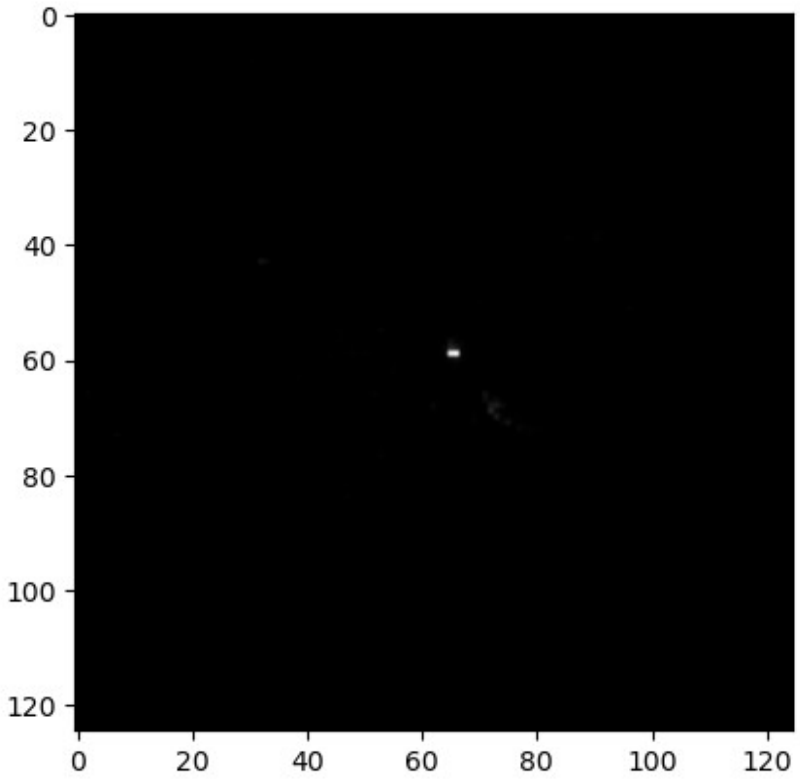
```
import numpy as np
import h5py

import tensorflow as tf
from tensorflow.keras.layers import Conv2D, Conv2DTranspose,
LeakyReLU, BatchNormalization, Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

PATH = r"/Users/suyashsachdeva/Desktop/gsoc_data.hdf5"
with h5py.File(PATH, 'r') as f:
    data = f['X_jets'][:]

dataset = np.zeros((data.shape[0], 128, 128, 3))
for c in range(data.shape[0]):
    dataset[c, 1:126, 1:126, :] = data[c]

import matplotlib.pyplot as plt
plt.imshow(data[1, :, :, 1], cmap="grey")
```

```
<matplotlib.image.AxesImage at 0x31a7ad850>
```




```
# COnvolutional Block for the autoencoder
def Conv(x, filters, kernel, strides, pad="same", momentum=0.99,
```

```
alpha=0.1, transpose=False, gf=2):
    if transpose:     # condition for a convolutional or convolutional
transpose operation
        x = Conv2DTranspose(filters, kernel, padding=pad,
strides=strides)(x)
        filters = filters//2
    else:
        x = Conv2D(filters, kernel, padding=pad, strides=strides)(x)
        filters = filters * gf
    x = BatchNormalization(momentum=momentum)(x)
    x = LeakyReLU(alpha)(x)
    return x

# Function for autoencoder
def AutoEncodeer():
    inp = Input((128, 128, 3))
    x = Conv(inp, 64, 3, 2)
    x = Conv(x, 128, 3, 2)
    x = Conv(x, 256, 3, 2)
    x = Conv(x, 512, 3, 2)
    x = Conv(x, 256, 3, 2, transpose=True)
    x = Conv(x, 128, 3, 2, transpose=True)
    x = Conv(x, 64, 3, 2, transpose=True)
    x = Conv(x, 3, 3, 2, transpose=True)
    model = Model(inputs=inp, outputs=x, name="autoencoder")
    return model
model = AutoEncodeer()
model.summary()
```

```
Model: "autoencoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 128, 128, 3)]     0

 conv2d_8 (Conv2D)           (None, 64, 64, 64)        1792

 batch_normalization_16 (Bat  (None, 64, 64, 64)       256
 chNormalization)

 leaky_re_lu_16 (LeakyReLU)  (None, 64, 64, 64)        0

 conv2d_9 (Conv2D)           (None, 32, 32, 128)       73856

 batch_normalization_17 (Bat  (None, 32, 32, 128)      512
 chNormalization)

 leaky_re_lu_17 (LeakyReLU)  (None, 32, 32, 128)       0

 conv2d_10 (Conv2D)          (None, 16, 16, 256)       295168
```

```
 batch_normalization_18 (Bat  (None, 16, 16, 256)      1024
 chNormalization)

 leaky_re_lu_18 (LeakyReLU)  (None, 16, 16, 256)       0

 conv2d_11 (Conv2D)          (None, 8, 8, 512)         1180160

 batch_normalization_19 (Bat  (None, 8, 8, 512)        2048
 chNormalization)

 leaky_re_lu_19 (LeakyReLU)  (None, 8, 8, 512)         0

 conv2d_transpose_8 (Conv2DT  (None, 16, 16, 256)      1179904
 ranspose)

 batch_normalization_20 (Bat  (None, 16, 16, 256)      1024
 chNormalization)

 leaky_re_lu_20 (LeakyReLU)  (None, 16, 16, 256)       0

 conv2d_transpose_9 (Conv2DT  (None, 32, 32, 128)      295040
 ranspose)

 batch_normalization_21 (Bat  (None, 32, 32, 128)      512
 chNormalization)

 leaky_re_lu_21 (LeakyReLU)  (None, 32, 32, 128)       0

 conv2d_transpose_10 (Conv2D  (None, 64, 64, 64)       73792
 Transpose)

 batch_normalization_22 (Bat  (None, 64, 64, 64)       256
 chNormalization)

 leaky_re_lu_22 (LeakyReLU)  (None, 64, 64, 64)        0

 conv2d_transpose_11 (Conv2D  (None, 128, 128, 3)      1731
