Alexandria University

Faculty of Engineering

Computer and System Engineering Department

CSE312: Operations Research

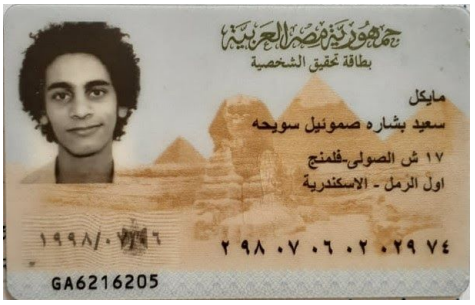
Students:

- Arsany Atef Abdo (10)
- Kirellos Malak Habib (35)
- Michael Said Beshara (38)
- Nada Salama (58)
- Yomna Gamal (63)

Professors:

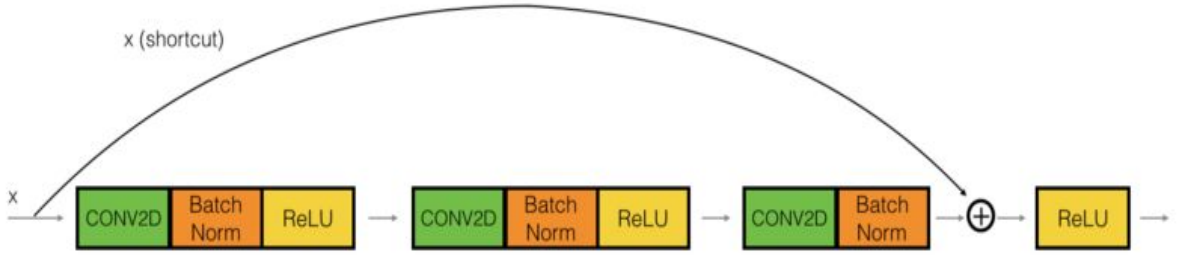
- Dr. Marwan Torky
- Dr. Ayman Khalaf-Allah

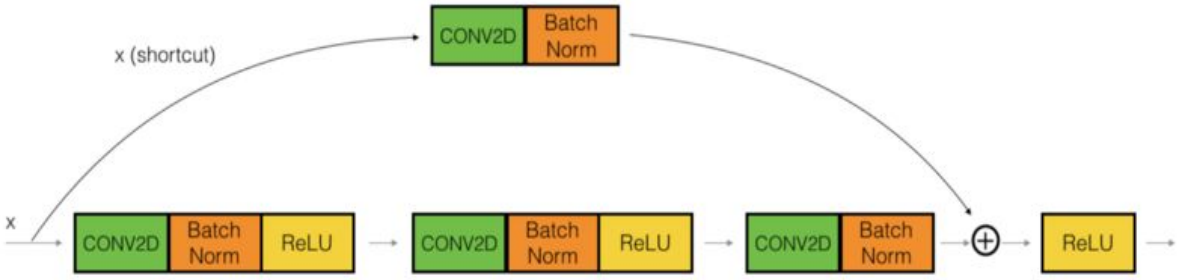
National IDs:

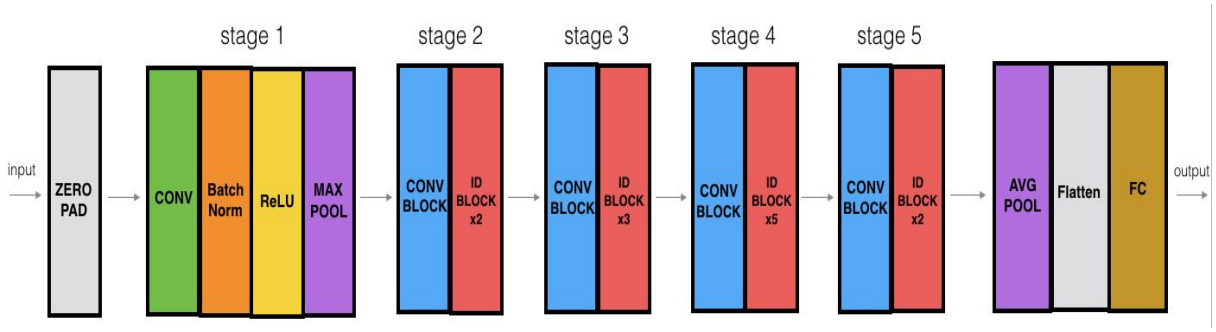


Description:

- **Models**
  - **ResNet 50**



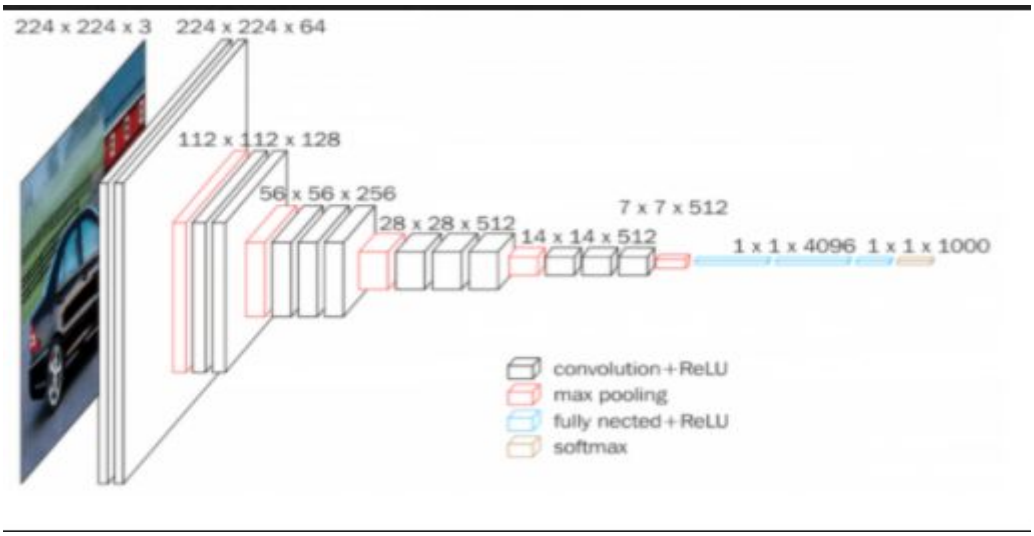




We build ResNet-50 model like this blocks in these images where we create class called ResNet-50 with three functions (init, call, getmodel)

- **In init, we define parameters of class like shape of input parameter,number of classes and include top or not.**
- **In call, we build all layers and residual blocks for the model.**
- **There are two types of residual blocks : Identity block and Convolutional block.**
- **In getmodel, we use it to get a model which in it takes input then calls a function call and passes input to it.**

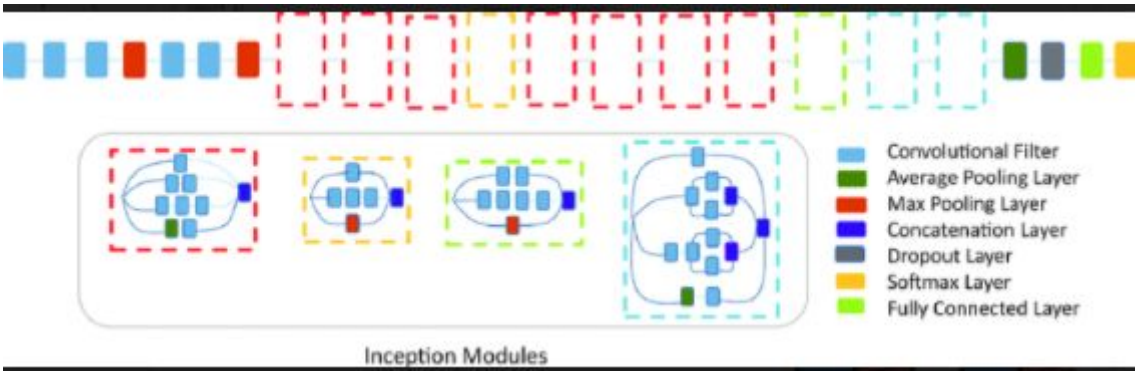
- **Vgg 16**



We build VGG model like this layer in this image where we create class called vgg16 with three functions (init, call, getmodel)

- **In init, we define parameters of class like shape of input parameter and include top or not.**
- **In call, we add all layers for input like conv, max pooling,..**
- **In getmodel, we use it to get a model which in it takes input then calls a function call and passes input to it.**

- ***Inception V3***



We build Inception model like this layer in this image where we create class called InceptionV3 with three functions (init, call, getmodel)

- **In init, like as vgg mode**
- **In call, we add all layers but at first we create 3 classes of blocks like following And then call them into a function call**

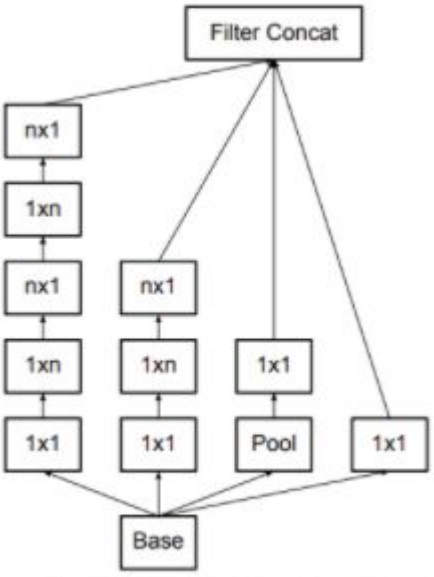


Figure 6. Inception modules after the factorization of the $n \times n$ convolutions. In our proposed architecture, we chose $n = 7$ for the 17×17 grid. (The filter sizes are picked using principle 3)

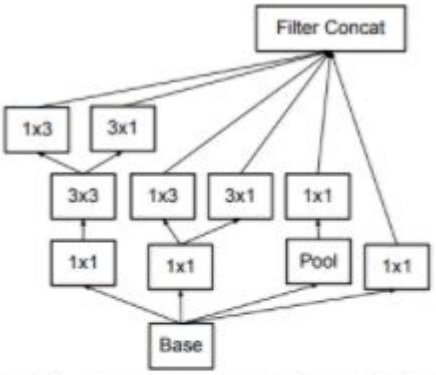


Figure 7. Inception modules with expanded the filter bank outputs. This architecture is used on the coarsest (8×8) grids to promote high dimensional representations, as suggested by principle 2 of Section 2. We are using this solution only on the coarsest grid, since that is the place where producing high dimensional sparse representation is the most critical as the ratio of local processing (by 1×1 convolutions) is increased compared to the spatial aggregation.

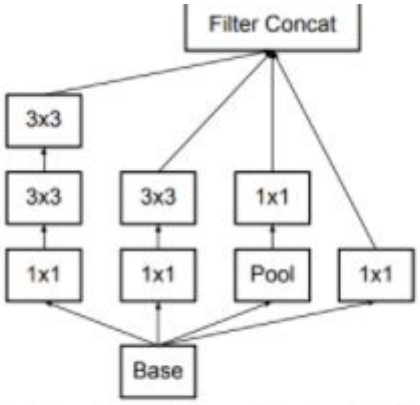


Figure 5. Inception modules where each 5×5 convolution is replaced by two 3×3 convolution, as suggested by principle 3 of Section 2.

The name of these classes are: block_1 , … , block_5.

Where each block class contain two function:

- **init : where defines our parameters which are filters of conv layers**
- **Call : where do the branches like images then at end do concatenate and then return the output (the output of concatenate)**

- **In getmodel, like as vgg model**

- # Dataset preparation
    1. We download the all dataset and compress the images of each plane in one numpy file instead of many files to reduce the time of the loading.
    2. We implement a data generator class which inherits keras.utils.Sequence to generate the batched dataset.
    3. We normalize the images during generation as :
        - Cropping the image to size (224, 224)
        - Standardize the image
        - Normalize the image using (MEAN = 58.09, STDDEV = 49.73)

- # Train details
    - **We train our models with some helper callbacks to achieve best trained models such as:**
        1. EarlyStopping callback: to stop training if there is no improvement in validation loss and overfitting occurs.
        2. ModelCheckpoint callback: to save the best model with the highest AUC Loss value.
        3. ReduceLROnPlateau callback: to reduce the learning rate if there is no improvement in validation loss.

    - **We define many metrics to evaluate the performance of the model such as:**
        1. TruePositives/FalsePositives
        2. TrueNegatives/FalseNegatives
        3. BinaryAccuracy
        4. Precision/Recall
        5. AUC(Area under the ROC-curve)

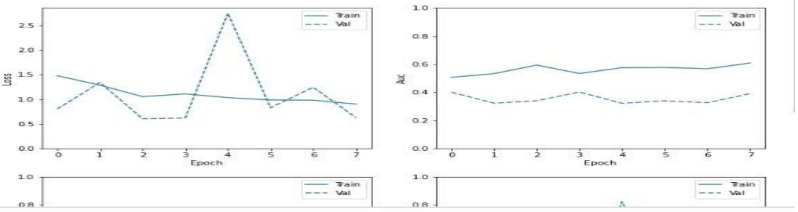
    - **We use the Adam optimizer with initial learning rate = 1e-4.**

    - **We overcome the problem of imbalance in the dataset using class weights. These will cause the model to "pay more attention" to examples from an under-represented class.**

- **Train results**
  - **ResNet 50**
    - **Acl - Sagittal**

```
[24] trainer.evaluate()

     120/120 [==============================
     loss :  1.3368421792984009
     tp :  0.0
     fp :  0.0
     tn :  66.0
     fn :  54.0
     accuracy :  0.550000011920929
     precision :  0.0
     recall :  0.0
     auc :  0.3291245698928833
```

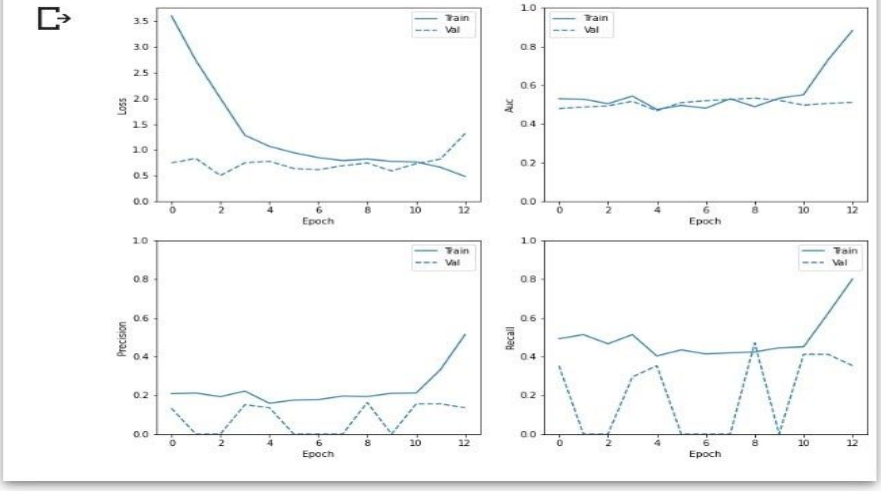
```
[25] trainer.plot_metrics()
```



    - **Acl - Coronal**

```
[20] trainer.evaluate()

     120/120 [==============================
     loss :  0.9699717164039612
     tp :  0.0
     fp :  0.0
     tn :  66.0
     fn :  54.0
     accuracy :  0.550000011920929
     precision :  0.0
     recall :  0.0
     auc :  0.6031144857406616
```

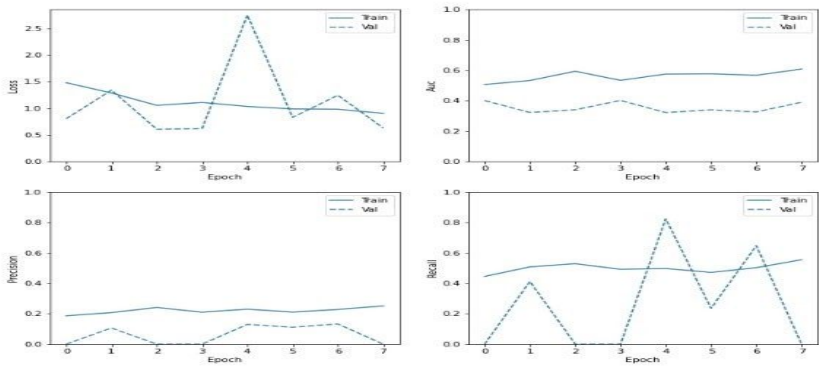
```
trainer.plot_metrics()
```



- **Acl - Axial**

```
[24]
120/120 [==============================:
loss :  1.3368421792984009
tp :  0.0
fp :  0.0
tn :  66.0
fn :  54.0
accuracy :  0.550000011920929
precision :  0.0
recall :  0.0
auc :  0.3291245698928833
```

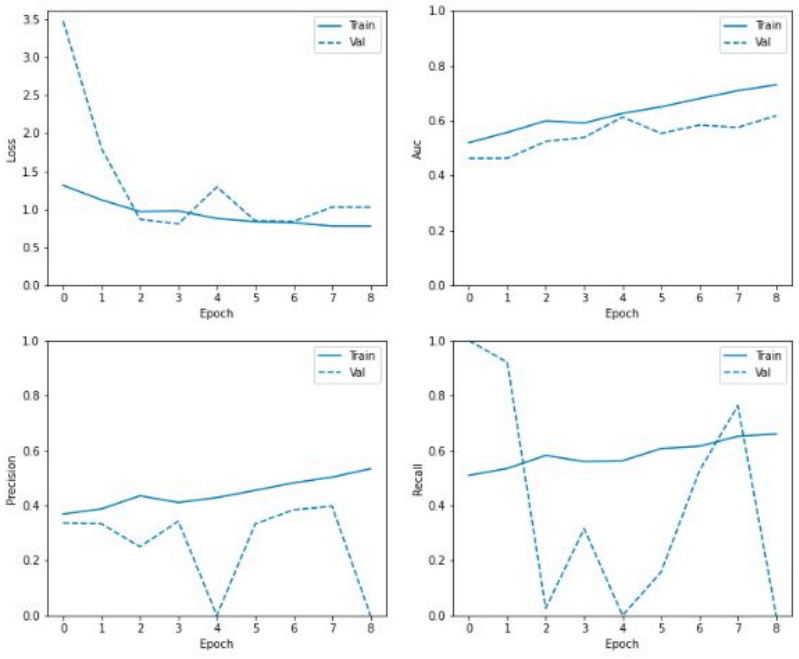
```
[25] trainer.plot_metrics()
```



- **Meniscus - Sagittal**

```
loss :  0.8650109767913818
tp :  20.0
fp :  14.0
tn :  54.0
fn :  32.0
accuracy :  0.6166666746139526
precision :  0.5882353186607361
recall :  0.38461539149284363
auc :  0.5325226187705994
```

