گزارش کار

شبکه های عصبی

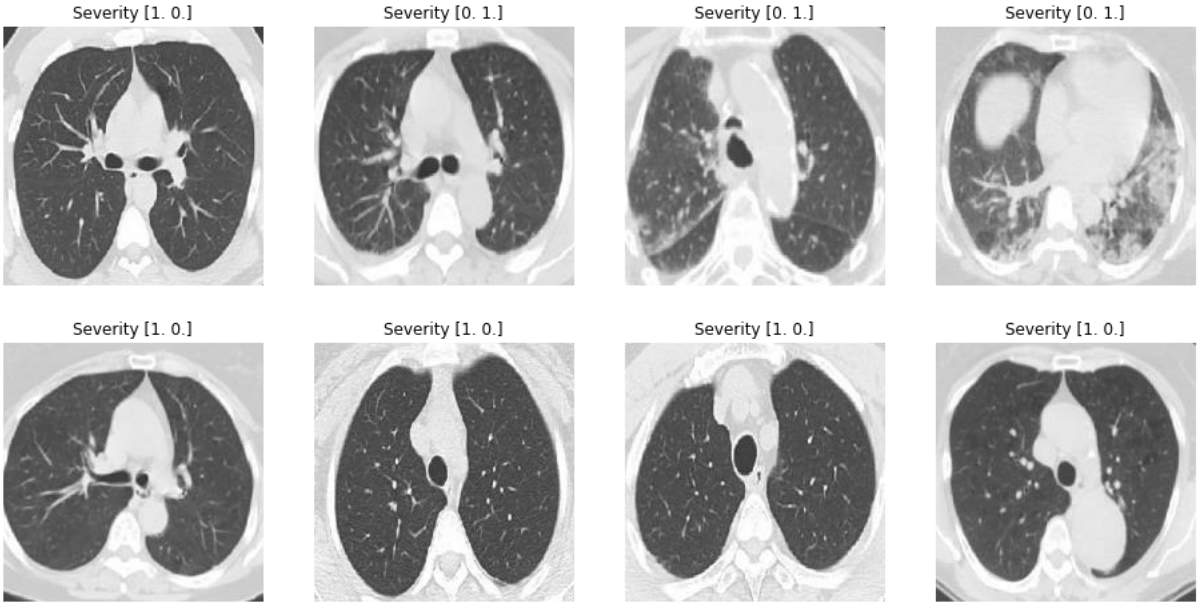
تمرین چهارم

آرمین خیاطی

9931153

فهرست

توضیحات



در این پروژه طبق موارد خواسته شده در فایل پروژه، تصاویر را به سایز 256*256 تغییر سایز داده و پیکسل ها را بین صفر و یک نرمال میکنیم. Augmentation هایی که در فاز ترین بخش اول پروژه استفاده شده به قرار زیر است. در پارت دوم Augmentation ای استفاده نشده.

```
rotation_range=20,
width_shift_range=0.1,
height_shift_range=0.1,
rescale=1./255,
shear_range=0.2,
zoom_range=[0.7, 1.0],
horizontal_flip=True,
vertical_flip=True,
fill_mode="constant",
cval=0.0
```

معماری پارت اول:

مدل پایه ای که برای بخش اول پروژه و دسته بندی تصاویر استفاده شده Resnet-50 است که در ادامه آن 2 لایه Dense به سایز 64 و 32 اضافه شده.

```
base_model = ResNet50(weights='imagenet', input_shape=input_shape, include_top=False)
base_model.trainable = True
base_model_input = base_model.input
x = GlobalAveragePooling2D()(base_model.output)
x = Dense(64, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
model = Dense(2, activation='softmax')(x)
model = Model(base_model_input, model)
model.compile(optimizer = Adam(), loss = 'categorical_crossentropy',
                           metrics = ['categorical_accuracy'])
```

معماری پارت دوم:

معماری Generator به ترتیب شامل یک لایه Dense به سایز 128*256*256 و چند لایه Conv2dTranspose به سایز های [128, 64, 32, 16, 8, 3] با تابع فعالساز LeakyRelu در لایه های میانی و تابع فعالساز سیگموید در لایه آخر می باشد.

```
def generator_(noise, labels, image_size):
  num_layers = int(np.log2(image_size)) - 1
  layer_filters = np.flip([2**i if 2**i > 4 else 3 for i in range(2, num_layers + 1) ]) # [128,  64,  32,  16,   8,  3]
  image_resize = image_size // 4
  inp = [noise, labels]
  x = concatenate(inp, axis=1)
  x = Dense(image_resize * image_resize * layer_filters[0])(x)
  x = Reshape((image_resize, image_resize, layer_filters[0]))(x)
  kernel_size = 5
  for f in layer_filters:
    strides = 2 if f > 32 else 1
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2DTranspose(filters=f, kernel_size=kernel_size, strides=strides, padding='same')(x)
  x = Activation('sigmoid')(x)
  return Model(inp, x, name='generator')
```

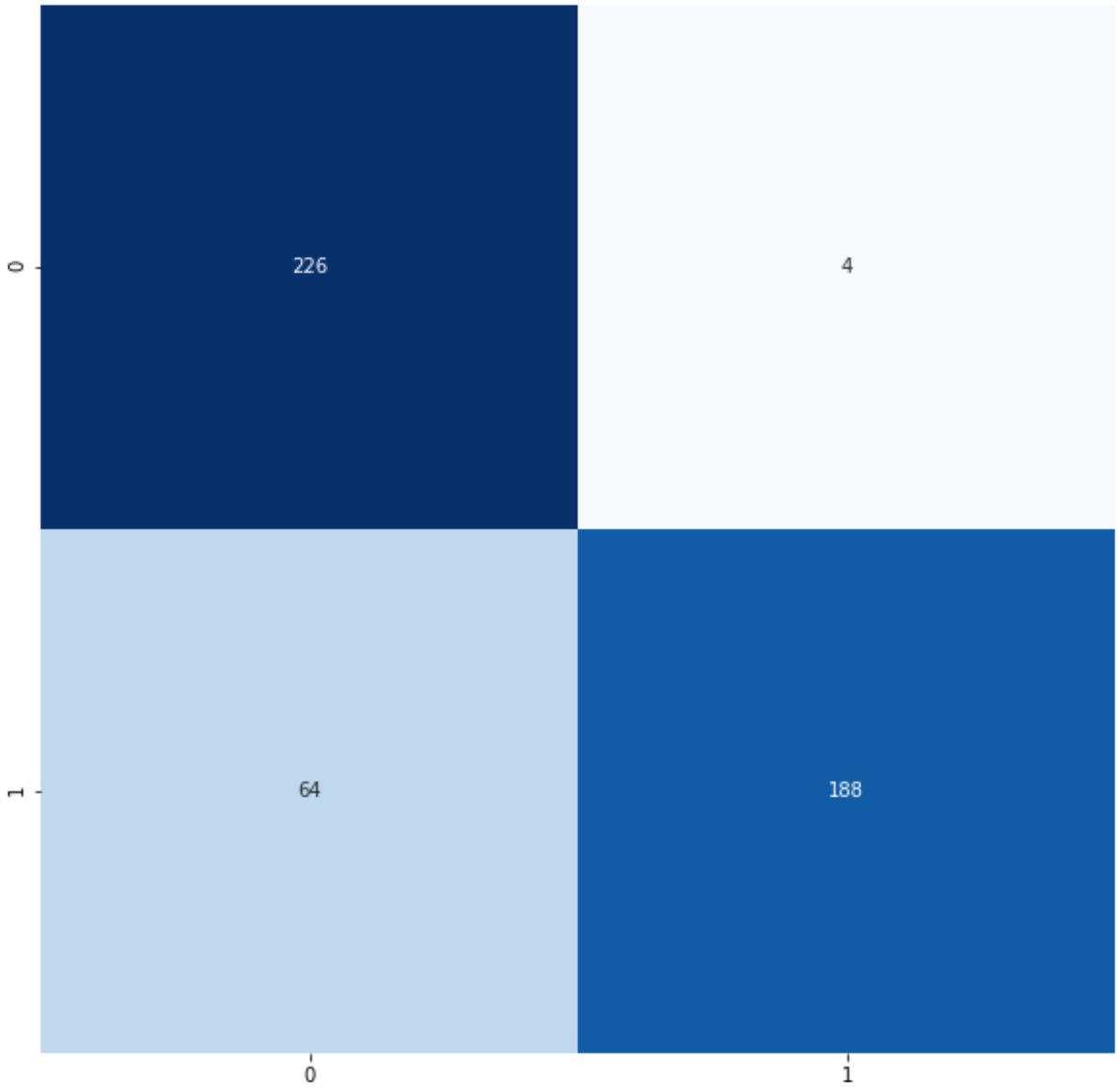
معماری Discriminator نیز به ترتیب شامل چند لایه Conv2D به سایز [16, 32, 64, 128, 256] با تابع فعالساز LeakyRelu در لایه های میانی است. برای تعیین Fake یا Real بودن تصاویر یک لایه Dense به سایز 1 و تابع فعالساز Sigmoid و برای برچسب تصویر نیز دو لایه Dense به ترتیب به سایز 128 و 64 قرار دادیم.

