# Event_Classification

July 28, 2021

[ ]:
```
#----------Running the code-------------#
"""
  Just run the script in the folder containing 'data.pickle' and 'data.csv file'

"""

#---------------OVERVIEW----------------#
"""
*  Input data is padded with a special value to make all the sequnces of equal
   length and an additional masking layer is used to mask these special value
   during training.

*  Since the dataset is imbalanced various techniques like stratification,class
   weighting and precision-recall curves are also used.

*  Model: Stacked LSTM layers are used for encoding the time series and then the
   encoding is passed through dense layers for classification.

*  Hyperparameter tuning is performed to find the best size of the network.

"""
```

[2]:
```
import tensorflow as tf
from tensorflow import keras
from keras.layers import LSTM, Dense, Masking, BatchNormalization, Activation
import keras_tuner as kt

import pickle

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

import sklearn
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
```

Helper Functions

[4]:
```python
def plot_loss(history):

    """
    Using the history logs of the trained model this function plots the training
    loss and validation loss on each epoch, using a log scale on y-axis to show
    the wide range of values.

    """

    # Training Loss vs Epoch
    plt.plot(history.epoch, history.history['loss'], label='Train ')
    # Validation Loss vs Epoch
    plt.plot(history.epoch, history.history['val_loss'], label='Val ',
             linestyle="--")

    plt.xlabel('Epoch')
    plt.ylabel('Loss')

    plt.legend()
```

[39]:
```python
def plot_metrics(history):
    """
    Using the history logs and metrics of the trained model this function plots
    4 subplots of precision, recall, loss and area under precision-recall curve
    for the training and validation data on each epoch.

    """

    metrics = ['loss', 'prc', 'precision', 'recall']

    for n, metric in enumerate(metrics):

      plt.subplot(2,2,n+1)

      plt.plot(history.epoch, history.history[metric], label='Train')
      plt.plot(history.epoch, history.history['val_'+metric],
               linestyle="--", label='Val')

      plt.xlabel('Epoch')
      plt.ylabel(metric)

      if metric == 'loss':
        plt.ylim([0, plt.ylim()[1]])
      else:
        plt.ylim([0,1])
    plt.legend()
```

[6]:
```
def plot_cm(true_labels, predictions, p=0.5):
    """
    Using the inbuilt function of confusion_matrix from sklearn this function
    plots a confusion  matrix with a particular threshold(p) with default value
    0.5.

    """
    # Confusion matrix
    cm = confusion_matrix(true_labels, predictions > p)

    plt.figure(figsize=(5,5))
    sns.heatmap(cm, annot=True, fmt="d")
    plt.title('Confusion matrix @{:.2f}'.format(p))
    plt.ylabel('Actual label')
    plt.xlabel('Predicted label')
    plt.legend()

    print('(True Negatives): ', cm[0][0])
    print('(False Positives): ', cm[0][1])
    print('(False Negatives): ', cm[1][0])
    print('(True Positives): ', cm[1][1])
    print('Total : ', np.sum(cm[1]))
```

[8]:
```
def plot_prc(name, true_labels, predictions, **kwargs):
    """
    Using the inbuilt function of precision_recall_curve from sklearn this
    function plots a PR curve

    """
    precision, recall, _ = sklearn.metrics.precision_recall_curve(true_labels,
                                                                  predictions)

    plt.plot(precision, recall, label=name, linewidth=2, **kwargs)
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.grid(True)
    plt.legend()
    ax = plt.gca()
    ax.set_aspect('equal')
```

[13]:
```
def visualize_event(data, e):
    """
    This function visualizes all the features across timesteps in the data
    for a particular event e.
    """

    fig, axes = plt.subplots(nrows=3, ncols=3, dpi=120, figsize=(10,6))
    idx = 0
    for i in range(9):

        ax = axes.flatten()[idx]
        ax.plot(range(len(data[e][:,i])), data[e][:,i], linewidth=1)
        idx += 1

        ax.set_xlabel('Timesteps')
        ax.set_ylabel("Feature_" + str(i))

    plt.tight_layout()
```

Building model

[26]:
```
def hyper_model(hp):
    """
    This function builds a model with four hyperparameters(lstm_units,
    dense1_units, dense2_units and learning_rate). This function is
    passed to a keras hypermeter tuner to select a best hyperparameter
    combination.

    """

    model = keras.Sequential()

    # Masks a sequence by using a mask_value to skip timesteps which were padded
    model.add(Masking(mask_value=special_value,
                      input_shape=(max_seq_len, dimension)))

    # Tune the number of units in the LSTM layer
    # Choose an optimal value between 128-256
    lstm_units = hp.Int('lstm_units', min_value=128, max_value=256, step=128)
    model.add(LSTM(lstm_units, return_sequences = True))
    model.add(LSTM(lstm_units))

    # Tune the number of units in the first dense layer
    # Choose an optimal value between 256-512
    dense1_units = hp.Int('Dense_1', min_value=256, max_value=512, step=256)
    model.add(Dense(dense1_units))
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))

    # Tune the number of units in the second dense layer
    # Choose an optimal value between 64-128
    dense2_units = hp.Int('Dense_2', min_value=64, max_value=128, step=64)
    model.add(Dense(dense2_units))
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))

    model.add(Dense(1, activation='sigmoid'))

    # Tune the learning rate for the optimizer
    # Choose an optimal value from 0.001, or 0.0001
    hp_learning_rate = hp.Choice('learning_rate', values=[1e-3, 1e-4])

    model.compile(optimizer=keras.optimizers.Adam(
                  learning_rate=hp_learning_rate),
                  loss=keras.losses.BinaryCrossentropy(),
                  metrics=METRICS)


    return model
```