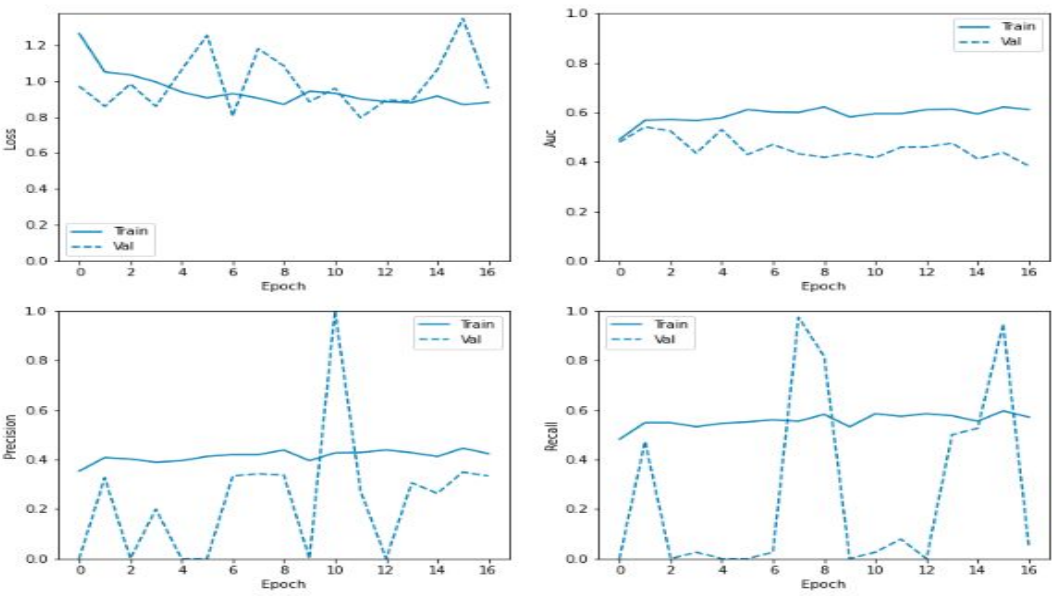
```
[19] trainer.plot_metrics()
```



■ Meniscus - Coronal

```
loss :  0.8878498077392578
tp :  8.0
fp :  13.0
tn :  55.0
fn :  44.0
accuracy :  0.5249999761581421
precision :  0.380952388048172
recall :  0.1538461595773697
auc :  0.45743778347969055
```

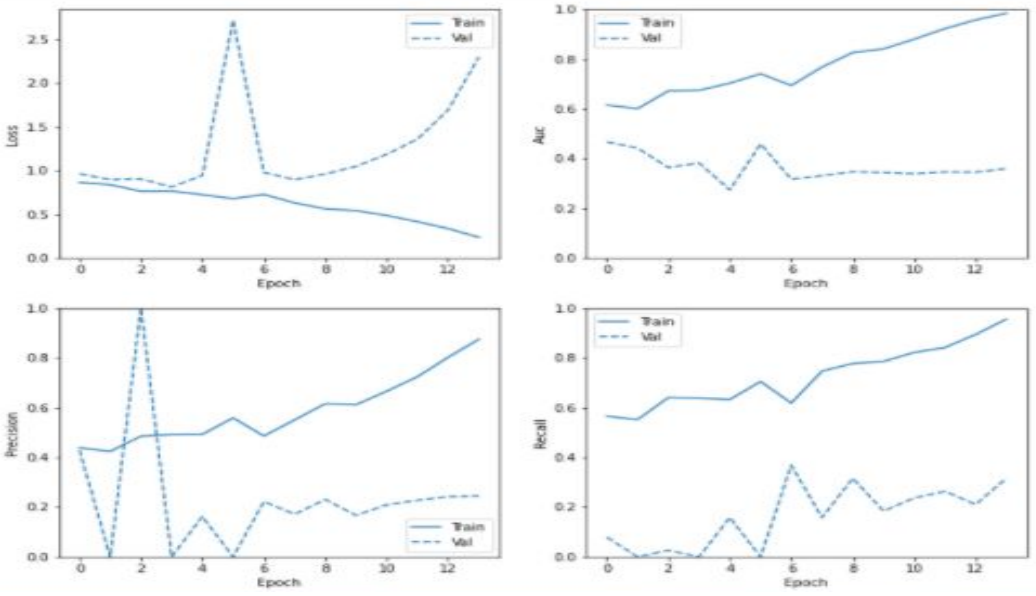
```
[35] trainer.plot_metrics()
```



■ Meniscus - Axial

```
[16] loss :  0.9461258053779602
tp :  0.0
fp :  2.0
tn :  66.0
fn :  52.0
accuracy :  0.550000011920929
precision :  0.0
recall :  0.0
auc :  0.41360294818878174
```

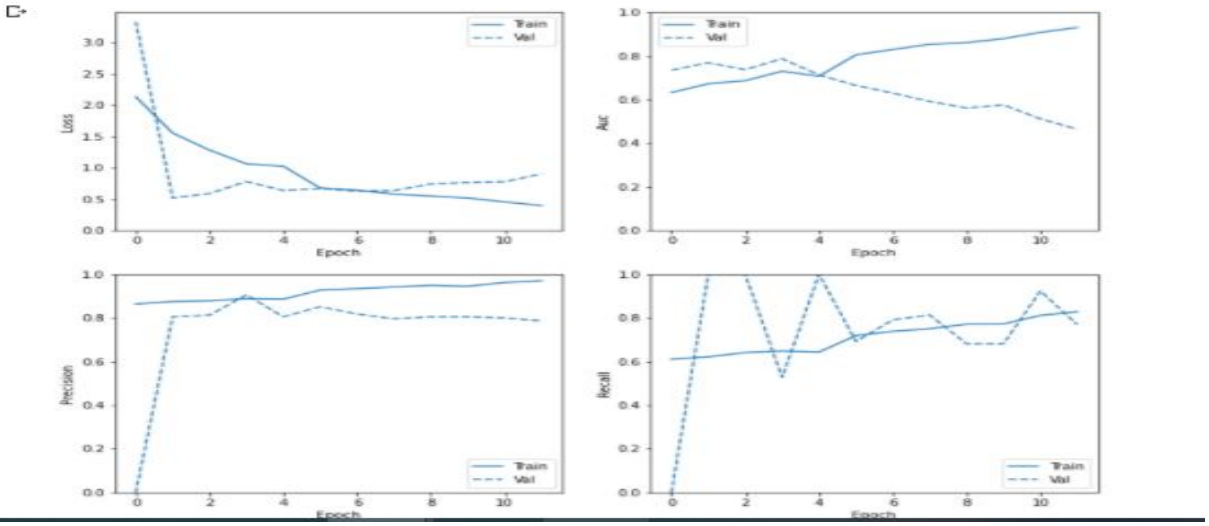
```
[17] trainer.plot_metrics()
```



- **Abnormal - Sagittal**

```
[16] 120/120 [==============================] - 9s 74ms/step - loss: 0.5274 - tp: 95.0000 - fp: 25.0000 - t
     loss :  0.5274019241333008
     tp :  95.0
     fp :  25.0
     tn :  0.0
     fn :  0.0
     accuracy :  0.7916666865348816
     precision :  0.7916666865348816
     recall :  1.0
     auc :  0.8412631750106812
```

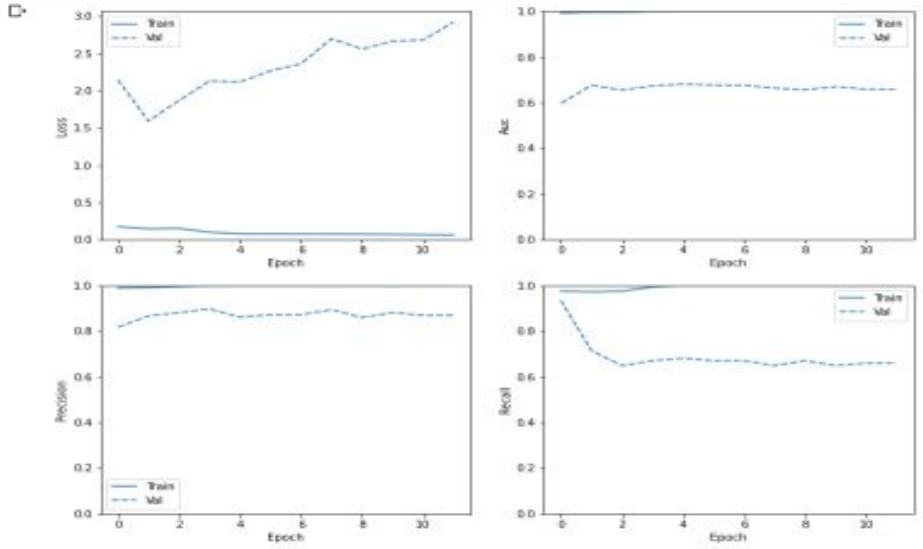
```
[17] trainer.plot_metrics()
```



- **Abnormal - Coronal**

```
[17] trainer.evaluate()

     120/120 [==============================] - 9s 72ms/step - loss: 1.7440 - tp: 68.0000 - fp: 16.0000 - tn: 9.0000
     loss :  1.7439670562744141
     tp :  68.0
     fp :  16.0
     tn :  9.0
     fn :  27.0
     accuracy :  0.6416666507720947
     precision :  0.8095238208770752
     recall :  0.7157894968986511
     auc :  0.5814736485481262
```

```
[18] trainer.plot_metrics()
```

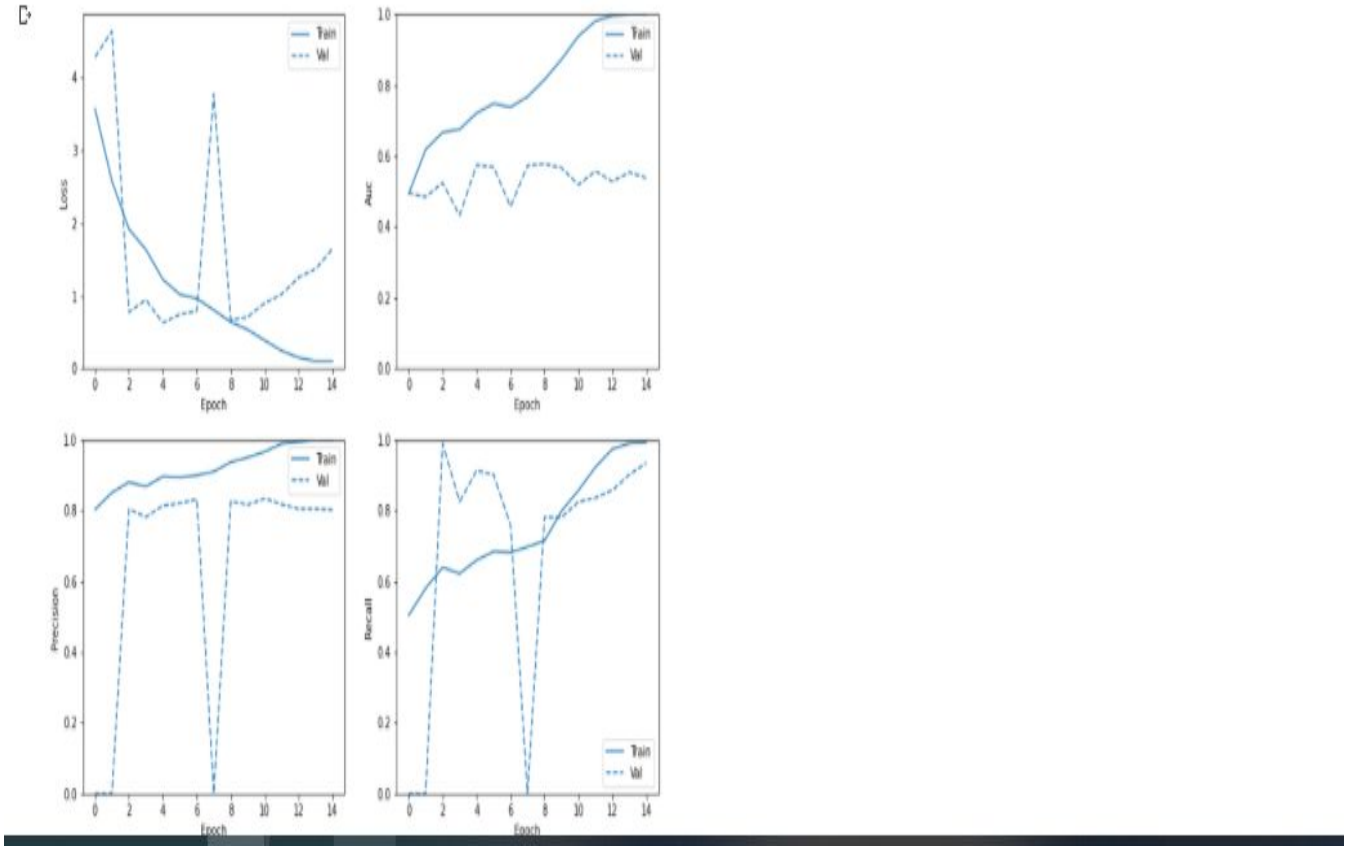



```
[ ]
```

■ Abnormal - Axial

```
[ ] 120/120 [==============================] - 10s 80ms/step - loss: 0.6015 - tp: 89.0000 - fp: 19.0000 - tn: 6.0000 - fn: 6.0000 - accuracy: 0.7917 - precision: 0.8241 - recall: 0.9368 - auc: 0.6251
    loss :  0.6015243530273438
    tp :  89.0
    fp :  19.0
    tn :  6.0
    fn :  6.0
    accuracy :  0.7916666865348816
    precision :  0.8240740895271301
    recall :  0.9368420839309692
    auc :  0.6250526309013367
```

```
[ ] trainer.plot_metrics()
```

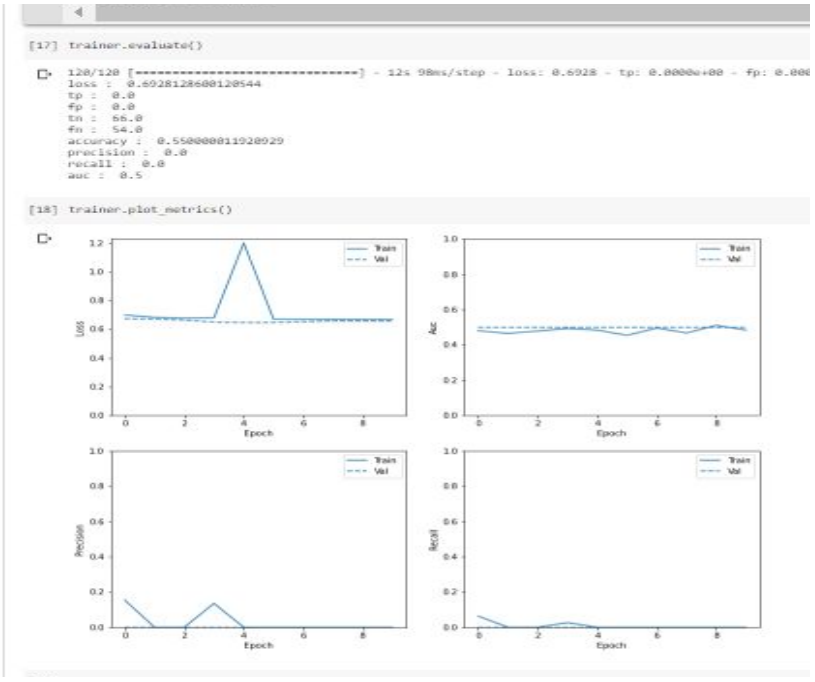


- ***Vgg 16***

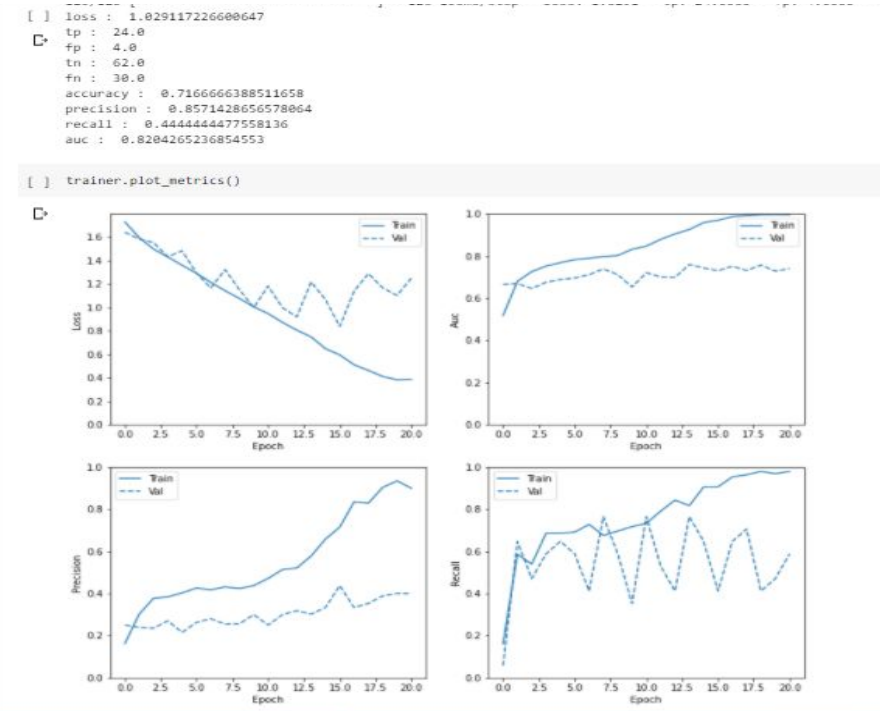
  We train with two versions. One time we do A global average pooling layer(old version) and another one we do flatten(new version)

