```
  import h5py
  import numpy as np
  import tensorflow as tf
  from tensorflow import keras
```

```
f1 = h5py.File('../input/electron-photon/download', 'r')
f2 = h5py.File('../input/electron-photon/download_1', 'r')

Electron_X = np.array(f1['X'])
Electron_y = np.array(f1['y'])
Parton_X = np.array(f2['X'])
Parton_y = np.array(f2['y'])
print(Electron_X.shape, Electron_y.shape, Parton_X.shape, Parton_y.shape)

All_X = np.concatenate((Electron_X, Parton_X), axis=0)
All_y = np.concatenate((Electron_y, Parton_y), axis=0)
print(All_X.shape, All_y.shape)
rand_seed = 263
index = np.random.permutation(len(All_y))
All_X, All_y = All_X[index][:,:,:,0], All_y[index]
print(All_X.shape, All_y.shape)

# clear cache to save memory
del Electron_X, Electron_y, Parton_X, Parton_y

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(All_X, All_y, test_size=0.2, random_state=12)
print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)

del All_X, All_y
```

```
(249000, 32, 32, 2) (249000,) (249000, 32, 32, 2) (249000,)
(498000, 32, 32, 2) (498000,)
(498000, 32, 32) (498000,)
(398400, 32, 32) (99600, 32, 32)
(398400,) (99600,)
```

```
from tensorflow import keras
from keras.models import Sequential, Model
from keras import layers
from keras import optimizers

num_classes = 1
input_shape = (32, 32, 1)
model = Sequential(
    [
        keras.Input(shape=input_shape),
        layers.BatchNormalization(),
        layers.Conv2D(32, kernel_size=(3,3), activation="relu"),
        layers.BatchNormalization(),
        layers.MaxPooling2D(pool_size=(2,2)),
        layers.Conv2D(64, kernel_size=(3,3), activation="relu"),
        layers.BatchNormalization(),
        layers.MaxPooling2D(pool_size=(2,2)),
        layers.Conv2D(128, kernel_size=(3,3), activation="relu"),
        layers.BatchNormalization(),
        layers.MaxPooling2D(pool_size=(2,2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="sigmoid"),
    ]
)
model.summary()
keras.utils.plot_model(model)
```

```
lr_schedule = keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=0.01,
    decay_steps=10000,
    decay_rate=0.9)
opt_func = keras.optimizers.Adam(learning_rate=lr_schedule)

checkpoint_filepath = 'saved_model'
checkpoint_callback = keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    monitor='val_binary_accuracy',
    save_weights_only=True,
    save_best_only=True)

model.compile(loss='binary_crossentropy',
              optimizer=opt_func,
              metrics=[
                keras.metrics.BinaryAccuracy(name="binary_accuracy", dtype=None, threshold=0.5),
                tf.keras.metrics.AUC(name="auc", from_logits=True),
              ],
             )

history = model.fit(X_train.reshape((-1,32,32,1)),
          y_train,
```

```
        epochs=30,
        validation_split=0.2,
        batch_size=32,
        shuffle=True,
        callbacks=[checkpoint_callback])
```

```
9960/9960 [==============================] - 53s 5ms/step - loss: 0.5910 - binary_accuracy: 0.6993 - auc:
Epoch 3/30
9960/9960 [==============================] - 52s 5ms/step - loss: 0.5807 - binary_accuracy: 0.7077 - auc:
Epoch 4/30
9960/9960 [==============================] - 53s 5ms/step - loss: 0.5741 - binary_accuracy: 0.7118 - auc:
Epoch 5/30
9960/9960 [==============================] - 53s 5ms/step - loss: 0.5687 - binary_accuracy: 0.7155 - auc:
Epoch 6/30
9960/9960 [==============================] - 54s 5ms/step - loss: 0.5654 - binary_accuracy: 0.7184 - auc:
Epoch 7/30
9960/9960 [==============================] - 53s 5ms/step - loss: 0.5613 - binary_accuracy: 0.7213 - auc:
