```
import numpy as np
import h5py


hf = h5py.File('SingleElectronPt50_IMGCROPS_n249k_RHv1.hdf5', 'r')
hf1 = h5py.File('SinglePhotonPt50_IMGCROPS_n249k_RHv1.hdf5', 'r')


print(hf.keys())
print(hf1.keys())
```

```
<KeysViewHDF5 ['X', 'y']>
<KeysViewHDF5 ['X', 'y']>
```

```
#For electrons
e_X = hf.get('X')
print(e_X)
e_y = hf.get('y')
print(e_y)

#for protons
p_X = hf1.get('X')
print(p_X)
p_y = hf1.get('y')
print(p_y)
```

```
<HDF5 dataset "X": shape (249000, 32, 32, 2), type "<f4">
<HDF5 dataset "y": shape (249000,), type "<f4">
<HDF5 dataset "X": shape (249000, 32, 32, 2), type "<f4">
<HDF5 dataset "y": shape (249000,), type "<f4">
```

```
e_X = np.array(e_X)
print(e_X.shape)
e_y = np.array(e_y)
print(e_y.shape)

p_X = np.array(p_X)
print(p_X.shape)
p_y = np.array(p_y)
print(p_y.shape)
```

```
(249000, 32, 32, 2)
(249000,)
(249000, 32, 32, 2)
(249000,)
```

```
hf.close()
hf1.close()
```

```python
e_train = []
p_train = []
e_val = []
p_val = []
e_test = []
p_test = []
for i in range(0,199200):
  e_train.append(e_X[i])
  p_train.append(p_X[i])


for j in range(199200, 224100):
  e_val.append(e_X[j])
  p_val.append(p_X[j])


for k in range(224100, 249000):
  e_test.append(e_X[k])
  p_test.append(p_X[k])



X_train = np.concatenate([e_train, p_train])
X_val = np.concatenate([e_val, p_val])
X_test = np.concatenate([e_test, p_test])


print((np.asarray(e_train)).shape)
print((np.asarray(p_train)).shape)
print((np.asarray(e_val)).shape)
print((np.asarray(p_val)).shape)
print((np.asarray(e_test)).shape)
print((np.asarray(p_test)).shape)

print((np.asarray(X_train)).shape)
print((np.asarray(X_val)).shape)
print((np.asarray(X_test)).shape)
```

```
    (199200, 32, 32, 2)
    (199200, 32, 32, 2)
    (24900, 32, 32, 2)
    (24900, 32, 32, 2)
    (24900, 32, 32, 2)
    (24900, 32, 32, 2)
    (398400, 32, 32, 2)
    (49800, 32, 32, 2)
    (49800, 32, 32, 2)
```

```python
ey_train = []
py_train = []
ey_val = []
py_val = []
ey_test = []
py_test = []
for i in range(0,199200):
  ey_train.append(e_y[i])
```

```
      py_train.append(p_y[i])


    for j in range(199200, 224100):
      ey_val.append(e_y[j])
      py_val.append(p_y[j])


    for k in range(224100, 249000):
      ey_test.append(e_y[k])
      py_test.append(p_y[k])



Y_train = np.concatenate([ey_train, py_train])
Y_val = np.concatenate([ey_val, py_val])
Y_test = np.concatenate([ey_test, py_test])


print((np.asarray(ey_train)).shape)
print((np.asarray(py_train)).shape)
print((np.asarray(ey_val)).shape)
print((np.asarray(py_val)).shape)
print((np.asarray(ey_test)).shape)
print((np.asarray(py_test)).shape)


print((np.asarray(Y_train)).shape)
print((np.asarray(Y_val)).shape)
print((np.asarray(Y_test)).shape)
```

```
    (199200,)
    (199200,)
    (24900,)
    (24900,)
    (24900,)
    (24900,)
    (398400,)
    (49800,)
    (49800,)
```

```
import keras
from keras import backend as K
from keras.applications import InceptionV3
from keras.layers import Input, Dense, Reshape, Flatten, Dropout, MaxPooling2D, GlobalMaxPool
import tensorflow as tf
from keras import layers
from keras.models import Sequential, Model
from keras.optimizers import RMSprop, SGD



ip=Input(shape=(32, 32, 2))

x = layers.Conv2D(256, (3, 3), activation='relu', padding='same', name='block5d_conv4')(ip)
x=BatchNormalization(momentum=0.8)(x)

x = layers.Conv2D(256, (3, 3), activation='relu', padding='same', name='block3d_conv4')(x)
```

```
x = layers.Conv2D(256, (3, 3), activation='relu', padding='same', name='block5d_conv4')(x)
x=BatchNormalization(momentum=0.8)(x)
x=Dropout(.3)(x)


x = layers.Conv2D(128, (3, 3), activation='relu', padding='same', name='block2d_conv2')(x)
x=BatchNormalization(momentum=0.8)(x)


x = layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='block1d_conv2')(x)
x=BatchNormalization(momentum=0.8)(x)
x=Dropout(.5)(x)


x=layers.Conv2D(3, (3, 3), activation='relu', padding='same')(x)
x=layers.Flatten()(x)
out=layers.Dense(1, activation="sigmoid")(x)
model=Model(ip,out)

print(model.summary())
```

```
Model: "model"

_____
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 32, 32, 2)]       0

block5d_conv4 (Conv2D)       (None, 32, 32, 256)       4864

batch_normalization (BatchNo (None, 32, 32, 256)       1024

block3d_conv4 (Conv2D)       (None, 32, 32, 256)       590080

batch_normalization_1 (Batch (None, 32, 32, 256)       1024

dropout (Dropout)            (None, 32, 32, 256)       0