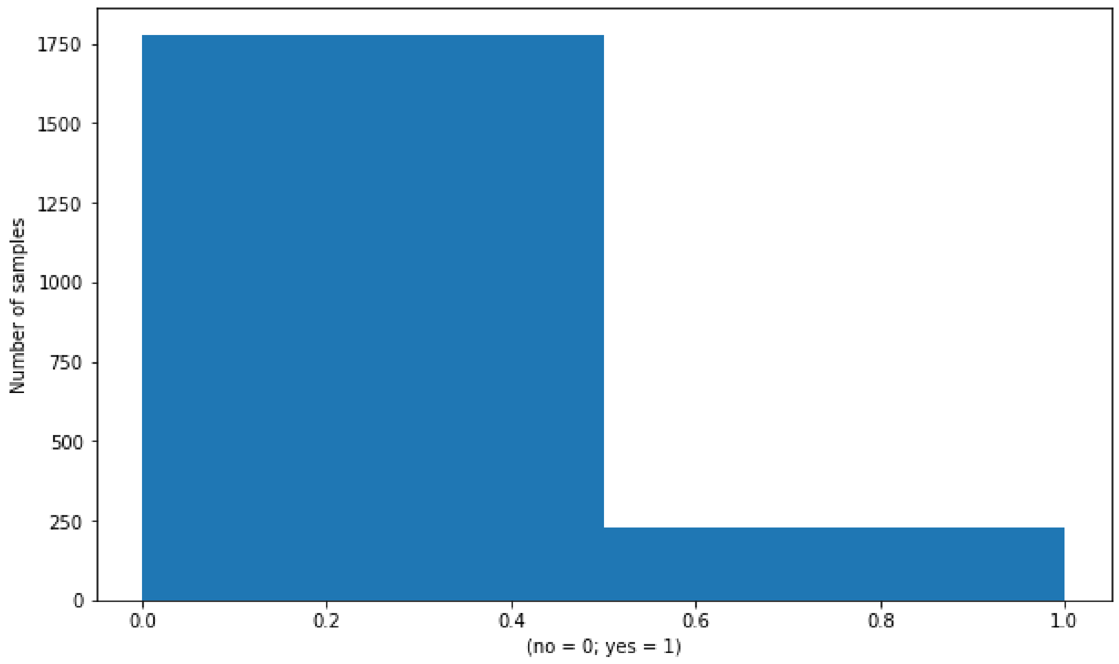
Load Data

[15]:
```
# Load data.pickle as raw_data
filename = 'data.pickle'
infile = open(filename,'rb')
raw_data = pickle.load(infile)
infile.close()

# Load data.csv as dataframe
df = pd.read_csv('data.csv')
```

Check the class distribution

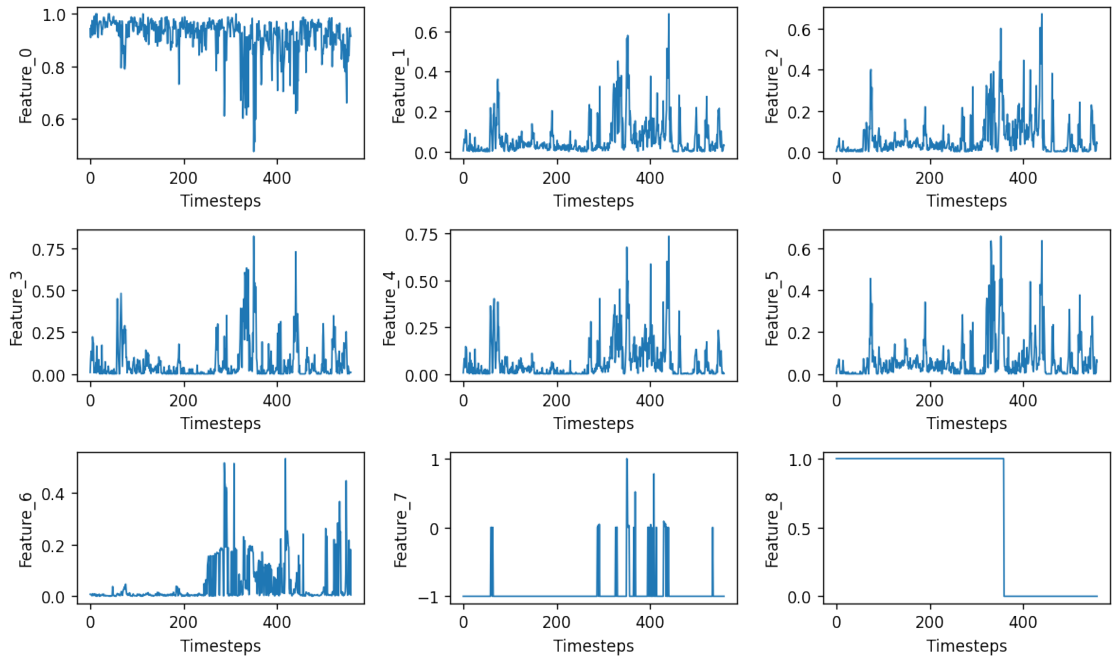
[16]:
```
# output label
y = np.array(df['selected_event'])

# count number of each class in the dataset
unique, counts = np.unique(y, return_counts=True)
print(f'Number of each output label are {dict(zip(unique, counts))}')

# plot the histogram for each class
plt.rcParams["figure.figsize"] = (10,6)
plt.hist(y, bins=2, density=False)
plt.ylabel('Number of samples')
plt.xlabel('(no = 0; yes = 1)')
plt.show()
```

```
Number of each output label are {0: 1775, 1: 225}
```



Visualizes all the features across timesteps for a particular event

```
[18]: data = np.array(raw_data, dtype=object)

      event = 100 #event to be visulaized
      visualize_event(data, event)
```