 Transpose)

 batch_normalization_23 (Bat  (None, 128, 128, 3)      12
 chNormalization)

 leaky_re_lu_23 (LeakyReLU)  (None, 128, 128, 3)       0

=================================================================
Total params: 3,107,087
Trainable params: 3,104,265
```

```
Non-trainable params: 2,822
_________________________________________________________________
```

```python
# PSNR calculation function
def psnr(img1, img2):
  return tf.image.psnr(img1, img2, max_val=1.0)

# SSIM calculation function
def ssim(image1, image2):
    image1 = tf.image.convert_image_dtype(image1, tf.float32)
    image2 = tf.image.convert_image_dtype(image2, tf.float32)
    ssim = tf.image.ssim(image1, image2, max_val=1.0)
    return ssim

traindata = dataset[:100000]
validdata = dataset[100000:]

model.compile(loss="mae", optimizer="adam", metrics=[ "mse", psnr,
ssim])
history = model.fit(traindata, traindata, epochs=20, verbose=1,
batch_size=64, validation_data=[validdata, validdata],
validation_batch_size=64,)
```

```
Epoch 1/20

2024-04-02 01:57:36.713684: I
tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc
:114] Plugin optimizer for device_type GPU is enabled.

1563/1563 [==============================] - ETA: 0s - loss: 0.0010 -
mse: 0.0269 - psnr: 67.8820 - ssim: 0.9926

2024-04-02 02:02:04.521477: I
tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc
:114] Plugin optimizer for device_type GPU is enabled.

1563/1563 [==============================] - 312s 199ms/step - loss:
0.0010 - mse: 0.0269 - psnr: 67.8820 - ssim: 0.9926 - val_loss:
7.4645e-04 - val_mse: 1.2150 - val_psnr: 54.3878 - val_ssim: 0.9935
Epoch 2/20
1563/1563 [==============================] - 301s 193ms/step - loss:
2.8068e-04 - mse: 0.0053 - psnr: 72.0186 - ssim: 0.9982 - val_loss:
4.2124e-04 - val_mse: 0.0215 - val_psnr: 63.0592 - val_ssim: 0.9971
Epoch 3/20
1563/1563 [==============================] - 298s 191ms/step - loss:
1.9840e-04 - mse: 0.0023 - psnr: 74.0360 - ssim: 0.9985 - val_loss:
2.5567e-04 - val_mse: 0.0050 - val_psnr: 62.9795 - val_ssim: 0.9970
Epoch 4/20
1563/1563 [==============================] - 296s 189ms/step - loss:
8.1740e-05 - mse: 2.8038e-04 - psnr: 80.6480 - ssim: 0.9993 -
val_loss: 5.4107e-05 - val_mse: 3.0474e-04 - val_psnr: 74.1888 -
```

```
val_ssim: 0.9989
Epoch 5/20
1563/1563 [==============================] - 302s 193ms/step - loss:
4.0912e-05 - mse: 9.5064e-06 - psnr: 84.7899 - ssim: 0.9997 -
val_loss: 1.1764e-04 - val_mse: 5.1468e-04 - val_psnr: 66.0800 -
val_ssim: 0.9982
Epoch 6/20
1563/1563 [==============================] - 296s 190ms/step - loss:
4.2731e-05 - mse: 2.3324e-06 - psnr: 84.4097 - ssim: 0.9997 -
val_loss: 4.2862e-05 - val_mse: 3.0466e-04 - val_psnr: 75.9434 -
val_ssim: 0.9989
Epoch 7/20
1563/1563 [==============================] - 296s 189ms/step - loss:
3.9296e-05 - mse: 2.2561e-06 - psnr: 84.9990 - ssim: 0.9997 -
val_loss: 4.7988e-05 - val_mse: 3.0468e-04 - val_psnr: 75.1662 -
val_ssim: 0.9989
Epoch 8/20
1563/1563 [==============================] - 299s 191ms/step - loss:
3.9341e-05 - mse: 2.2597e-06 - psnr: 84.9892 - ssim: 0.9997 -
val_loss: 5.2309e-05 - val_mse: 3.0466e-04 - val_psnr: 74.5061 -
val_ssim: 0.9989
Epoch 9/20
1563/1563 [==============================] - 306s 196ms/step - loss:
