```
import numpy as np
import h5py


hf = h5py.File('/content/drive/MyDrive/data/SingleElectronPt50_IMGCROPS_n249k_RHv1.hdf5','r')
x = hf.get('X')[:50000]
y = hf.get('y')[:50000]

x_electron = np.asarray(x)
y_electron = np.asarray(y)
x = None
y = None


from keras.optimizers.schedules.learning_rate_schedule import NoisyLinearCosineDecay
hf1 = h5py.File('/content/drive/MyDrive/data/SinglePhotonPt50_IMGCROPS_n249k_RHv1.hdf5','r')
x1 = hf1.get('X')[:50000]
y1 = hf1.get('y')[:50000]

x_photon = np.asarray(x1)
y_photon = np.asarray(y1)
x1 = None
y1 = None


x_data = np.concatenate((x_electron,x_photon),axis=0)
y_data = np.concatenate((y_electron,y_photon),axis=0)
avg_channel = np.mean(x_data[:,:, :, :2], axis=-1, keepdims=True)

# Concatenate the average channel with the original image
x_data = np.concatenate((x_data, avg_channel), axis=-1)


import tensorflow as tf
from tensorflow import keras
from keras import layers
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
%matplotlib inline


x_train, x_validate, y_train, y_validate = train_test_split(x_data, y_data, test_size=0.2, random_state=42)
x_test, x_val, y_test, y_val = train_test_split(x_validate, y_validate, test_size=0.5, random_state=42)

x_data = None
y_data = None


import tensorflow as tf

input_shape = (32, 32, 3)
num_classes = 1

def residual_block(inputs, filters):
    x = tf.keras.layers.Conv2D(filters, 3, padding='same', activation='relu')(inputs)
    x = tf.keras.layers.Conv2D(filters, 3, padding='same', activation=None)(x)
    if inputs.shape[-1] != filters:
        inputs = tf.keras.layers.Conv2D(filters, 1, padding='same', activation=None)(inputs)
    x = tf.keras.layers.add([x, inputs])
    return tf.keras.layers.Activation('relu')(x)

inputs = tf.keras.layers.Input(shape=input_shape)
x = tf.keras.layers.Conv2D(16, 3, padding='same', activation='relu')(inputs)
x = residual_block(x, 16)
x = residual_block(x, 16)
x = tf.keras.layers.MaxPooling2D(pool_size=2, strides=2)(x)
x = residual_block(x, 32)
x = residual_block(x, 32)
x = tf.keras.layers.MaxPooling2D(pool_size=2, strides=2)(x)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dense(256, activation='relu')(x)
outputs = tf.keras.layers.Dense(num_classes, activation='sigmoid')(x)

model1 = tf.keras.models.Model(inputs=inputs, outputs=outputs)
model1.summary()
```

```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_2 (InputLayer)           [(None, 32, 32, 3)]  0           []
```

```
 conv2d_10 (Conv2D)             (None, 32, 32, 16)   448         ['input_2[0][0]']

 conv2d_11 (Conv2D)             (None, 32, 32, 16)   2320        ['conv2d_10[0][0]']

 conv2d_12 (Conv2D)             (None, 32, 32, 16)   2320        ['conv2d_11[0][0]']

 add_4 (Add)                    (None, 32, 32, 16)   0           ['conv2d_12[0][0]',
                                                                  'conv2d_10[0][0]']

 activation_4 (Activation)      (None, 32, 32, 16)   0           ['add_4[0][0]']

 conv2d_13 (Conv2D)             (None, 32, 32, 16)   2320        ['activation_4[0][0]']

 conv2d_14 (Conv2D)             (None, 32, 32, 16)   2320        ['conv2d_13[0][0]']

 add_5 (Add)                    (None, 32, 32, 16)   0           ['conv2d_14[0][0]',
                                                                  'activation_4[0][0]']

 activation_5 (Activation)      (None, 32, 32, 16)   0           ['add_5[0][0]']

 max_pooling2d_2 (MaxPooling2D)  (None, 16, 16, 16)  0           ['activation_5[0][0]']

 conv2d_15 (Conv2D)             (None, 16, 16, 32)   4640        ['max_pooling2d_2[0][0]']

 conv2d_16 (Conv2D)             (None, 16, 16, 32)   9248        ['conv2d_15[0][0]']

 conv2d_17 (Conv2D)             (None, 16, 16, 32)   544         ['max_pooling2d_2[0][0]']

 add_6 (Add)                    (None, 16, 16, 32)   0           ['conv2d_16[0][0]',
                                                                  'conv2d_17[0][0]']

 activation_6 (Activation)      (None, 16, 16, 32)   0           ['add_6[0][0]']

 conv2d_18 (Conv2D)             (None, 16, 16, 32)   9248        ['activation_6[0][0]']

 conv2d_19 (Conv2D)             (None, 16, 16, 32)   9248        ['conv2d_18[0][0]']

 add_7 (Add)                    (None, 16, 16, 32)   0           ['conv2d_19[0][0]',
                                                                  'activation_6[0][0]']

 activation_7 (Activation)      (None, 16, 16, 32)   0           ['add_7[0][0]']

 max_pooling2d_3 (MaxPooling2D)  (None, 8, 8, 32)    0           ['activation_7[0][0]']

