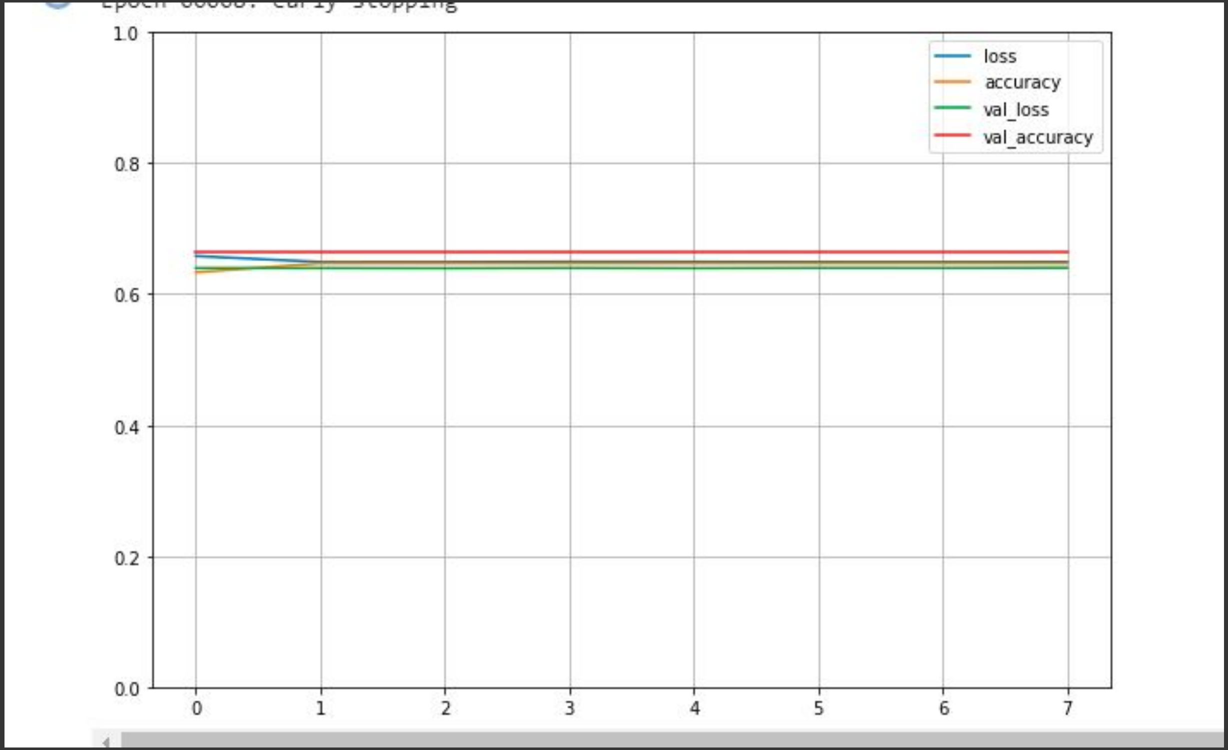
  Loss : 0.7051
  Accuracy : 0.5500



RESNet

- **Meniscal**

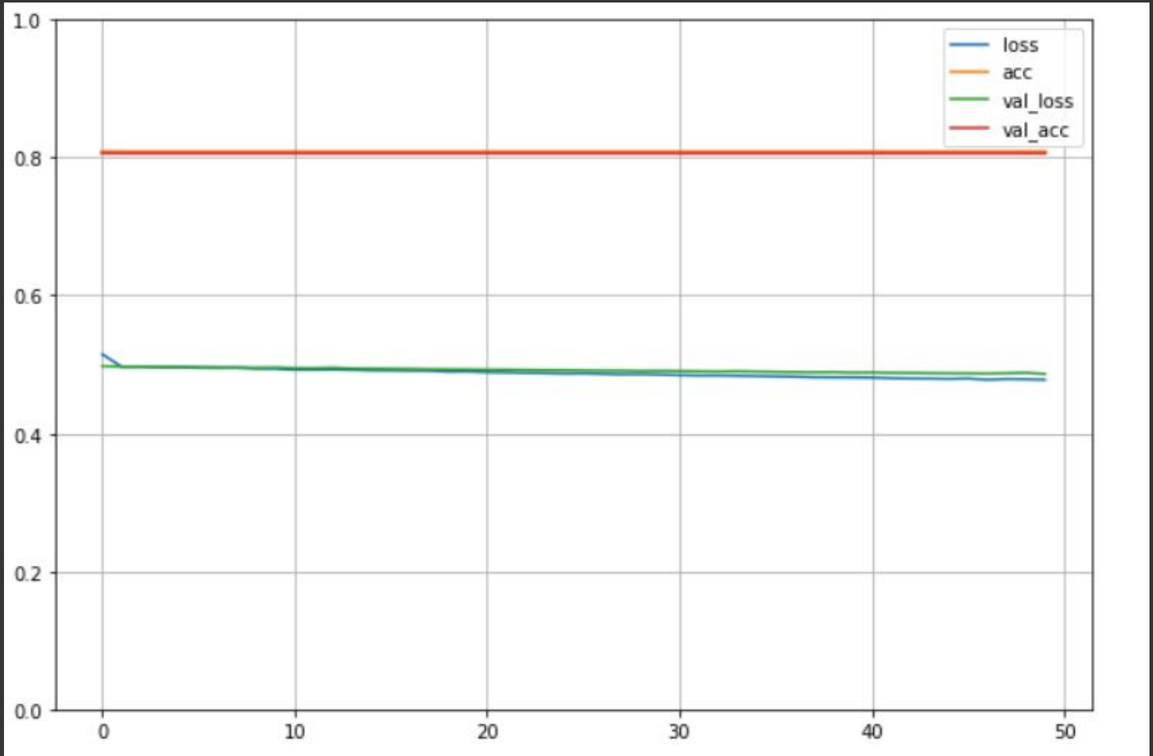
  Loss : 0.7024
  Accuracy : 0.5667



Inception V3

- **Abnormal**

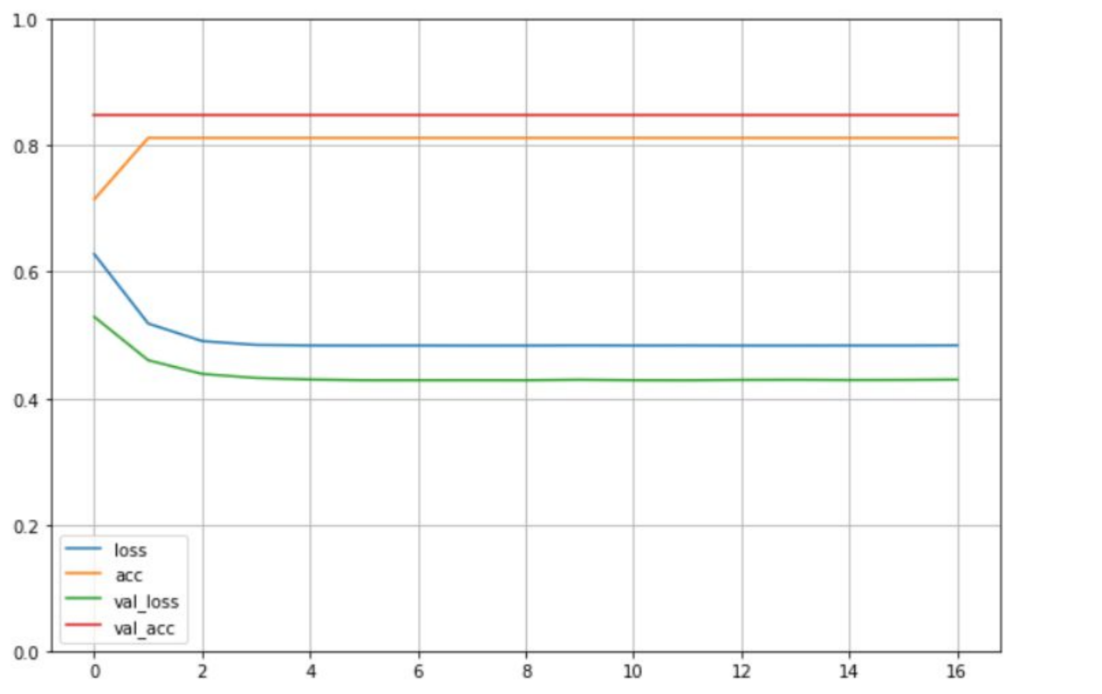
  Loss : 0.5167
  Accuracy : 0.7917



Inception V3

- **ACL**

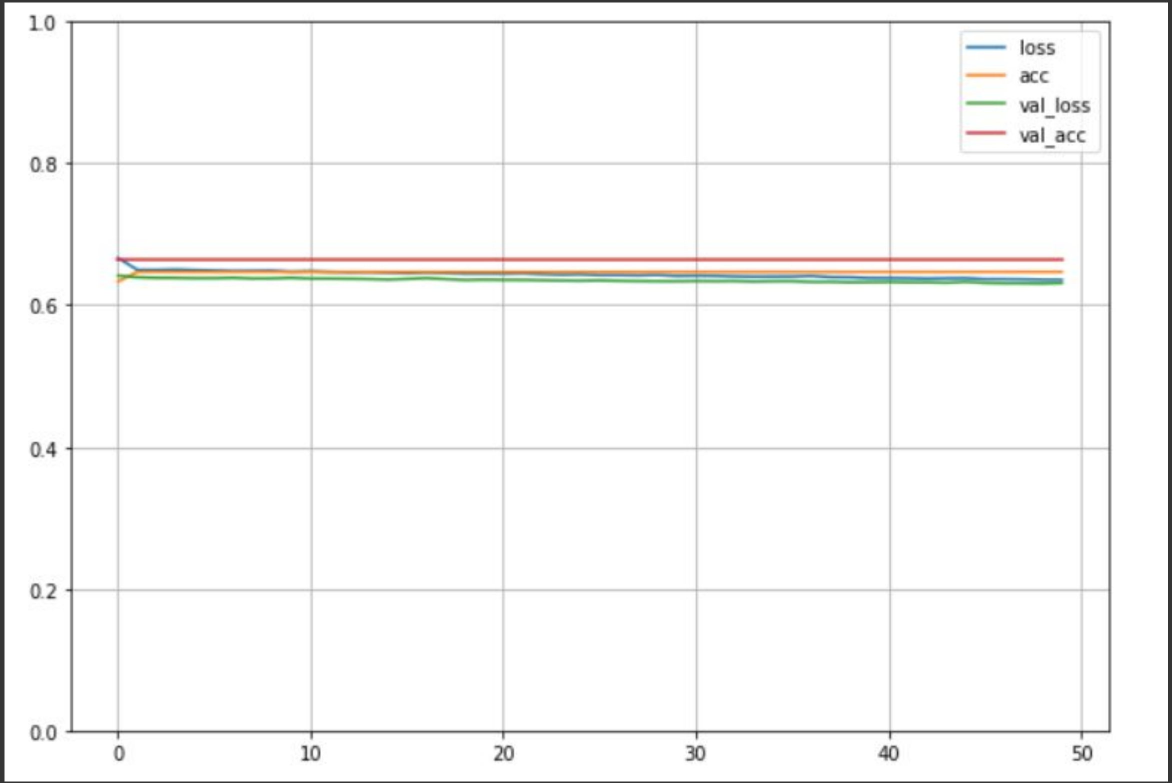
  Loss : 0.7071
  Accuracy : 0.5500



Inception V3

- **Meniscal**

Loss : 0.6912
Accuracy : 0.5667



models summary

	Anomaly	Accuracy	Loss
VGG	Abnormal	0.7917	0.5356
	Acl	0.5500	0.6903
	Meniscal	0.5667	0.6982
RESNet	Abnormal	0.7917	0.5182
	Acl	0.5500	0.7051
	Meniscal	0.5667	0.7024
Inception V3	Abnormal	0.7917	0.5167
	Acl	0.5500	0.7071
	Meniscal	0.5667	0.6912

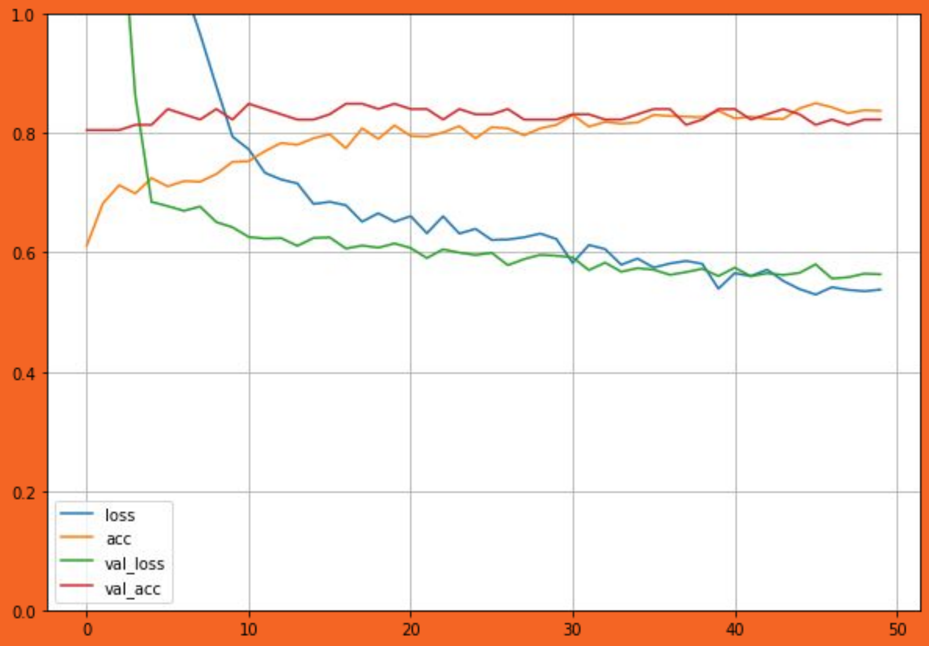
Transfer learning

We trained the three feature extractors (VGG , RESNet and Inception V3) by the models implemented in keras by training them on ImageNet which contains more than 14 million different images and then start training of classifiers and regressors by our Dataset.