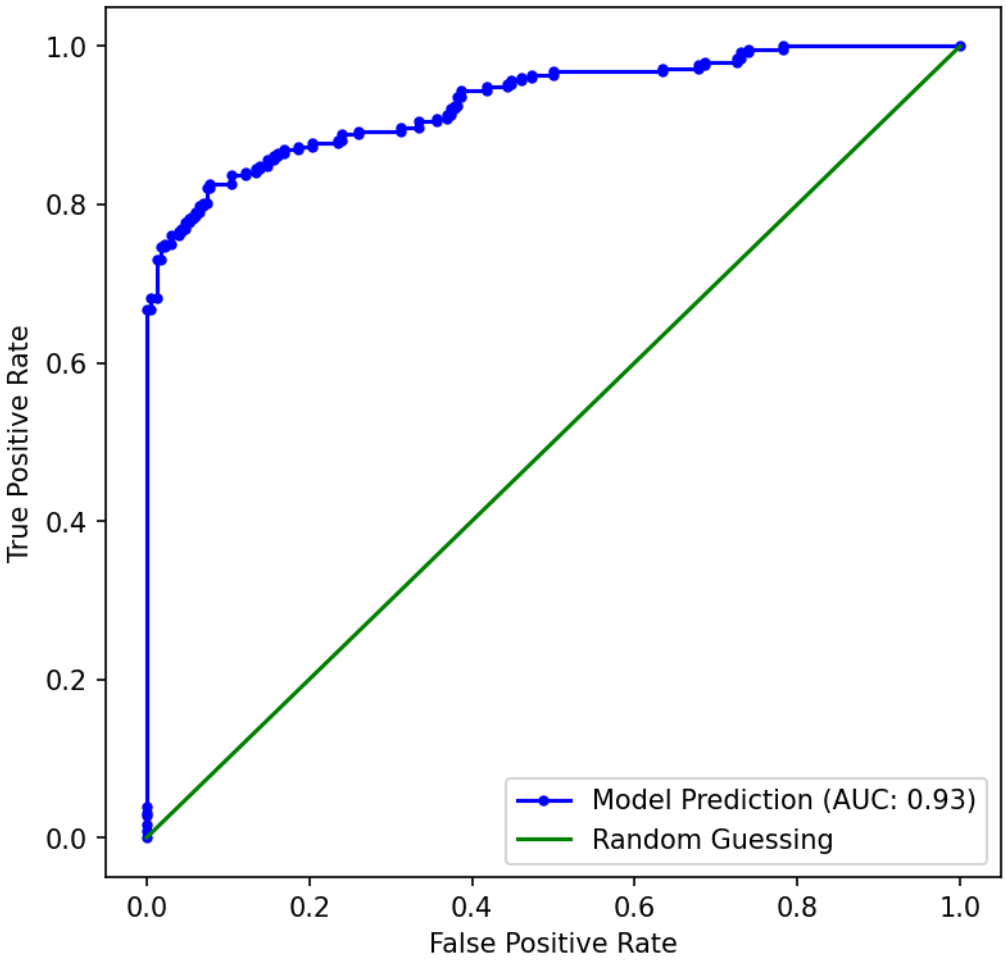
```
def discriminator_(inputs, num_labels, image_size):
  kernel_size = 5
  num_layers = int(np.log2(image_size))
  layer_filters = [2**i  for i in range(2, num_layers + 1) if 2**i > 8 ] # [16, 32, 64, 128, 256]
  x = inputs
  for f in layer_filters:
    strides = 2 if f < 128 else 1
    x = LeakyReLU(alpha=0.2)(x)
    x = Conv2D(filters=f, kernel_size=kernel_size, strides=strides, padding='same')(x)

  x = Flatten()(x)
  # probability that the image is real
  prob = Dense(1)(x)
  prob = Activation('sigmoid', name='Real_Fake')(prob)

  # Classification part
  layer = Dense(128)(x)
  layer = Dense(64)(layer)
  labels = Dense(num_labels)(layer)
  labels = Activation('softmax', name='label')(labels)

  # Concat Outputs
  outputs = [prob, labels]
  return Model(inputs, outputs, name='discriminator')
```