Pre-process the Data

[19]:
```
# Padding the data with special_value(which will be masked later in LSTM layers)
# to make all the sequences all equal length.
max_seq_len = df['X_length'].max()
special_value = -10.0

# Value 2000 and 10 are hard coded. These are number of events and feature size
# respectively.
X = np.full((2000,max_seq_len,10), special_value, dtype=float)

for i in range(data.size):
    tmp = data[i].shape
    X[i,:tmp[0],:] = data[i]

# Check for missing anf inf vlaues
X = np.nan_to_num(X, copy=True, nan=0.0, posinf=None, neginf=None)
```

Calculating class weights

[20]:
```
# Since the data considered here is imbalanced, the weighted loss function is
# used where minority class is penalized more according to the class weights.
# This block calculates the class_weight which will passed to the model's loss
# function.

total = counts[0] + counts[1]
weight_for_0 = (1 / counts[0]) * (total / 2.0)
weight_for_1 = (1 / counts[1]) * (total / 2.0)

class_weight = {0: weight_for_0, 1: weight_for_1}

print('Weight for class 0: {:.2f}'.format(weight_for_0))
print('Weight for class 1: {:.2f}'.format(weight_for_1))
```

```
Weight for class 0: 0.56
Weight for class 1: 4.44
```

Train-Validation-Test split

[21]:
```
# Split the dataset into train, validation and test set. Using stratify to make
# the split balanced with both classes.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,
                                                    stratify = y)

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,
                                                  test_size=0.2,
                                                  stratify = y_train)
```

Define Parameters

[22]:
```
# All the metrics which will be monitored during training.
METRICS = [
      keras.metrics.TruePositives(name='tp'),
      keras.metrics.FalsePositives(name='fp'),
      keras.metrics.TrueNegatives(name='tn'),
      keras.metrics.FalseNegatives(name='fn'),
      keras.metrics.BinaryAccuracy(name='accuracy'),
      keras.metrics.Precision(name='precision'),
      keras.metrics.Recall(name='recall'),
      keras.metrics.AUC(name='auc'),
      keras.metrics.AUC(name='prc', curve='PR'), # precision-recall curve
]
```

[23]:
```
# Parameters
dimension = 10
verbosity = 1
EPOCHS = 200
BATCH_SIZE = 256

# Callbacks used during training
# Validation precision-recall curve is monitored instead of accuracy since
# the data is imbalanced
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_prc',
                                                  verbose=1,
                                                  patience=10, mode='max')

# Callback for saving the best model
chk = tf.keras.callbacks.ModelCheckpoint('best_model',
                                         monitor='val_prc',
                                         save_best_only=True,
                                         mode='max', verbose=1)
```

Hyperparameter Tuning using Baysian Optimization

[27]:
```
# Hyperparameter tuning using Bayesian Optimization for hyper_model
# defined above
tuner = kt.BayesianOptimization(hyper_model, max_trials=10,
                                objective=kt.Objective("val_prc",
                                                       direction="max"))


# Hyperparameter search according to the BayesianOptimization tuner
tuner.search(X_train, y_train, epochs=10, batch_size=BATCH_SIZE,
             class_weight=class_weight,
             validation_data=(X_val, y_val), callbacks=[early_stopping])

# Get the optimal hyperparameters
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]

print(f"""
The hyperparameter search is complete. The optimal number of units
in the lstm output is {best_hps.get('lstm_units')}, first densely-connected
layer is {best_hps.get('Dense_1')} and second densely-connected layer is
{best_hps.get('Dense_2')} and the optimal learning rate for the optimizer
is {best_hps.get('learning_rate')}.
""")
```

```
Trial 10 Complete [00h 22m 33s]
val_prc: 0.17031022906303406

Best val_prc So Far: 0.8057265877723694
Total elapsed time: 03h 14m 04s
INFO:tensorflow:Oracle triggered exit

The hyperparameter search is complete. The optimal number of units in the lstm
output is 256, first densely-connected
layer is 256 and second densely-connected layer is 128 and the optimal learning
rate for the optimizer
is 0.001.
```