  - **Acl - Sagittal**

Old version



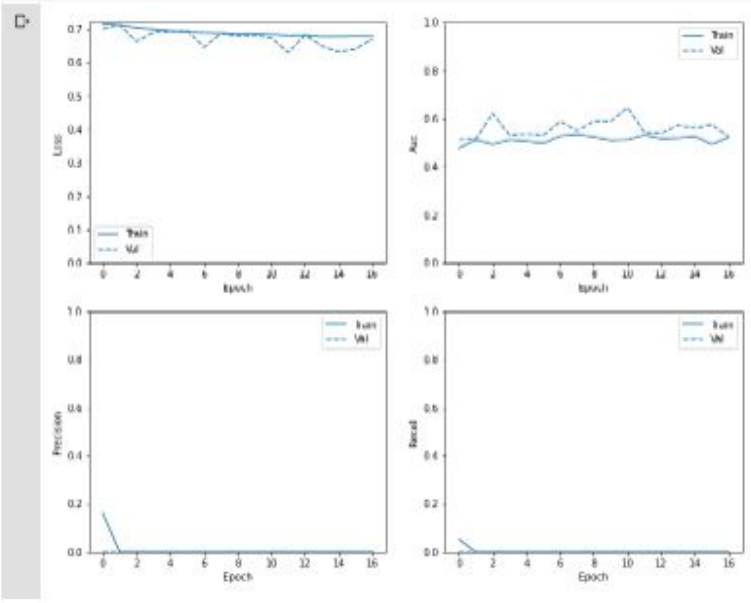
New version



- **Acl - Coronal**

Old version

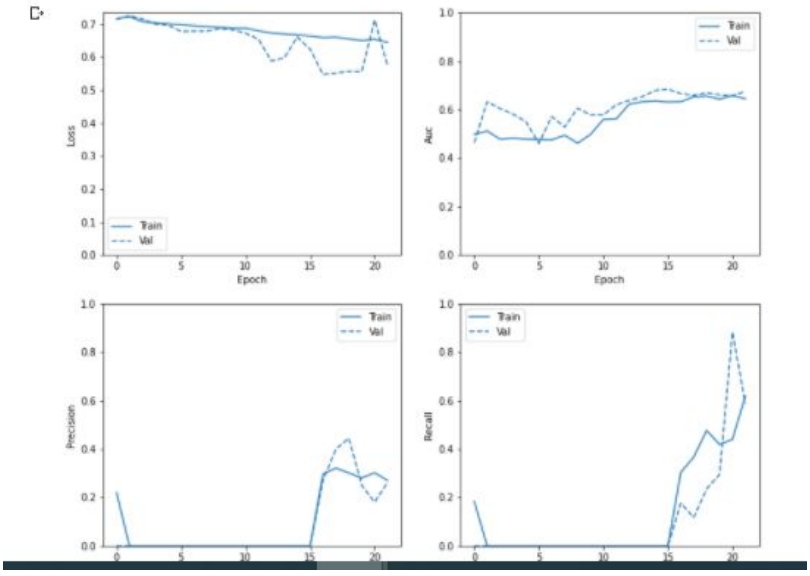
```
[17] trainer.evaluate()

120/120 [==============================] - 20s 168ms/step - loss: 0.6995 - tp: 0.0000e+00 - fp: 0.0000e+00 -
loss :  0.6994817852973938
tp :  0.0
fp :  0.0
tn :  66.0
fn :  54.0
accuracy :  0.550000011920929
precision :  0.0
recall :  0.0
auc :  0.4983164668083191

[18] trainer.plot_metrics()
```



- **Acl - Axial**

Old version

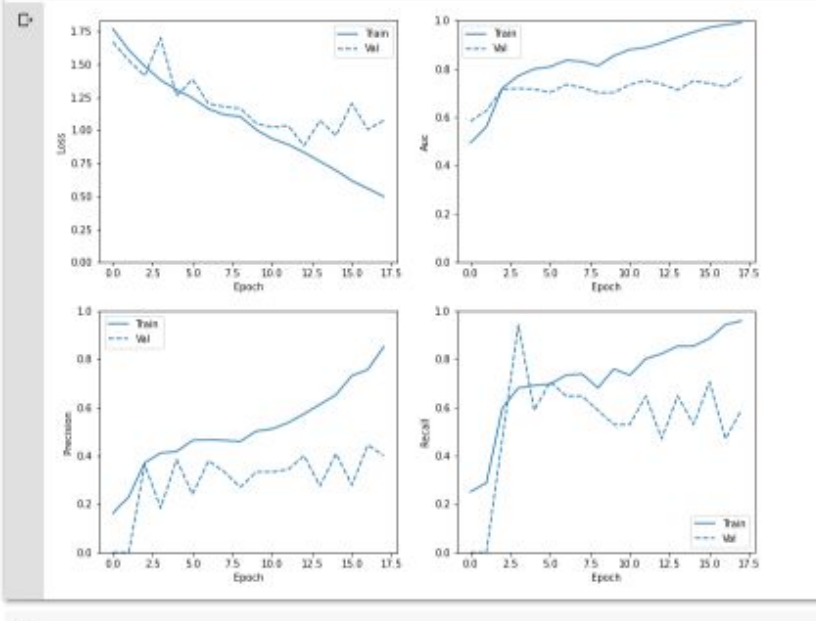
```
loss :  0.7073988318443298
tp :  14.0
fp :  8.0
tn :  58.0
fn :  40.0
accuracy :  0.6000000238418579
precision :  0.6363636255264282
recall :  0.25925925374031067
auc :  0.596520721912384

[18] trainer.plot_metrics()
```



New version

```
loss :  0.0413861632347107
tp :  31.0
fp :  8.0
tn :  58.0
fn :  23.0
accuracy :  0.7416666746139526
precision :  0.7948718070983887
recall :  0.5740740895271301
auc :  0.8518518209457397
```

```
trainer.plot_metrics()
```

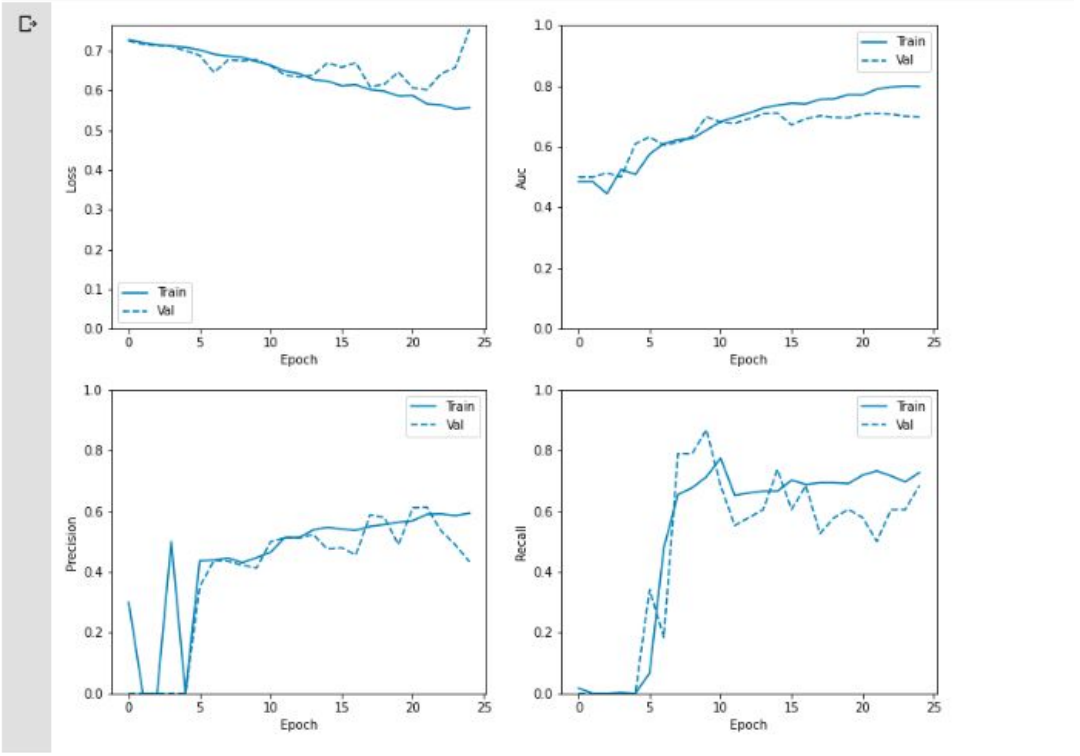


- **Meniscus - Sagittal**

Old Version

```
loss :  0.6552752852439888
tp :  43.0
fp :  33.0
tn :  35.0
fn :  9.0
accuracy :  0.6499999761581421
precision :  0.5657894611358643
recall :  0.8269230723381042
auc :  0.7642816305160522
```

```
[18] trainer.plot_metrics()
```

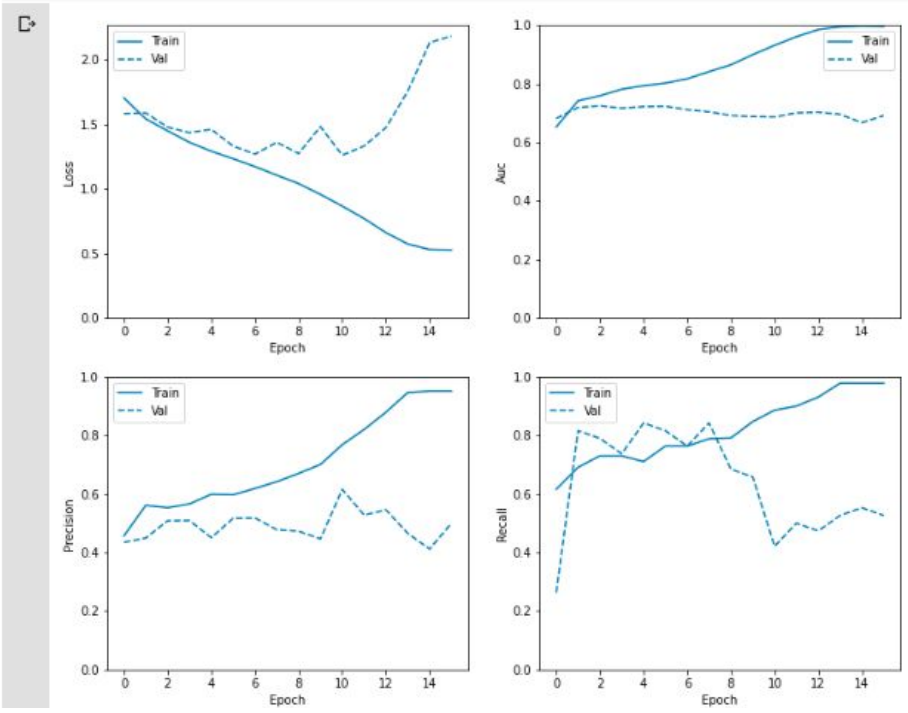


New Version

```
trainer.evaluate()
```

```
120/120 [==============================] -
loss :  1.4991611242294312
tp :  16.0
fp :  17.0
tn :  51.0
fn :  36.0
accuracy :  0.5583333373069763
precision :  0.4848484992980957
recall :  0.3076923191547394
auc :  0.5718324780464172
```

```
[18] trainer.plot_metrics()
```

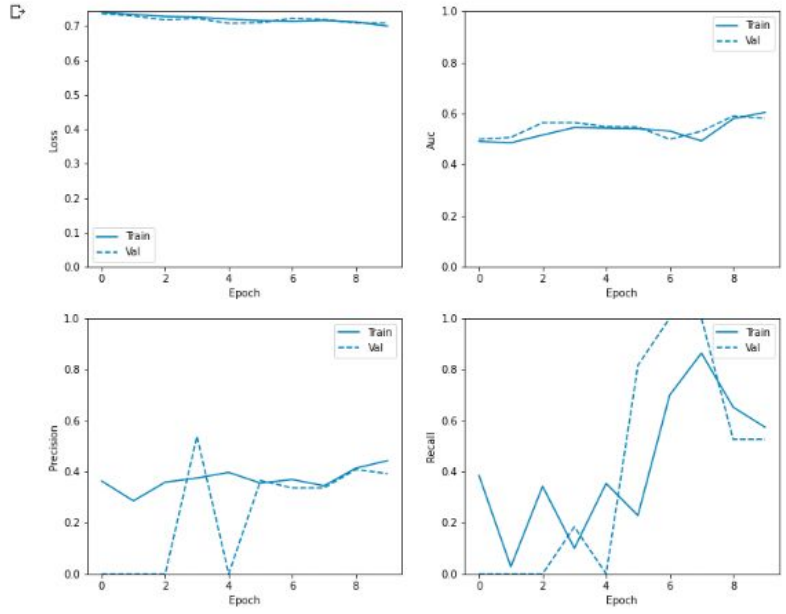


- **Meniscus - Coronal**

Old version

```
loss :  0.7138941884040833
tp :  0.0
fp :  0.0
tn :  68.0
fn :  52.0
accuracy :  0.5666666626930237
precision :  0.0
recall :  0.0
auc :  0.5370475053787231
```

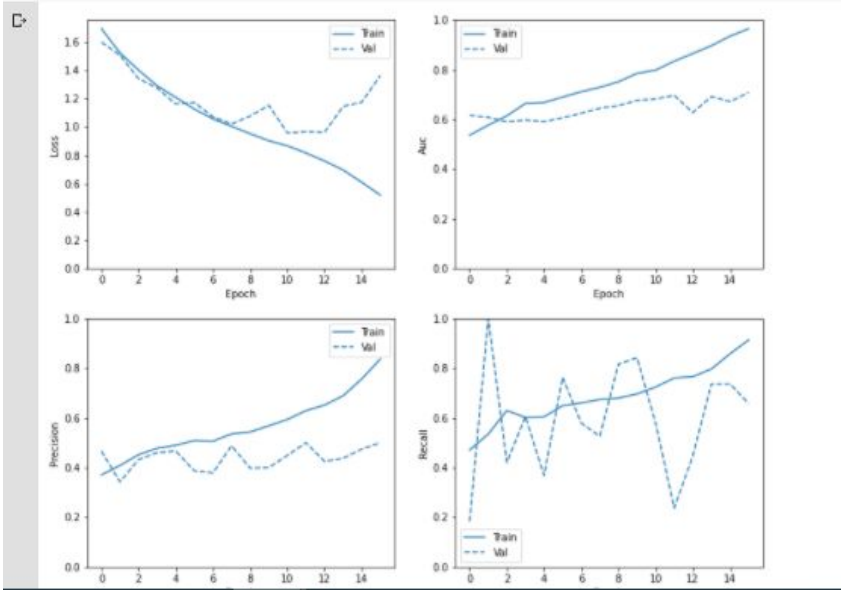
```
[44] trainer.plot_metrics()
```



New version

```
loss :  0.9761182069778442
tp :  35.0
fp :  22.0
tn :  46.0
fn :  17.0
accuracy :  0.675000011920929
precision :  0.6140350699424744
recall :  0.6730769276618958
auc :  0.7061651945114136
```

```
[18] trainer.plot_metrics()
```

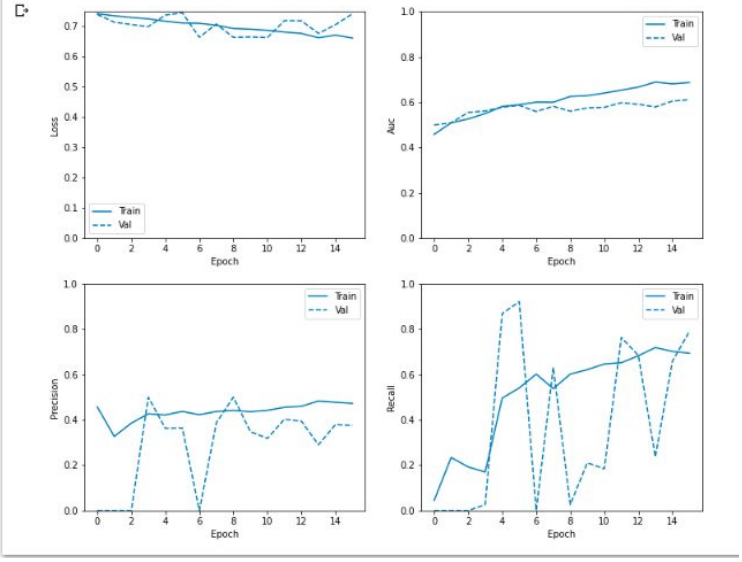


- **Meniscus - Axial**

Old version

```
loss :  0.6946406960487366
tp :  13.0
fp :  13.0
tn :  55.0
fn :  39.0
accuracy :  0.5666666626930237
precision :  0.5
recall :  0.25
auc :  0.6134049296379089
```

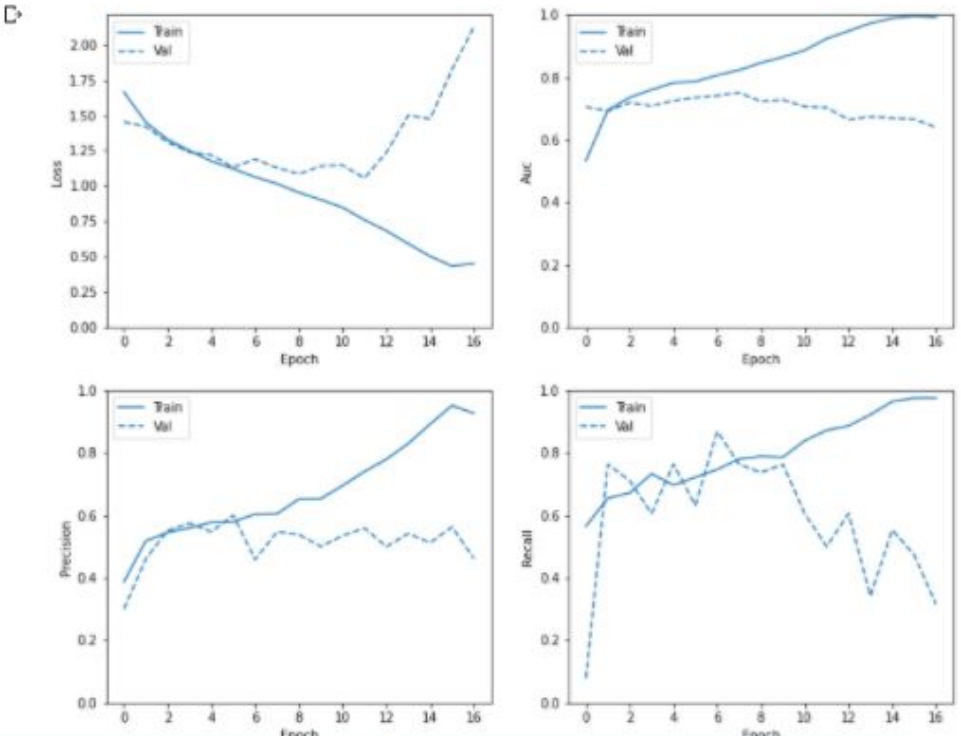
```
trainer.plot_metrics()
```



New version

```
[ ] loss :  1.0914644002914429
    tp :  30.0
    fp :  23.0
    tn :  45.0
    fn :  22.0
    accuracy :  0.625
    precision :  0.5660377144813538
    recall :  0.5769230723381042
    auc :  0.7072963714599609
```

```
[ ] trainer.plot_metrics()
```