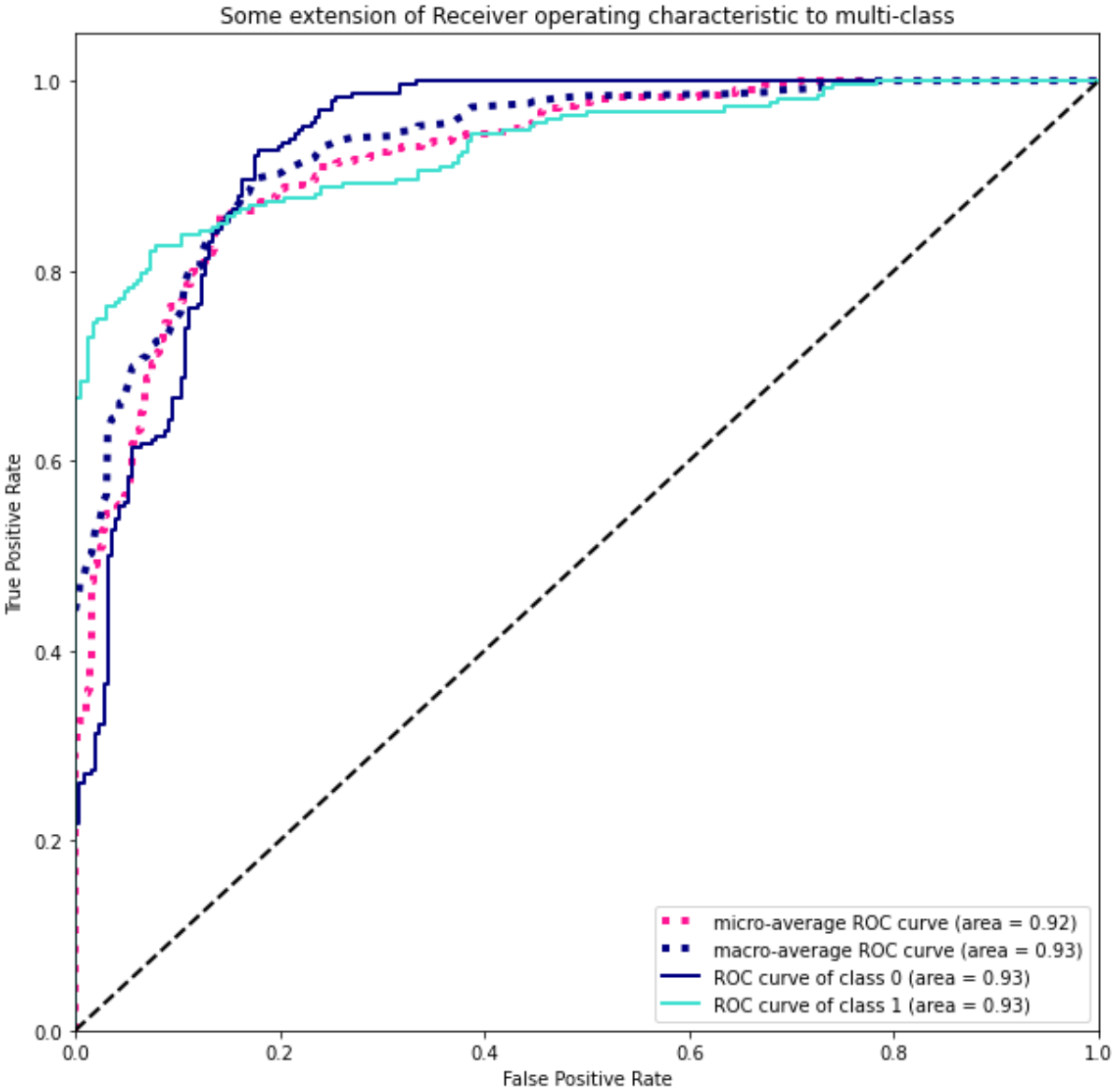
نتایج بخش اول

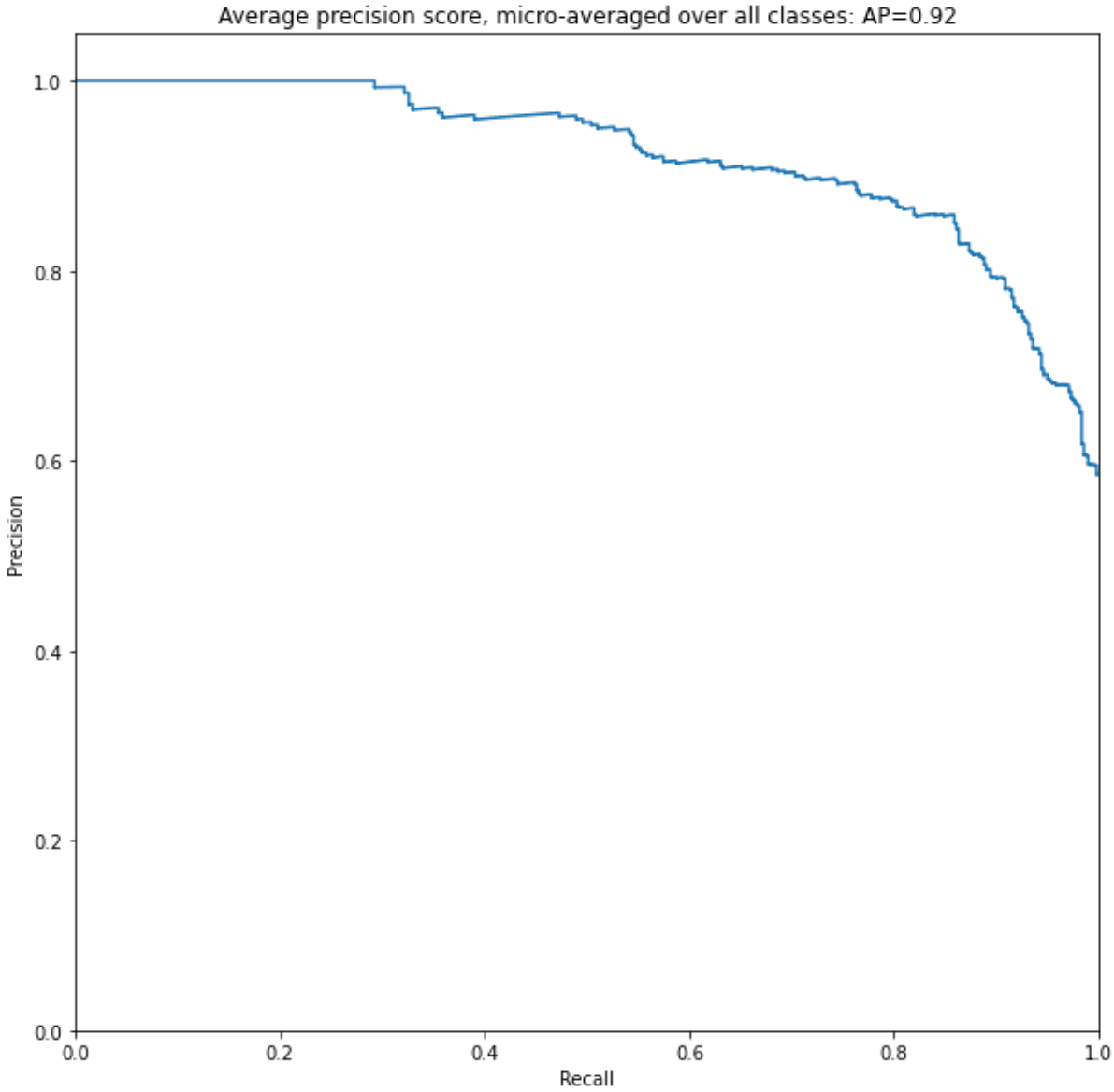
برای این بخش چندین مدل را چندین بار با پارامتر های مختلف تست کردیم. که بهترین آن این نتایج را به دنبال داشت.

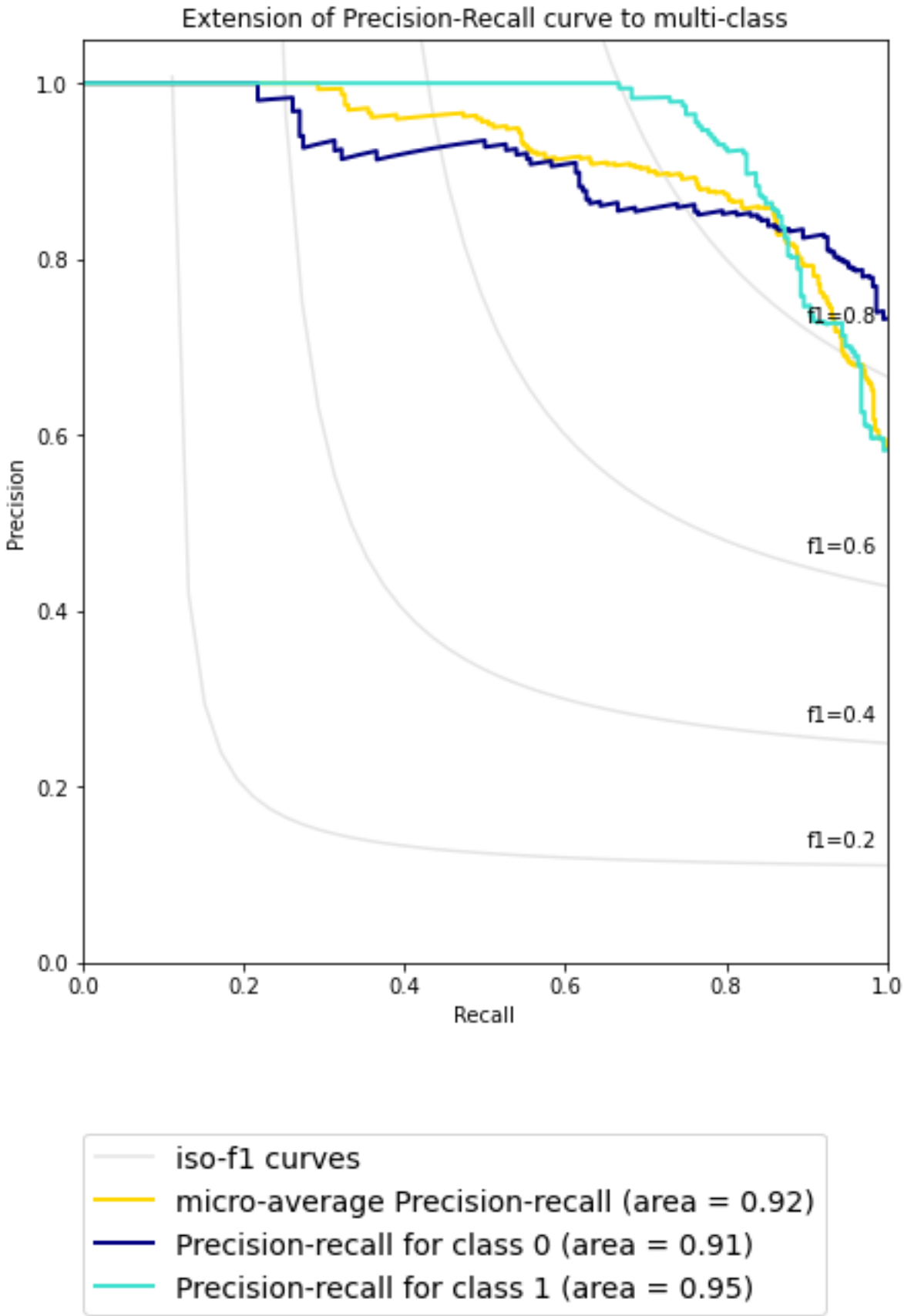
```
31/31 [==============================] - 4s 145ms/step
Accuracy on Test Data: 0.86%
              precision    recall  f1-score   support

           0       0.78      0.98      0.87       230
           1       0.98      0.75      0.85       252

    accuracy                           0.86       482
   macro avg       0.88      0.86      0.86       482
weighted avg       0.88      0.86      0.86       482
```




1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
Model Prediction (AUC: 0.93)
Random Guessing