Model Training

[33]:
```
# Build the model with the optimal hyperparameters and train it on the data
model = tuner.hypermodel.build(best_hps)
history = model.fit(X_train, y_train, batch_size=BATCH_SIZE, epochs=100,
                    validation_data=(X_val, y_val),
                    class_weight=class_weight, callbacks=[early_stopping, chk])
```

```
Epoch 1/100
6/6 [==============================] - 154s 24s/step - loss: 0.5346 - tp:
153.0000 - fp: 708.0000 - tn: 889.0000 - fn: 50.0000 - accuracy: 0.5789 -
precision: 0.1777 - recall: 0.7537 - auc: 0.7574 - prc: 0.4443 - val_loss:
0.6110 - val_tp: 1.0000 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
40.0000 - val_accuracy: 0.8889 - val_precision: 1.0000 - val_recall: 0.0244 -
val_auc: 0.7979 - val_prc: 0.6883

Epoch 00001: val_prc improved from -inf to 0.68828, saving model to
/content/drive/MyDrive/Experiments/Experiment_4/best_model

WARNING:absl:Found untraced functions such as
masking_layer_call_and_return_conditional_losses, masking_layer_call_fn,
dense_layer_call_and_return_conditional_losses, dense_layer_call_fn,
activation_layer_call_and_return_conditional_losses while saving (showing 5 of
40). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

Epoch 2/100
6/6 [==============================] - 147s 24s/step - loss: 0.3735 - tp:
138.0000 - fp: 207.0000 - tn: 1071.0000 - fn: 24.0000 - accuracy: 0.8396 -
precision: 0.4000 - recall: 0.8519 - auc: 0.9261 - prc: 0.7045 - val_loss:
0.4228 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.8840 - val_prc: 0.5224

Epoch 00002: val_prc did not improve from 0.68828
Epoch 3/100
6/6 [==============================] - 154s 25s/step - loss: 0.3806 - tp:
121.0000 - fp: 141.0000 - tn: 1137.0000 - fn: 41.0000 - accuracy: 0.8736 -
precision: 0.4618 - recall: 0.7469 - auc: 0.9055 - prc: 0.7155 - val_loss:
0.4170 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.8801 - val_prc: 0.6697

Epoch 00003: val_prc did not improve from 0.68828
Epoch 4/100
6/6 [==============================] - 157s 26s/step - loss: 0.3549 - tp:
137.0000 - fp: 160.0000 - tn: 1118.0000 - fn: 25.0000 - accuracy: 0.8715 -
precision: 0.4613 - recall: 0.8457 - auc: 0.9267 - prc: 0.7267 - val_loss:
0.4263 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.8024 - val_prc: 0.6616

Epoch 00004: val_prc did not improve from 0.68828
Epoch 5/100
6/6 [==============================] - 149s 25s/step - loss: 0.3303 - tp:
139.0000 - fp: 161.0000 - tn: 1117.0000 - fn: 23.0000 - accuracy: 0.8722 -
precision: 0.4633 - recall: 0.8580 - auc: 0.9343 - prc: 0.7544 - val_loss:
0.3639 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.7900 - val_prc: 0.6627

Epoch 00005: val_prc did not improve from 0.68828
Epoch 6/100
6/6 [==============================] - 152s 25s/step - loss: 0.3062 - tp:
139.0000 - fp: 161.0000 - tn: 1117.0000 - fn: 23.0000 - accuracy: 0.8722 -
precision: 0.4633 - recall: 0.8580 - auc: 0.9468 - prc: 0.7568 - val_loss:
```

```
0.3381 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.8459 - val_prc: 0.7266

Epoch 00006: val_prc improved from 0.68828 to 0.72660, saving model to
/content/drive/MyDrive/Experiments/Experiment_4/best_model

WARNING:absl:Found untraced functions such as
masking_layer_call_and_return_conditional_losses, masking_layer_call_fn,
dense_layer_call_and_return_conditional_losses, dense_layer_call_fn,
activation_layer_call_and_return_conditional_losses while saving (showing 5 of
40). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

Epoch 7/100
6/6 [==============================] - 147s 25s/step - loss: 0.2880 - tp:
136.0000 - fp: 121.0000 - tn: 1157.0000 - fn: 26.0000 - accuracy: 0.8979 -
precision: 0.5292 - recall: 0.8395 - auc: 0.9501 - prc: 0.8017 - val_loss:
0.3295 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.8487 - val_prc: 0.7309

Epoch 00007: val_prc improved from 0.72660 to 0.73085, saving model to
/content/drive/MyDrive/Experiments/Experiment_4/best_model

WARNING:absl:Found untraced functions such as
masking_layer_call_and_return_conditional_losses, masking_layer_call_fn,
dense_layer_call_and_return_conditional_losses, dense_layer_call_fn,
activation_layer_call_and_return_conditional_losses while saving (showing 5 of
40). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

Epoch 8/100
6/6 [==============================] - 139s 23s/step - loss: 0.2755 - tp:
143.0000 - fp: 121.0000 - tn: 1157.0000 - fn: 19.0000 - accuracy: 0.9028 -
precision: 0.5417 - recall: 0.8827 - auc: 0.9557 - prc: 0.7921 - val_loss:
0.3621 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.7918 - val_prc: 0.6727

Epoch 00008: val_prc did not improve from 0.73085
Epoch 9/100
```