3.9524e-05 - mse: 4.6691e-06 - psnr: 84.9992 - ssim: 0.9997 -
val_loss: 6.4394e-05 - val_mse: 3.0466e-04 - val_psnr: 73.1939 -
val_ssim: 0.9989
Epoch 10/20
1563/1563 [==============================] - 267s 170ms/step - loss:
3.9385e-05 - mse: 2.2603e-06 - psnr: 85.2135 - ssim: 0.9997 -
val_loss: 5.3882e-05 - val_mse: 3.0467e-04 - val_psnr: 73.6647 -
val_ssim: 0.9989
Epoch 11/20
1563/1563 [==============================] - 265s 169ms/step - loss:
3.9271e-05 - mse: 2.2601e-06 - psnr: 85.0290 - ssim: 0.9997 -
val_loss: 5.5498e-05 - val_mse: 3.0476e-04 - val_psnr: 74.2556 -
val_ssim: 0.9989
Epoch 12/20
1563/1563 [==============================] - 263s 169ms/step - loss:
3.9503e-05 - mse: 2.4469e-06 - psnr: 84.9013 - ssim: 0.9997 -
val_loss: 5.5738e-05 - val_mse: 3.0465e-04 - val_psnr: 73.8372 -
val_ssim: 0.9989
Epoch 13/20
1563/1563 [==============================] - 264s 169ms/step - loss:
3.9452e-05 - mse: 2.3429e-06 - psnr: 85.0016 - ssim: 0.9997 -
val_loss: 5.9011e-05 - val_mse: 3.0466e-04 - val_psnr: 73.5655 -
val_ssim: 0.9989
Epoch 14/20
1563/1563 [==============================] - 264s 169ms/step - loss:
3.9290e-05 - mse: 2.2582e-06 - psnr: 85.0269 - ssim: 0.9997 -
```

```
val_loss: 4.8312e-05 - val_mse: 3.0466e-04 - val_psnr: 75.2453 -
val_ssim: 0.9989
Epoch 15/20
1563/1563 [==============================] - 264s 169ms/step - loss:
3.9227e-05 - mse: 2.2594e-06 - psnr: 84.9916 - ssim: 0.9997 -
val_loss: 5.3768e-05 - val_mse: 3.0468e-04 - val_psnr: 74.4952 -
val_ssim: 0.9989
Epoch 16/20
1563/1563 [==============================] - 264s 169ms/step - loss:
3.9651e-05 - mse: 2.2601e-06 - psnr: 84.8512 - ssim: 0.9997 -
val_loss: 5.2239e-05 - val_mse: 3.0476e-04 - val_psnr: 74.8289 -
val_ssim: 0.9989
Epoch 17/20
1563/1563 [==============================] - 264s 169ms/step - loss:
3.9446e-05 - mse: 2.5827e-06 - psnr: 85.1884 - ssim: 0.9997 -
val_loss: 5.5690e-05 - val_mse: 3.0467e-04 - val_psnr: 73.8660 -
val_ssim: 0.9990
Epoch 18/20
1563/1563 [==============================] - 265s 169ms/step - loss:
3.9159e-05 - mse: 2.2588e-06 - psnr: 84.9814 - ssim: 0.9997 -
val_loss: 6.3735e-05 - val_mse: 3.0471e-04 - val_psnr: 73.0354 -
val_ssim: 0.9989
Epoch 19/20
1563/1563 [==============================] - 264s 169ms/step - loss:
3.9278e-05 - mse: 2.2623e-06 - psnr: 85.0912 - ssim: 0.9997 -
val_loss: 7.9856e-05 - val_mse: 3.0477e-04 - val_psnr: 69.9411 -
val_ssim: 0.9989
Epoch 20/20
1563/1563 [==============================] - 265s 169ms/step - loss:
4.1163e-05 - mse: 6.6978e-06 - psnr: 84.9864 - ssim: 0.9996 -
val_loss: 6.2313e-05 - val_mse: 3.1343e-04 - val_psnr: 67.5627 -
val_ssim: 0.9987
```

```
import matplotlib.pyplot as plt
true = dataset[100101]
output = model(dataset[100101].reshape(-1, 128, 128, 3))

plt.figure(figsize= (15, 3*5))

# Loop through each channel and create a subplot
for i in range(3):
    plt.subplot( 3, 2, i*2+1)  # 1 row, 3 columns, ith subplot
    plt.imshow(output[:, :, :, i][0], cmap='gray')  # Display ith
channel
    plt.axis('off')  # Hide axes for clarity

    plt.subplot(3, 2, i*2+2)  # 1 row, 3 columns, ith subplot
    plt.imshow(true[:, :, i], cmap='gray')  # Display ith channel
    plt.axis('off')  # Hide axes for clarity
```

```
plt.show()
```