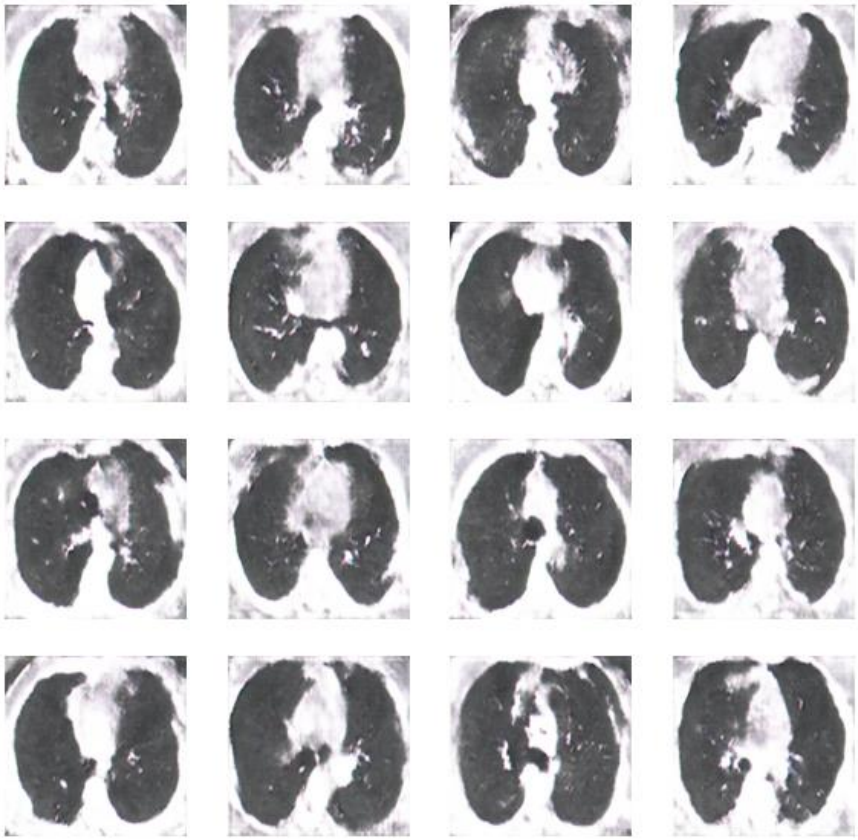

Some extension of Receiver operating characteristic to multi-class
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 0.92)
macro-average ROC curve (area = 0.93)
ROC curve of class 0 (area = 0.93)
ROC curve of class 1 (area = 0.93)


Average precision score, micro-averaged over all classes: AP=0.92
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Extension of Precision-Recall curve to multi-class
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
f1=0.8
f1=0.6
f1=0.4
f1=0.2
iso-f1 curves
micro-average Precision-recall (area = 0.92)
Precision-recall for class 0 (area = 0.91)
Precision-recall for class 1 (area = 0.95)

نتایج بخش دوم

در این بخش ابتدا یک مدل ACGAN برای تولید تصاویر سی تی اسکن ریه ترین میکنیم که خروجی آن بصورت زیر است. یک فایل GIF نیز همراه پروژه ارسال شده که روند بهبود تصاویر تولیدی توسط GAN را نمایش میدهد.