block2d_conv2 (Conv2D)       (None, 32, 32, 128)       295040

batch_normalization_2 (Batch (None, 32, 32, 128)       512

block1d_conv2 (Conv2D)       (None, 32, 32, 64)        73792

batch_normalization_3 (Batch (None, 32, 32, 64)        256

dropout_1 (Dropout)          (None, 32, 32, 64)        0

conv2d (Conv2D)              (None, 32, 32, 3)         1731

flatten (Flatten)            (None, 3072)              0

dense (Dense)                (None, 1)                 3073
=================================================================
Total params: 971,396
Trainable params: 969,988
Non-trainable params: 1,408
_____
None
```

```
opt=RMSprop(lr=0.0001)
model.compile(optimizer=opt,loss='binary_crossentropy',metrics=['accuracy'])
```

```
model.load_weights("t1_20.h5")
model.fit(X_train, Y_train, validation_data=(X_val, Y_val), batch_size=32,epochs=10,verbose=1
```

```
    Epoch 1/10
    12450/12450 [==============================] - 834s 67ms/step - loss: 0.5614 - accuracy
    Epoch 2/10
    12450/12450 [==============================] - 832s 67ms/step - loss: 0.5606 - accuracy
    Epoch 3/10
    12450/12450 [==============================] - 831s 67ms/step - loss: 0.5598 - accuracy
    Epoch 4/10
    12450/12450 [==============================] - 832s 67ms/step - loss: 0.5593 - accuracy
    Epoch 5/10
    12450/12450 [==============================] - 829s 67ms/step - loss: 0.5587 - accuracy
    Epoch 6/10
    12450/12450 [==============================] - 823s 66ms/step - loss: 0.5581 - accuracy
    Epoch 7/10
    12450/12450 [==============================] - 828s 66ms/step - loss: 0.5576 - accuracy
    Epoch 8/10
    12450/12450 [==============================] - 828s 67ms/step - loss: 0.5563 - accuracy
    Epoch 9/10
    12450/12450 [==============================] - 830s 67ms/step - loss: 0.5561 - accuracy
    Epoch 10/10
    12450/12450 [==============================] - 830s 67ms/step - loss: 0.5554 - accuracy
    <tensorflow.python.keras.callbacks.History at 0x7fdc7f765b10>
```

```
model.save_weights("t1_30.h5")
```

```
model.load_weights("t1_30.h5")
model.fit(X_train, Y_train, validation_data=(X_val, Y_val), batch_size=32,epochs=20,verbose=1
```

```
    Epoch 1/20
    12450/12450 [==============================] - 889s 69ms/step - loss: 0.5552 - accuracy
    Epoch 2/20
    12450/12450 [==============================] - 852s 68ms/step - loss: 0.5534 - accuracy
    Epoch 3/20
    12450/12450 [==============================] - 843s 68ms/step - loss: 0.5542 - accuracy
    Epoch 4/20
    12450/12450 [==============================] - 839s 67ms/step - loss: 0.5532 - accuracy
    Epoch 5/20
    12450/12450 [==============================] - 841s 68ms/step - loss: 0.5509 - accuracy
    Epoch 6/20
    12450/12450 [==============================] - 841s 68ms/step - loss: 0.5517 - accuracy
    Epoch 7/20
    12450/12450 [==============================] - 841s 68ms/step - loss: 0.5517 - accuracy
    Epoch 8/20
    12450/12450 [==============================] - 841s 68ms/step - loss: 0.5506 - accuracy
    Epoch 9/20
    12450/12450 [==============================] - 840s 67ms/step - loss: 0.5508 - accuracy
    Epoch 10/20
```

```
12450/12450 [==============================] - 841s 68ms/step - loss: 0.5493 - accuracy
Epoch 11/20
12450/12450 [==============================] - 840s 68ms/step - loss: 0.5510 - accuracy
Epoch 12/20
12450/12450 [==============================] - 840s 67ms/step - loss: 0.5499 - accuracy
Epoch 13/20
12450/12450 [==============================] - 838s 67ms/step - loss: 0.5486 - accuracy
Epoch 14/20
12450/12450 [==============================] - 839s 67ms/step - loss: 0.5481 - accuracy
Epoch 15/20
12450/12450 [==============================] - 837s 67ms/step - loss: 0.5464 - accuracy
Epoch 16/20
12450/12450 [==============================] - 837s 67ms/step - loss: 0.5473 - accuracy
Epoch 17/20
12450/12450 [==============================] - 838s 67ms/step - loss: 0.5468 - accuracy
Epoch 18/20
12450/12450 [==============================] - 839s 67ms/step - loss: 0.5458 - accuracy
Epoch 19/20
12450/12450 [==============================] - 839s 67ms/step - loss: 0.5463 - accuracy
Epoch 20/20
12450/12450 [==============================] - 837s 67ms/step - loss: 0.5436 - accuracy
<tensorflow.python.keras.callbacks.History at 0x7f0a90671190>
```

```
model.save_weights("t1_50.h5")
```