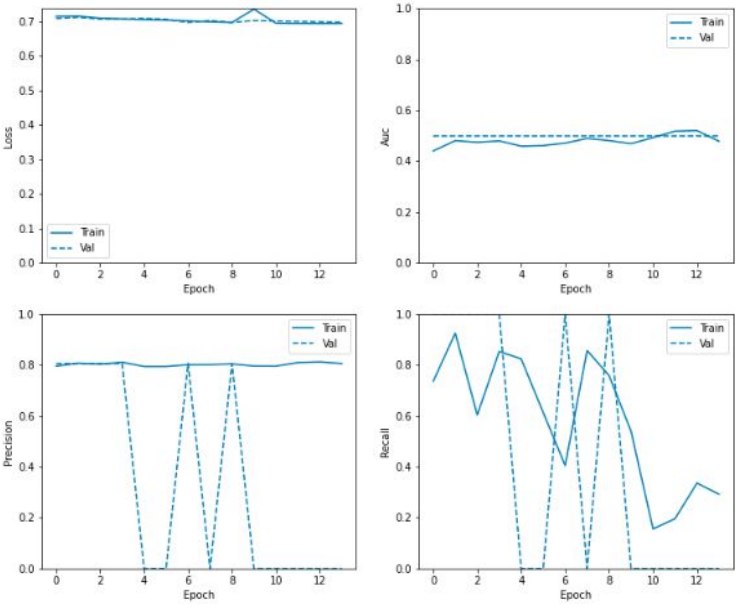


- **Abnormal - Sagittal**

Old version

```
loss :  0.6972228288650513
tp :  95.0
fp :  25.0
tn :  0.0
fn :  0.0
accuracy :  0.7916666865348816
precision :  0.7916666865348816
recall :  1.0
auc :  0.5
```

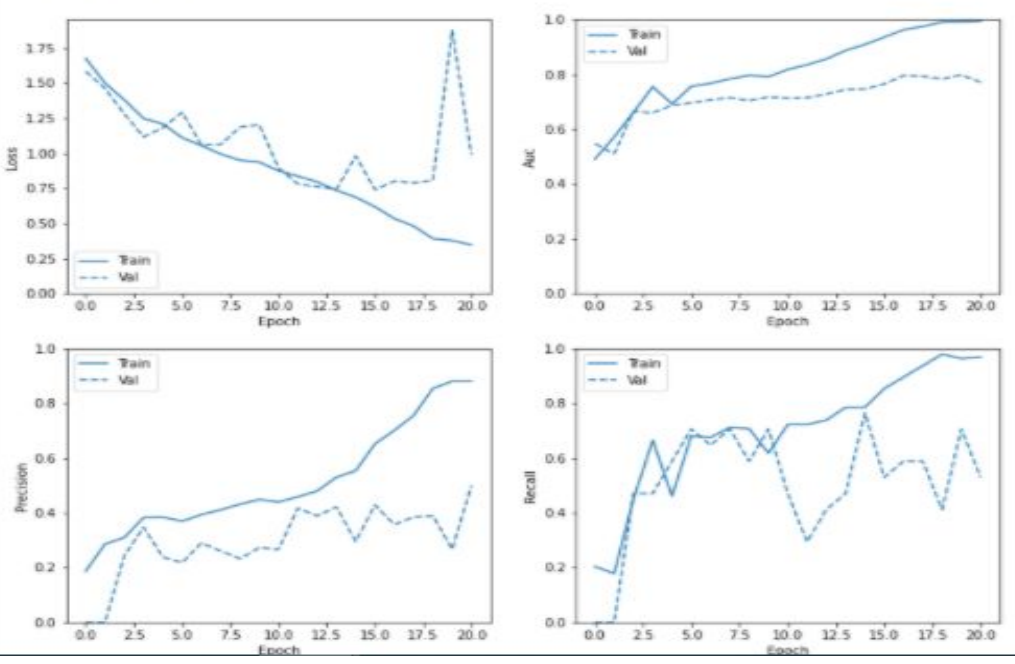
```
trainer.plot_metrics()
```



New version

```
loss :  0.933467447757721
tp :  29.0
fp :  6.0
tn :  60.0
fn :  25.0
accuracy :  0.7416666746139526
precision :  0.8285714387893677
recall :  0.5370370149612427
auc :  0.8218293786048889
```

```
trainer.plot_metrics()
```

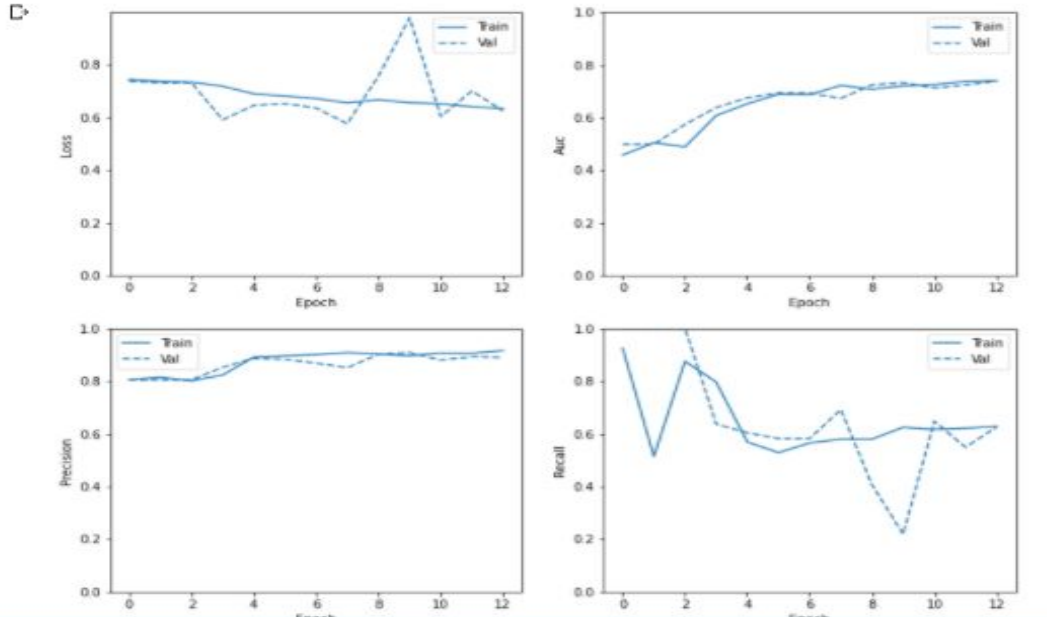


- **Abnormal - Coronal**

Old version

```
loss :  0.6050395965576172
tp :  69.0
fp :  11.0
tn :  14.0
fn :  26.0
accuracy :  0.6916666626930237
precision :  0.862500011920929
recall :  0.7263157963752747
auc :  0.6338947415351868
```

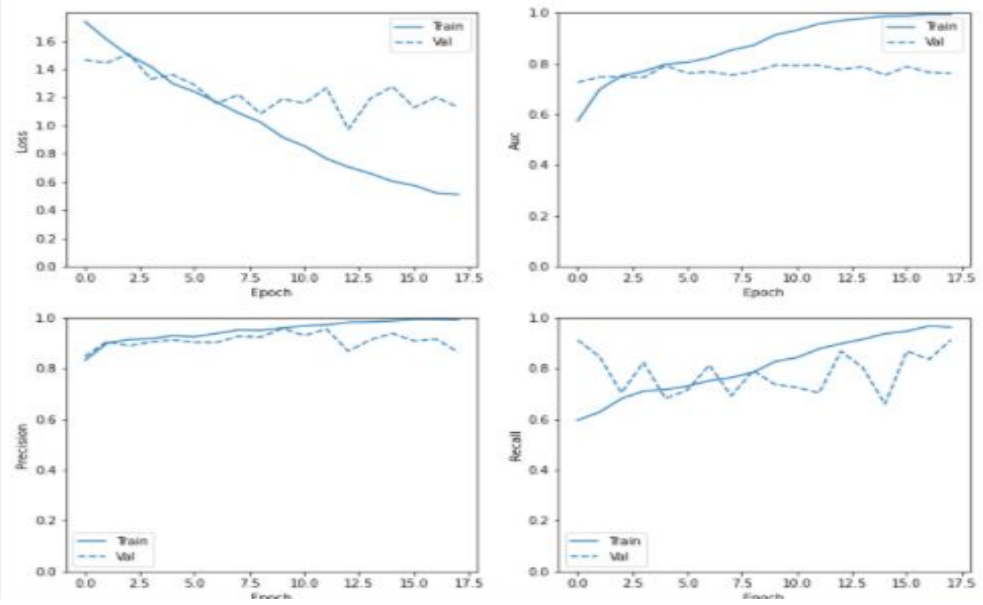
```
[18] trainer.plot_metrics()
```



New version

```
loss :  1.0757982730865479
tp :  90.0
fp :  21.0
tn :  4.0
fn :  5.0
accuracy :  0.7833333611488342
precision :  0.8108108043670654
recall :  0.9473684430122375
auc :  0.7122105360031128
```

```
[ ] trainer.plot_metrics()
```

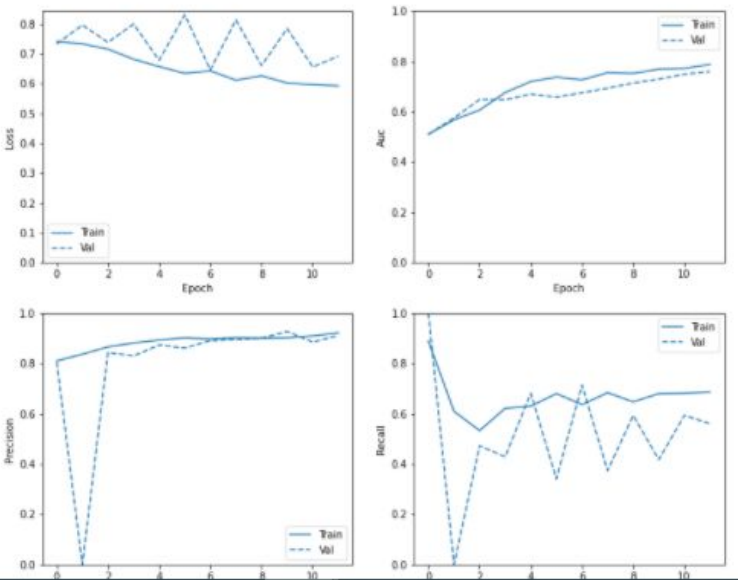


- **Abnormal - Axial**

Old version

```
120/120 [==============================] - 23s 193ms/step - loss: 0.6580 - tp: 72.0000 - fp: 14.0000 -
loss :  0.6580488085746765
tp :  72.0
fp :  14.0
tn :  11.0
fn :  23.0
accuracy :  0.6916666626930237
precision :  0.8372092843055725
recall :  0.75789475440979
auc :  0.5993683934211731
```

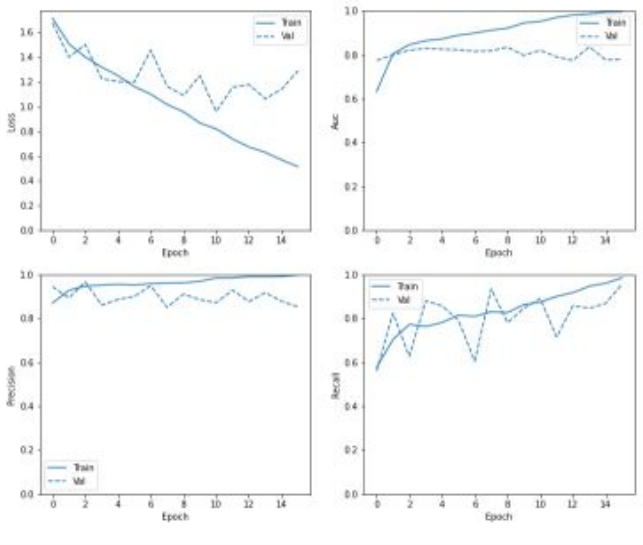
```
[18] trainer.plot_metrics()
```



New version

```
loss :  0.8616342544555664
tp :  94.0
fp :  14.0
tn :  11.0
fn :  1.0
accuracy :  0.875
precision :  0.8703703880310059
recall :  0.9894737005233765
auc :  0.9025263786315918
```

```
[36] trainer.plot_metrics()
```



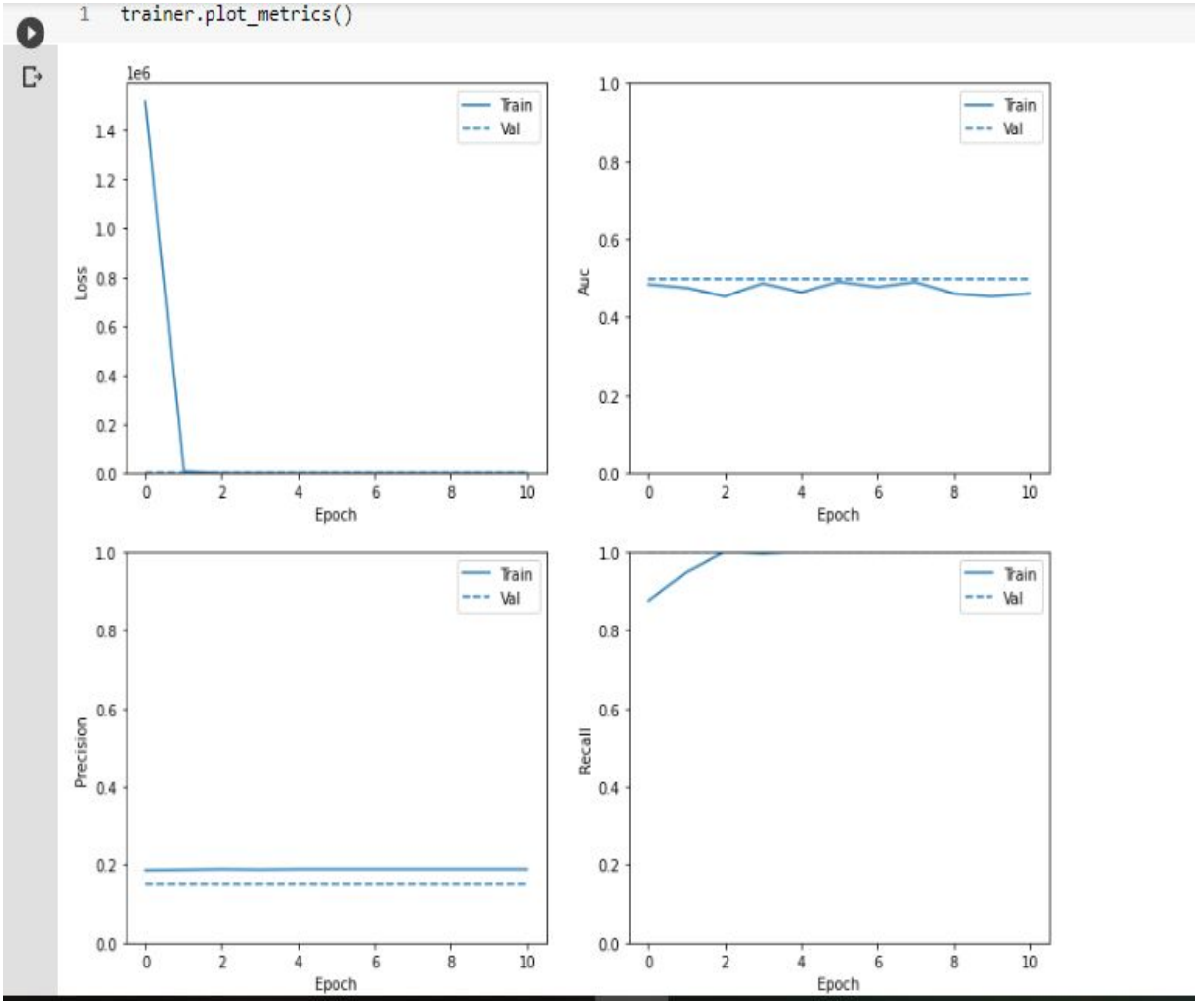

```
[36]
```

- **Inception V3**
  - **Acl - Sagittal**

    First time which is a failed one before doing normalization on models.

```
[32]  1  trainer.evaluate()

120/120 [==============================] -
loss :  0.7078002095222473
tp :  54.0
fp :  66.0
tn :  0.0
fn :  0.0
accuracy :  0.44999998807907104
precision :  0.44999998807907104
recall :  1.0
auc :  0.5
```

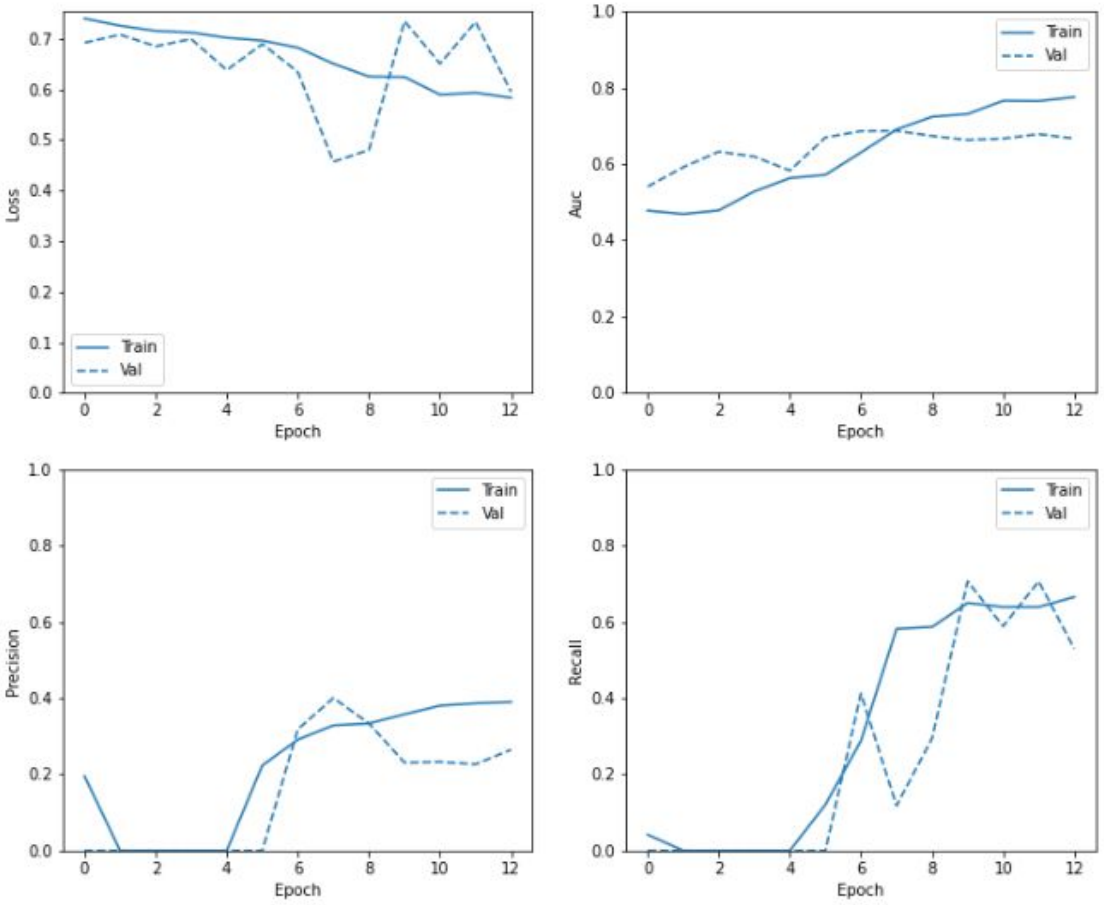
```
1  trainer.plot_metrics()
```



Second time after normalization.

