1.Load data

train

In [2]:

```
import keras.backend as K
import os
print(os.listdir())
```

```
['HW12test', '.DS_Store', 'HW12test.zip', 'Untitled.ipynb', 'Hw12.pd
f', 'HW12train.zip', '.ipynb_checkpoints', 'HW12train']
```

In [9]:

```
import numpy as np
import cv2
import os
import glob
img_dir = "HW12train/" # Enter Directory of all images
data_path = os.path.join(img_dir,'*g')
files = glob.glob(data_path)
train_x = []
for f1 in files:
    img = cv2.imread(f1)
    train_x.append(img)


print(np.shape(train_x))
```

```
(18059, 64, 64, 3)
```

In [32]:

```
y=os.listdir('HW12train/')
# extract 1st 2 letters as labels
train_y=[ int(i[:2]) for i in y]
print(np.shape(y),np.shape(train_y))
```

```
(18059,) (18059,)
```

test

In [28]:

```
import cv2
import os
import glob
img_dir = "HW12test/" # Enter Directory of all images
data_path = os.path.join(img_dir,'*g')
files = glob.glob(data_path)
test_x = []
for f1 in files:
    img = cv2.imread(f1)
    test_x.append(img)
print(np.shape(test))
```

```
(2261, 64, 64, 3)
```

In [140]:

```
y=os.listdir('HW12test/')
# extract 1st 2 letters as labels
test_y=[ int(i[:2]) for i in y]
print(np.shape(y),np.shape(test_y))
```

```
(2261,) (2261,)
```

Scale the data

In [31]:

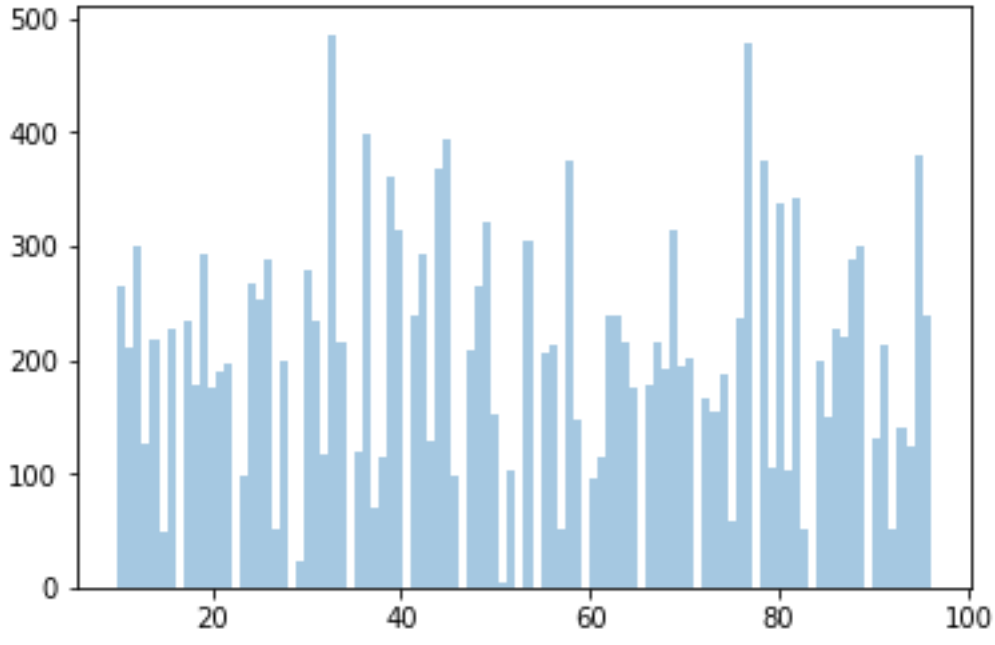
```
train_x=np.array(train_x)/255
test=np.array(test)/255
train_x.shape,test.shape
```

Out[31]:

```
((18059, 64, 64, 3), (2261, 64, 64, 3))
```

In [73]:

```
# check the distribution of target
import seaborn as sns
import matplotlib.pyplot as plt
sns.distplot(train_y,bins=100,kde=False)
plt.show()
```