Training Classifiers For VGG

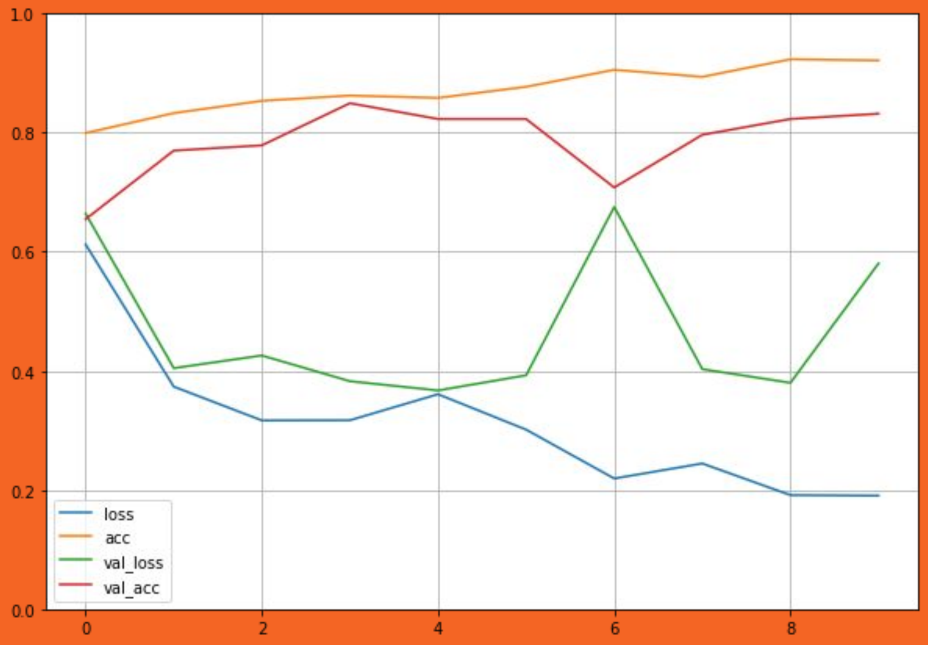
Classifier 1

Using less neurons in dense layers (128 neurons then 64 neurons with regularization and dropout layers)



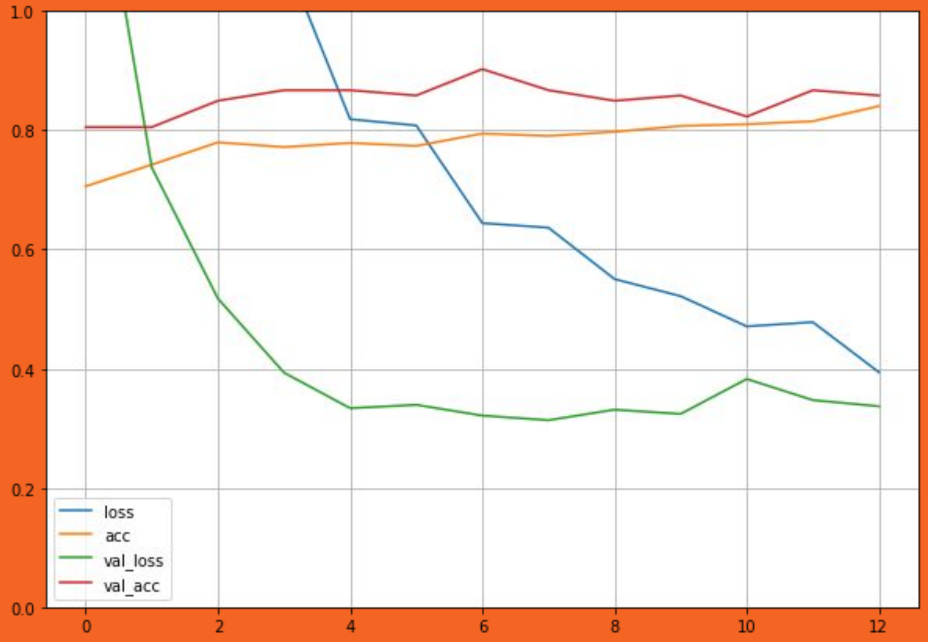
Classifier 2

Removing regularization and dropout layers and use more neurons in dense layers (1024 neurons then 512)



Classifier 3

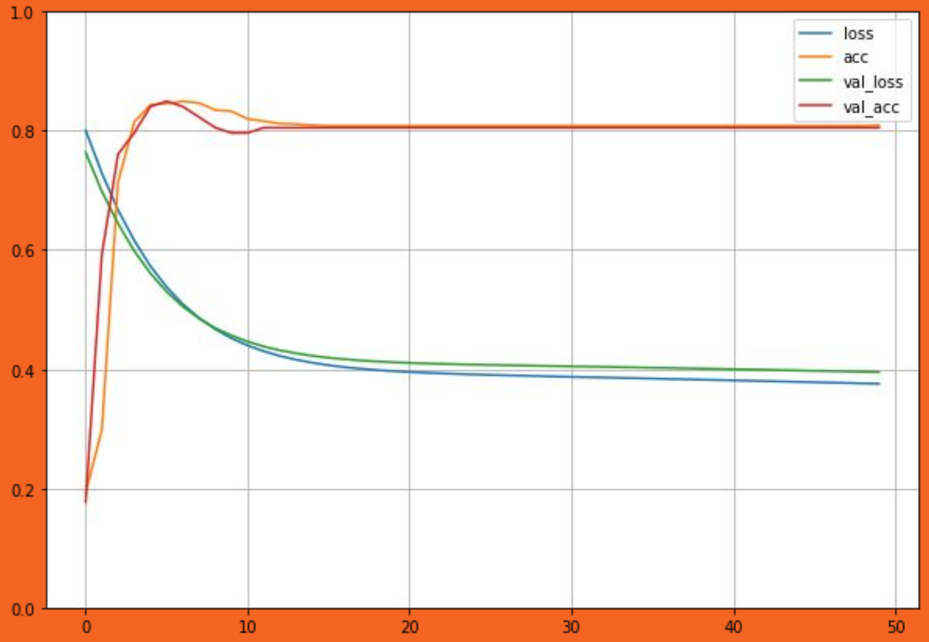
It is the final model for classifier by using more neurons in dense layers (1024 neurons then 512) and add dropout layers to overcome the overfitting



Training Regressors For VGG

Regressor 1

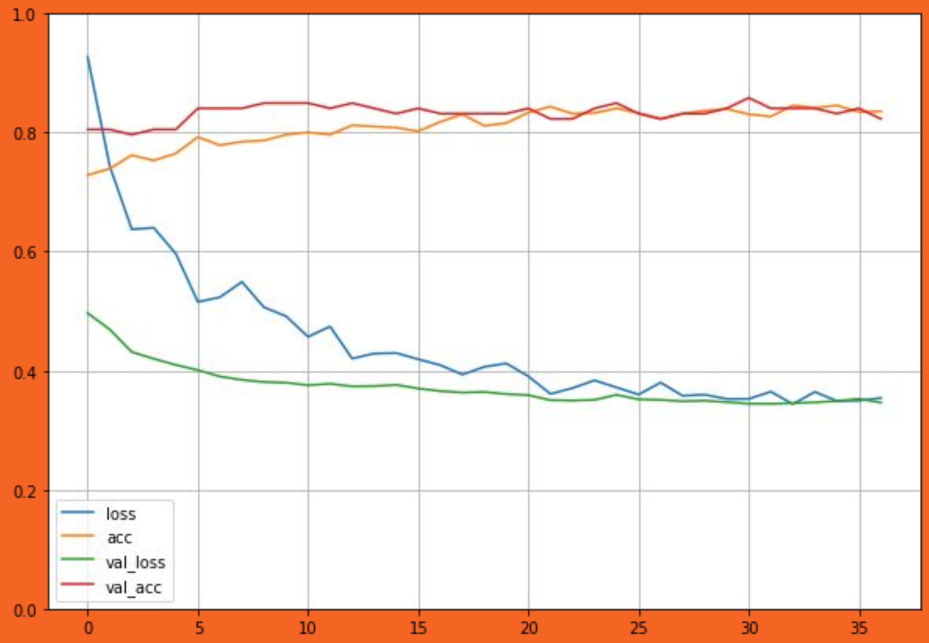
It is the final model for regressor using the adam optimizer with learning rate = 0.01



Training Classifiers For RESNet

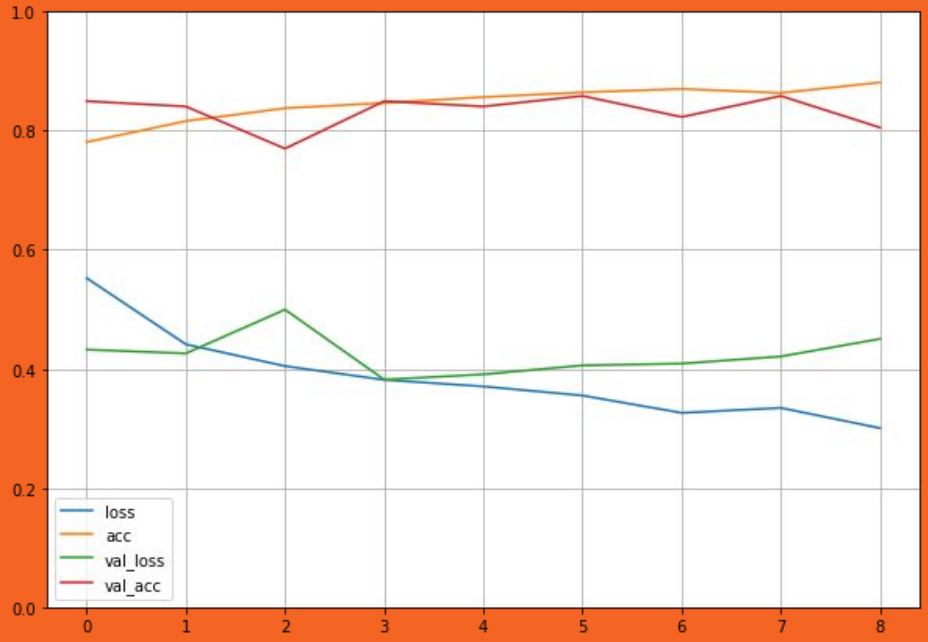
Classifier 1

Using less neurons in dense layers (512 neurons then 256 neurons with dropout layers)



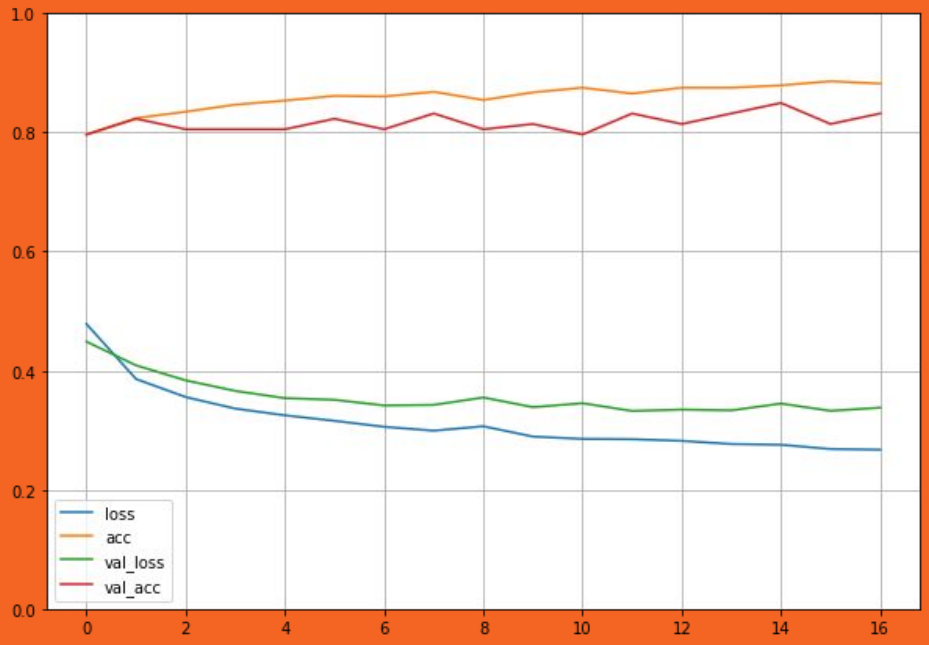
Classifier 2

Using learning rate = 10^-4 and this test is for Axial ACL classifier not Axial Abnormal like the other ones