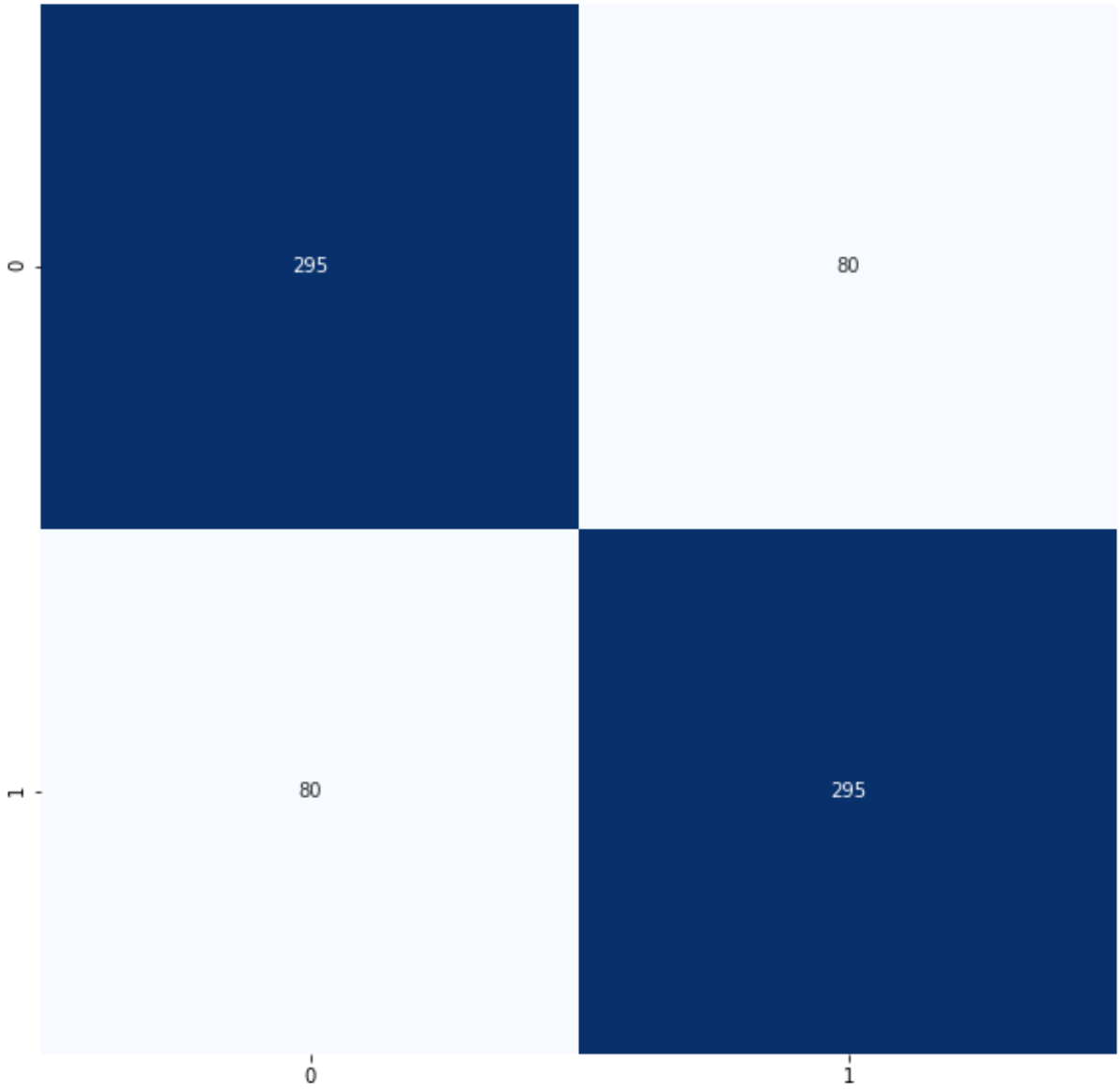


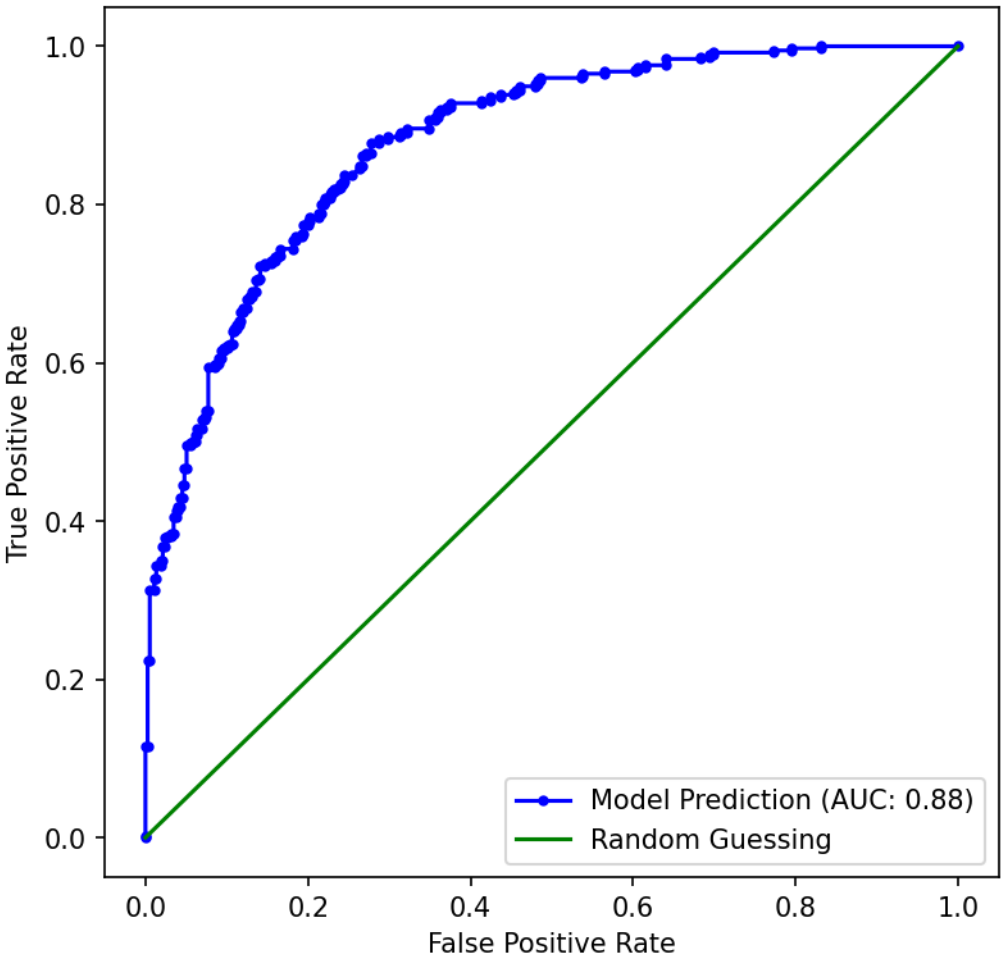
Test Classifier on Generated images

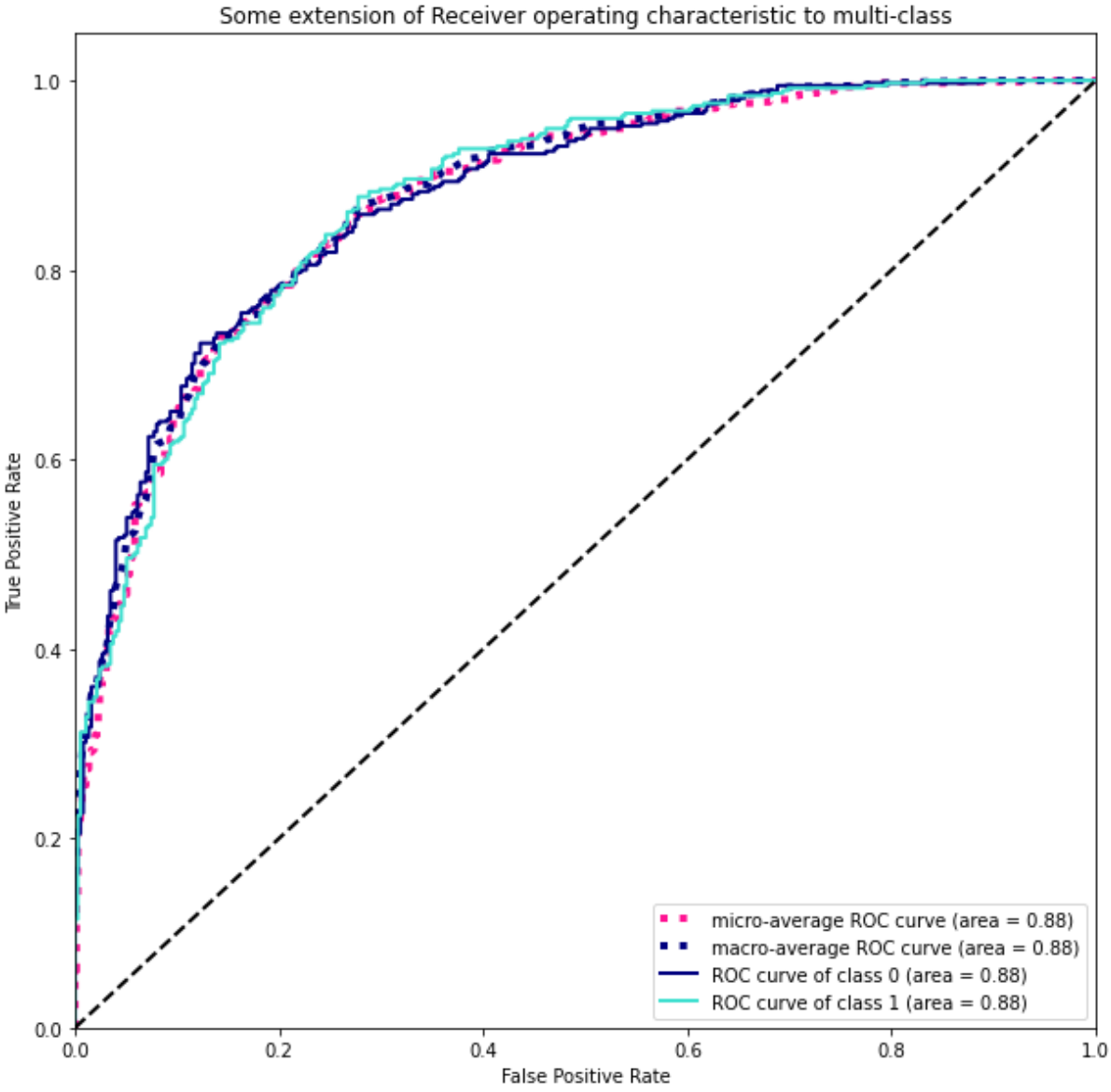
در ادامه مدلی که در بخش قبلی برای کلاسیفای کردن تصاویر ترین کردیم را روی 750 تصویر تولیدی تست میکنیم که نتایج زیر حاصل میشود.