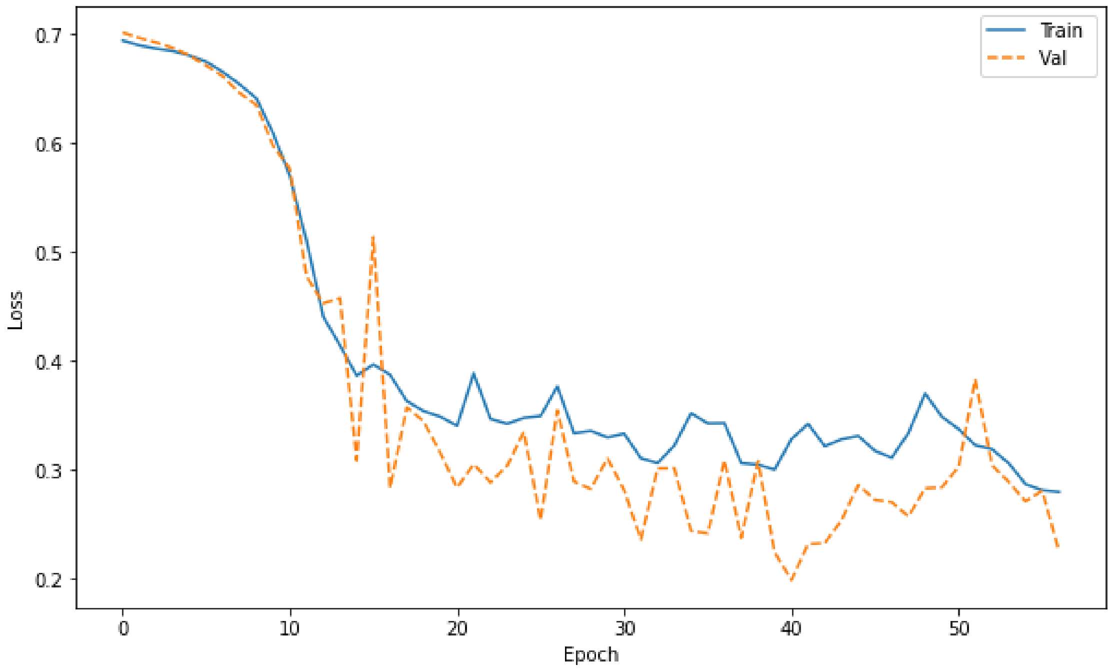
```
6/6 [==============================] - 147s 25s/step - loss: 0.2673 - tp:
141.0000 - fp: 118.0000 - tn: 1160.0000 - fn: 21.0000 - accuracy: 0.9035 -
precision: 0.5444 - recall: 0.8704 - auc: 0.9595 - prc: 0.7937 - val_loss:
0.4120 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.8579 - val_prc: 0.6128

Epoch 00009: val_prc did not improve from 0.73085
Epoch 10/100
6/6 [==============================] - 144s 24s/step - loss: 0.3174 - tp:
136.0000 - fp: 140.0000 - tn: 1138.0000 - fn: 26.0000 - accuracy: 0.8847 -
precision: 0.4928 - recall: 0.8395 - auc: 0.9386 - prc: 0.7585 - val_loss:
0.3547 - val_tp: 7.0000 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
34.0000 - val_accuracy: 0.9056 - val_precision: 1.0000 - val_recall: 0.1707 -
val_auc: 0.7860 - val_prc: 0.6359

Epoch 00010: val_prc did not improve from 0.73085
Epoch 11/100
6/6 [==============================] - 138s 23s/step - loss: 0.3012 - tp:
136.0000 - fp: 147.0000 - tn: 1131.0000 - fn: 26.0000 - accuracy: 0.8799 -
precision: 0.4806 - recall: 0.8395 - auc: 0.9464 - prc: 0.7660 - val_loss:
0.2849 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.9195 - val_prc: 0.6510

Epoch 00011: val_prc did not improve from 0.73085
Epoch 12/100
6/6 [==============================] - 139s 23s/step - loss: 0.2883 - tp:
136.0000 - fp: 105.0000 - tn: 1173.0000 - fn: 26.0000 - accuracy: 0.9090 -
precision: 0.5643 - recall: 0.8395 - auc: 0.9540 - prc: 0.7734 - val_loss:
0.3802 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.8798 - val_prc: 0.6111

Epoch 00012: val_prc did not improve from 0.73085
Epoch 13/100
6/6 [==============================] - 141s 24s/step - loss: 0.3003 - tp:
146.0000 - fp: 217.0000 - tn: 1061.0000 - fn: 16.0000 - accuracy: 0.8382 -
precision: 0.4022 - recall: 0.9012 - auc: 0.9497 - prc: 0.7938 - val_loss:
0.3219 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.9125 - val_prc: 0.7526

Epoch 00013: val_prc improved from 0.73085 to 0.75259, saving model to
/content/drive/MyDrive/Experiments/Experiment_4/best_model

WARNING:absl:Found untraced functions such as
masking_layer_call_and_return_conditional_losses, masking_layer_call_fn,
dense_layer_call_and_return_conditional_losses, dense_layer_call_fn,
activation_layer_call_and_return_conditional_losses while saving (showing 5 of
40). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

Epoch 14/100
6/6 [==============================] - 138s 23s/step - loss: 0.2883 - tp:
139.0000 - fp: 98.0000 - tn: 1180.0000 - fn: 23.0000 - accuracy: 0.9160 -
precision: 0.5865 - recall: 0.8580 - auc: 0.9495 - prc: 0.7748 - val_loss:
0.3293 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.7735 - val_prc: 0.4724

Epoch 00014: val_prc did not improve from 0.75259
Epoch 15/100
6/6 [==============================] - 138s 23s/step - loss: 0.2927 - tp:
141.0000 - fp: 150.0000 - tn: 1128.0000 - fn: 21.0000 - accuracy: 0.8813 -
precision: 0.4845 - recall: 0.8704 - auc: 0.9476 - prc: 0.7858 - val_loss:
0.2809 - val_tp: 5.0000 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
36.0000 - val_accuracy: 0.9000 - val_precision: 1.0000 - val_recall: 0.1220 -
