Setup

```
import h5py
import gc
import numpy as np
import tensorflow as tf
import tensorflow.keras as keras
from keras.models import Sequential, Model, load_model
from keras import optimizers
from keras import layers
from keras import regularizers
```

Data preparation

```
import h5py
import numpy as np
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 = 12
index = np.random.permutation(len(All_y))
# here the dataset is flattened
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)
X_train = np.expand_dims(X_train, -1)
X_test = np.expand_dims(X_test, -1)
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) (498000,)
(398400, 32, 32, 1) (99600, 32, 32, 1)
(398400,) (99600,)
```

Build ResNet Block

```
def convolutional_block(X, f, filters, stage, block, s=2):
```

```
      conv_name_base = 'res' + str(stage) + block + '_branch'
      bn_name_base =·'bn'·+·str(stage)·+·block·+·'_branch'

  ····F1,·F2,·F3·=·filters

      X_shortcut = X

      X = layers.Conv2D(filters=F1, kernel_size=(1, 1), strides=(s, s), padding='valid', name=conv_na
      X = layers.BatchNormalization(axis=3, name=bn_name_base + '2a')(X)
      X = layers.Activation('relu')(X)

      X = layers.Conv2D(filters=F2, kernel_size=(f, f), strides=(1, 1), padding='same', name=conv_nar
      X = layers.BatchNormalization(axis=3, name=bn_name_base + '2b')(X)
      X = layers.Activation('relu')(X)

      X = layers.Conv2D(filters=F3, kernel_size=(1, 1), strides=(1, 1), padding='valid', name=conv_na
      X = layers.BatchNormalization(axis=3, name=bn_name_base + '2c')(X)

      X_shortcut = layers.Conv2D(filters=F3, kernel_size=(1, 1), strides=(s, s), padding='valid', nar
      X_shortcut = layers.BatchNormalization(axis=3, name=bn_name_base + '1')(X_shortcut)

      X = layers.Add()([X, X_shortcut])
      X = layers.Activation('relu')(X)

      return X
```

```
def identity_block(X, f, filters, stage, block):

    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'
    F1, F2, F3 = filters

    X_shortcut = X

    X = layers.Conv2D(filters=F1, kernel_size=(1, 1), strides=(1, 1), padding='valid', name=conv_na
    X = layers.BatchNormalization(axis=3, name=bn_name_base + '2a')(X)
    X = layers.Activation('relu')(X)

    X = layers.Conv2D(filters=F2, kernel_size=(f, f), strides=(1, 1), padding='same', name=conv_nam
    X = layers.BatchNormalization(axis=3, name=bn_name_base + '2b')(X)
    X = layers.Activation('relu')(X)

    X = layers.Conv2D(filters=F3, kernel_size=(1, 1), strides=(1, 1), padding='valid', name=conv_na
    X = layers.BatchNormalization(axis=3, name=bn_name_base + '2c')(X)

    X = layers.Add()([X, X_shortcut])# SKIP Connection
    X = layers.Activation('relu')(X)

    return X
```

```
def ResNet(input_shape=(32, 32, 1)):

    X_input = layers.Input(input_shape)

    X = layers.ZeroPadding2D((3, 3))(X_input)

    X = layers.Conv2D(64, (7, 7), strides=(2, 2), name='conv1')(X)
    X = layers.BatchNormalization(axis=3, name='bn_conv1')(X)
```

```
    X = layers.Activation('relu')(X)
    X = layers.MaxPooling2D((3, 3), strides=(2, 2))(X)

    X = convolutional_block(X, f=3, filters=[32, 32, 64], stage=2, block='a', s=1)
    X = identity_block(X, 3, [32, 32, 64], stage=2, block='b')

    X = layers.GlobalMaxPooling2D()(X)

    model = Model(inputs=X_input, outputs=X, name='ResNet')

    return model
```