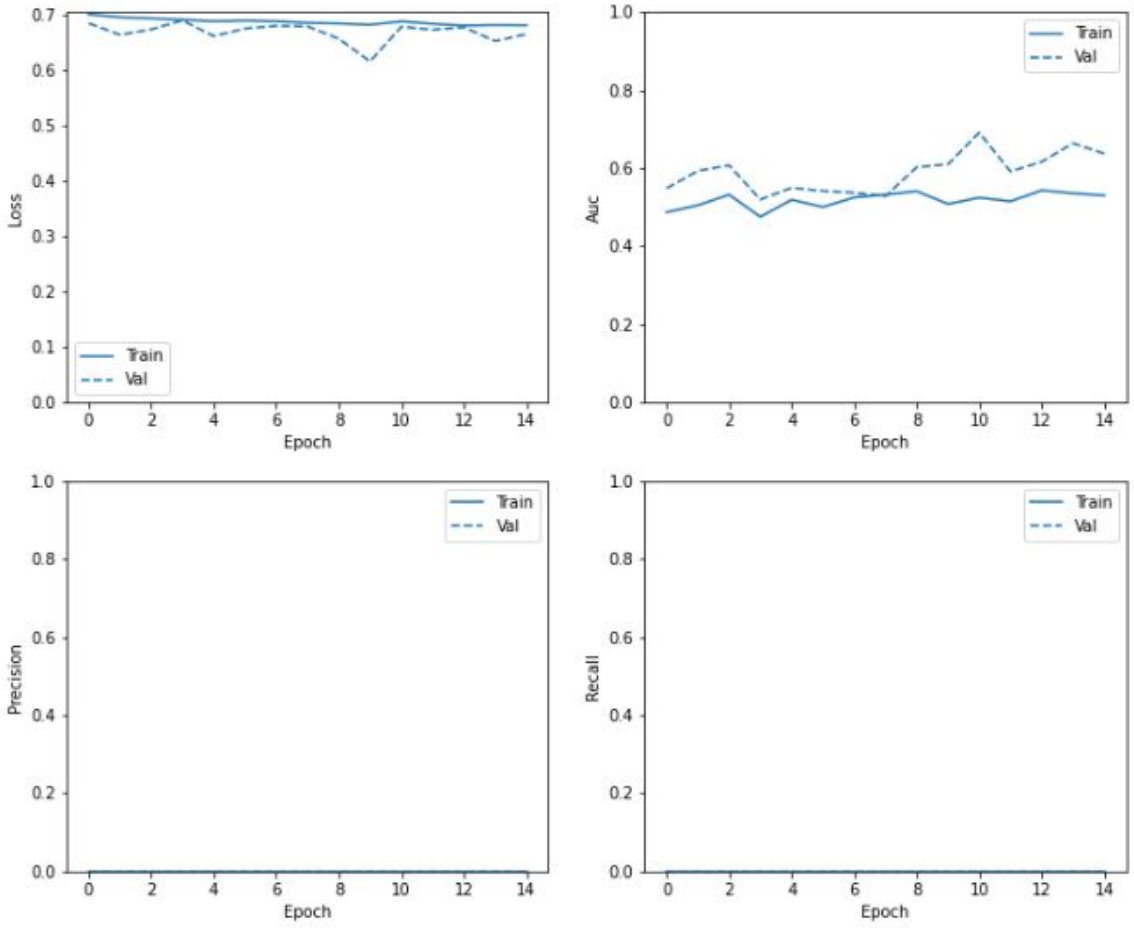
```
[19]  1  trainer.evaluate()
```

```
120/120 [==============================] -
loss :  0.6847073435783386
tp :  7.0
fp :  0.0
tn :  66.0
fn :  47.0
accuracy :  0.6083333492279053
precision :  1.0
recall :  0.12962962687015533
auc :  0.8208473324775696
```



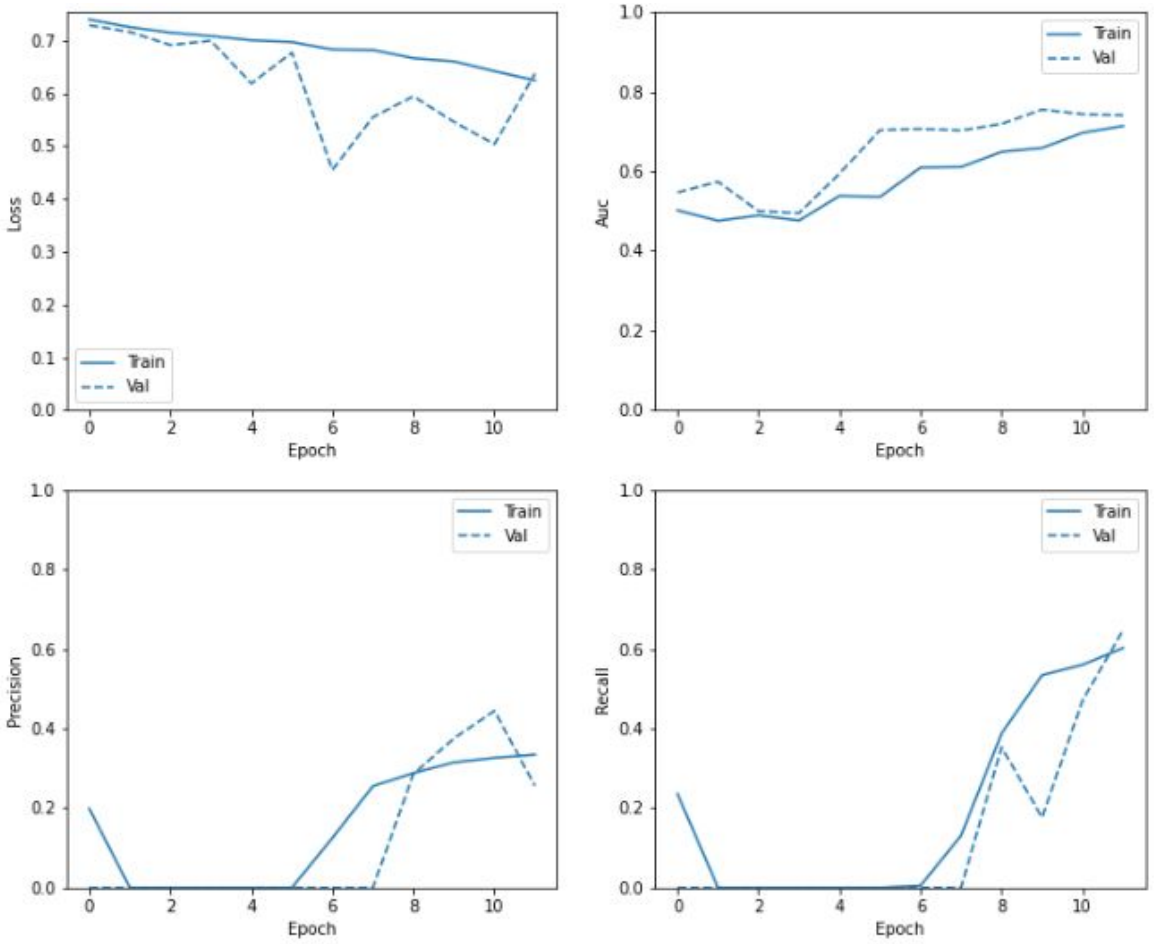
- **Acl - Coronal**

```
120/120 [==============================]
loss :  0.7035849690437317
tp :  0.0
fp :  0.0
tn :  66.0
fn :  54.0
accuracy :  0.550000011920929
precision :  0.0
recall :  0.0
auc :  0.523849606513977
```



- **Acl - Axial**

```
120/120 [==============================]
loss :  0.7465319037437439
tp :  0.0
fp :  0.0
tn :  66.0
fn :  54.0
accuracy :  0.550000011920929
precision :  0.0
recall :  0.0
auc :  0.746632993221283
```

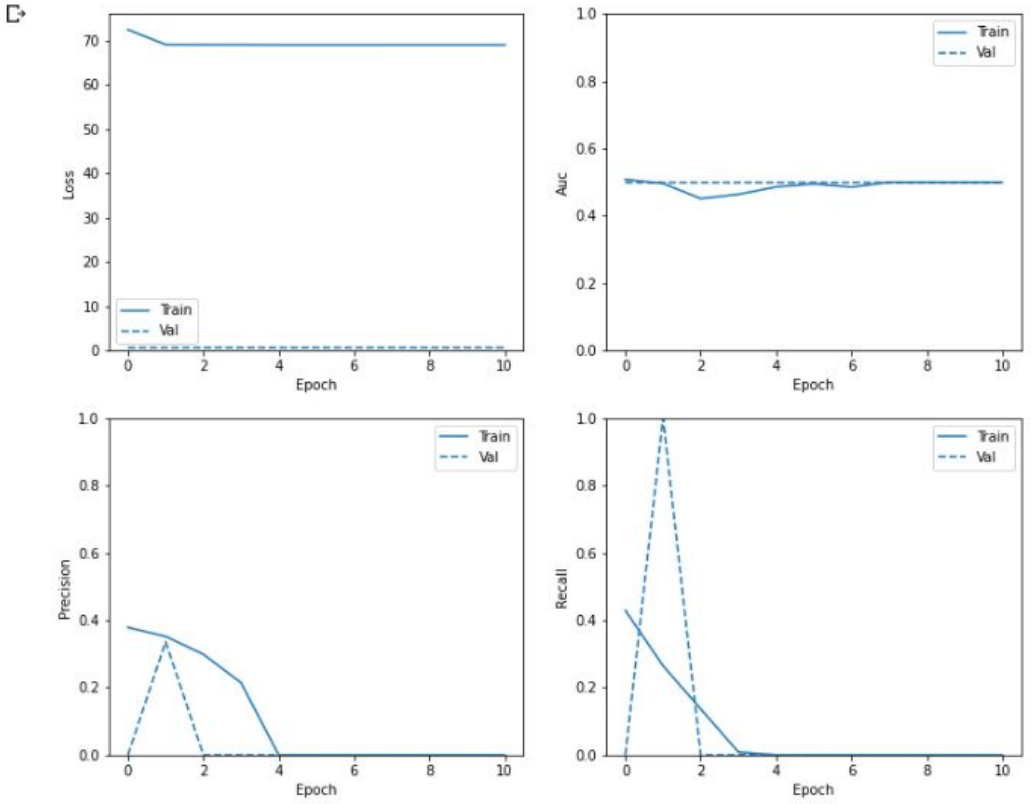


- **Meniscus - Sagittal**

  First time which is a failed one before doing normalization on models.

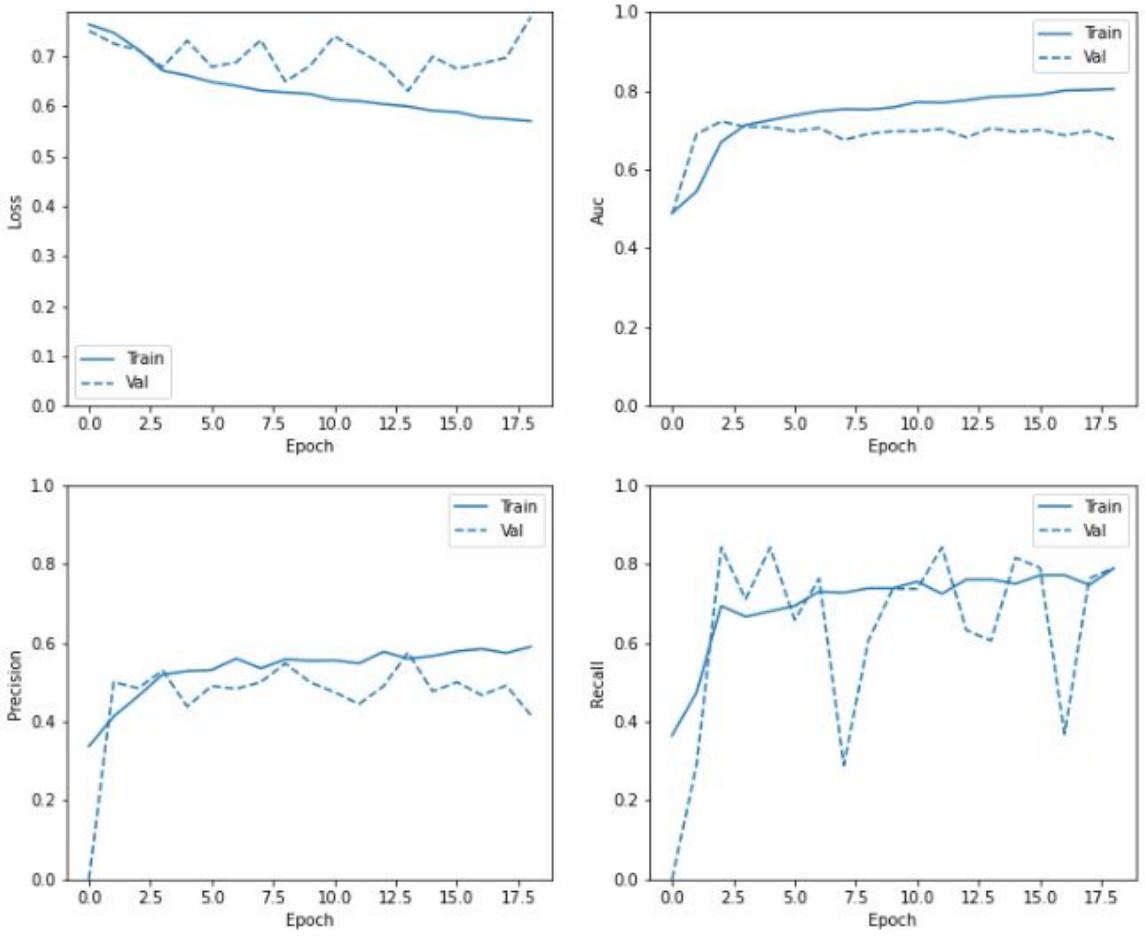
```
[ ]  1  trainer.evaluate()

     120/120 [==============================] -
     loss :  0.6892433762550354
     tp :  0.0
     fp :  0.0
     tn :  68.0
     fn :  52.0
     accuracy :  0.5666666626930237
     precision :  0.0
     recall :  0.0
     auc :  0.5
```

```
[ ]  1  trainer.plot_metrics()
```



Second time after normalization

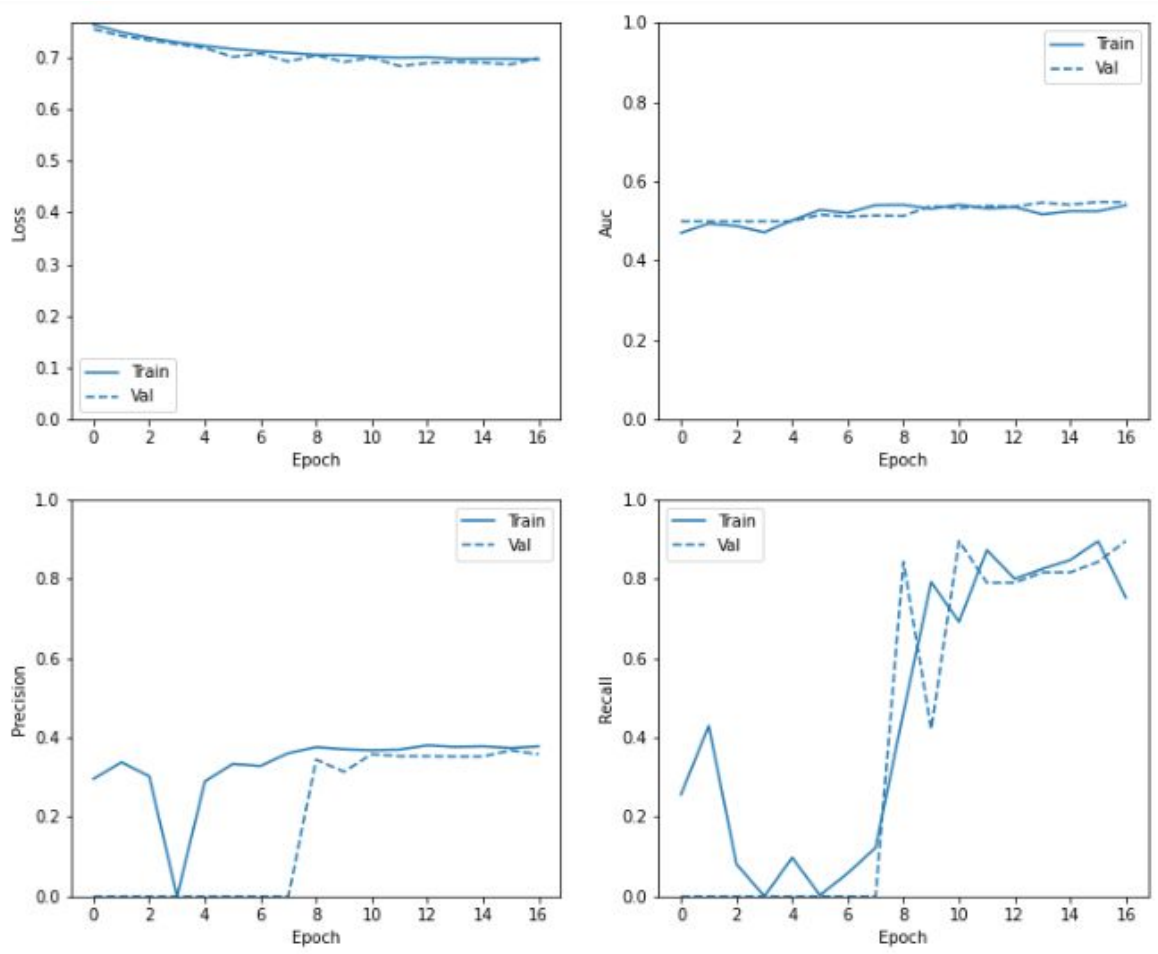
```
120/120 [==============================] -
loss :  0.6201345324516296
tp :  37.0
fp :  17.0
tn :  51.0
fn :  15.0
accuracy :  0.7333333492279053
precision :  0.6851851940155029
recall :  0.7115384340286255
auc :  0.7918551564216614
```