val_auc: 0.9267 - val_prc: 0.7534

Epoch 00015: val_prc improved from 0.75259 to 0.75336, saving model to
/content/drive/MyDrive/Experiments/Experiment_4/best_model

WARNING:absl:Found untraced functions such as
masking_layer_call_and_return_conditional_losses, masking_layer_call_fn,
dense_layer_call_and_return_conditional_losses, dense_layer_call_fn,
activation_layer_call_and_return_conditional_losses while saving (showing 5 of
40). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

Epoch 16/100
6/6 [==============================] - 136s 23s/step - loss: 0.2657 - tp:
143.0000 - fp: 132.0000 - tn: 1146.0000 - fn: 19.0000 - accuracy: 0.8951 -
precision: 0.5200 - recall: 0.8827 - auc: 0.9582 - prc: 0.8119 - val_loss:
0.2928 - val_tp: 11.0000 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
30.0000 - val_accuracy: 0.9167 - val_precision: 1.0000 - val_recall: 0.2683 -
val_auc: 0.9485 - val_prc: 0.8182

Epoch 00016: val_prc improved from 0.75336 to 0.81823, saving model to
```

```
/content/drive/MyDrive/Experiments/Experiment_4/best_model

WARNING:absl:Found untraced functions such as
masking_layer_call_and_return_conditional_losses, masking_layer_call_fn,
dense_layer_call_and_return_conditional_losses, dense_layer_call_fn,
activation_layer_call_and_return_conditional_losses while saving (showing 5 of
40). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

Epoch 17/100
6/6 [==============================] - 143s 24s/step - loss: 0.2822 - tp:
141.0000 - fp: 121.0000 - tn: 1157.0000 - fn: 21.0000 - accuracy: 0.9014 -
precision: 0.5382 - recall: 0.8704 - auc: 0.9502 - prc: 0.8049 - val_loss:
0.2863 - val_tp: 22.0000 - val_fp: 1.0000 - val_tn: 318.0000 - val_fn: 19.0000 -
val_accuracy: 0.9444 - val_precision: 0.9565 - val_recall: 0.5366 - val_auc:
0.9539 - val_prc: 0.8336

Epoch 00017: val_prc improved from 0.81823 to 0.83359, saving model to
/content/drive/MyDrive/Experiments/Experiment_4/best_model

WARNING:absl:Found untraced functions such as
masking_layer_call_and_return_conditional_losses, masking_layer_call_fn,
dense_layer_call_and_return_conditional_losses, dense_layer_call_fn,
activation_layer_call_and_return_conditional_losses while saving (showing 5 of
40). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

Epoch 18/100
6/6 [==============================] - 134s 22s/step - loss: 0.2785 - tp:
137.0000 - fp: 107.0000 - tn: 1171.0000 - fn: 25.0000 - accuracy: 0.9083 -
precision: 0.5615 - recall: 0.8457 - auc: 0.9536 - prc: 0.7945 - val_loss:
0.2871 - val_tp: 6.0000 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
35.0000 - val_accuracy: 0.9028 - val_precision: 1.0000 - val_recall: 0.1463 -
val_auc: 0.9354 - val_prc: 0.7713

Epoch 00018: val_prc did not improve from 0.83359
Epoch 19/100
6/6 [==============================] - 135s 23s/step - loss: 0.2881 - tp:
142.0000 - fp: 172.0000 - tn: 1106.0000 - fn: 20.0000 - accuracy: 0.8667 -
precision: 0.4522 - recall: 0.8765 - auc: 0.9510 - prc: 0.7864 - val_loss:
0.3325 - val_tp: 11.0000 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
30.0000 - val_accuracy: 0.9167 - val_precision: 1.0000 - val_recall: 0.2683 -
```