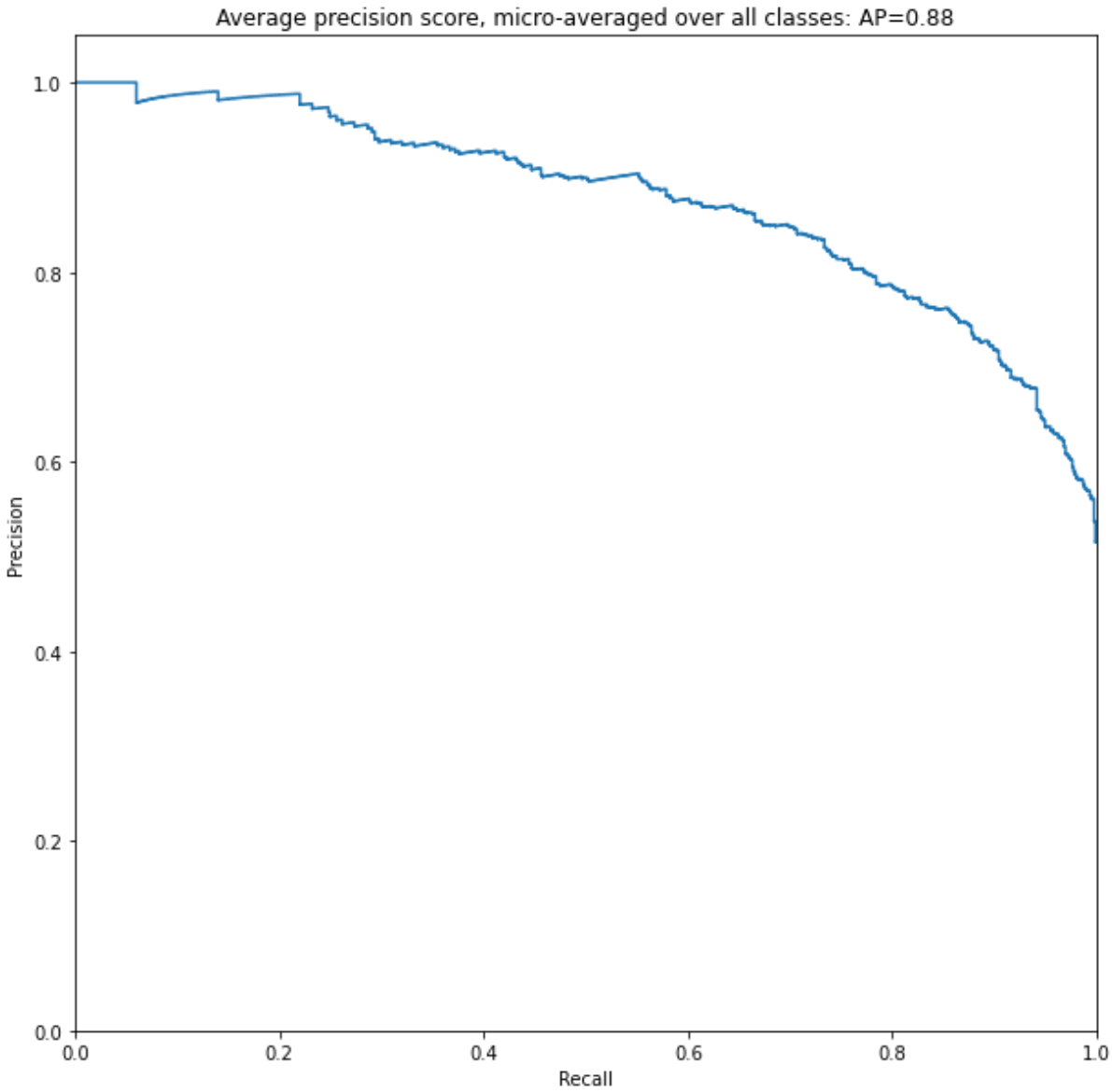
Accuracy on Test Data: 0.79%

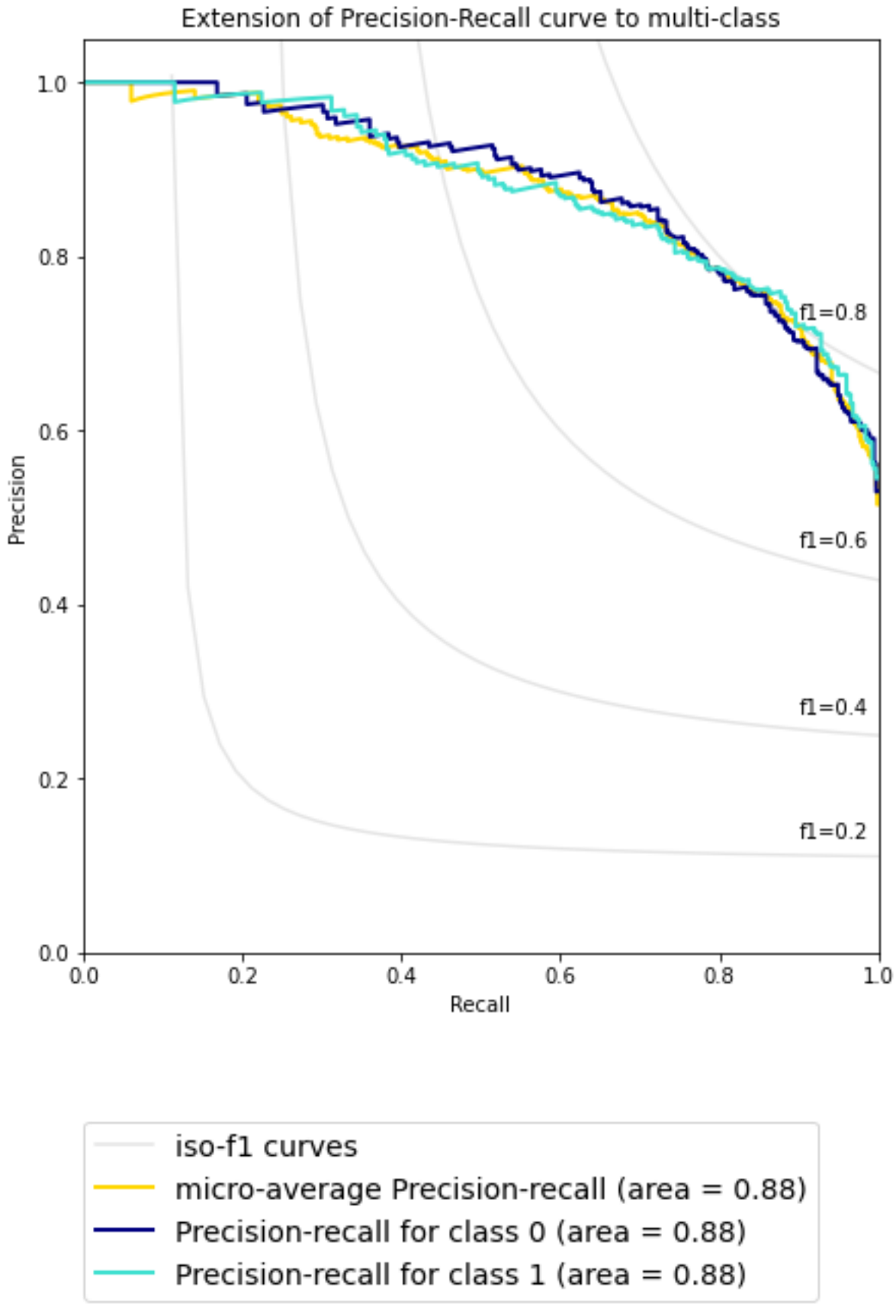
	precision	recall	f1-score	support
0	0.79	0.79	0.79	375
1	0.79	0.79	0.79	375
accuracy			0.79	750
macro avg	0.79	0.79	0.79	750
weighted avg	0.79	0.79	0.79	750


0
1
295
80
80
295
0
1


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
Model Prediction (AUC: 0.88)
Random Guessing


Some extension of Receiver operating characteristic to multi-class
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 0.88)
macro-average ROC curve (area = 0.88)
ROC curve of class 0 (area = 0.88)
ROC curve of class 1 (area = 0.88)

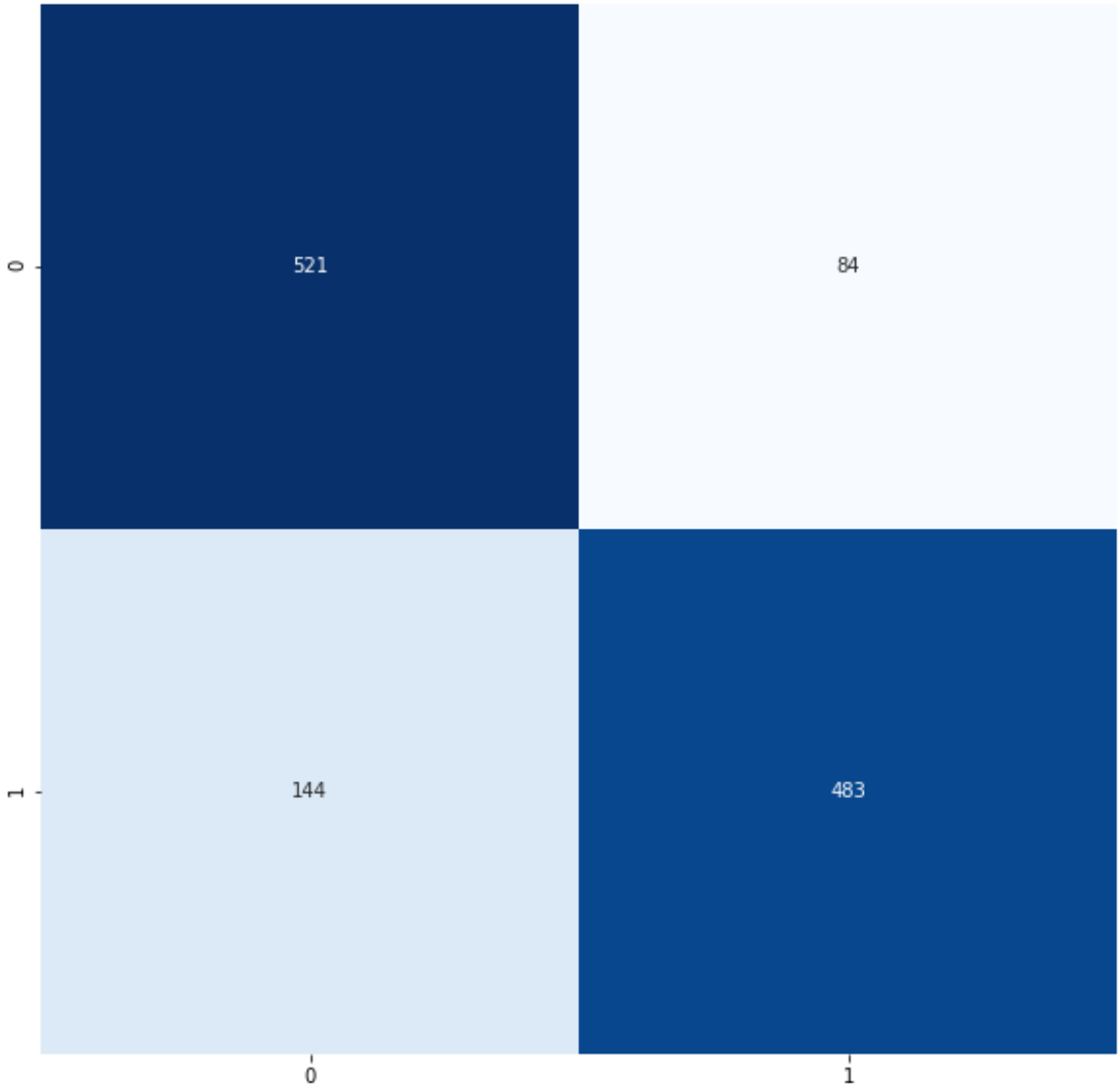

Average precision score, micro-averaged over all classes: AP=0.88
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0