 flatten_1 (Flatten)            (None, 2048)         0           ['max_pooling2d_3[0][0]']

 dense_2 (Dense)                (None, 256)          524544      ['flatten_1[0][0]']

 dense_3 (Dense)                (None, 1)            257         ['dense_2[0][0]']

==================================================================================================
```

```
model1.compile(loss='binary_crossentropy',
               optimizer='nadam',
               metrics=['accuracy'])
model1.fit(x_train,y_train,validation_data=(x_val,y_val),epochs=20)
```

```
Epoch 1/20
2500/2500 [==============================] - 49s 11ms/step - loss: 0.6547 - accuracy: 0.6145 - val_loss: 0.6435 - val_acc
Epoch 2/20
2500/2500 [==============================] - 27s 11ms/step - loss: 0.6279 - accuracy: 0.6524 - val_loss: 0.6074 - val_acc
Epoch 3/20
2500/2500 [==============================] - 28s 11ms/step - loss: 0.5988 - accuracy: 0.6857 - val_loss: 0.5887 - val_acc
Epoch 4/20
2500/2500 [==============================] - 27s 11ms/step - loss: 0.5840 - accuracy: 0.7003 - val_loss: 0.5817 - val_acc
Epoch 5/20
2500/2500 [==============================] - 28s 11ms/step - loss: 0.5782 - accuracy: 0.7053 - val_loss: 0.5792 - val_acc
Epoch 6/20
2500/2500 [==============================] - 26s 11ms/step - loss: 0.5716 - accuracy: 0.7103 - val_loss: 0.5768 - val_acc
Epoch 7/20
2500/2500 [==============================] - 26s 10ms/step - loss: 0.5681 - accuracy: 0.7133 - val_loss: 0.5804 - val_acc
Epoch 8/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.5658 - accuracy: 0.7153 - val_loss: 0.5738 - val_acc
Epoch 9/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.5620 - accuracy: 0.7187 - val_loss: 0.5695 - val_acc
Epoch 10/20
2500/2500 [==============================] - 24s 10ms/step - loss: 0.5598 - accuracy: 0.7198 - val_loss: 0.5752 - val_acc
Epoch 11/20
2500/2500 [==============================] - 24s 10ms/step - loss: 0.5576 - accuracy: 0.7217 - val_loss: 0.5800 - val_acc
Epoch 12/20
2500/2500 [==============================] - 24s 10ms/step - loss: 0.5541 - accuracy: 0.7242 - val_loss: 0.5805 - val_acc
Epoch 13/20
2500/2500 [==============================] - 24s 10ms/step - loss: 0.5528 - accuracy: 0.7253 - val_loss: 0.5654 - val_acc
Epoch 14/20
2500/2500 [==============================] - 26s 10ms/step - loss: 0.5505 - accuracy: 0.7263 - val_loss: 0.5653 - val_acc
Epoch 15/20
2500/2500 [==============================] - 24s 9ms/step - loss: 0.5484 - accuracy: 0.7283 - val_loss: 0.5728 - val_accu
Epoch 16/20
2500/2500 [==============================] - 24s 10ms/step - loss: 0.5460 - accuracy: 0.7306 - val_loss: 0.5788 - val_acc
```

```
Epoch 17/20
2500/2500 [==============================] - 24s 10ms/step - loss: 0.5438 - accuracy: 0.7305 - val_loss: 0.5663 - val_acc
Epoch 18/20
2500/2500 [==============================] - 24s 9ms/step - loss: 0.5422 - accuracy: 0.7333 - val_loss: 0.5633 - val_accu
Epoch 19/20
2500/2500 [==============================] - 24s 9ms/step - loss: 0.5392 - accuracy: 0.7357 - val_loss: 0.5680 - val_accu
Epoch 20/20
2500/2500 [==============================] - 24s 9ms/step - loss: 0.5367 - accuracy: 0.7379 - val_loss: 0.5681 - val_accu
<keras.callbacks.History at 0x7f2b98107910>
```

```
from sklearn.metrics import roc_auc_score
pred_prob1 = model1.predict(x_test)
auc_score1 = roc_auc_score(y_test, pred_prob1[:])
auc_score1
```

```
313/313 [==============================] - 1s 3ms/step
0.7813479721317492
```

```
import tensorflow as tf

input_shape = (32, 32, 3)
num_classes = 1

def residual_block(inputs, filters):
    x = tf.keras.layers.Conv2D(filters, 3, padding='same', activation='relu')(inputs)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Conv2D(filters, 3, padding='same', activation=None)(x)
    x = tf.keras.layers.BatchNormalization()(x)
    if inputs.shape[-1] != filters:
        inputs = tf.keras.layers.Conv2D(filters, 1, padding='same', activation=None)(inputs)
        inputs = tf.keras.layers.BatchNormalization()(inputs)
    x = tf.keras.layers.add([x, inputs])
    return tf.keras.layers.Activation('relu')(x)

inputs = tf.keras.layers.Input(shape=input_shape)
x = tf.keras.layers.Conv2D(32, 3, padding='same', activation='relu')(inputs)
x = tf.keras.layers.BatchNormalization()(x)
x = residual_block(x, 32)
x = residual_block(x, 32)
x = tf.keras.layers.MaxPooling2D(pool_size=2, strides=2)(x)
x = residual_block(x, 64)
x = residual_block(x, 64)