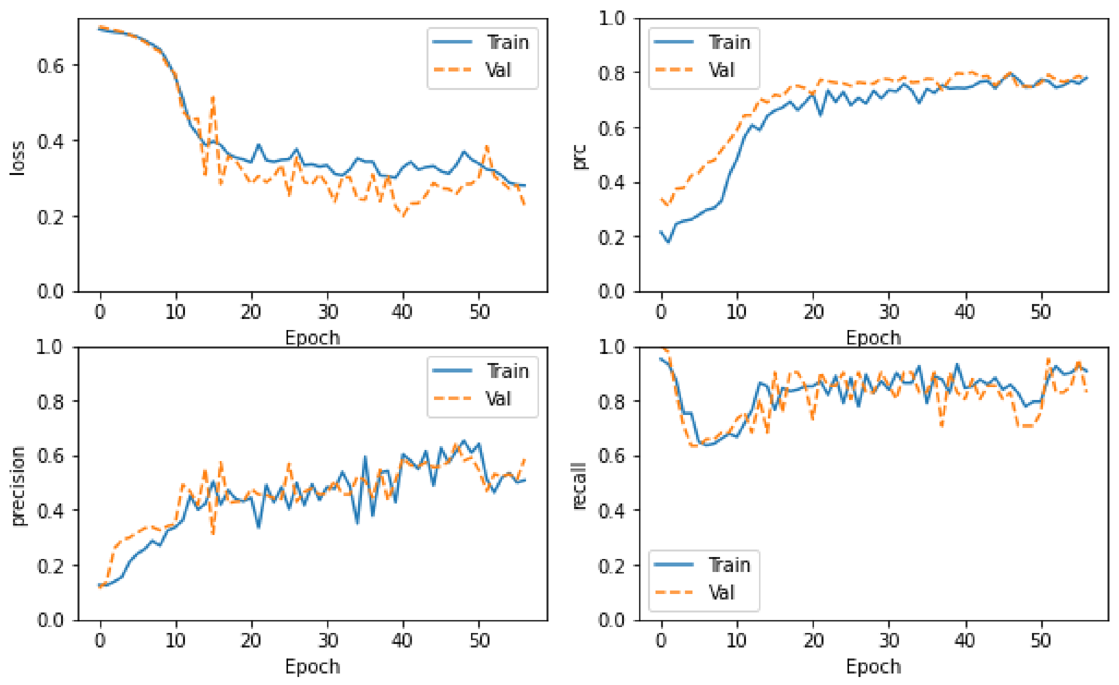
```
val_auc: 0.9574 - val_prc: 0.8448

Epoch 00019: val_prc improved from 0.83359 to 0.84482, saving model to
/content/drive/MyDrive/Experiments/Experiment_4/best_model

WARNING:absl:Found untraced functions such as
masking_layer_call_and_return_conditional_losses, masking_layer_call_fn,
dense_layer_call_and_return_conditional_losses, dense_layer_call_fn,
activation_layer_call_and_return_conditional_losses while saving (showing 5 of
40). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

INFO:tensorflow:Assets written to:
/content/drive/MyDrive/Experiments/Experiment_4/best_model/assets

Epoch 20/100
6/6 [==============================] - 129s 22s/step - loss: 0.2759 - tp:
144.0000 - fp: 182.0000 - tn: 1096.0000 - fn: 18.0000 - accuracy: 0.8611 -
precision: 0.4417 - recall: 0.8889 - auc: 0.9549 - prc: 0.8107 - val_loss:
0.3306 - val_tp: 3.0000 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
38.0000 - val_accuracy: 0.8944 - val_precision: 1.0000 - val_recall: 0.0732 -
val_auc: 0.9530 - val_prc: 0.8198

Epoch 00020: val_prc did not improve from 0.84482
Epoch 21/100
6/6 [==============================] - 138s 23s/step - loss: 0.2521 - tp:
145.0000 - fp: 125.0000 - tn: 1153.0000 - fn: 17.0000 - accuracy: 0.9014 -
precision: 0.5370 - recall: 0.8951 - auc: 0.9631 - prc: 0.8105 - val_loss:
0.2070 - val_tp: 19.0000 - val_fp: 1.0000 - val_tn: 318.0000 - val_fn: 22.0000 -
val_accuracy: 0.9361 - val_precision: 0.9500 - val_recall: 0.4634 - val_auc:
0.9562 - val_prc: 0.8350

Epoch 00021: val_prc did not improve from 0.84482
Epoch 22/100
6/6 [==============================] - 130s 22s/step - loss: 0.3083 - tp:
132.0000 - fp: 86.0000 - tn: 1192.0000 - fn: 30.0000 - accuracy: 0.9194 -
precision: 0.6055 - recall: 0.8148 - auc: 0.9507 - prc: 0.7680 - val_loss:
0.3581 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.8267 - val_prc: 0.6374

Epoch 00022: val_prc did not improve from 0.84482
Epoch 23/100
6/6 [==============================] - 135s 22s/step - loss: 0.3071 - tp:
130.0000 - fp: 110.0000 - tn: 1168.0000 - fn: 32.0000 - accuracy: 0.9014 -
precision: 0.5417 - recall: 0.8025 - auc: 0.9425 - prc: 0.7820 - val_loss:
0.3154 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
```

```
0.0000e+00 - val_auc: 0.8229 - val_prc: 0.4491

Epoch 00023: val_prc did not improve from 0.84482
Epoch 24/100
6/6 [==============================] - 131s 22s/step - loss: 0.2925 - tp:
144.0000 - fp: 170.0000 - tn: 1108.0000 - fn: 18.0000 - accuracy: 0.8694 -
precision: 0.4586 - recall: 0.8889 - auc: 0.9481 - prc: 0.7946 - val_loss:
0.3199 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.8235 - val_prc: 0.3568

Epoch 00024: val_prc did not improve from 0.84482
Epoch 25/100
6/6 [==============================] - 136s 23s/step - loss: 0.2752 - tp:
147.0000 - fp: 181.0000 - tn: 1097.0000 - fn: 15.0000 - accuracy: 0.8639 -
precision: 0.4482 - recall: 0.9074 - auc: 0.9559 - prc: 0.8171 - val_loss:
0.2971 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.8571 - val_prc: 0.4360

Epoch 00025: val_prc did not improve from 0.84482
Epoch 26/100
6/6 [==============================] - 144s 24s/step - loss: 0.2732 - tp:
140.0000 - fp: 108.0000 - tn: 1170.0000 - fn: 22.0000 - accuracy: 0.9097 -
precision: 0.5645 - recall: 0.8642 - auc: 0.9543 - prc: 0.7969 - val_loss:
0.3008 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.8659 - val_prc: 0.4358

Epoch 00026: val_prc did not improve from 0.84482
Epoch 27/100
6/6 [==============================] - 133s 22s/step - loss: 0.2906 - tp:
135.0000 - fp: 120.0000 - tn: 1158.0000 - fn: 27.0000 - accuracy: 0.8979 -
precision: 0.5294 - recall: 0.8333 - auc: 0.9483 - prc: 0.7797 - val_loss:
0.3196 - val_tp: 0.0000e+00 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
41.0000 - val_accuracy: 0.8861 - val_precision: 0.0000e+00 - val_recall:
0.0000e+00 - val_auc: 0.8409 - val_prc: 0.5445

Epoch 00027: val_prc did not improve from 0.84482
Epoch 28/100
6/6 [==============================] - 135s 22s/step - loss: 0.2756 - tp:
139.0000 - fp: 115.0000 - tn: 1163.0000 - fn: 23.0000 - accuracy: 0.9042 -
precision: 0.5472 - recall: 0.8580 - auc: 0.9554 - prc: 0.7997 - val_loss:
0.2939 - val_tp: 9.0000 - val_fp: 1.0000 - val_tn: 318.0000 - val_fn: 32.0000 -
val_accuracy: 0.9083 - val_precision: 0.9000 - val_recall: 0.2195 - val_auc:
0.9257 - val_prc: 0.6689

Epoch 00028: val_prc did not improve from 0.84482
Epoch 29/100
6/6 [==============================] - 139s 23s/step - loss: 0.2611 - tp:
145.0000 - fp: 135.0000 - tn: 1143.0000 - fn: 17.0000 - accuracy: 0.8944 -
precision: 0.5179 - recall: 0.8951 - auc: 0.9585 - prc: 0.8156 - val_loss:
0.2738 - val_tp: 3.0000 - val_fp: 0.0000e+00 - val_tn: 319.0000 - val_fn:
38.0000 - val_accuracy: 0.8944 - val_precision: 1.0000 - val_recall: 0.0732 -
val_auc: 0.8595 - val_prc: 0.7224

Epoch 00029: val_prc did not improve from 0.84482
Epoch 00029: early stopping
```