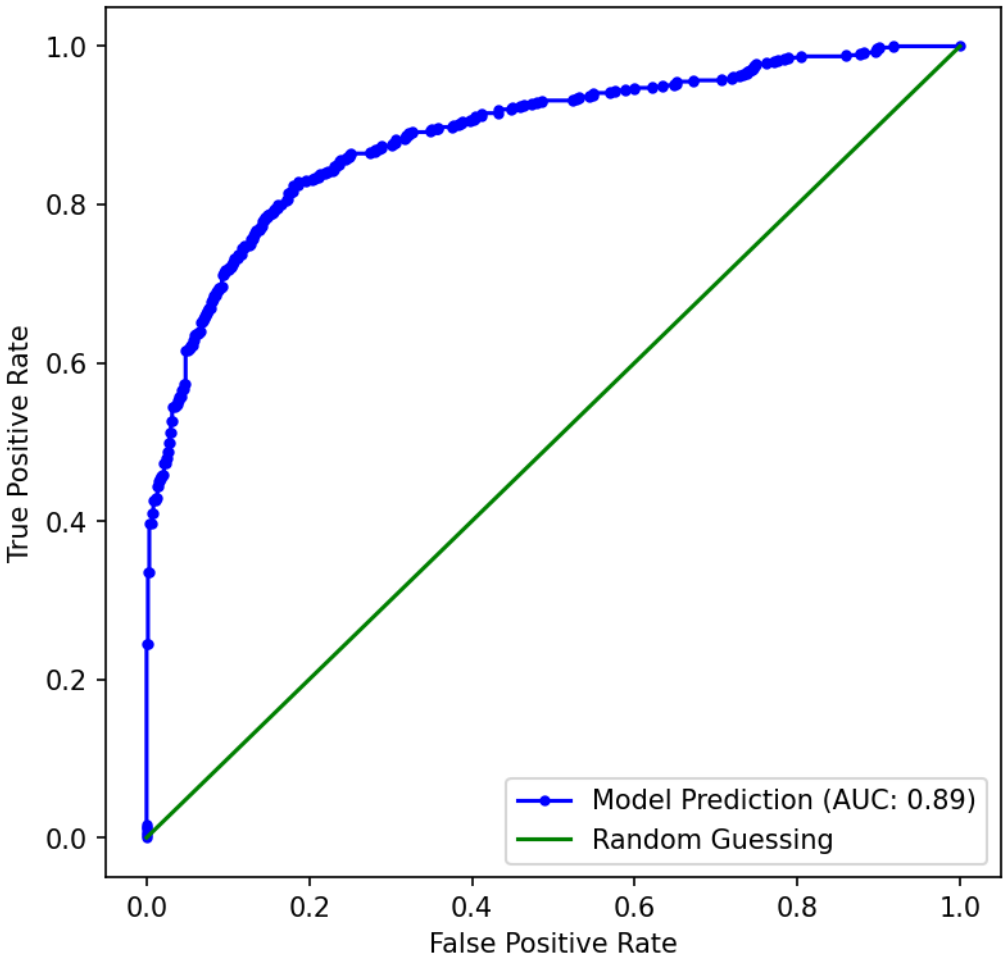

Extension of Precision-Recall curve to multi-class
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
f1=0.8
f1=0.6
f1=0.4
f1=0.2
iso-f1 curves
micro-average Precision-recall (area = 0.88)
Precision-recall for class 0 (area = 0.88)
Precision-recall for class 1 (area = 0.88)