- **Meniscus - Coronal**

```
[34]  1  trainer.evaluate()
```

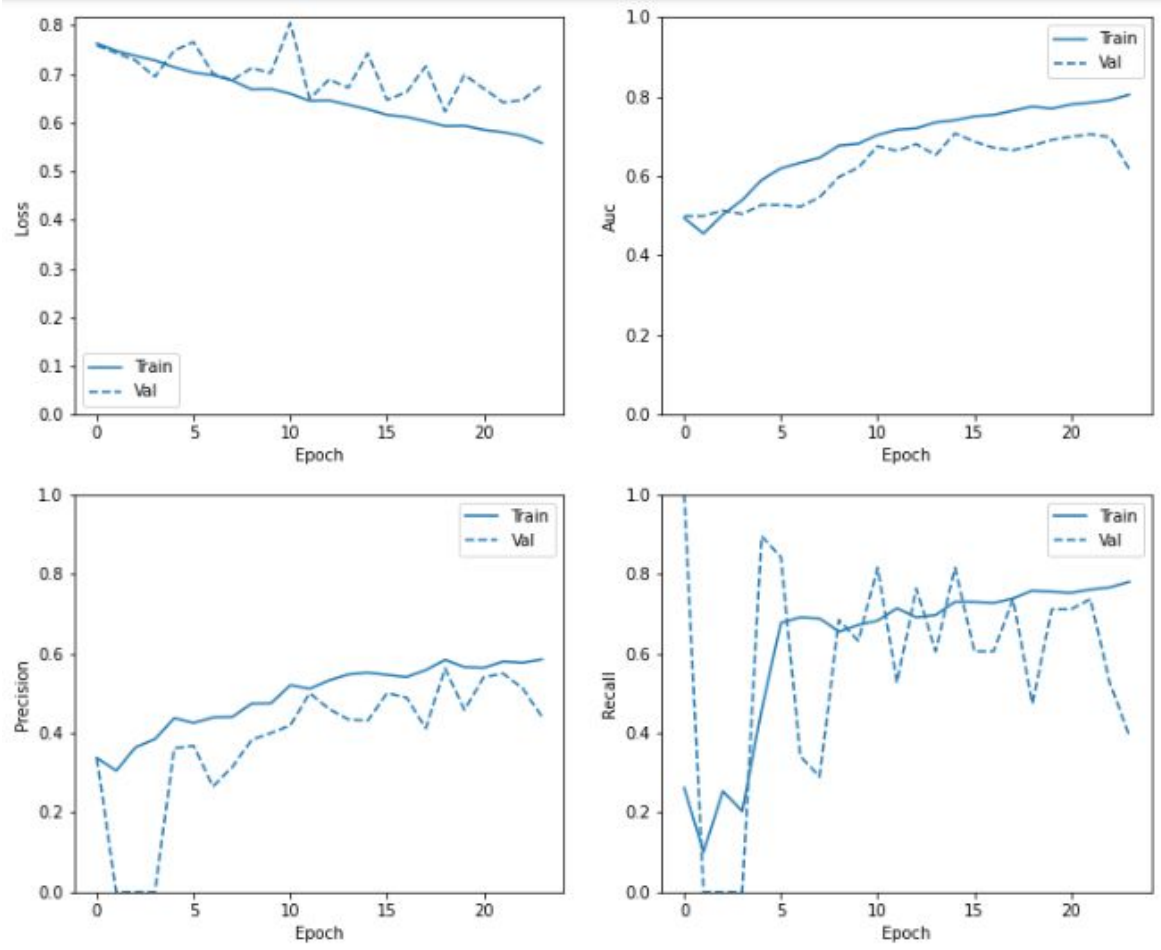
```
120/120 [==============================]
loss :  0.6895701289176941
tp :  45.0
fp :  50.0
tn :  18.0
fn :  7.0
accuracy :  0.5249999761581421
precision :  0.4736842215061188
recall :  0.8653846383094788
auc :  0.5616515874862671
```



- **Meniscus - Axial**

```
[45]  1  trainer.evaluate()
```

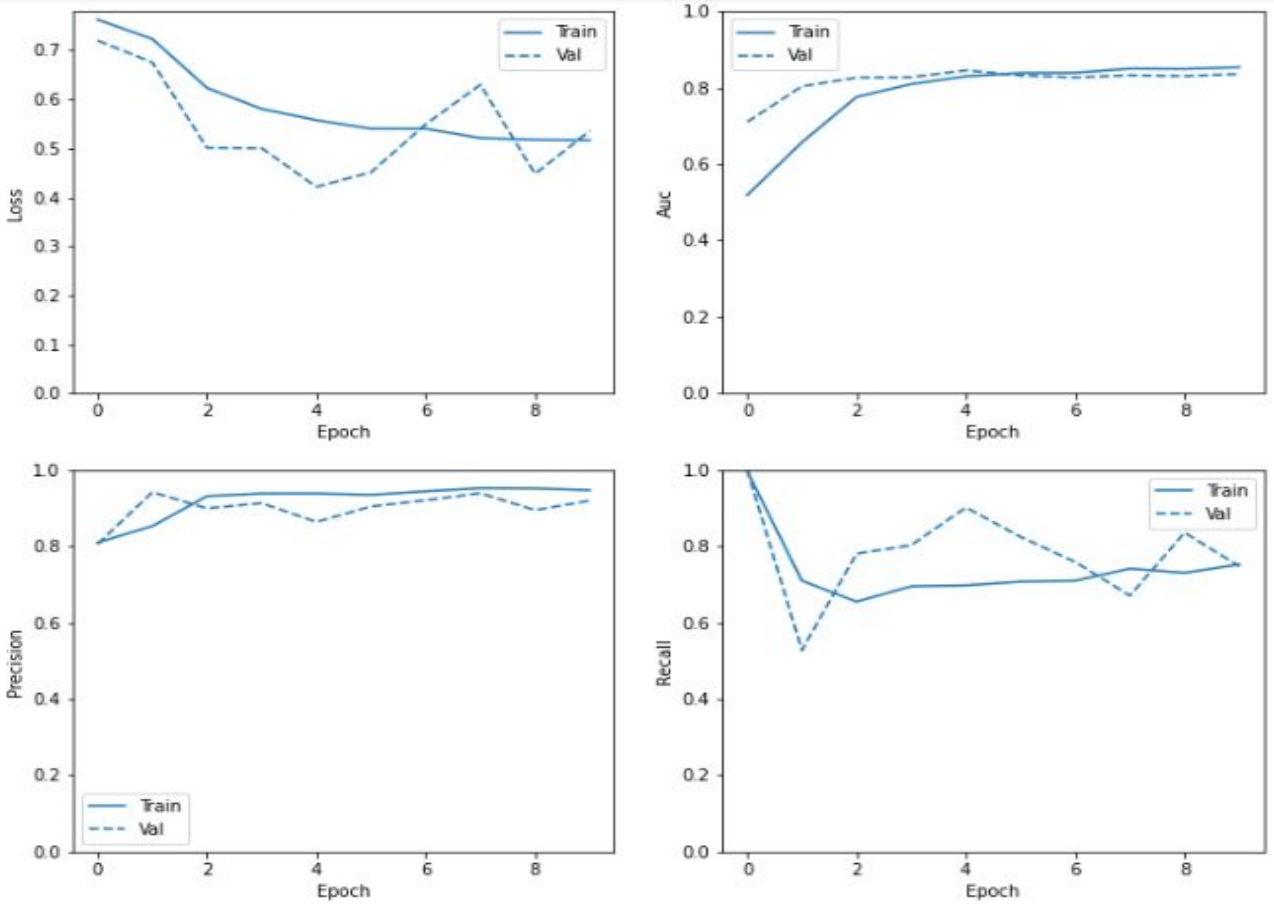
```
120/120 [==============================]
loss :  0.6191207766532898
tp :  32.0
fp :  16.0
tn :  52.0
fn :  20.0
accuracy :  0.699999988079071
precision :  0.6666666865348816
recall :  0.6153846383094788
auc :  0.7427884340286255
```



- **Abnormal - Sagittal**

```
trainer.evaluate()
```

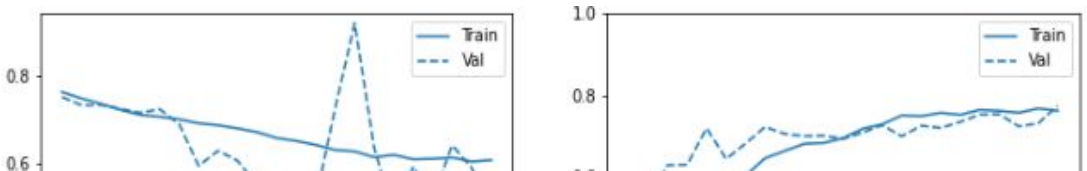
```
120/120 [==============================] - 8s 6
loss :  0.4004831612110138
tp :  91.0
fp :  15.0
tn :  10.0
fn :  4.0
accuracy :  0.8416666388511658
precision :  0.8584905862808228
recall :  0.9578947424888611
auc :  0.8919999599456787
```



- **Abnormal - Axial**

```
trainer.evaluate()
```

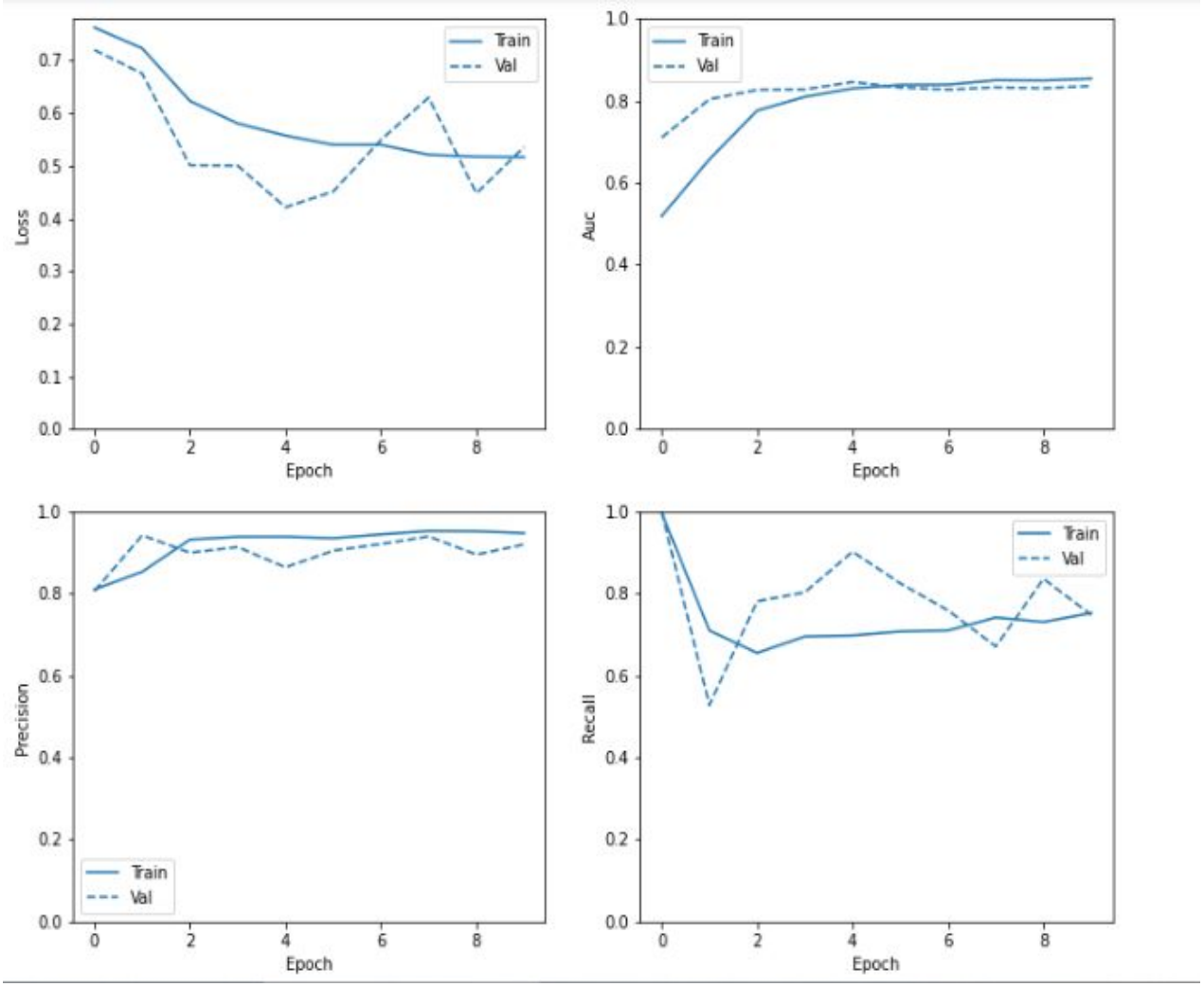
```
120/120 [==============================] -
loss :  0.4004831612110138
tp :  91.0
fp :  15.0
tn :  10.0
fn :  4.0
accuracy :  0.8416666388511658
precision :  0.8584905862808228
recall :  0.9578947424888611
auc :  0.8919999599456787
```



- **Abnormal - Coronal**

```
trainer.evaluate()
```

```
120/120 [==============================]
loss :  0.564805805683136
tp :  80.0
fp :  16.0
tn :  9.0
fn :  15.0
accuracy :  0.7416666746139526
precision :  0.8333333134651184
recall :  0.8421052694320679
auc :  0.6227368116378784
```

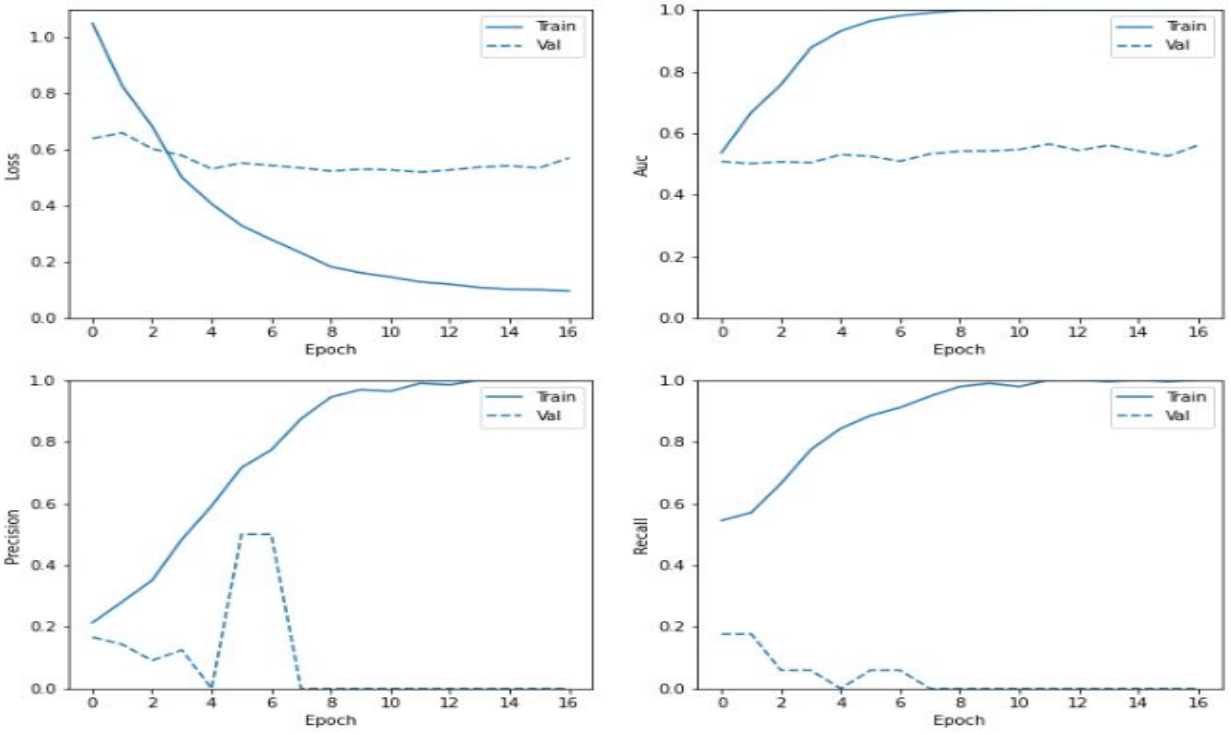


- **Transfer learning**
  - We made a full train on the three diseases with the three planes on the three models.
  - We made transfer learning on inception because it is the best result in AUC.