[47]:
```
# Loss plot
plot_loss(history)
```



[48]:
```
# Metric plot
plot_metrics(history)
```



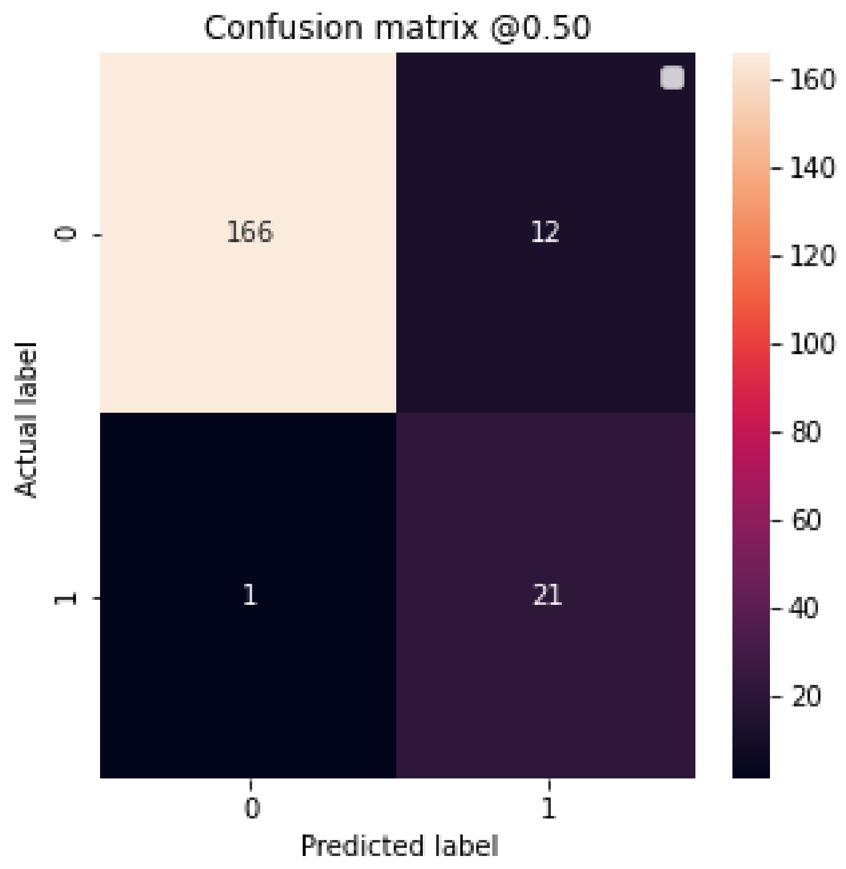
Evaluate Model

[41]:
```
# Predict for train and test data using trained model
train_predictions = model.predict(X_train, batch_size=BATCH_SIZE)
test_predictions = model.predict(X_test, batch_size=BATCH_SIZE)
```

[51]:
```
# Plot Confusion Matrix
plot_cm(y_test, test_predictions)
```

```
WARNING:matplotlib.legend:No handles with labels found to put in legend.
```

```
(True Negatives):  166
(False Positives):  12
(False Negatives):  1
(True Positives):  21
Total :  22
```

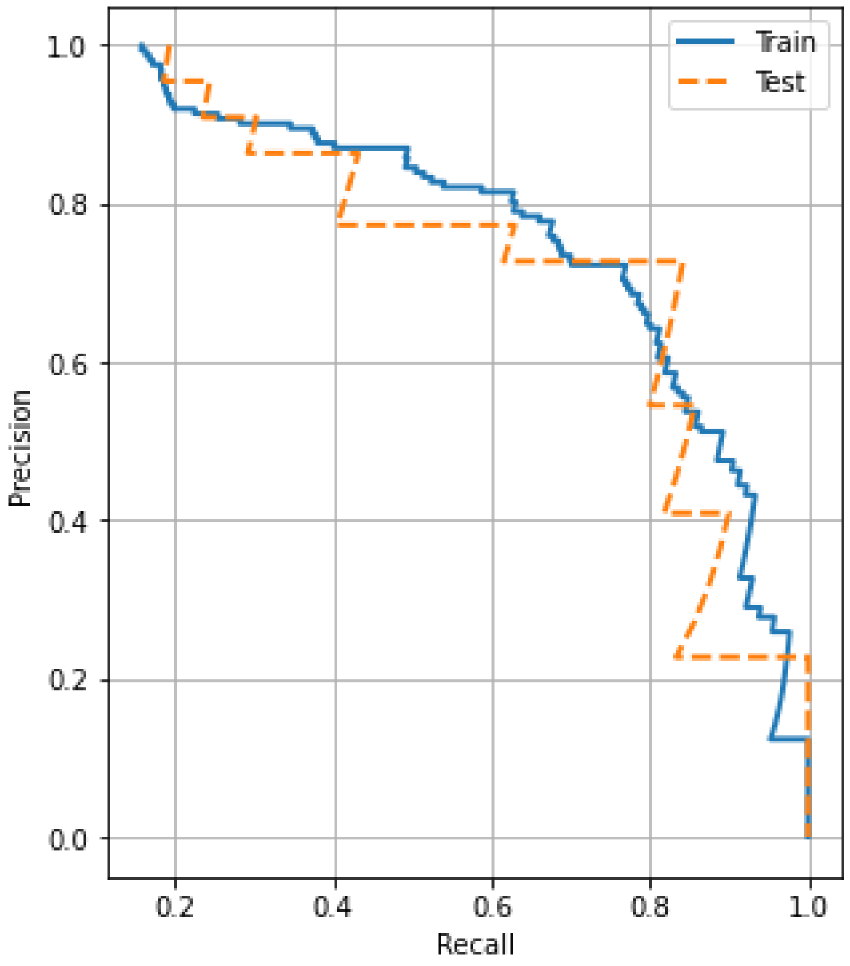


[50]:
```
# Evaluate the model and output the various metrics
results = model.evaluate(X_test, y_test, batch_size=BATCH_SIZE, verbose=0)

for name, value in zip(model.metrics_names, results):
  print(name, ': ', value)
print()
```

```
loss :  0.18149976432323456
tp :  21.0
fp :  12.0
tn :  166.0
fn :  1.0
accuracy :  0.9350000023841858
precision :  0.6363636255264282
recall :  0.9545454382896423
auc :  0.9802093505859375
prc :  0.8792988061904907
```

```
[45]: # Plot Precision-recall curve
      plot_prc("Train", y_train, train_predictions)
      plot_prc("Test", y_test, test_predictions, linestyle='--')
```

```
[45]: <matplotlib.legend.Legend at 0x7f940fec9b10>
```


Train
Test
Precision
Recall