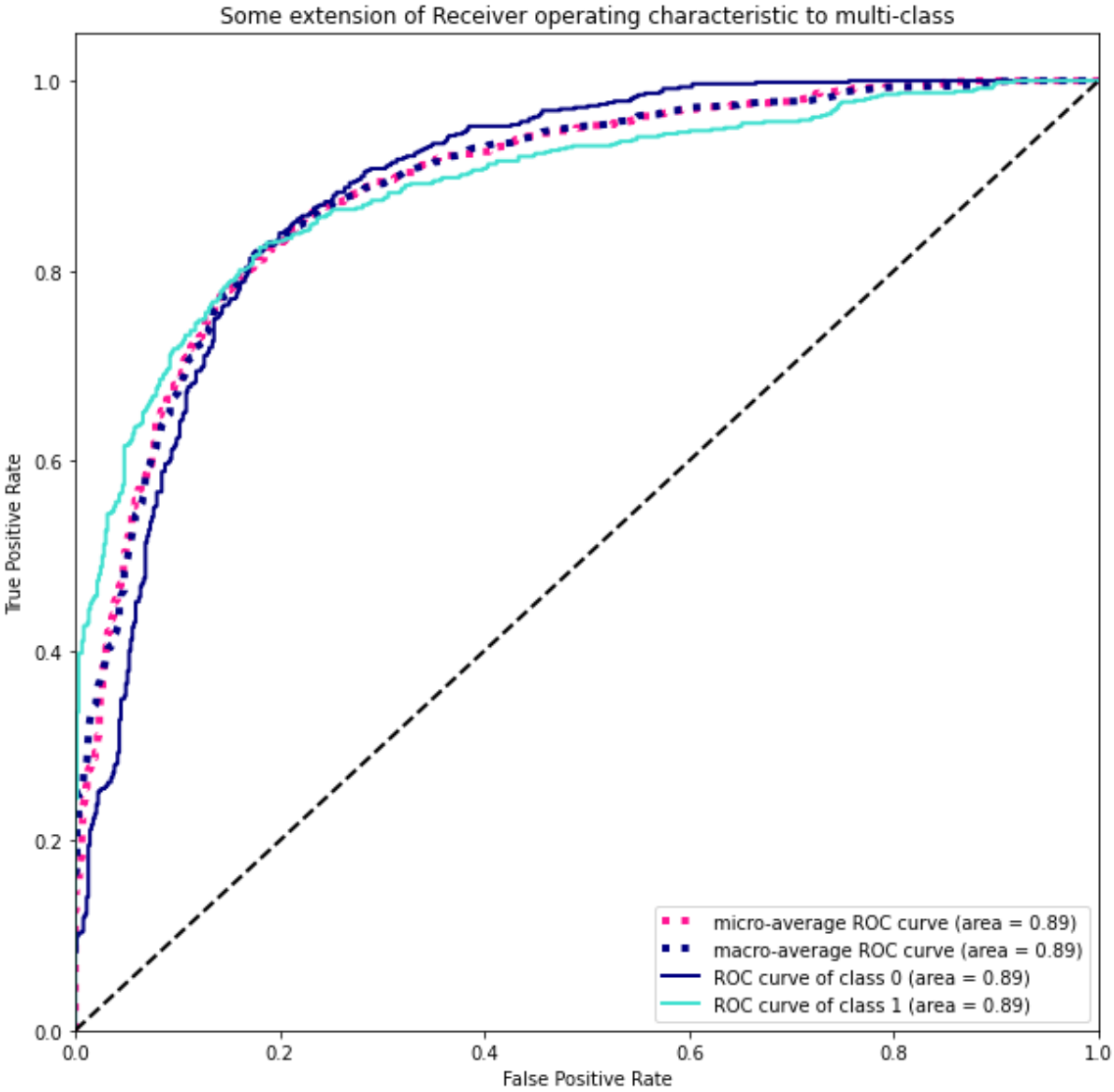
Test Classifier on Real and Fake images

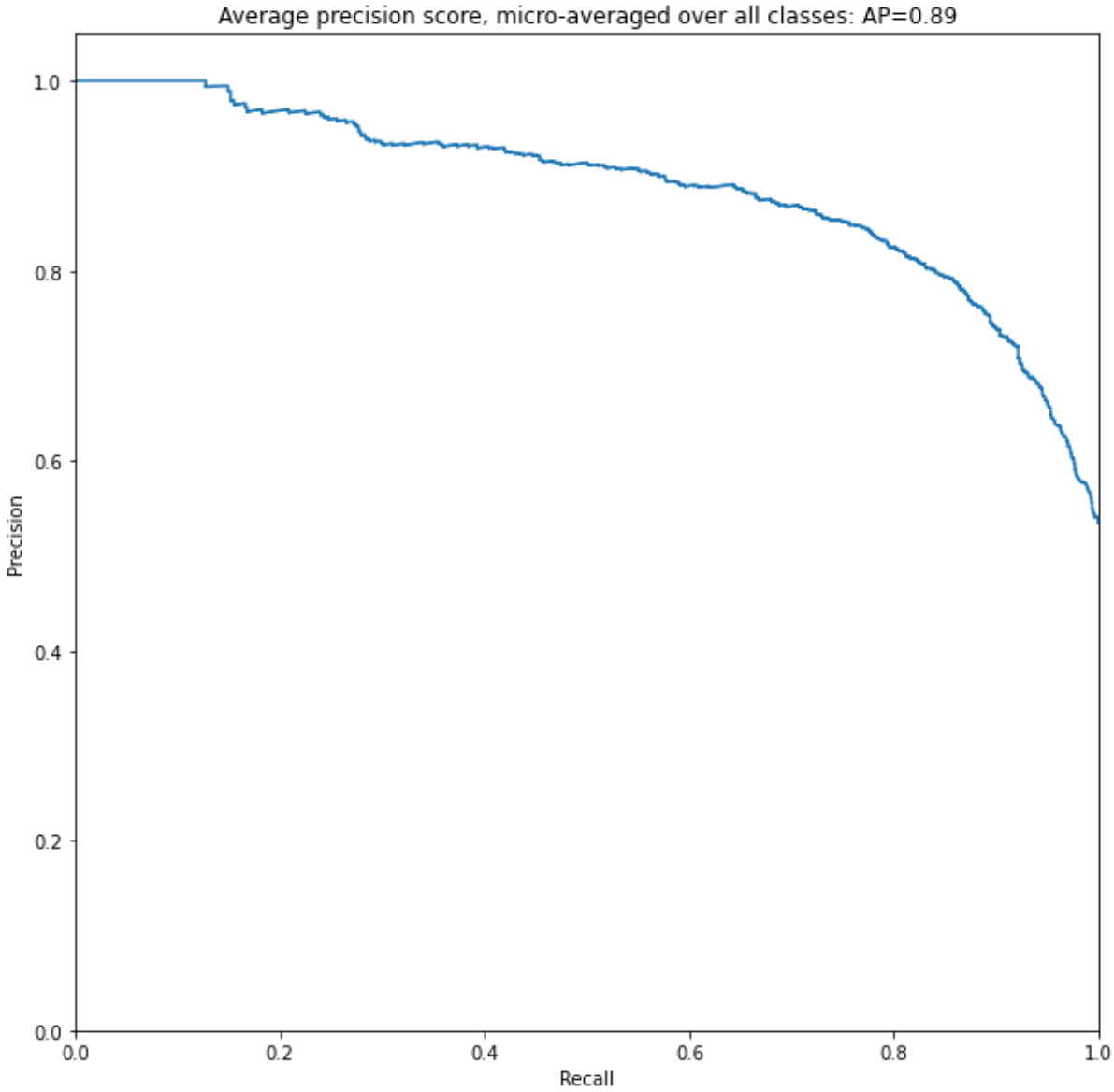
سپس همین مدل را روی ترکیب تصاویر تولیدی و تصاویر اصلی تست میکنیم که نتایج زیر حاصل میشود.

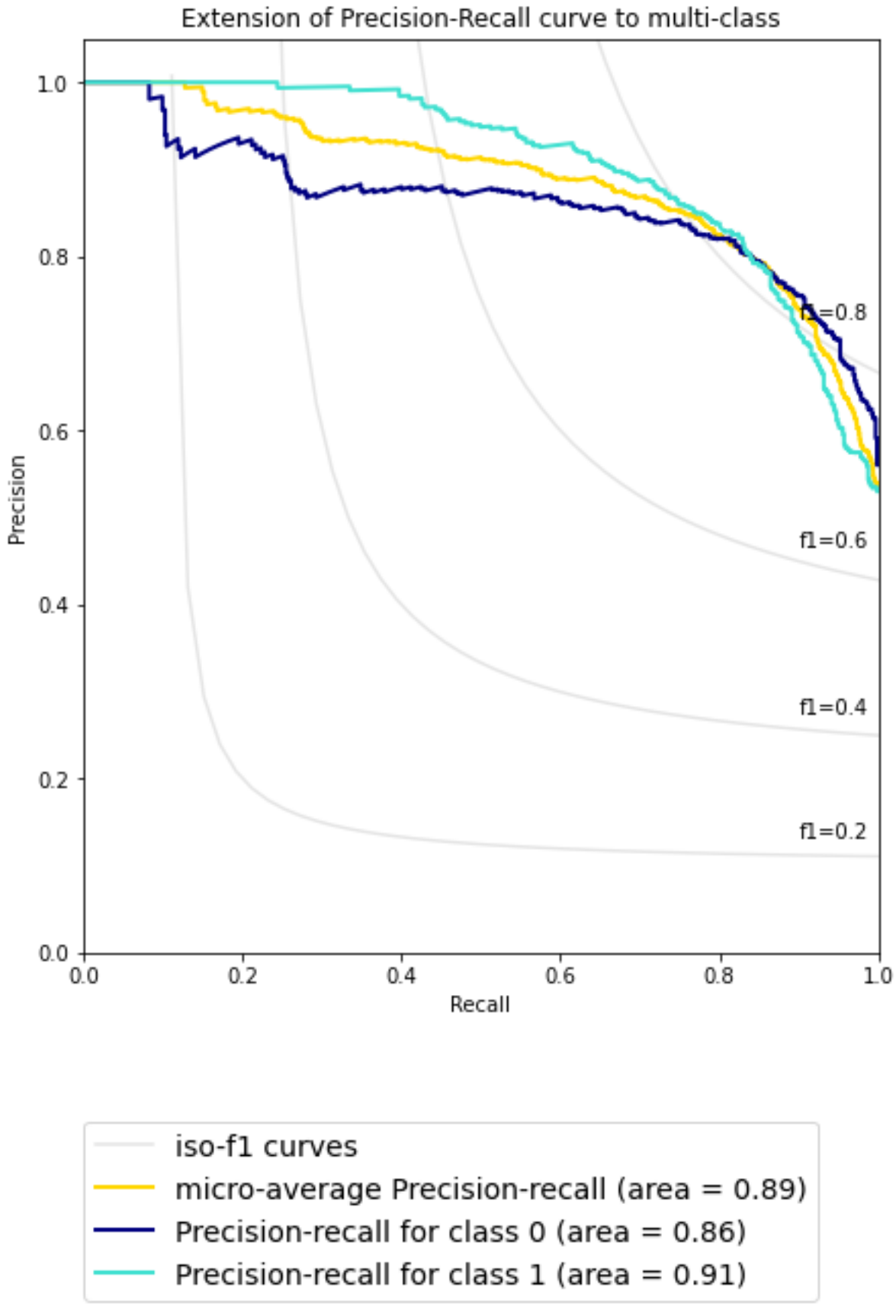
```
24/24 [==============================] - 11s 496ms/step
24/24 [==============================] - 6s 252ms/step
31/31 [==============================] - 4s 143ms/step
Accuracy on Test Data: 0.81%
              precision    recall  f1-score   support

           0       0.78      0.86      0.82       605
           1       0.85      0.77      0.81       627

    accuracy                           0.81      1232
   macro avg       0.82      0.82      0.81      1232
weighted avg       0.82      0.81      0.81      1232
```


0
1
521
84
144
483
0
1


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
Model Prediction (AUC: 0.89)
Random Guessing


Some extension of Receiver operating characteristic to multi-class
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 0.89)
macro-average ROC curve (area = 0.89)
ROC curve of class 0 (area = 0.89)
ROC curve of class 1 (area = 0.89)


Average precision score, micro-averaged over all classes: AP=0.89
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Extension of Precision-Recall curve to multi-class
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
f1=0.8
f1=0.6
f1=0.4
f1=0.2
iso-f1 curves
micro-average Precision-recall (area = 0.89)
Precision-recall for class 0 (area = 0.86)
Precision-recall for class 1 (area = 0.91)