Points of comparisons	ResNet 50		Inception V3		Vgg 16	
	Loss	Auc	Loss	Auc	Loss	Auc
Acl	0.6798	0.596941	0.6459625	0.829963	0.83132	0.39843
Meniscus	0.676249	0.6455034	0.6382402	0.7164875	0.682755	0.53139
Abnormal	0.51309	0.658526	0.4849827	0.765684	0.65385	0.33557

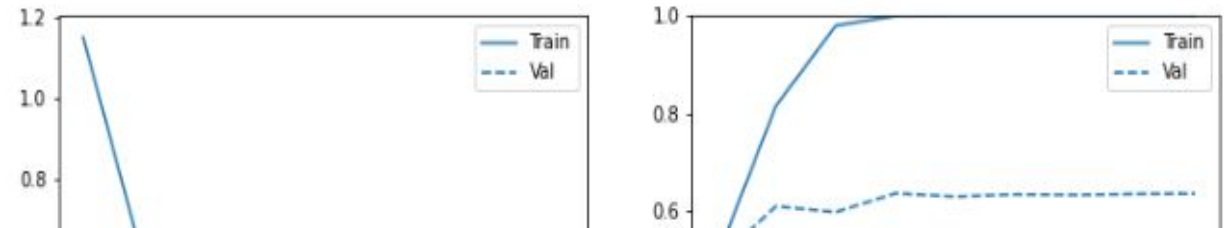
    - ACl - Sagittal

```
120/120 [==============================]
loss :  1.091561198234558
tp :  2.0
fp :  0.0
```



- ACl - Coronal

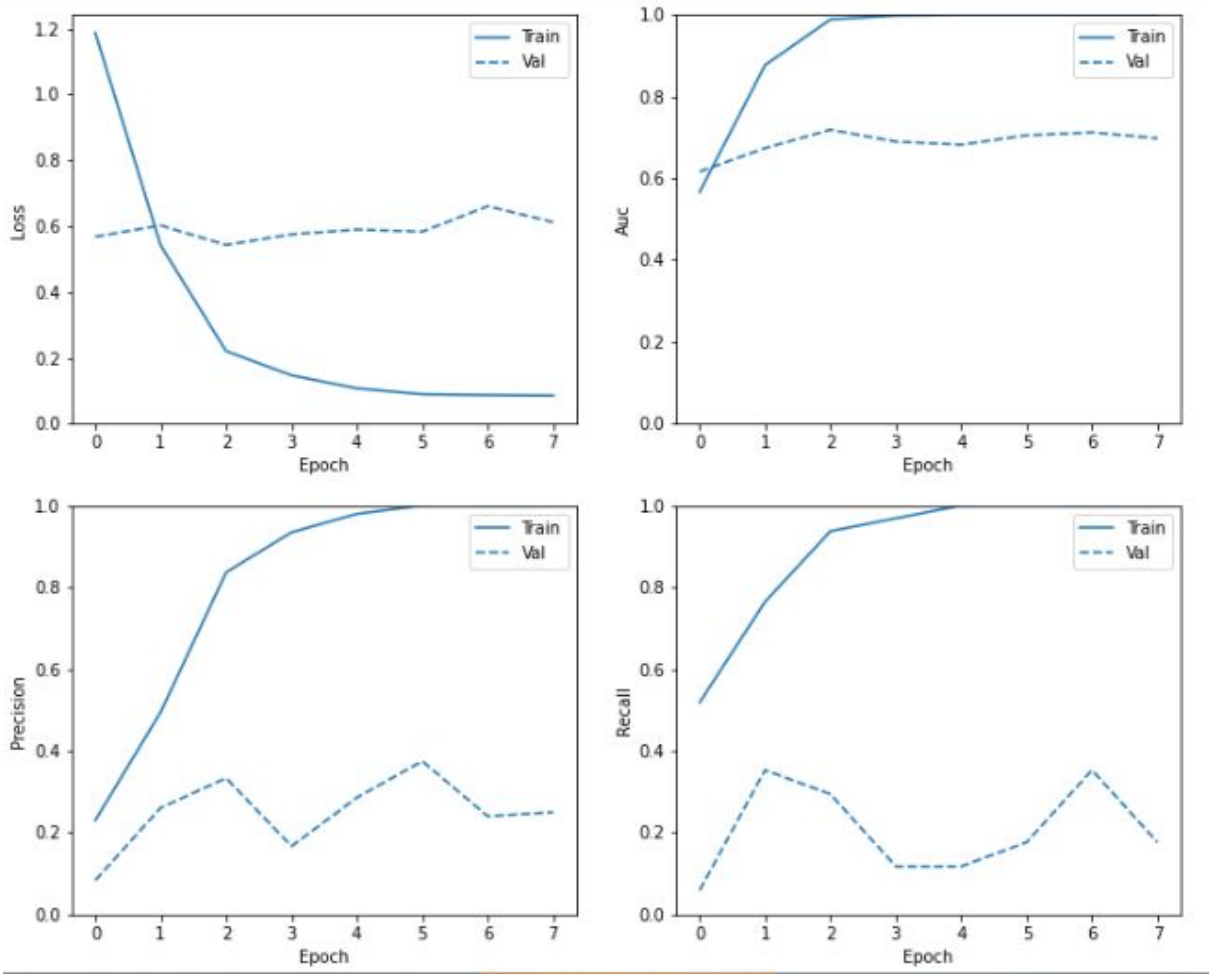
```
120/120 [==============================]
loss :  0.8144945502281189
tp :  8.0
fp :  4.0
tn :  62.0
fn :  46.0
accuracy :  0.5833333134651184
precision :  0.6666666865348816
recall :  0.14814814925193787
auc :  0.7215208411216736
```



- ACI - Axial

```
trainer.evaluate()
```

```
120/120 [==============================] - 10s 82ms/step
loss :  0.6254660487174988
tp :  27.0
fp :  4.0
tn :  62.0
fn :  27.0
accuracy :  0.7416666746139526
precision :  0.8709677457809448
recall :  0.5
auc :  0.8703703284263611
```

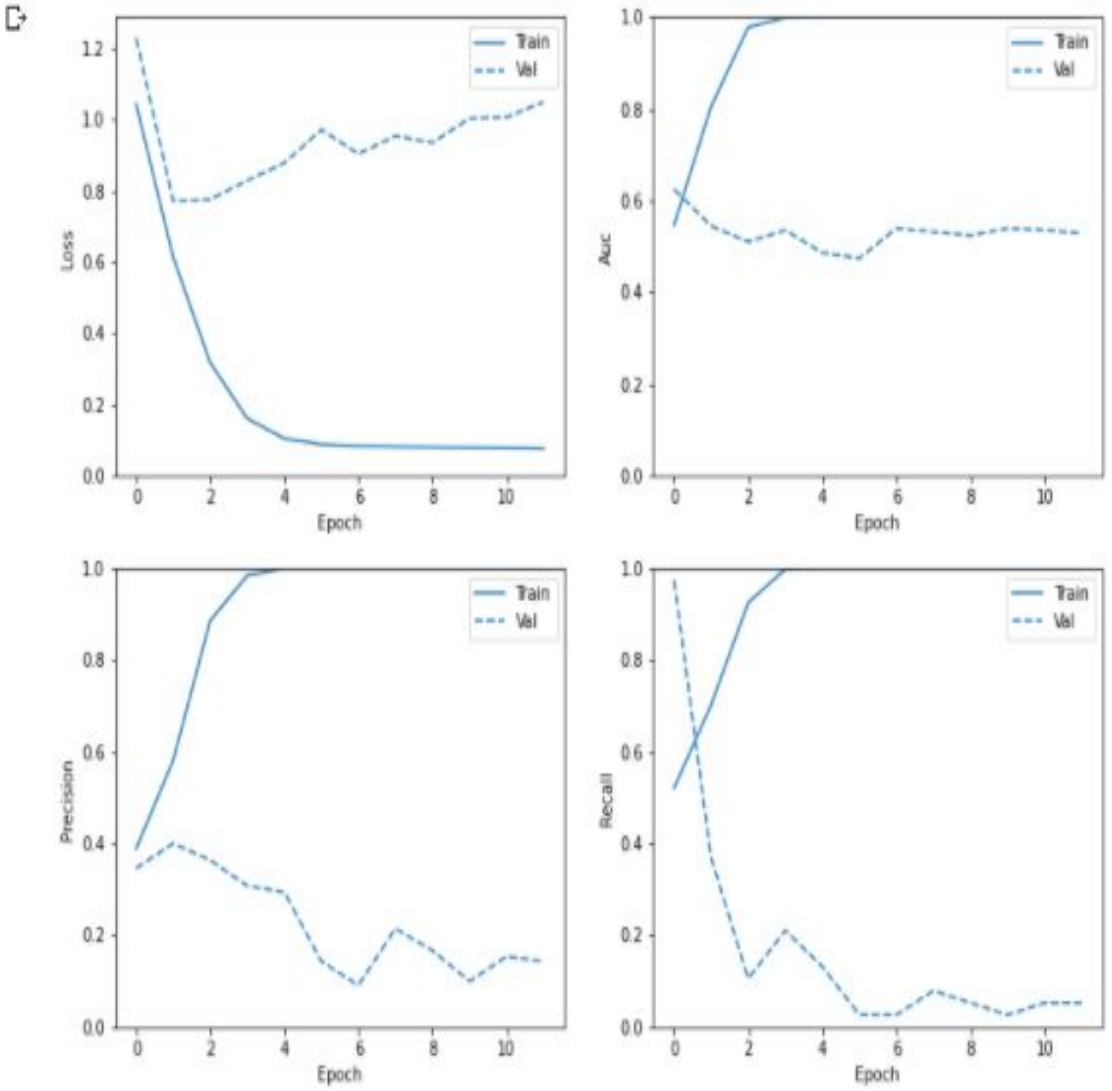


- Meniscus - Sagittal

```
[16] trainer.evaluate()
```

```
120/120 [==============================] - 9s 77ms/step
loss :  0.8061665296554565
tp :  20.0
fp :  21.0
tn :  47.0
fn :  32.0
accuracy :  0.5583333373069763
precision :  0.4878048896789551
recall :  0.38461539149284363
auc :  0.5302602052688599
```

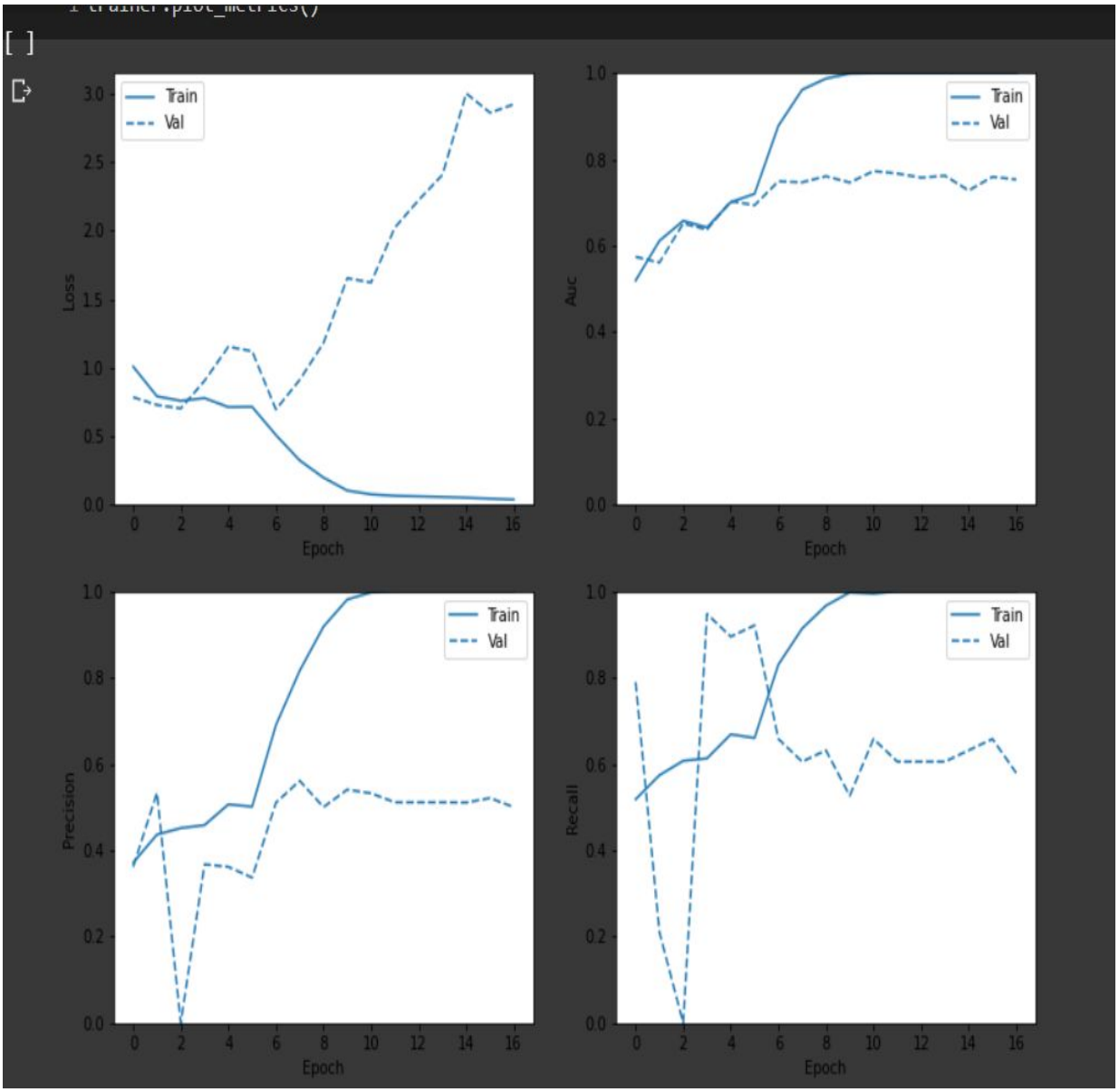
```
[17] trainer.plot_metrics()
```



- Meniscus - coronal

```
[17]  1 trainer.evaluate()

120/120 [==============================] - 7s 61ms/step - loss: 0.6979
loss :  0.6978656053543091
tp :  43.0
fp :  29.0
tn :  39.0
fn :  9.0
accuracy :  0.6833333373069763
precision :  0.5972222089767456
recall :  0.8269230723381042
auc :  0.7515553832054138
```

```
[ ]
```

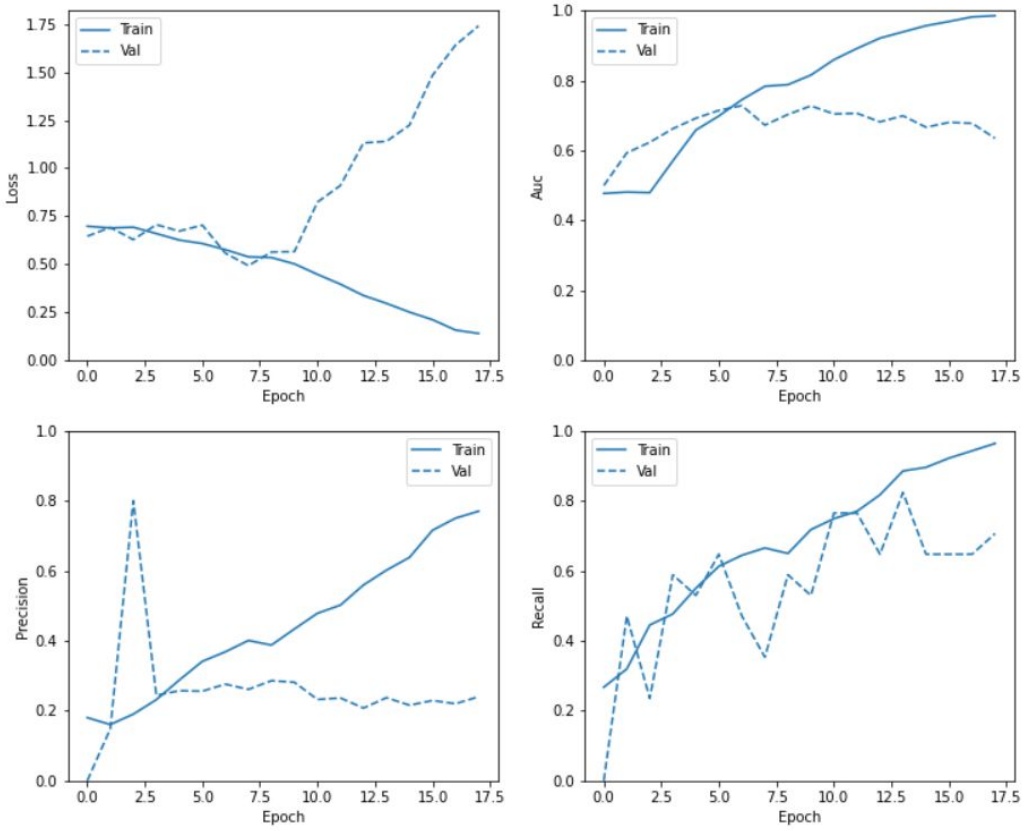


- Meniscus - Axial

```
[19]  1 trainer.evaluate()
```

```
120/120 [==============================] - 9s 73ms/step - loss: 0.5290 -
loss :  0.528984010219574
tp :  34.0
fp :  7.0
tn :  59.0
fn :  20.0
accuracy :  0.7749999761581421
precision :  0.8292682766914368
recall :  0.6296296119689941
auc :  0.8118686676025391
```

[20]