Model Compiling and Training

```
base_model = ResNet(input_shape=(32,32,1))
head_model = base_model.output
head_model = layers.Flatten()(head_model)
#·head_model·=·layers.Dense(128,·activation='relu',·name='fc1',·kernel_regularizer=regularizers.l2(
head_model·=·layers.Dense(1, activation='sigmoid', name='fc2', kernel_regularizer=regularizers.l2(0
model = Model(inputs=base_model.input, outputs=head_model)
gc.collect()
```

```
2022-03-26 11:14:07.254843: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] success
2022-03-26 11:14:07.331644: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] success
2022-03-26 11:14:07.332462: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] success
2022-03-26 11:14:07.333660: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlo
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-03-26 11:14:07.334017: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] success
2022-03-26 11:14:07.334764: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] success
2022-03-26 11:14:07.335410: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] success
2022-03-26 11:14:08.775983: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] success
2022-03-26 11:14:08.776826: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] success
2022-03-26 11:14:08.777487: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] success
2022-03-26 11:14:08.778068: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] Created de
146
```

```
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=0.01)

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

opt_func = keras.optimizers.Adam(learning_rate=1e-2)
model.compile(loss='binary_crossentropy',
              optimizer=opt_func,
              metrics=['accuracy',
                       keras.metrics.AUC(name="auc", from_logits=True),
                      ])


history = model.fit(X_train,
```

```
          y_train,
          epochs=20,
          validation_split=0.2,
          batch_size=32,
          shuffle=False,
          callbacks=[checkpoint_callback])
gc.collect()
```

```
activation_6 (Activation)       (None, 7, 7, 64)     0           add_1[0][0]
__________________________________________________________________________________________________
global_max_pooling2d (GlobalMax (None, 64)           0           activation_6[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 64)           0           global_max_pooling2d[0][0]
__________________________________________________________________________________________________
fc2 (Dense)                     (None, 1)            65          flatten[0][0]
==================================================================================================
Total params: 35,841
Trainable params: 35,073
Non-trainable params: 768
__________________________________________________________________________________________________
2022-03-26 11:14:15.374903: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation
2022-03-26 11:14:16.725141: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation
2022-03-26 11:14:17.709214: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185]
Epoch 1/20
2022-03-26 11:14:19.622428: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cuDNN
9960/9960 [==============================] - 78s 7ms/step - loss: 0.6094 - accuracy: 0.6800
Epoch 2/20
9960/9960 [==============================] - 73s 7ms/step - loss: 0.5769 - accuracy: 0.7076
Epoch 3/20
9960/9960 [==============================] - 75s 7ms/step - loss: 0.5669 - accuracy: 0.7157
Epoch 4/20
9960/9960 [==============================] - 76s 8ms/step - loss: 0.5623 - accuracy: 0.7191
Epoch 5/20
9960/9960 [==============================] - 73s 7ms/step - loss: 0.5585 - accuracy: 0.7216
Epoch 6/20
9960/9960 [==============================] - 73s 7ms/step - loss: 0.5563 - accuracy: 0.7233
Epoch 7/20
9960/9960 [==============================] - 72s 7ms/step - loss: 0.5551 - accuracy: 0.7250
Epoch 8/20
9960/9960 [==============================] - 72s 7ms/step - loss: 0.5533 - accuracy: 0.7257
Epoch 9/20
9960/9960 [==============================] - 72s 7ms/step - loss: 0.5519 - accuracy: 0.7273
Epoch 10/20
9960/9960 [==============================] - 73s 7ms/step - loss: 0.5510 - accuracy: 0.7279
Epoch 11/20
9960/9960 [==============================] - 73s 7ms/step - loss: 0.5501 - accuracy: 0.7286
Epoch 12/20
9960/9960 [==============================] - 72s 7ms/step - loss: 0.5490 - accuracy: 0.7291
Epoch 13/20
9960/9960 [==============================] - 73s 7ms/step - loss: 0.5479 - accuracy: 0.7296
Epoch 14/20
9960/9960 [==============================] - 75s 8ms/step - loss: 0.5470 - accuracy: 0.7304
Epoch 15/20
9960/9960 [==============================] - 72s 7ms/step - loss: 0.5462 - accuracy: 0.7305
Epoch 16/20
9960/9960 [==============================] - 73s 7ms/step - loss: 0.5454 - accuracy: 0.7307
Epoch 17/20
9960/9960 [==============================] - 72s 7ms/step - loss: 0.5449 - accuracy: 0.7317
Epoch 18/20
9960/9960 [==============================] - 72s 7ms/step - loss: 0.5439 - accuracy: 0.7323
Epoch 19/20
9960/9960 [==============================] - 75s 8ms/step - loss: 0.5432 - accuracy: 0.7330
Epoch 20/20
9960/9960 [==============================] - 72s 7ms/step - loss: 0.5433 - accuracy: 0.7328
1543
```