Classifier 3

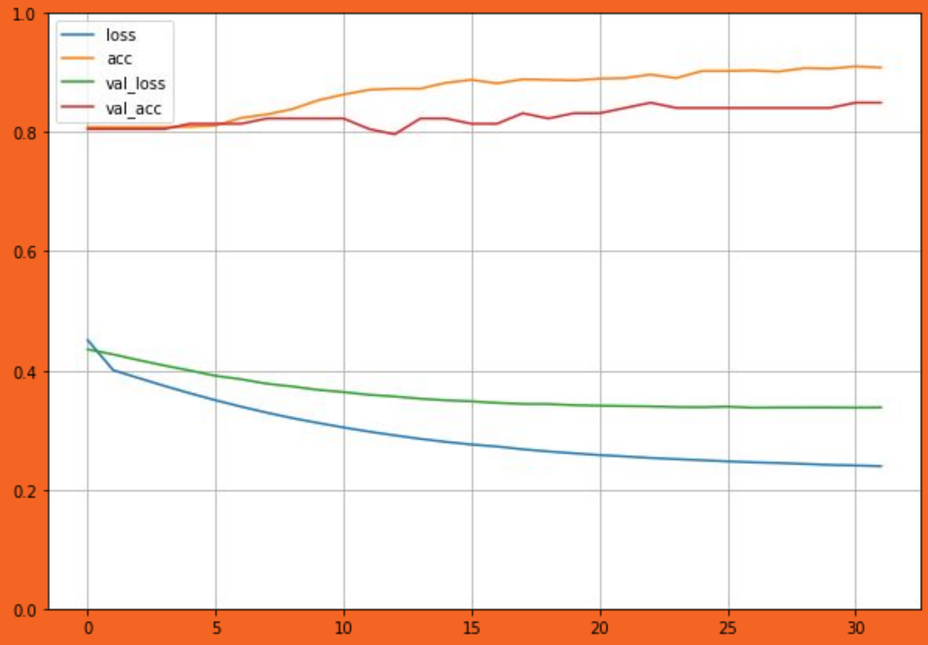
It is the final model for the classifier by using learning rate = 10^-5 for adam optimizer with (512 neurons in first dense and 256 in the next dense layer)



Training Regressors For RESNet

Regressor 1

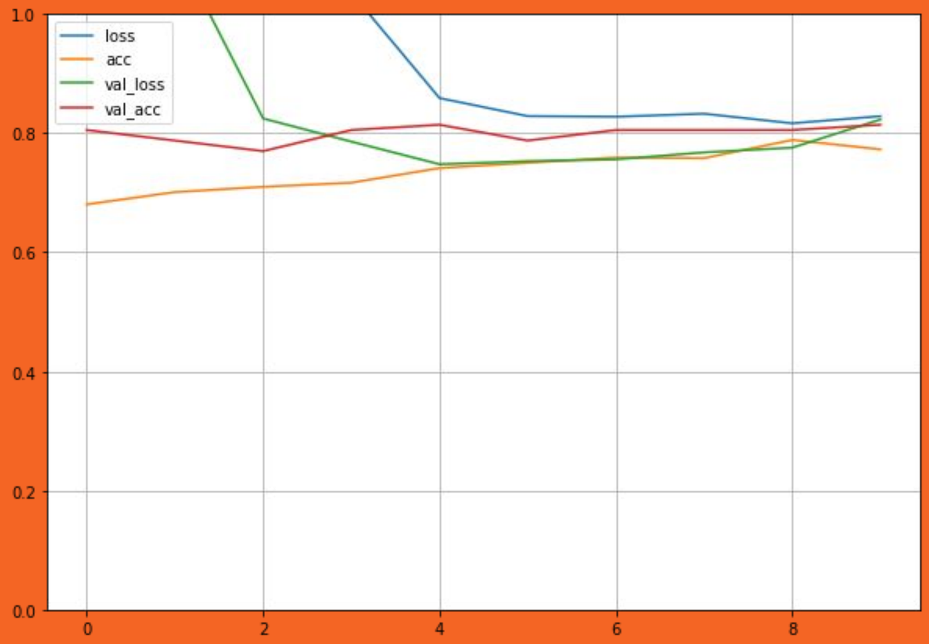
It the final model for regressor using the learning rate = 0.01 by adam optimizer



Training Classifiers For Inception V3

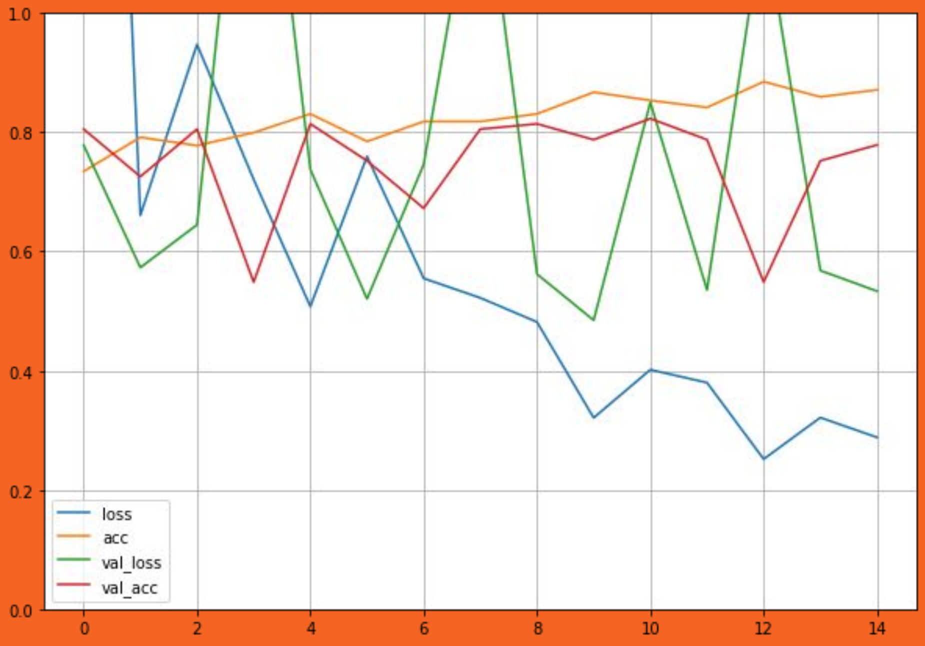
Classifier 1

Using less neurons in dense layers (128 neurons then 64 neurons as MRNet paper with regularization and dropout layers)



Classifier 2

Removing regularization and dropout layers and use more neurons in dense layers (1024 neurons then 512)



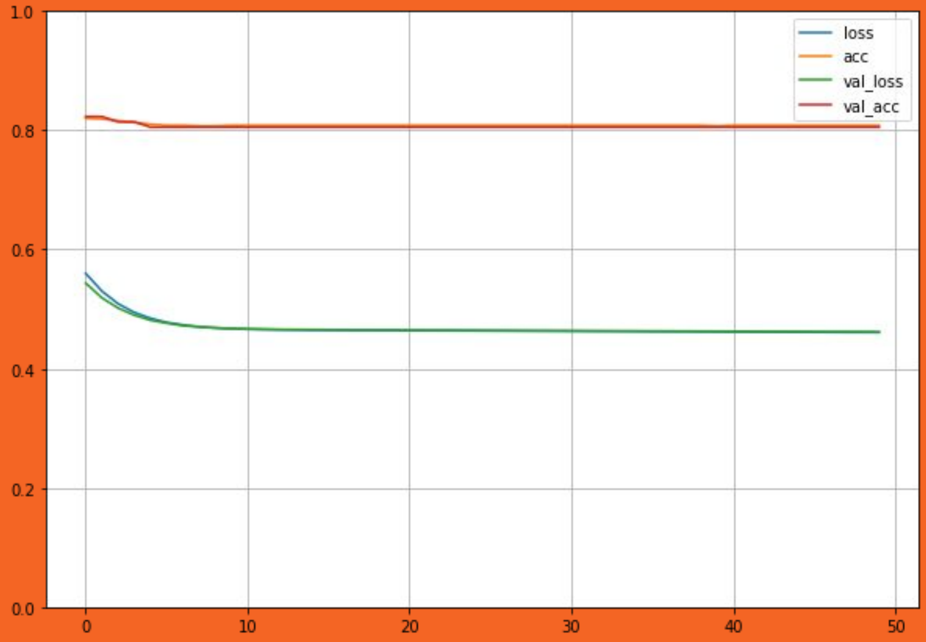
Classifier 3

It is the final model for classifier by using more neurons in dense layers (1024 neurons then 512) and add dropout layers to overcome the overfitting

Training Regressors For Inception V3

Regressor 1

It the final model for regressor using the learning rate = 0.01 by adam optimizer