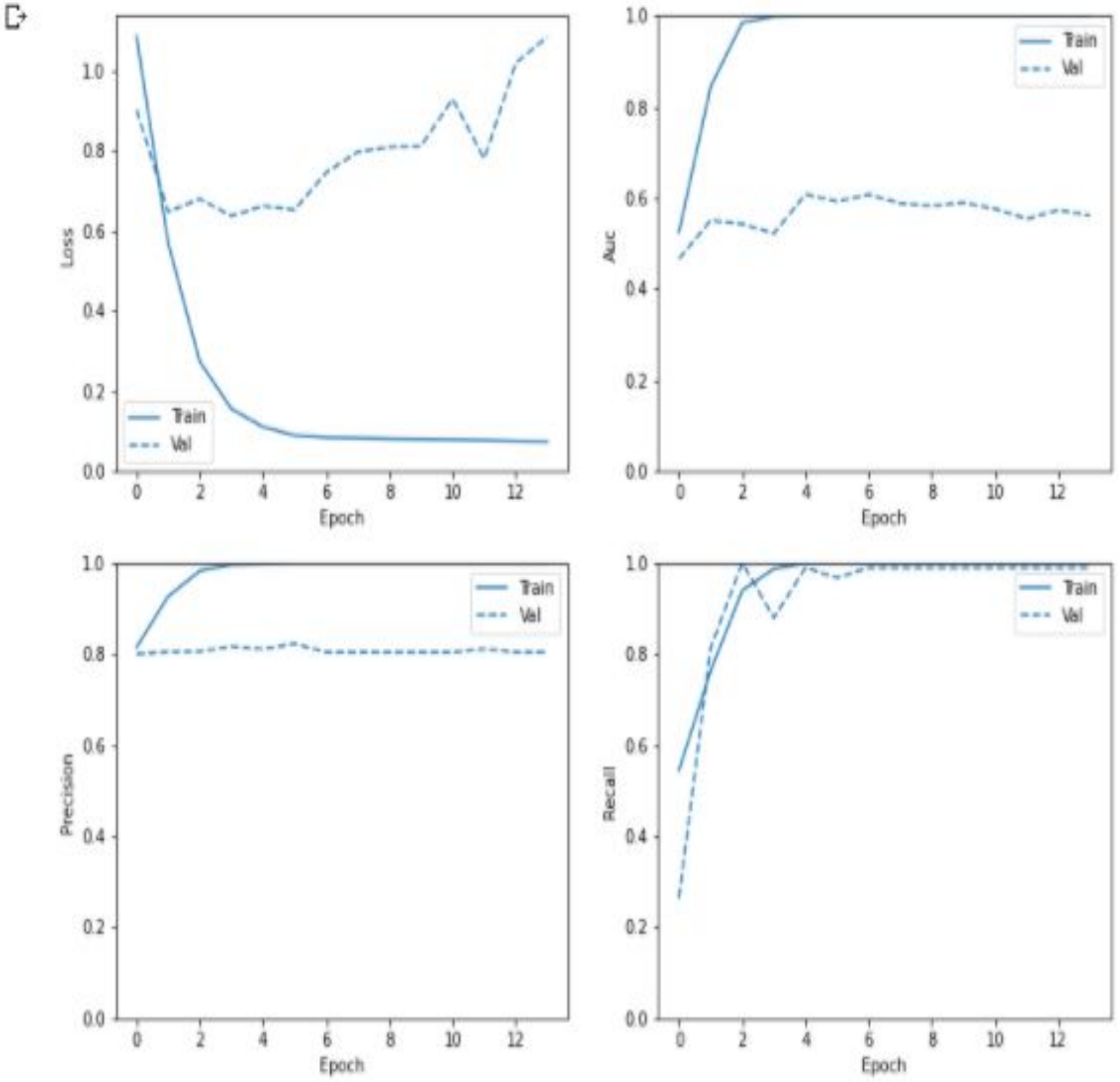


- Abnormal - Sagittal

```
[16] trainer.evaluate()
```

```
120/120 [==============================] -
loss :  0.622832715511322
tp :  85.0
fp :  21.0
tn :  4.0
fn :  10.0
accuracy :  0.7416666746139526
precision :  0.801886796951294
recall :  0.8947368264198303
auc :  0.6035789847373962
```

```
[17] trainer.plot_metrics()
```

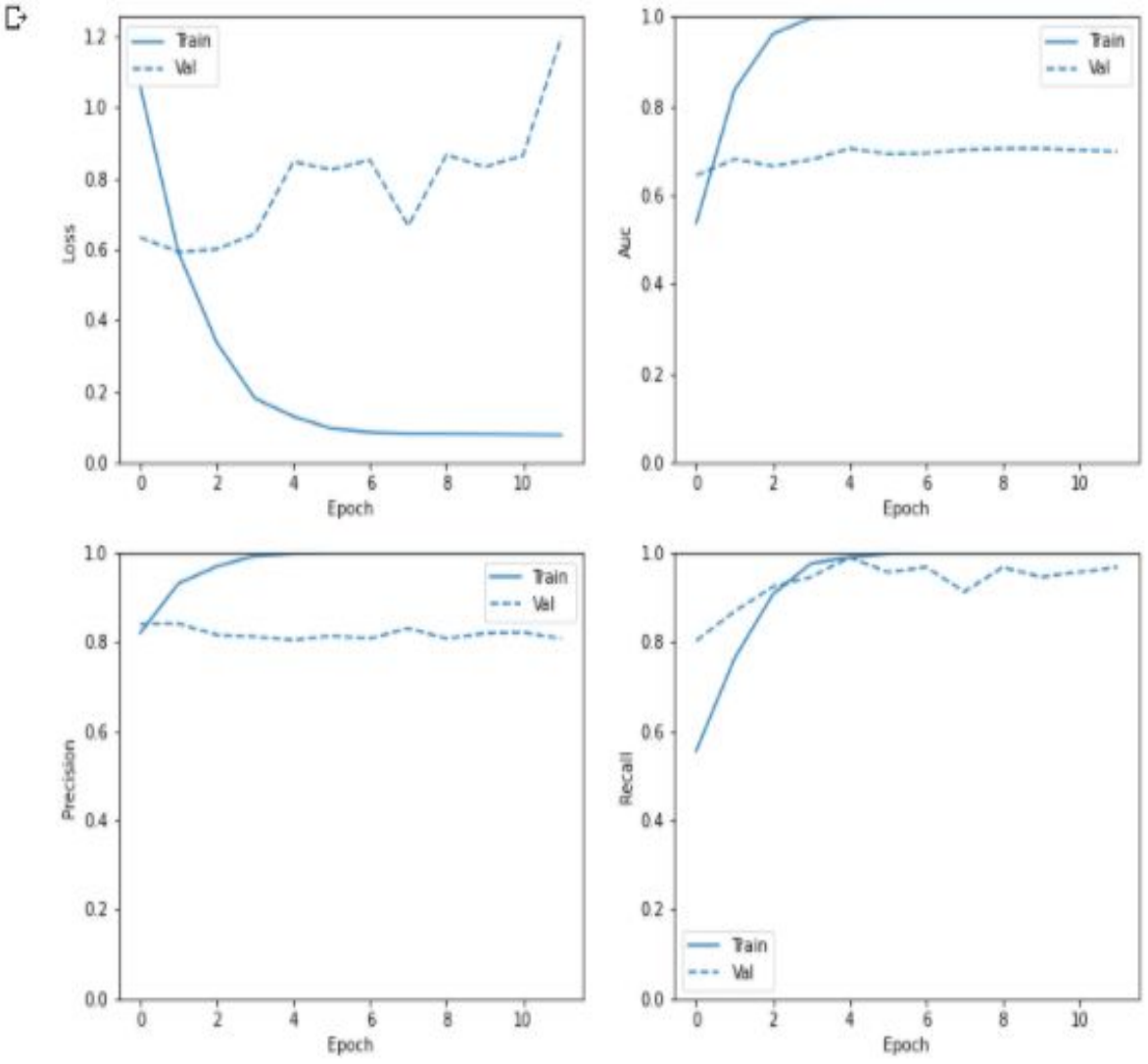


- Abnormal - Coronal

```
[16] trainer.evaluate()
```

```
120/120 [==============================] - 2
loss :  0.6065647006034851
tp :  86.0
fp :  20.0
tn :  5.0
fn :  9.0
accuracy :  0.7583333253860474
precision :  0.8113207817077637
recall :  0.9052631855010986
auc :  0.6614736914634705
```

```
[17] trainer.plot_metrics()
```

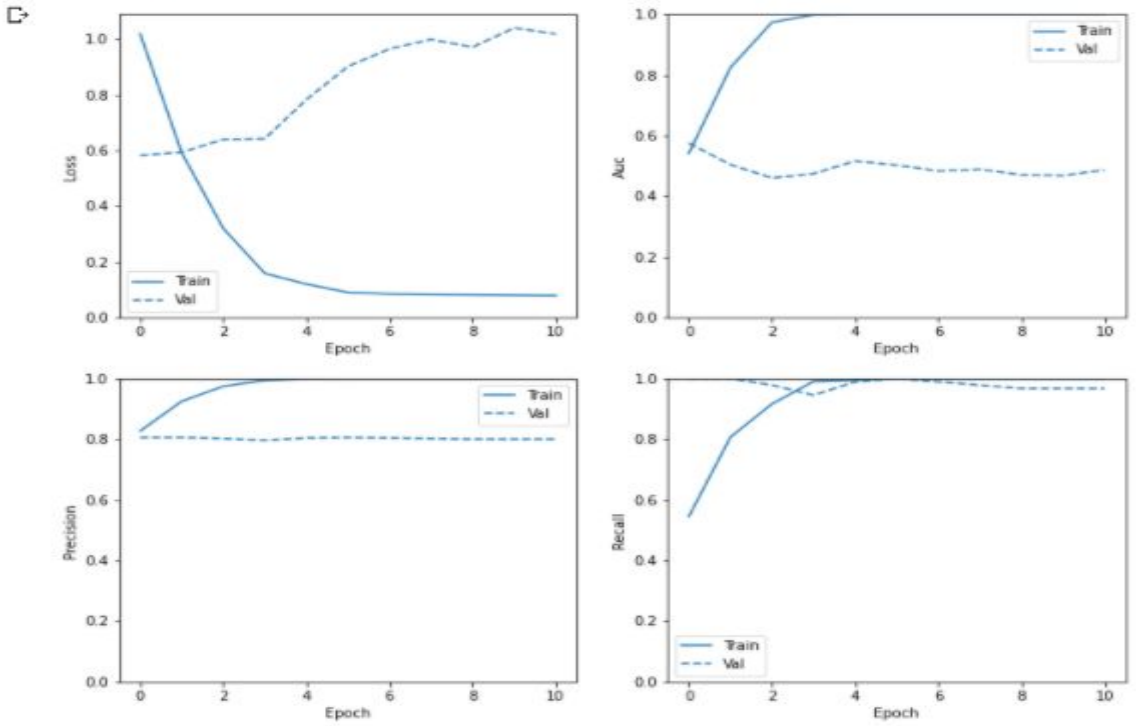


- Abnormal - Axial

```
[16] trainer.evaluate()
```

```
120/120 [==============================] -
loss :  0.6643935441970825
tp :  95.0
fp :  25.0
tn :  0.0
fn :  0.0
accuracy :  0.7916666865348816
precision :  0.7916666865348816
recall :  1.0
auc :  0.4648420810699463
```

```
[17] trainer.plot_metrics()
```



- **Logistic regression**

Points of comparisons	Inception V3	
	Loss	Auc
Acl	0.6282893419265747	0.8787879347801208
Meniscus	0.6265704035758972	0.6751980185508728
Abnormal	0.5122967958450317	0.5612631440162659

Paper Summary:

- The goal of this project is to form a model that can determine the disease in a human knee, since the diagnoses in some time may be hard and result errors to the patients, there is a deep learning model to detect diagnoses on exams.
- Our dataset consisted of 1,370 knee MRI exams performed at Stanford University Medical Center (mean age 38.0 years; 569 [41.5%] female patients).
- We developed MRNet, a convolutional neural network for classifying MRI series and combined predictions from 3 series per exam using logistic regression.
- The exam was about three types of diseases: acl, meniscus and abnormal.
- The exam was of three types of views: sagittal, coronal or axial.
- Using a 2-sided Pearson's chi-squared test with adjustment for multiple comparisons, we found no significant differences between the performance of the model and that of unassisted general radiologists in detecting abnormalities.
- The primary limitations of our study include lack of surgical ground truth and the small size of the panel of clinical experts.
- The deep learning model results in accurate classification of exams and results of model improve execution of clinical experts therapy.
- **External validation:** the authors had extracted labels for 3 levels of ACL disease: non-injured (690 exams), partially injured (172 exams), and completely ruptured (55 exams). At the beginning we applied MRNet without retraining on the external data, then subsequently optimized MRNet using the external training and tuning sets. Classification task was to discriminate between non-injured ACLs and injured ACLs.
- **Model**: there are four steps to move on (preprocessing - MRNet - MRNet interpretation - Combining MRNet predictions).
    - ***Preprocessing***:
        - Convert images from DICOM to PNG and scale them to 256 X 256 pixels using Python.
        - Some transformation done on the pixel intensities and regulate these intensities by (training, tuning, and validation).
        - At the end all the pixel values were between 0 and 255 as it is the standard for the .PNG images.

- ***MRNet***:
  - Building block which is CNN maps 3D MRI series.
  - Input dimension (*s (number of images) x 3 (number of color channels) x 256 x 256*).
  - Each 2D image enters AlexNet to obtain (*s x 256 x 7 x 7*) tensor with the features of each image.
  - Apply global average pooling layer to reduce these features to *s x 256*, then apply max pooling to obtain 256D vector, then passed through FCL and activation (sigmoid function) to obtain prediction in 0 to 1 range, then optimized using binary cross-entropy.
  - Apply backpropagation algorithm to compute gradient of loss while training and each training example rotated, shifted, and flipped.
  - Evaluate the model with lowest average loss from the saved model parameters after every full pass during training.
  - We initialized weights of AlexNet to values optimized on ImageNet dataset as the training from scratch needs large dataset then fine-tuned these weights to fit MRI dataset, this allows earlier difficult layers to optimize to recognize features like lines and edges.
- ***MRNet interpretation*:**
  - Generate class activation mappings (CAMs) to ensure the model learned the relevant features by computing weighted average through 256 CNN maps using weights from the classification layer to obtain a 7 x 7 image.
  - The CAM mapped to unsampled 256 x 256 pixel mist with original input image.
  - Parameters from the final layer are used to weight the feature map, so, more predictive feature maps appear brighter which are with higher effect of model's prediction.
- **Combining MRNet predictions:**
  - Train a logistic regression to weight the predictions from the 3 series and generate a single output for each exam.
  - The most beneficial series, determined from the coefficients of the fitted logistic regression.

- The logistic regression was applied to the predictions of the 3 MRNets for the internal validation set to obtain the final predictions.
- These models were implemented in Python.
- use the prediction from a single MRNet directly as the final output.

- **Evaluation:** Definitions for labels were as follows:
    - Abnormality: normal (all images reviewed are free of abnormalities) or abnormal (the abnormal findings in the internal validation set that were not ACL tear or meniscal tear included osteoarthritis, effusion, iliotibial band syndrome, posterior cruciate ligament tear, fracture, contusion, plica, and medial collateral ligament sprain).
    - ACL: intact (normal, mucoid degeneration, ganglion cyst, sprain) or tear (low-grade partial tear with <50% of fibers torn, high-grade partial tear with >50% of fibers torn, complete tear.
    - Meniscus: intact (normal, degenerative changes without tear, postsurgical changes without tear) or tear (increased signal reaching the articular surface on at least 2 slices or morphologic deformity).
- **Statistical methods:** We computed the micro-average of these statistics across general radiologists only and across all clinical experts, we performed multiple comparisons in this study to assess the model's performance against that of the practicing general radiologists and also to assess the clinical utility of providing model assistance, To assess model performance against that of general radiologists, we used a 2-sided Pearson's chi-squared test to evaluate whether there were significant differences in specificity. A robust hypothesis tests have been performed to assess if the clinical experts demonstrated statistically significant improvement with model assistance.All statistical analyses were completed in the R environment for statistical computing, using the irr, pROC, binom, and qvalue packages, and R code was provided with submission.

Statistic	Training	Tuning	Validation		
			All	Prospective labels[1]	Reference standard labels[2]
Number of exams	1,130	120	120		
Number of patients	1,088[3]	111	113		
Number of female patients (%)	480 (42.5)	50 (41.7)	39 (32.5)		
Age, mean (SD)	38.3 (16.9)	36.3 (16.9)	37.1 (14.8)		
Number with abnormality (%)	913 (80.8)	95 (79.2)		96 (80.0)	99 (82.5)
Number with ACL tear (%)	208 (18.4)	54 (45.0)		57 (47.5)	58 (48.3)
Number with meniscal tear (%)	397 (35.1)	52 (43.3)		59 (49.2)	65 (54.2)
Number with ACL and meniscal tear (%)	125 (11.1)	31 (25.8)		38 (31.7)	40 (33.3)

- **Results:** the arrangement was s 0.508 for detecting abnormalities, 0.800 for detecting ACL tears, and 0.745 for detecting meniscal tears.
- **Model performance:**
    - For abnormality detection: ACL tear detection, and meniscal tear detection, the model achieved AUCs of 0.937, 0.965, and 0.847, respectively.
    - The model was highly specific for ACL tear detection, achieving a specificity of 0.968, which is higher than the micro-average of general radiologists, at 0.933, but this difference was not statistically significant.
- **Clinical utility of model assistance:** Fleiss kappa was computed for the 9 clinical experts with and without model assistance, and while we did not assess statistical significance, we observed that model assistance increased the Fleiss kappa measure of inter-rater reliability for all 3 tasks. With model assistance, the Fleiss kappa measure for abnormality detection increased from 0.571 to 0.640, for ACL tear detection it increased from 0.754 to 0.840, and for meniscal tear detection it increased from 0.526 to 0.621.

Colab Links:

Dataset loader:

https://colab.research.google.com/drive/1BCG8M_BrNHDuyowb9j5yMkcJeRCSVS7a?usp=sharing

Models:

https://colab.research.google.com/drive/1NwrdSavpvAU-iOvIeEvrOb56gZbqzmgM?usp=sharing

Trainer:

https://colab.research.google.com/drive/1D_YI88uDFj3eZvjkyndLmLgpBguqkUxg?usp=sharing

Logistic regression trainer:

https://colab.research.google.com/drive/1Woppj5rgm6ncvx2cPHG_ZXLKCYPPgtxu?usp=sharing

Individual Reports:

1- Arsany Atef Abdo:

- **Building ResNet 50 model.**
- **Summarizing the online paper.**
- **Training models.**

2- Kirellos Malak Habib:

- **Logistic regression implementation**
- **Summarizing the online paper.**
- **Training models.**

3- Michael Said Beshara:

- **Dataset preparation implementation.**
- **Training models implementation.**
- **Training models.**

4- Nada Salama:

- **Building Inception v3 model.**
- **Training models.**

5- Yomna Gamal:

- **Building VGG16 model.**
- **Training models.**