Model Test

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

```
3113/3113 [==============================] - 15s 5ms/step - loss: 0.5516 - accuracy: 0.7269 - a
Test accuracy: 0.7268574237823486
Test AUC: 0.7931223511695862
```

```
model.summary()
```

```
Model: "model"
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 32, 32, 1)]	0	
zero_padding2d (ZeroPadding2D)	(None, 38, 38, 1)	0	input_1[0][0]
conv1 (Conv2D)	(None, 16, 16, 64)	3200	zero_padding2d[0][0]
bn_conv1 (BatchNormalization)	(None, 16, 16, 64)	256	conv1[0][0]
activation (Activation)	(None, 16, 16, 64)	0	bn_conv1[0][0]
max_pooling2d (MaxPooling2D)	(None, 7, 7, 64)	0	activation[0][0]
res2a_branch2a (Conv2D)	(None, 7, 7, 32)	2080	max_pooling2d[0][0]
bn2a_branch2a (BatchNormalizati	(None, 7, 7, 32)	128	res2a_branch2a[0][0]
activation_1 (Activation)	(None, 7, 7, 32)	0	bn2a_branch2a[0][0]
res2a_branch2b (Conv2D)	(None, 7, 7, 32)	9248	activation_1[0][0]
bn2a_branch2b (BatchNormalizati	(None, 7, 7, 32)	128	res2a_branch2b[0][0]
activation_2 (Activation)	(None, 7, 7, 32)	0	bn2a_branch2b[0][0]
res2a_branch2c (Conv2D)	(None, 7, 7, 64)	2112	activation_2[0][0]
res2a_branch1 (Conv2D)	(None, 7, 7, 64)	4160	max_pooling2d[0][0]
bn2a_branch2c (BatchNormalizati	(None, 7, 7, 64)	256	res2a_branch2c[0][0]
bn2a_branch1 (BatchNormalizatio	(None, 7, 7, 64)	256	res2a_branch1[0][0]
add (Add)	(None, 7, 7, 64)	0	bn2a_branch2c[0][0] bn2a_branch1[0][0]
activation_3 (Activation)	(None, 7, 7, 64)	0	add[0][0]
res2b_branch2a (Conv2D)	(None, 7, 7, 32)	2080	activation_3[0][0]
bn2b_branch2a (BatchNormalizati	(None, 7, 7, 32)	128	res2b_branch2a[0][0]
activation_4 (Activation)	(None, 7, 7, 32)	0	bn2b_branch2a[0][0]
res2b_branch2b (Conv2D)	(None, 7, 7, 32)	9248	activation_4[0][0]

bn2b_branch2b (BatchNormalizati	(None, 7, 7, 32)	128	res2b_branch2b[0][0]
activation_5 (Activation)	(None, 7, 7, 32)	0	bn2b_branch2b[0][0]
res2b_branch2c (Conv2D)	(None, 7, 7, 64)	2112	activation_5[0][0]
bn2b_branch2c (BatchNormalizati	(None, 7, 7, 64)	256	res2b_branch2c[0][0]