Transfer Learning Statistics

Regressors

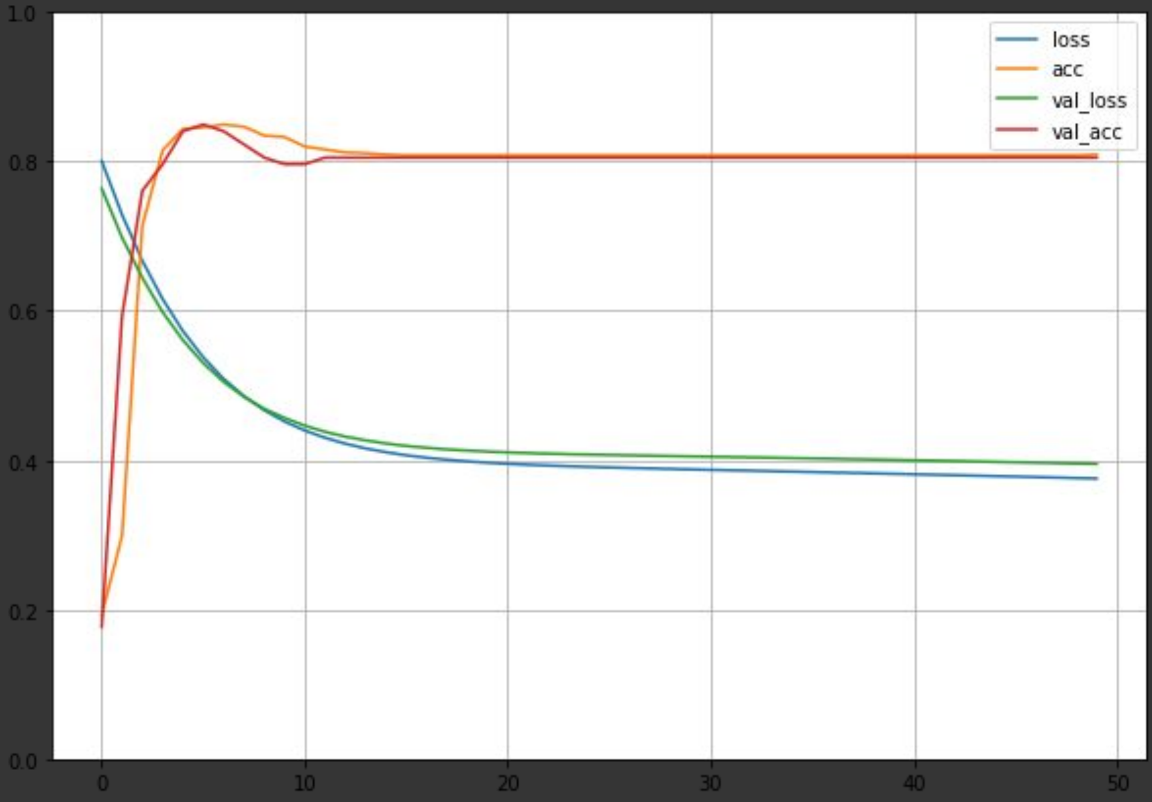
We judge the model by two methods

- ➔ Loss
- ➔ Accuracy

VGG

- **Abnormal**

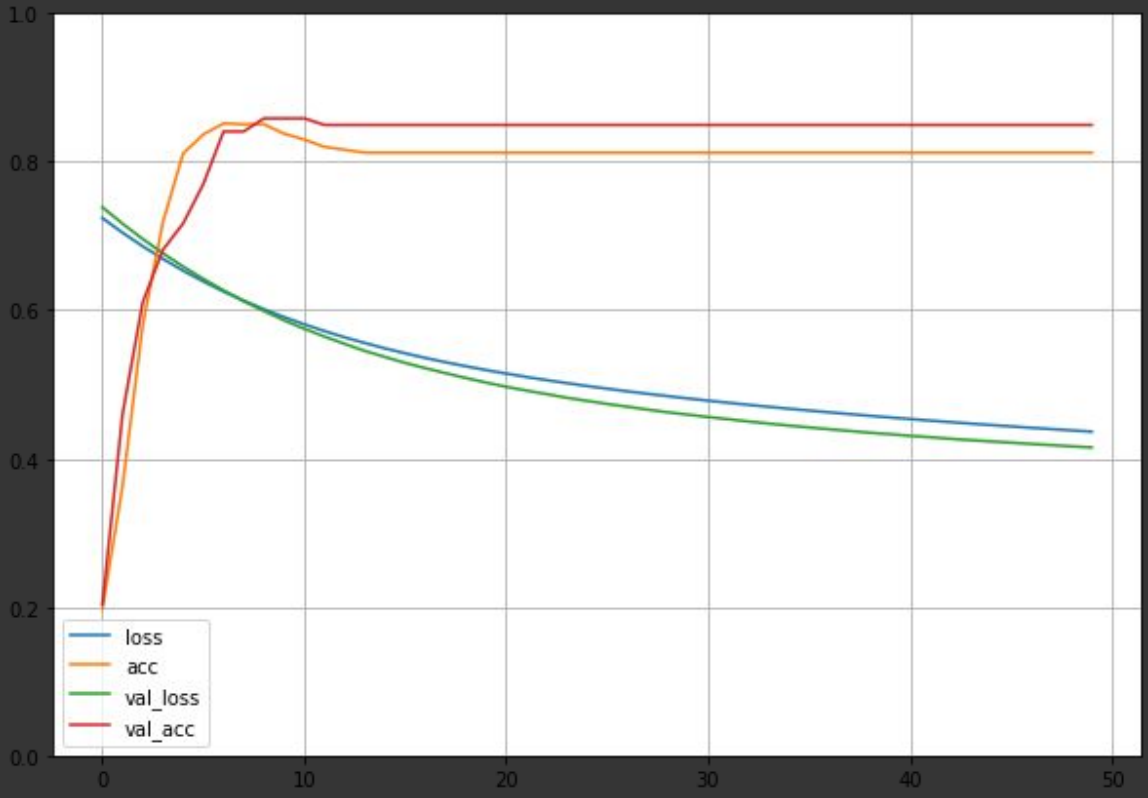
Loss :   0.5338
Accuracy : 0.8083



VGG

- **ACL**

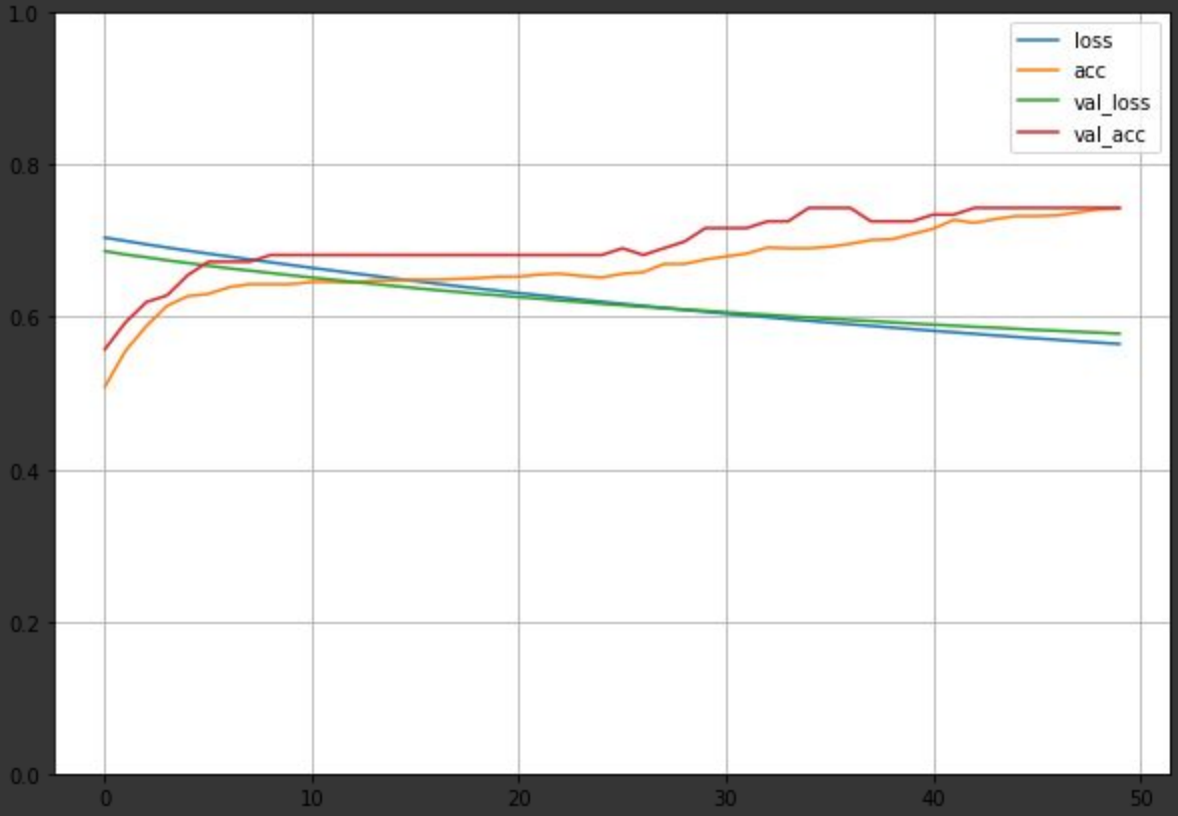
  Loss : 0.659
  Accuracy : 0.625



VGG

- **Meniscal**

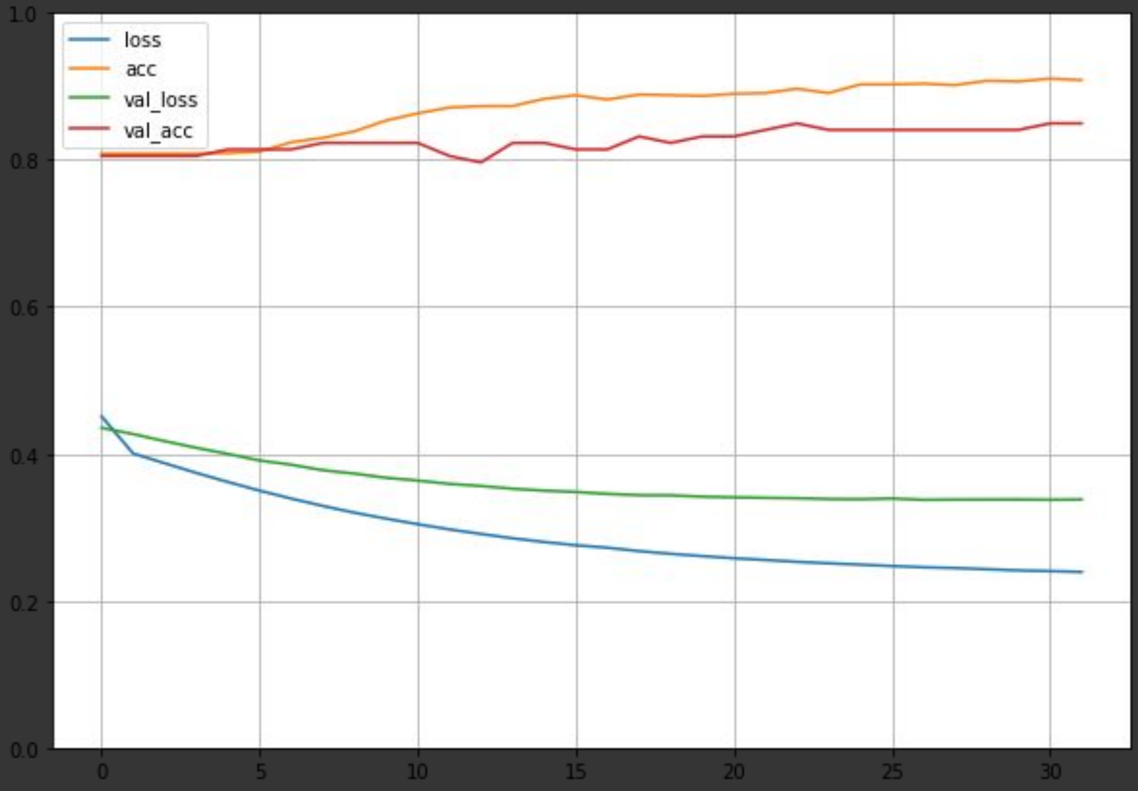
Loss :   0.6374
Accuracy : 0.5917



RESNet

- **Abnormal**

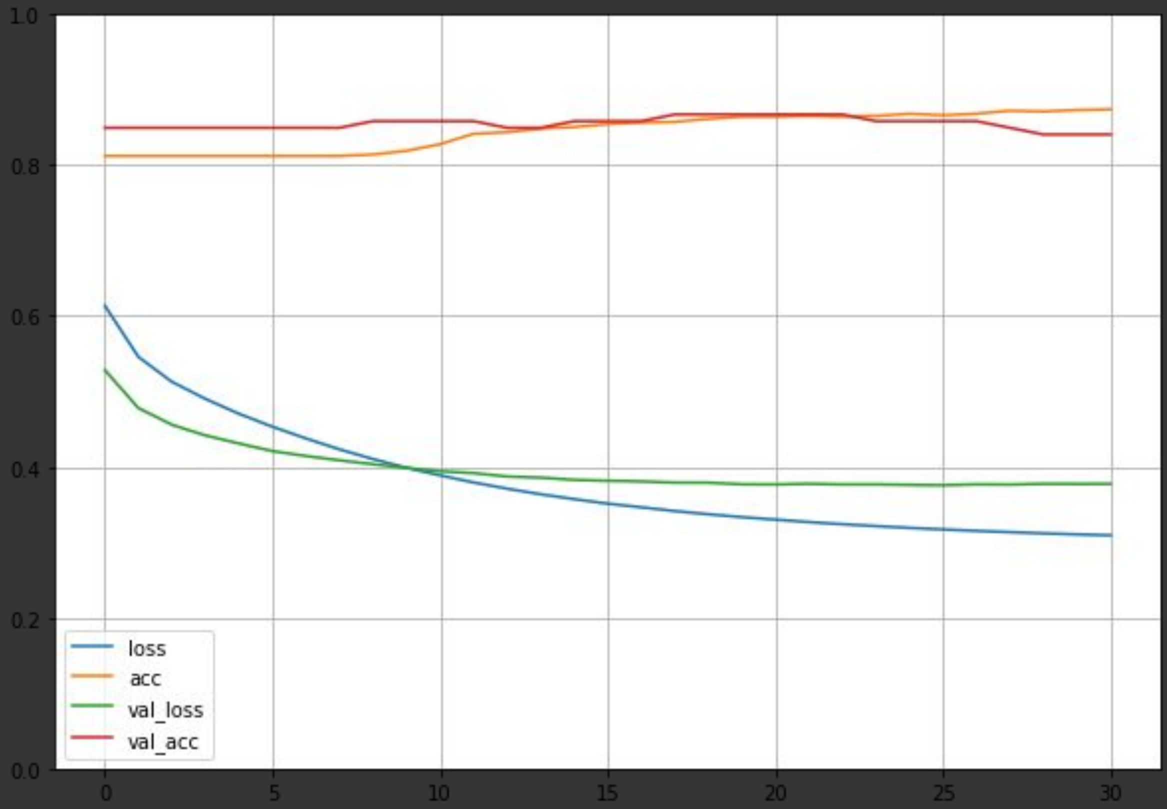
  Loss : 0.3940
  Accuracy : 0.8333



RESNet

- **ACL**

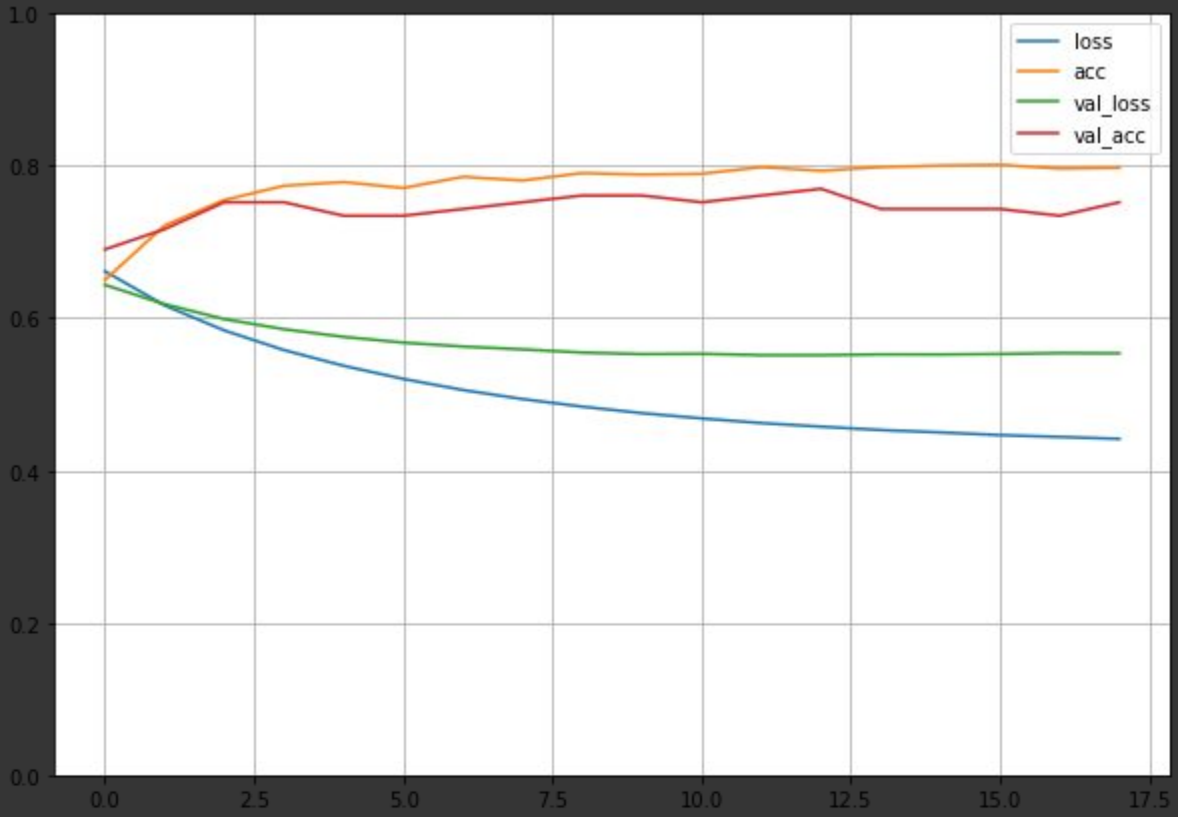
  Loss : 0.6274
  Accuracy : 0.6583



RESNet

- **Meniscal**

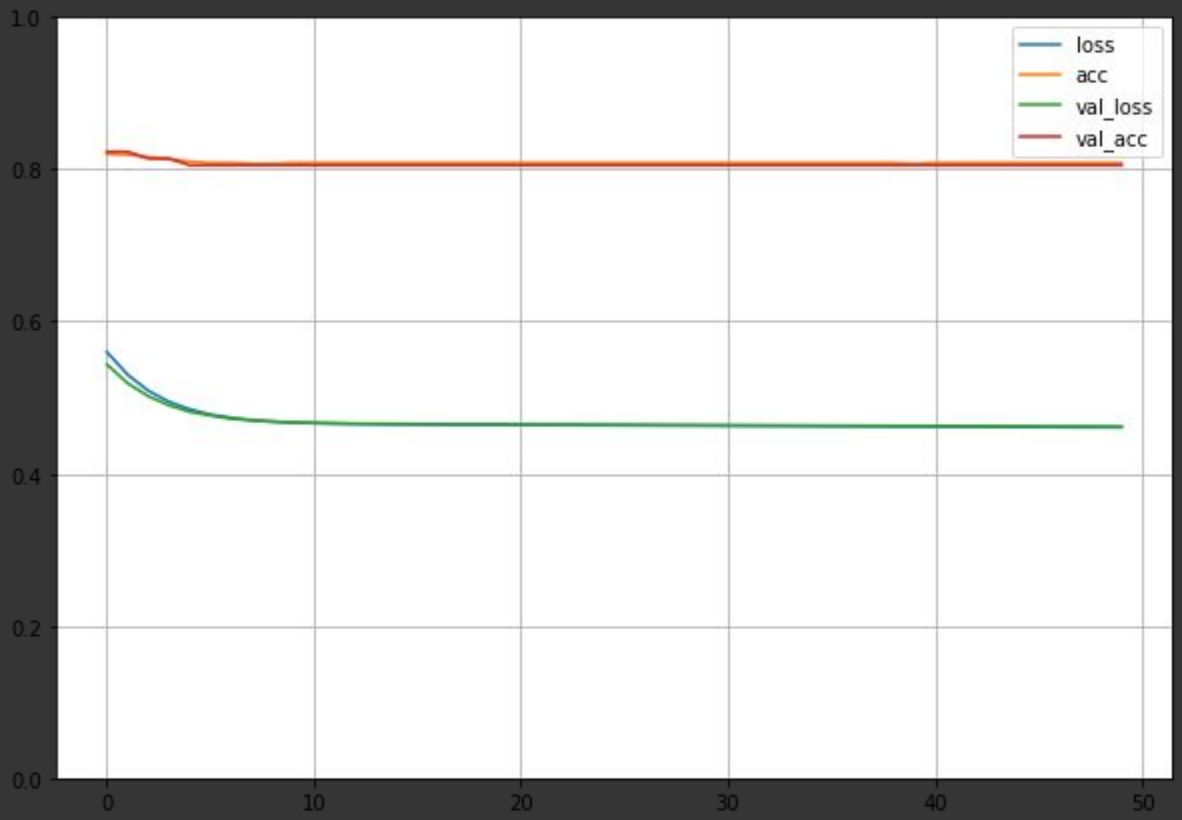
  Loss : 0.5947
  Accuracy : 0.7083



Inception V3

- **Abnormal**

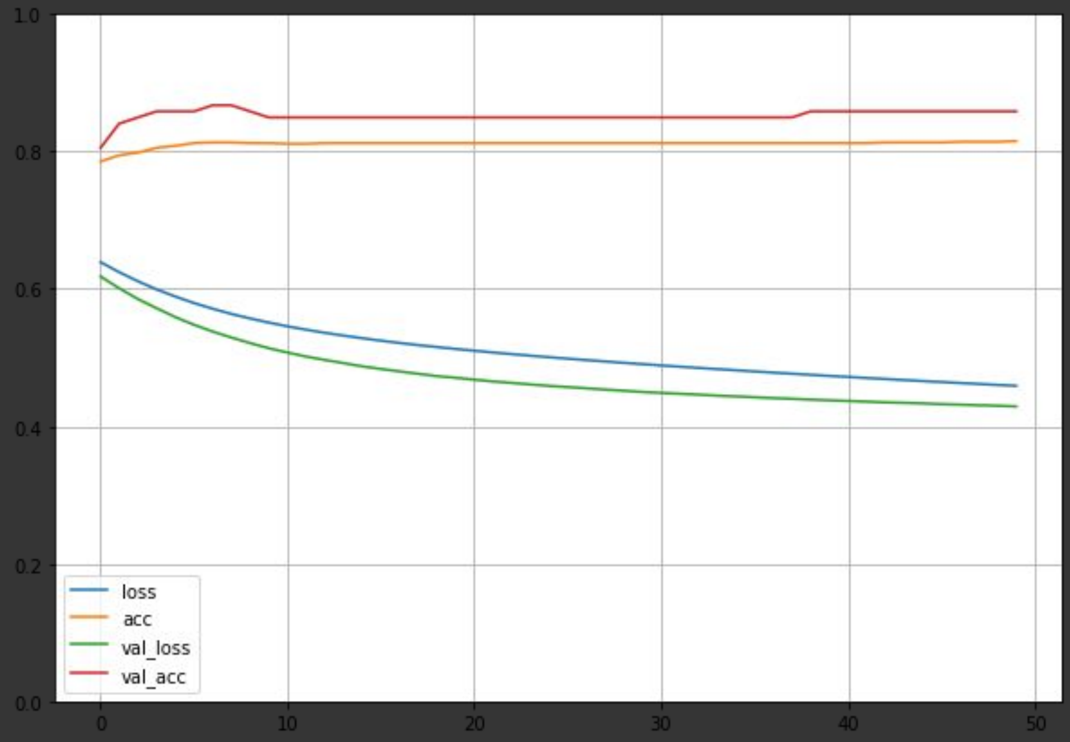
Loss : 0.5504
Accuracy : 0.7917



Inception V3

- **ACL**

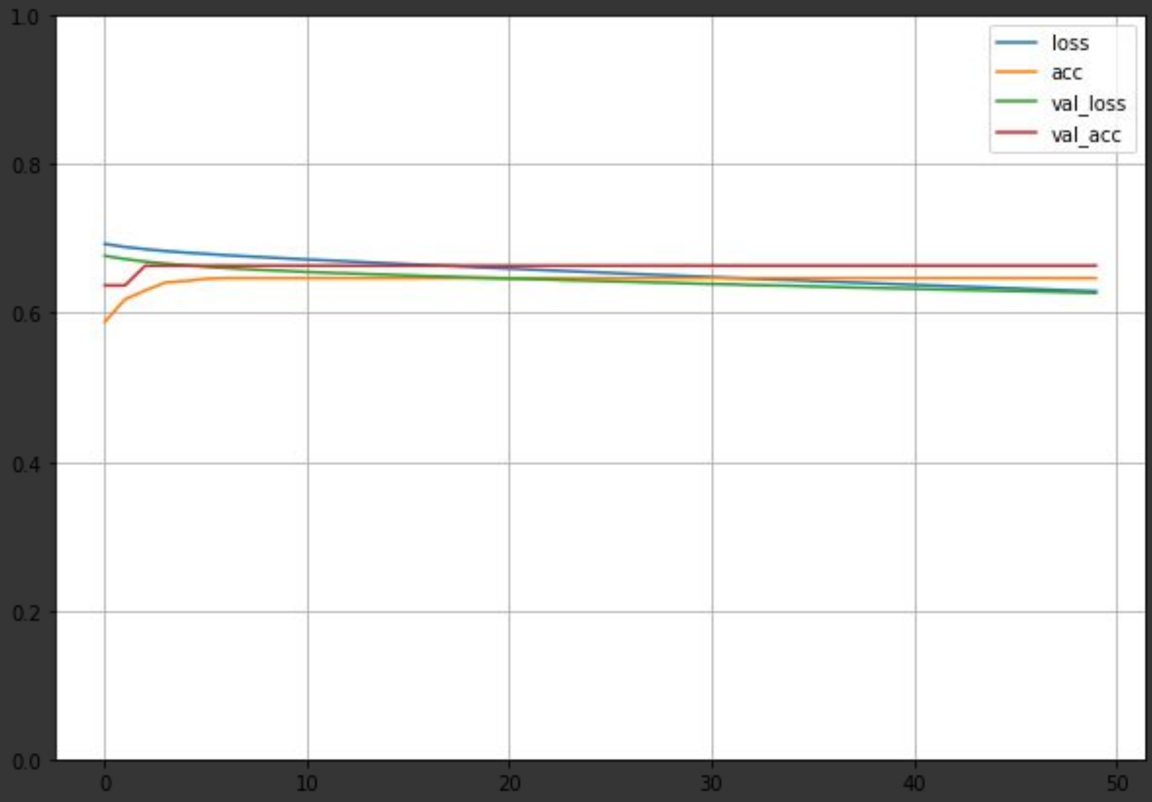