Epoch 8/30
9960/9960 [==============================] - 54s 5ms/step - loss: 0.5591 - binary_accuracy: 0.7227 - auc:
Epoch 9/30
9960/9960 [==============================] - 58s 6ms/step - loss: 0.5568 - binary_accuracy: 0.7240 - auc:
Epoch 10/30
9960/9960 [==============================] - 55s 6ms/step - loss: 0.5545 - binary_accuracy: 0.7263 - auc:
Epoch 11/30
9960/9960 [==============================] - 53s 5ms/step - loss: 0.5527 - binary_accuracy: 0.7271 - auc:
Epoch 12/30
9960/9960 [==============================] - 55s 6ms/step - loss: 0.5511 - binary_accuracy: 0.7281 - auc:
Epoch 13/30
9960/9960 [==============================] - 54s 5ms/step - loss: 0.5499 - binary_accuracy: 0.7294 - auc:
Epoch 14/30
9960/9960 [==============================] - 54s 5ms/step - loss: 0.5482 - binary_accuracy: 0.7303 - auc:
Epoch 15/30
9960/9960 [==============================] - 54s 5ms/step - loss: 0.5469 - binary_accuracy: 0.7311 - auc:
Epoch 16/30
9960/9960 [==============================] - 55s 5ms/step - loss: 0.5459 - binary_accuracy: 0.7313 - auc:
Epoch 17/30
9960/9960 [==============================] - 55s 6ms/step - loss: 0.5445 - binary_accuracy: 0.7328 - auc:
Epoch 18/30
9960/9960 [==============================] - 51s 5ms/step - loss: 0.5437 - binary_accuracy: 0.7330 - auc:
Epoch 19/30
9960/9960 [==============================] - 56s 6ms/step - loss: 0.5426 - binary_accuracy: 0.7341 - auc:
Epoch 20/30
9960/9960 [==============================] - 55s 6ms/step - loss: 0.5416 - binary_accuracy: 0.7345 - auc:
Epoch 21/30
9960/9960 [==============================] - 55s 6ms/step - loss: 0.5416 - binary_accuracy: 0.7348 - auc:
Epoch 22/30
9960/9960 [==============================] - 52s 5ms/step - loss: 0.5410 - binary_accuracy: 0.7349 - auc:
Epoch 23/30
9960/9960 [==============================] - 55s 6ms/step - loss: 0.5401 - binary_accuracy: 0.7354 - auc:
Epoch 24/30
9960/9960 [==============================] - 56s 6ms/step - loss: 0.5395 - binary_accuracy: 0.7364 - auc:
Epoch 25/30
9960/9960 [==============================] - 56s 6ms/step - loss: 0.5392 - binary_accuracy: 0.7360 - auc:
Epoch 26/30
9960/9960 [==============================] - 52s 5ms/step - loss: 0.5389 - binary_accuracy: 0.7362 - auc:
Epoch 27/30
9960/9960 [==============================] - 56s 6ms/step - loss: 0.5381 - binary_accuracy: 0.7370 - auc:
Epoch 28/30
9960/9960 [==============================] - 56s 6ms/step - loss: 0.5377 - binary_accuracy: 0.7371 - auc:
Epoch 29/30
9960/9960 [==============================] - 55s 6ms/step - loss: 0.5374 - binary_accuracy: 0.7378 - auc:
Epoch 30/30
9960/9960 [==============================] - 56s 6ms/step - loss: 0.5368 - binary_accuracy: 0.7378 - auc:
```

```
model.load_weights(checkpoint_filepath)
_, accuracy, auc = model.evaluate(X_test.reshape((-1,32,32,1)), y_test)
print(f"Test accuracy: {accuracy}")
print(f"Test AUC: {auc}")
```

```
3113/3113 [==============================] - 11s 4ms/step - loss: 0.5513 - binary_accuracy: 0.7324 - auc: 0.
Test accuracy: 0.7324497699737549
Test AUC: 0.8002513647079468
```

```
import matplotlib.pyplot as plt
print(history.history.keys())
# summarize history for accuracy
plt.plot(history.history['binary_accuracy'])
plt.plot(history.history['val_binary_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'valid'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'valid'], loc='upper left')
plt.show()
```

```
dict_keys(['loss', 'binary_accuracy', 'auc', 'val_loss', 'val_binary_accuracy', 'val_auc'])
```