x = tf.keras.layers.MaxPooling2D(pool_size=2, strides=2)(x)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dense(512, activation='relu')(x)
x = tf.keras.layers.Dropout(0.5)(x)
outputs = tf.keras.layers.Dense(num_classes, activation='sigmoid')(x)

model2 = tf.keras.models.Model(inputs=inputs, outputs=outputs)
model2.summary()
```

```
 batch_normalization_9 (BatchNo  (None, 16, 16, 64)  256         ['conv2d_29[0][0]']
 rmalization)

 add_11 (Add)                   (None, 16, 16, 64)   0           ['batch_normalization_9[0][0]',
                                                                  'activation_10[0][0]']

 activation_11 (Activation)     (None, 16, 16, 64)   0           ['add_11[0][0]']

 max_pooling2d_5 (MaxPooling2D)  (None, 8, 8, 64)    0           ['activation_11[0][0]']

 flatten_2 (Flatten)            (None, 4096)         0           ['max_pooling2d_5[0][0]']

 dense_4 (Dense)                (None, 512)          2097664     ['flatten_2[0][0]']

 dropout (Dropout)              (None, 512)          0           ['dense_4[0][0]']

 dense_5 (Dense)                (None, 1)            513         ['dropout[0][0]']

==================================================================================================
Total params: 2,269,377
Trainable params: 2,268,417
Non-trainable params: 960
__________________________________________________________________________________________________
```

```
from keras.optimizers import Adam

model2.compile(loss='binary_crossentropy',
               optimizer=Adam(learning_rate=0.009),
               metrics=['accuracy'])
model2.fit(x_train,y_train,validation_data=(x_val,y_val),epochs=20)
```

```
Epoch 1/20
2500/2500 [==============================] - 27s 10ms/step - loss: 0.8181 - accuracy: 0.5881 - val_loss: 0.6537 - val_acc
Epoch 2/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.6352 - accuracy: 0.6543 - val_loss: 0.6343 - val_acc
Epoch 3/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.6129 - accuracy: 0.6826 - val_loss: 0.8784 - val_acc
Epoch 4/20
2500/2500 [==============================] - 24s 10ms/step - loss: 0.6013 - accuracy: 0.6913 - val_loss: 0.6171 - val_acc
Epoch 5/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.5955 - accuracy: 0.6978 - val_loss: 0.6079 - val_acc
Epoch 6/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.5915 - accuracy: 0.6990 - val_loss: 0.5807 - val_acc
Epoch 7/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.5890 - accuracy: 0.7009 - val_loss: 0.5875 - val_acc
Epoch 8/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.5838 - accuracy: 0.7055 - val_loss: 0.6005 - val_acc
Epoch 9/20
2500/2500 [==============================] - 26s 10ms/step - loss: 0.5822 - accuracy: 0.7063 - val_loss: 0.6452 - val_acc
Epoch 10/20
2500/2500 [==============================] - 24s 10ms/step - loss: 0.5794 - accuracy: 0.7082 - val_loss: 0.5741 - val_acc
Epoch 11/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.5770 - accuracy: 0.7119 - val_loss: 0.6273 - val_acc
Epoch 12/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.5756 - accuracy: 0.7131 - val_loss: 0.5821 - val_acc
Epoch 13/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.5729 - accuracy: 0.7127 - val_loss: 0.5736 - val_acc
Epoch 14/20
2500/2500 [==============================] - 26s 10ms/step - loss: 0.5716 - accuracy: 0.7134 - val_loss: 0.5793 - val_acc
Epoch 15/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.5714 - accuracy: 0.7161 - val_loss: 1.4877 - val_acc
Epoch 16/20
2500/2500 [==============================] - 26s 10ms/step - loss: 0.5697 - accuracy: 0.7183 - val_loss: 0.6475 - val_acc
Epoch 17/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.5672 - accuracy: 0.7170 - val_loss: 0.8692 - val_acc
Epoch 18/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.5667 - accuracy: 0.7196 - val_loss: 0.5915 - val_acc
Epoch 19/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.5648 - accuracy: 0.7195 - val_loss: 0.6090 - val_acc
Epoch 20/20
2500/2500 [==============================] - 25s 10ms/step - loss: 0.5643 - accuracy: 0.7192 - val_loss: 0.5628 - val_acc
<keras.callbacks.History at 0x7f2b0cb22190>
```

```
from sklearn.metrics import roc_auc_score
pred_prob2 = model2.predict(x_test)
auc_score2 = roc_auc_score(y_test, pred_prob2[:])
auc_score2
```

```
313/313 [==============================] - 1s 4ms/step
0.7849721247491227
```

```
pred_final = 0.6*auc_score1 + 0.4*auc_score2
pred_final
```

```
0.7827976331786985
```

✓ 0s completed at 22:50