  Loss : 0.7141
  Accuracy : 0.5417



Inception V3

- **Meniscal**

  Loss : 0.7000
  Accuracy : 0.5667



Transfer learning models summary

	Anomaly	Accuracy	Loss
VGG	Abnormal	0.8083	0.5338
	Acl	0.6250	0.6590
	Meniscal	0.5917	0.6374
RESNet	Abnormal	0.8333	0.3940
	Acl	0.6583	0.6274
	Meniscal	0.7083	0.5947
Inception V3	Abnormal	0.7917	0.5504
	Acl	0.5417	0.7141
	Meniscal	0.5667	0.7000

Contribution

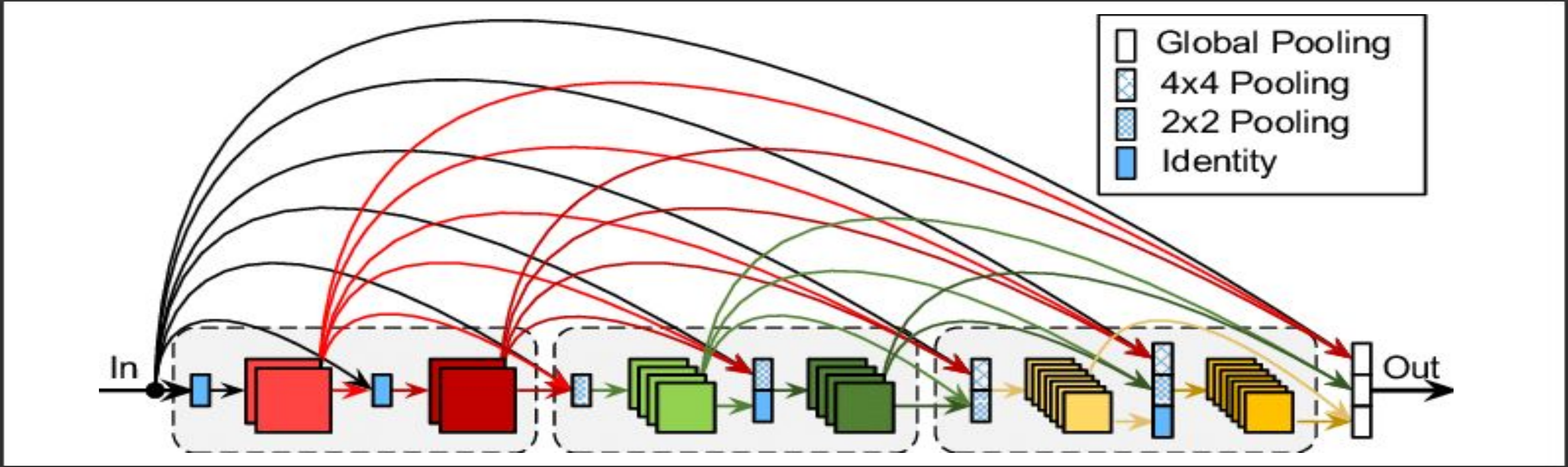
The first paper

“ Deep Learning for Musculoskeletal Image Analysis” suggests two approaches

- ➔ Train DenseNet model as feature extractor By transfer learning
- ➔ Use large amount of imaging data (sooo... challenging !!!)

DenseNet

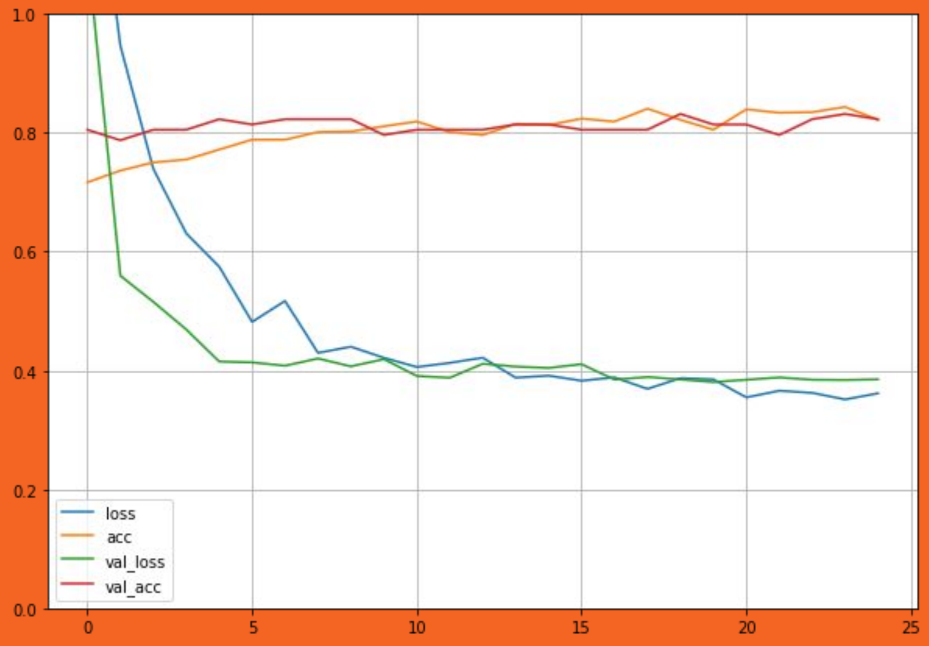
- It enhances the RESNet model by connecting output of a layer to all subsequent layers. This architecture helps in improving in the flow of information and gradient in the network.



Training Classifiers For DenseNet

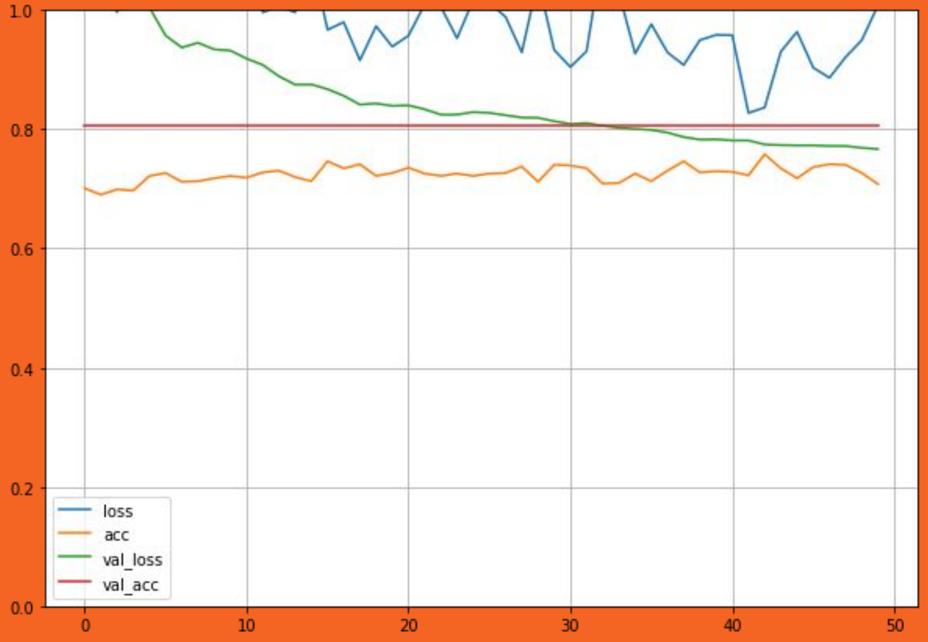
Classifier 1

Using less neurons in dense layers (512 neurons then 256 neurons with dropout layers)



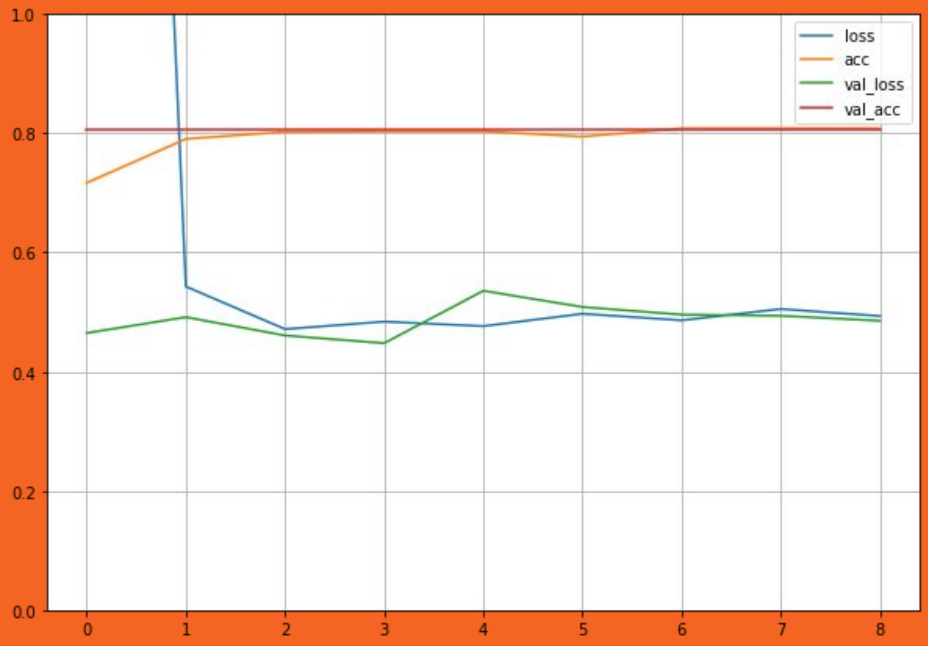
Classifier 2

Using learning rate = 10^-4 and decay rate = 0.1



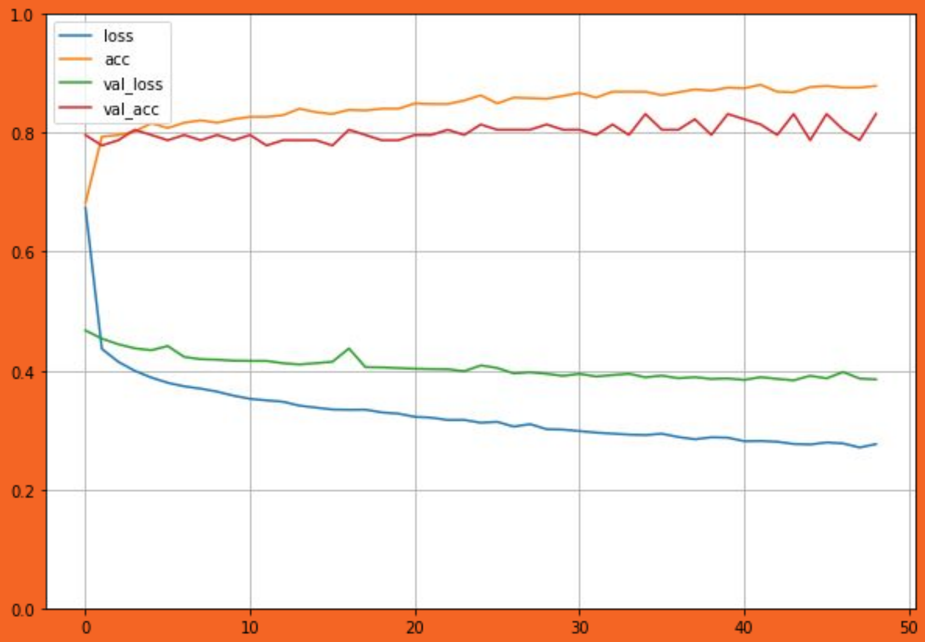
Classifier 3

Increase learning rate to 0.01



Classifier 4

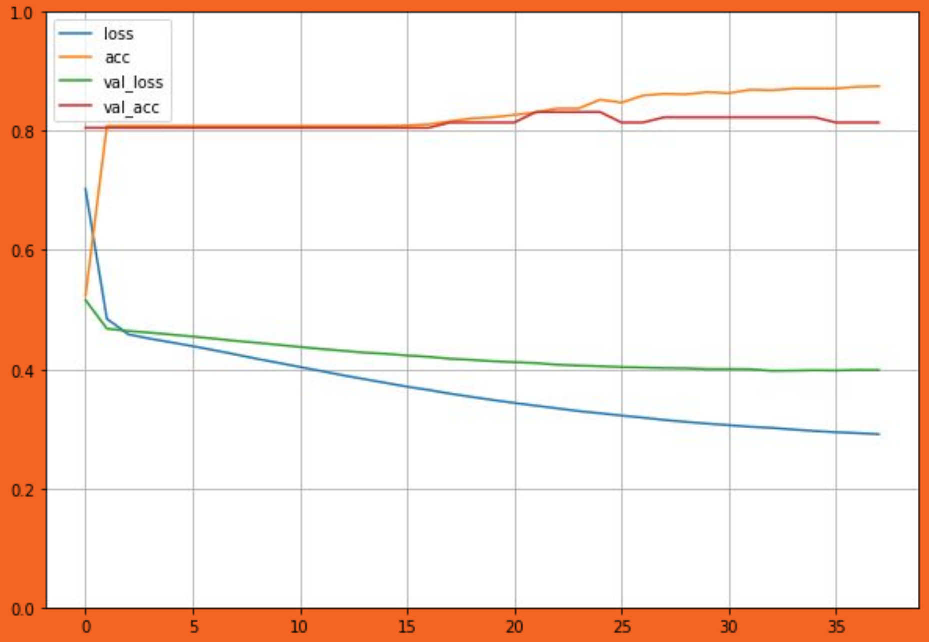
It is the final model for the classifier by using learning rate = 10^-5 for adam optimizer with (512 neurons in first dense and 256 in the next dense layer)



Training Regressors For DenseNet

Regressor 1

It the final model for regressor using the learning rate = 0.01 by adam optimizer



DenseNet Model Statistics

Regressors

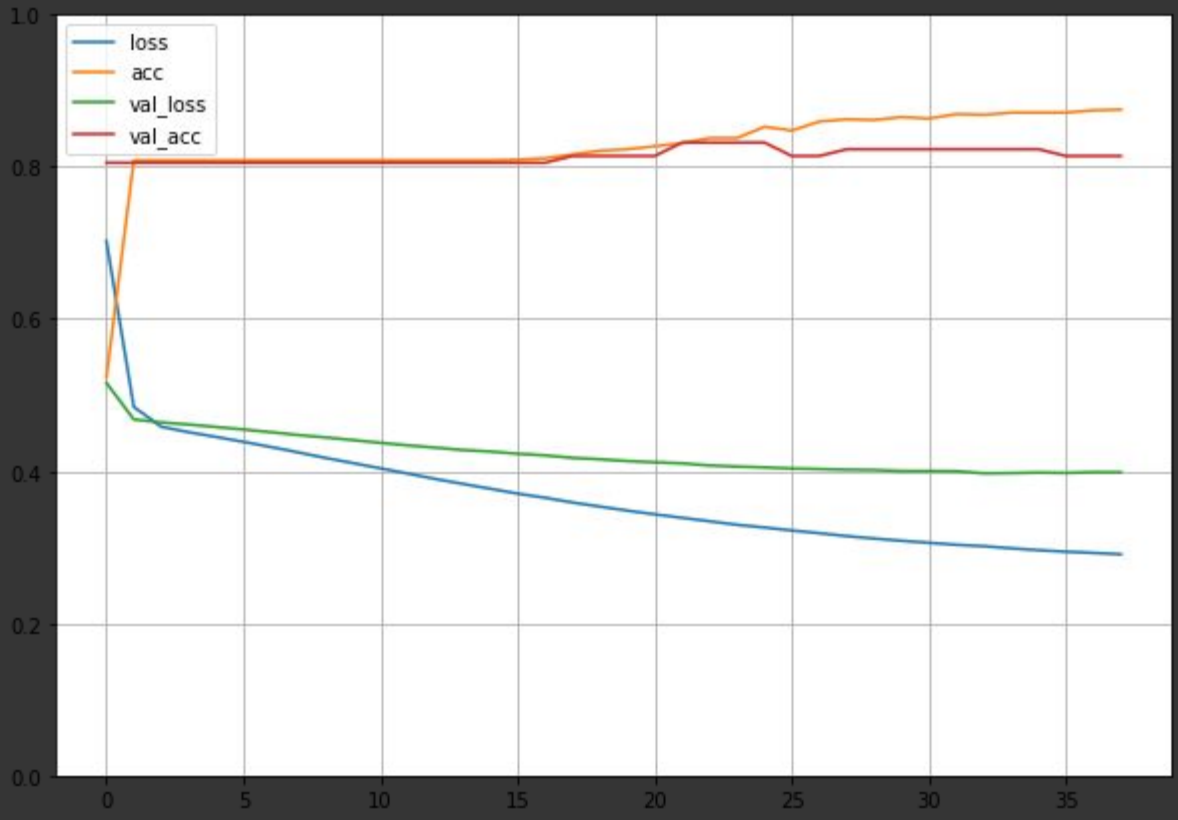
We judge the model by two methods

- ➔ Loss
- ➔ Accuracy

DenseNet

- **Abnormal**

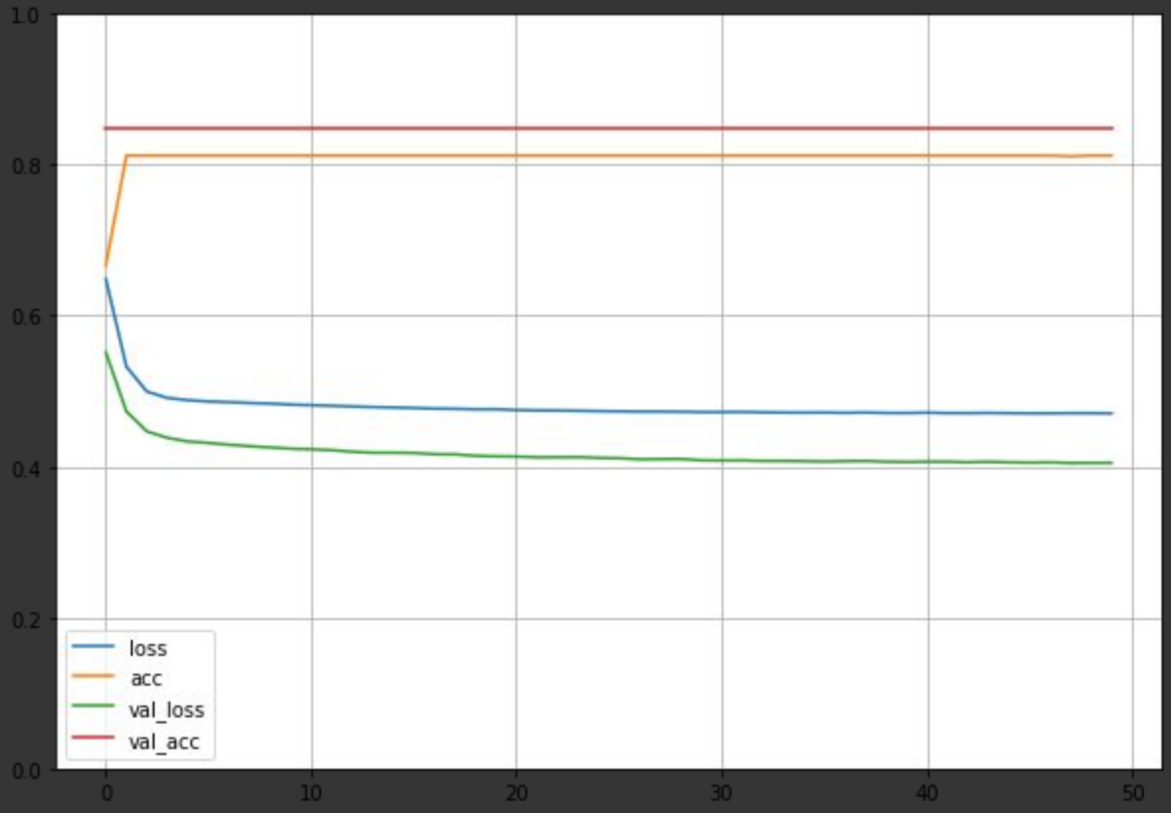
  Loss : 0.4418
  Accuracy : 0.8167



DenseNet

- **ACL**

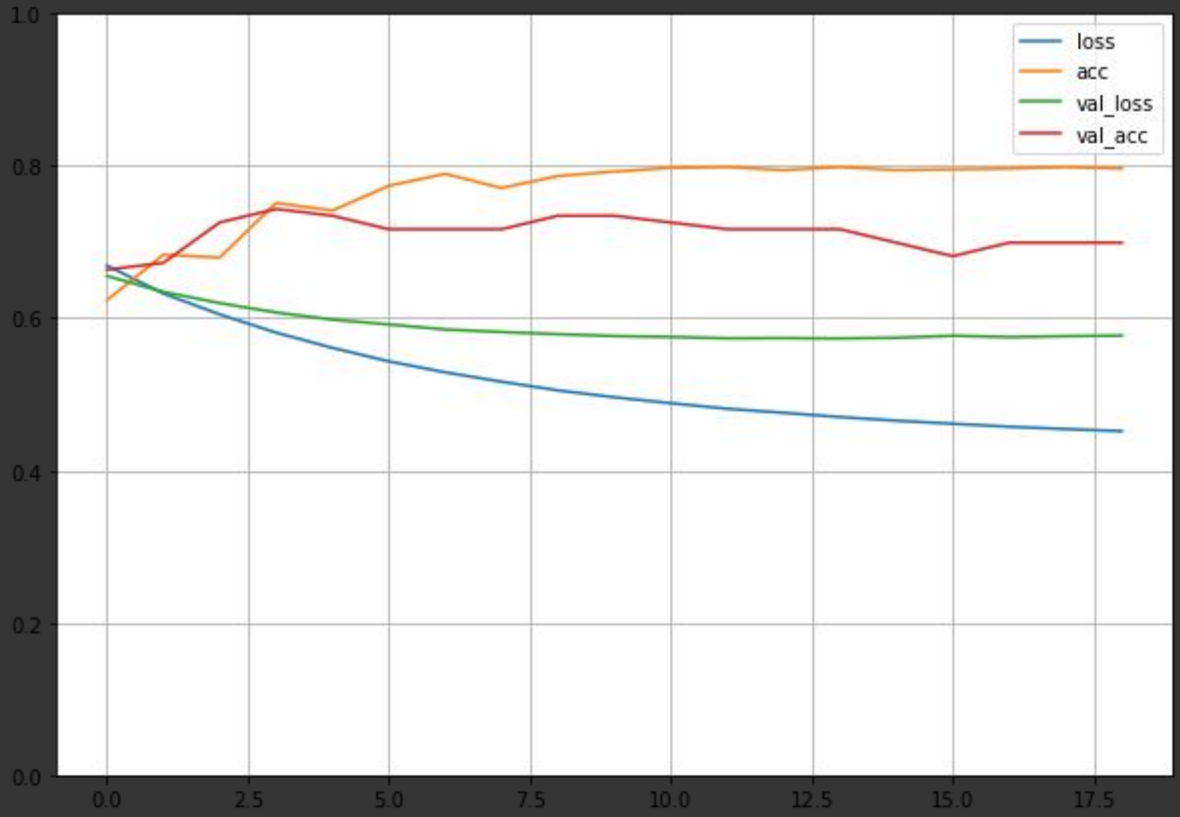
  Loss : 0.6972
  Accuracy : 0.5500



DenseNet

- **Meniscal**

  Loss : 0.6453
  Accuracy : 0.6417



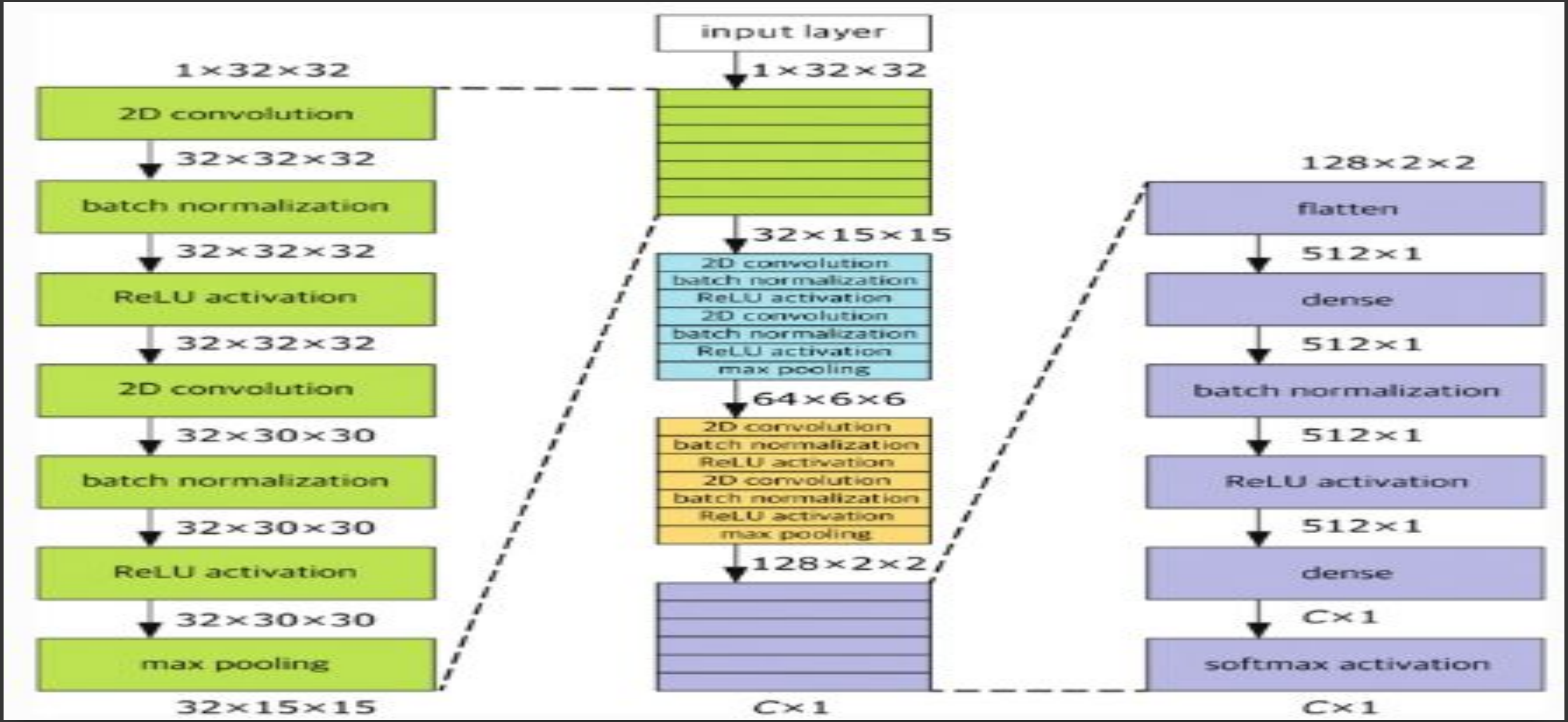
DenseNet model summary

	Accuracy	Loss
Abnormal	0.8167	0.4418
ACL	0.5500	0.6972
Meniscal	0.6417	0.6453

The second paper

Using Deep Learning Algorithms to Automatically Identify the Brain MRI Contrast: Implications for Managing Large Databases

Contrast

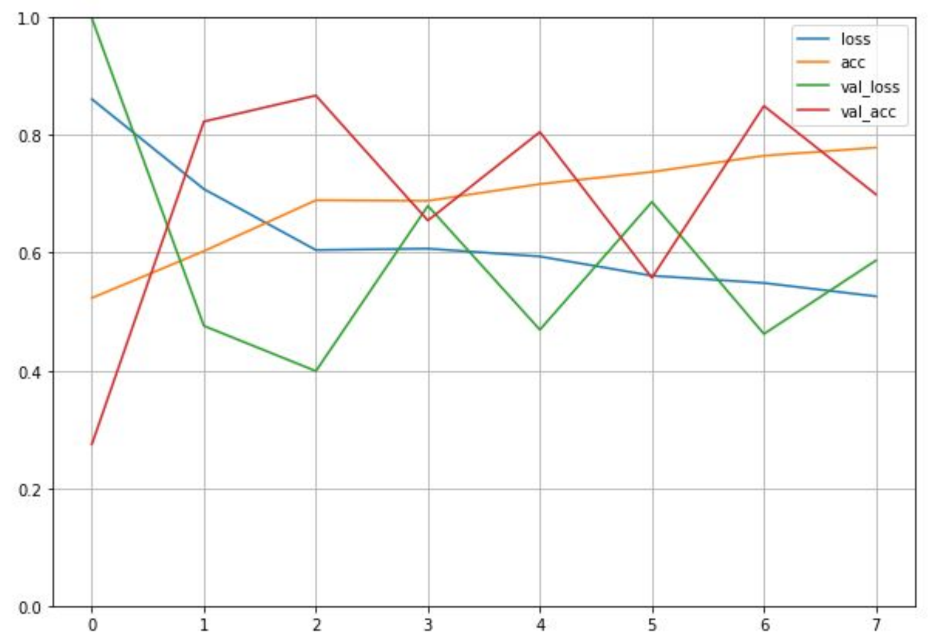


input layer
1×32×32
1×32×32
2D convolution
32×32×32
batch normalization
32×32×32
ReLU activation
32×32×32
2D convolution
32×30×30
batch normalization
32×30×30
ReLU activation
32×30×30
max pooling
32×15×15
32×15×15
2D convolution
batch normalization
ReLU activation
2D convolution
batch normalization
ReLU activation
max pooling
64×6×6
2D convolution
batch normalization
ReLU activation
2D convolution
batch normalization
ReLU activation
max pooling
128×2×2
C×1
128×2×2
flatten
512×1
dense
512×1
batch normalization
512×1
ReLU activation
512×1
dense
C×1
softmax activation
C×1

Training Extractors For Contrast

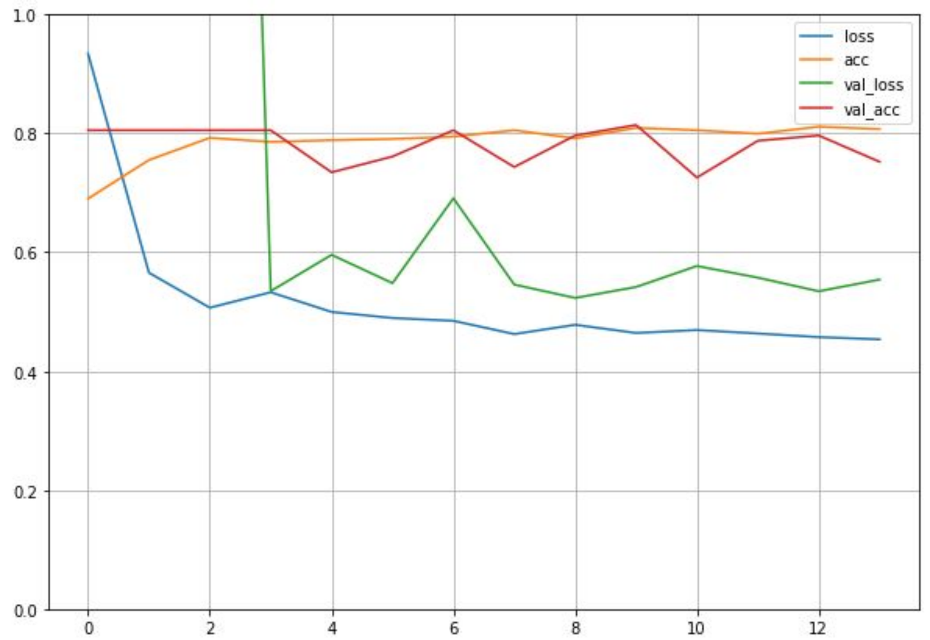
Extractor 1

Without using dropout layer and using adam optimizer there was overfitting problem



Extractor 2

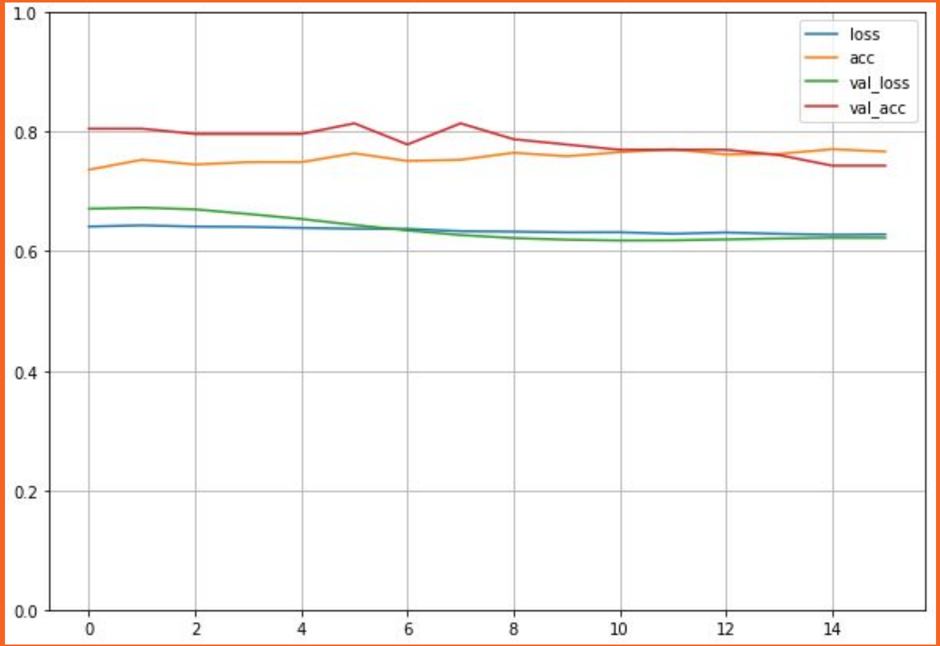
Using dropout layer reduces the overfitting problem also optimizer changed to nadam



Training Classifiers For Contrast

Classifier

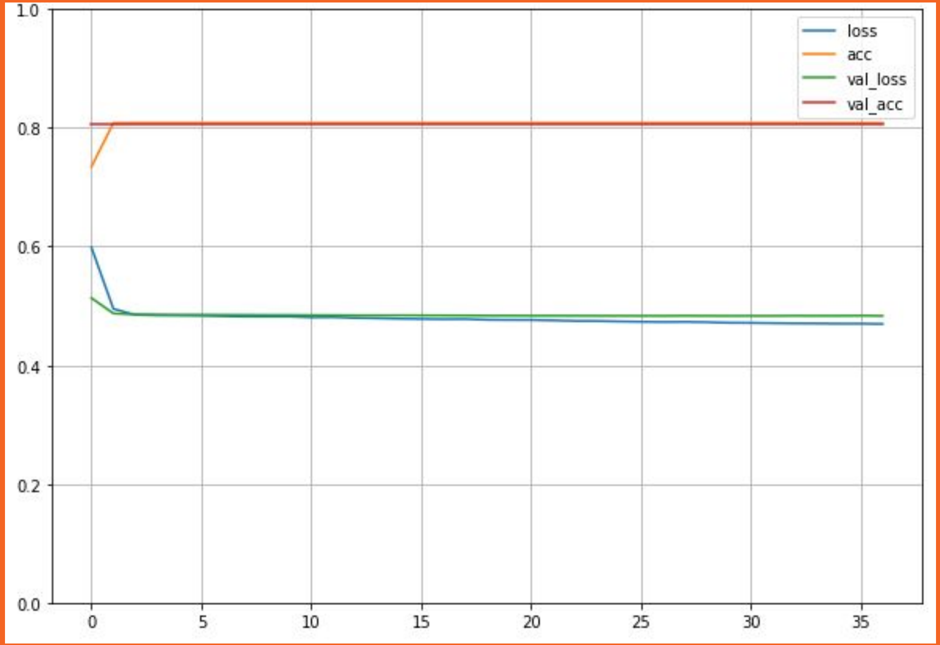
At the first trial we used the same optimizer used in extractor but there was overfitting so we changed the optimizer to adam with learning rate 10^-7 and it was much better



Training Regressors For Contrast

Regressor

At first we used the adam optimizer with learning rate 10^-7 but result was not good so we tried to increase the learning rate and make it 10^-2 and it was much better



Contrast Model Statistics

Regressors

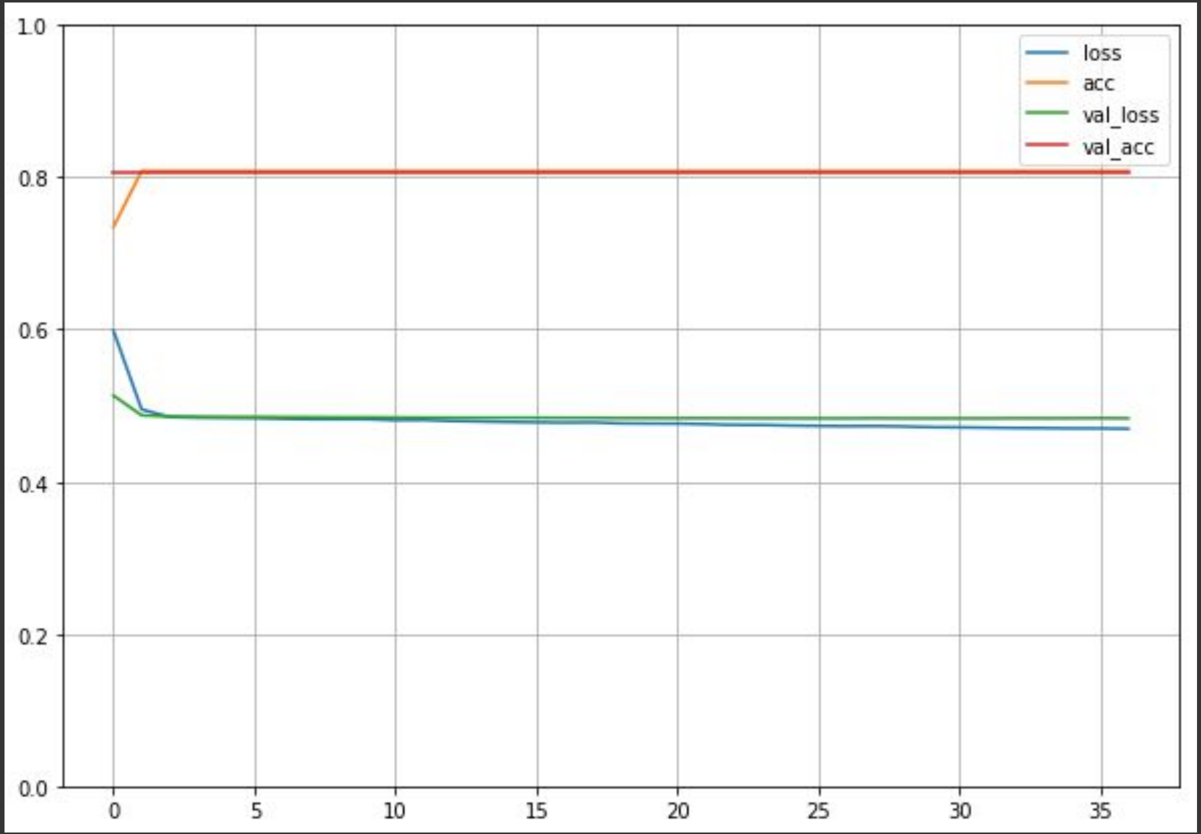
We judge the model by two methods

- ➔ Loss
- ➔ Accuracy

Contrast

- **Abnormal**

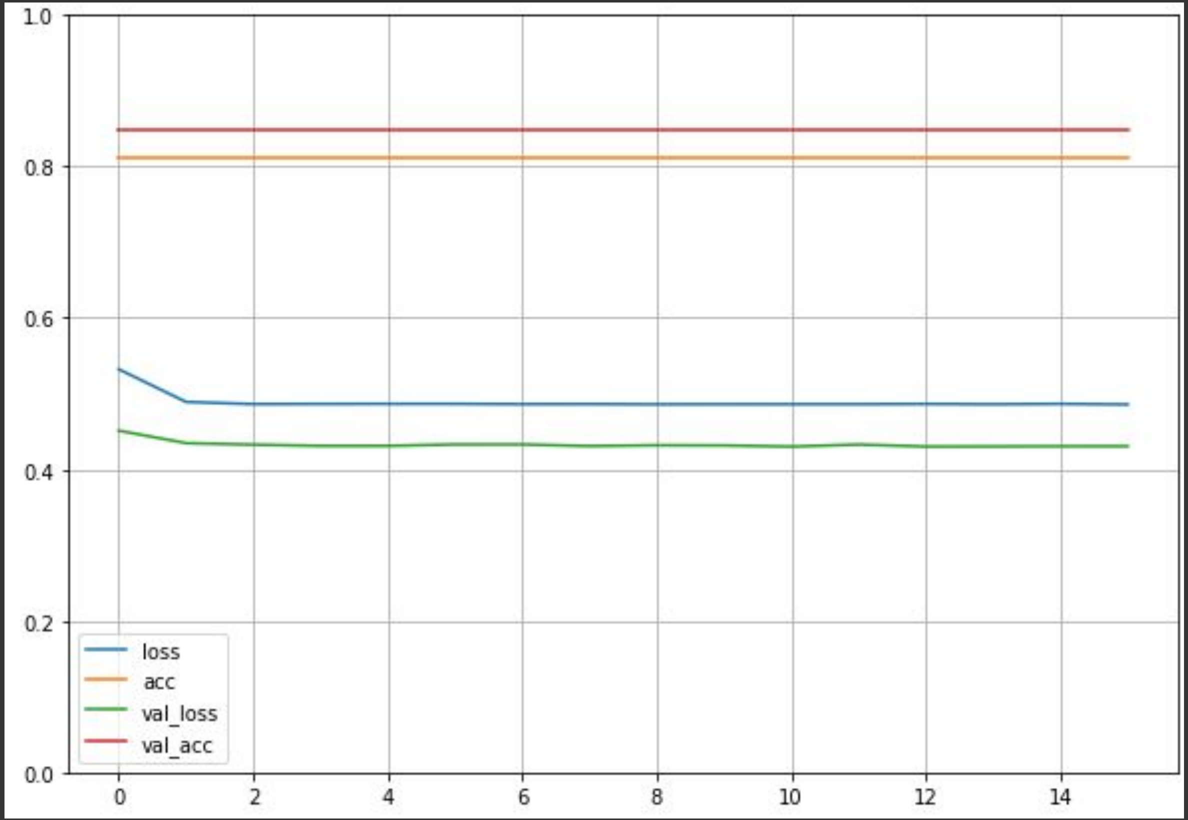
  Loss : 0.5096
  Accuracy : 0.7917



Contrast

- **ACL**

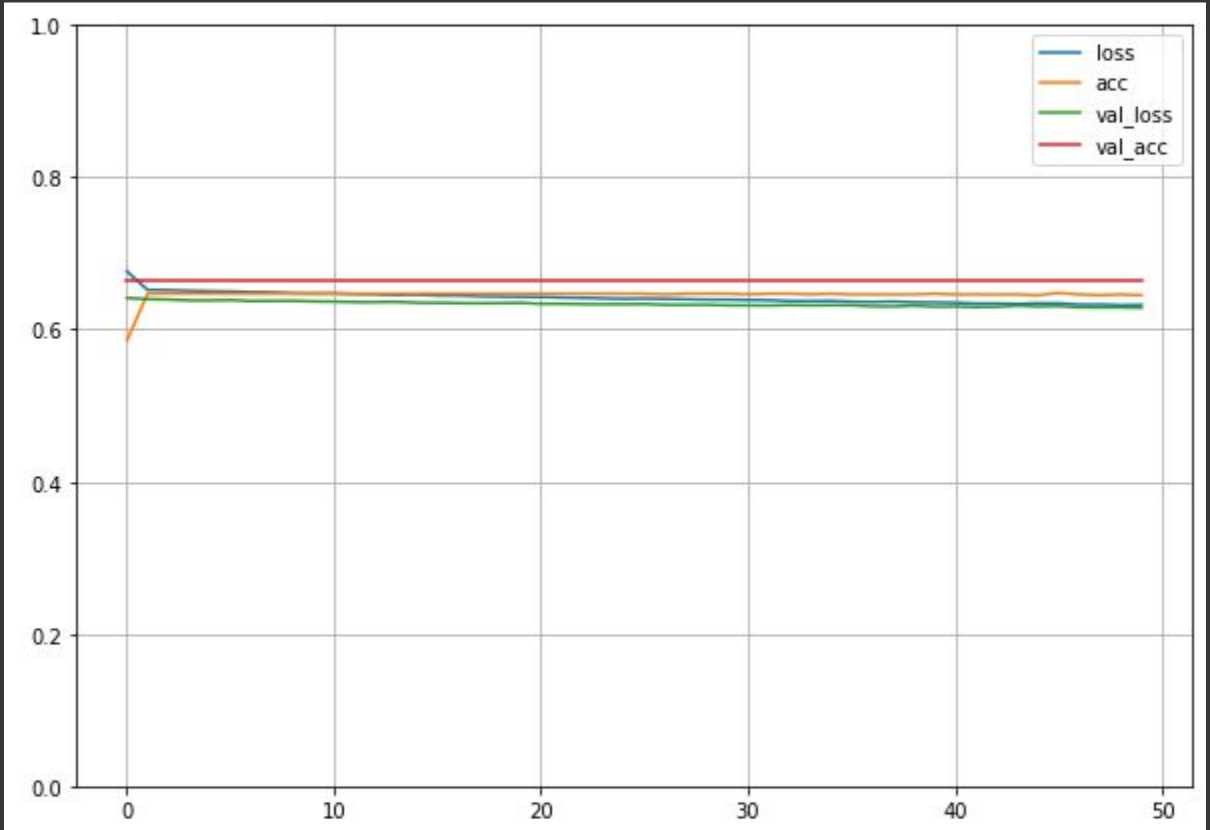
Loss : 0.8062
Accuracy : 0.5500



Contrast

- **Meniscal**

  Loss : 0.6925
  Accuracy : 0.5667



Contrast model summary

	Accuracy	Loss
Abnormal	0.7917	0.5096
ACL	0.5500	0.8062
Meniscal	0.5667	0.6925