Use keras build CNN

In [35]:

```
from keras import layers
from keras.layers import Input, Dense, Activation, ZeroPadding2D, Flatten, Conv2
D
from keras.layers import MaxPooling2D
from keras.models import Model
```

def the struture of CNN

In [302]:

```
def model(input_shape):
    # Define the input (64*64)*3. 3 means  R,G,B channels,  64*64 pixels. 18059
obs
    X_input = Input(input_shape)
    # (CONV -> pooling)*4 -> RELU
    X = Conv2D(32, (5,5), strides = (1, 1), name = 'conv1')(X_input)
    X = MaxPooling2D((2, 2),strides = (2, 2), name='max_pool1')(X)
    #
    X = Conv2D(32, (5, 5), strides = (1, 1), name = 'conv2')(X)
    X = MaxPooling2D((2, 2),strides = (1, 1), name='max_pool2')(X)
    #
    X = Conv2D(32, (5, 5), strides = (1, 1), name = 'conv3')(X)
    X = MaxPooling2D((2, 2),strides = (2, 2), name='max_pool3')(X)
    #
    X = Conv2D(64, (5, 5), strides = (1, 1), name = 'conv4')(X)
    X = Flatten()(X)
    X = Dense(1, activation='relu',kernel_initializer='random_normal', name='out
')(X)
    #X = Activation('relu')(X)
    # Create model. This creates your Keras model instance, you'll use this inst
ance to train/test the model.
    model = Model(inputs = X_input, outputs = X, name='resoluion')

    return model
```

In [188]:

```
def R_sq(y_true, y_pred):
    from keras import backend as K
    SS_res =  K.sum(K.square( y_true-y_pred ))
    SS_tot = K.sum(K.square( y_true - K.mean(y_true) ) )
    return ( 1 - SS_res/(SS_tot + K.epsilon()) )
```

In [ ]:

```
def residuals(y_true, y_pred):
    return (y_true- y_pred )
```

In [303]:

```
from keras import optimizers
rsmodel = model(train_x.shape[1:])
# define optimazier
ad = optimizers.Adam()
# compile model
rsmodel.compile(loss='mean_squared_error', optimizer=ad,metrics=[R_sq]) # SGD, A
dam.....
```

Mabey more epochs will better outcome. I will try it later

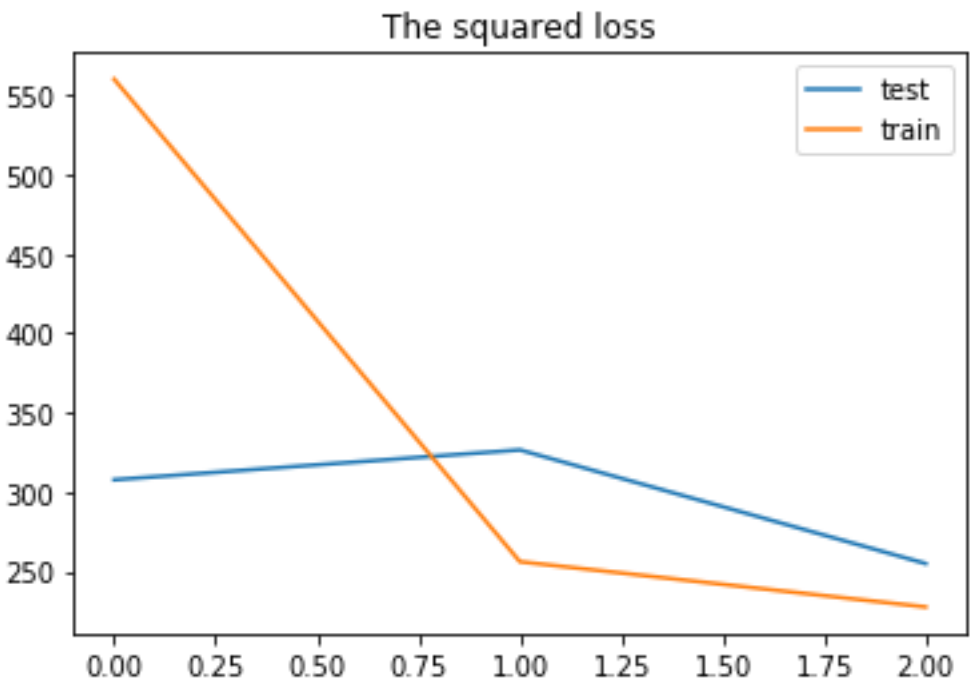
a)

In [ ]:

```
rsmodel.fit(train_x, train_y, epochs=3, batch_size=128,validation_data=(test_x,
test_y))
## change batch_size to 64,128,256,512, 1024 see what could happened.
```

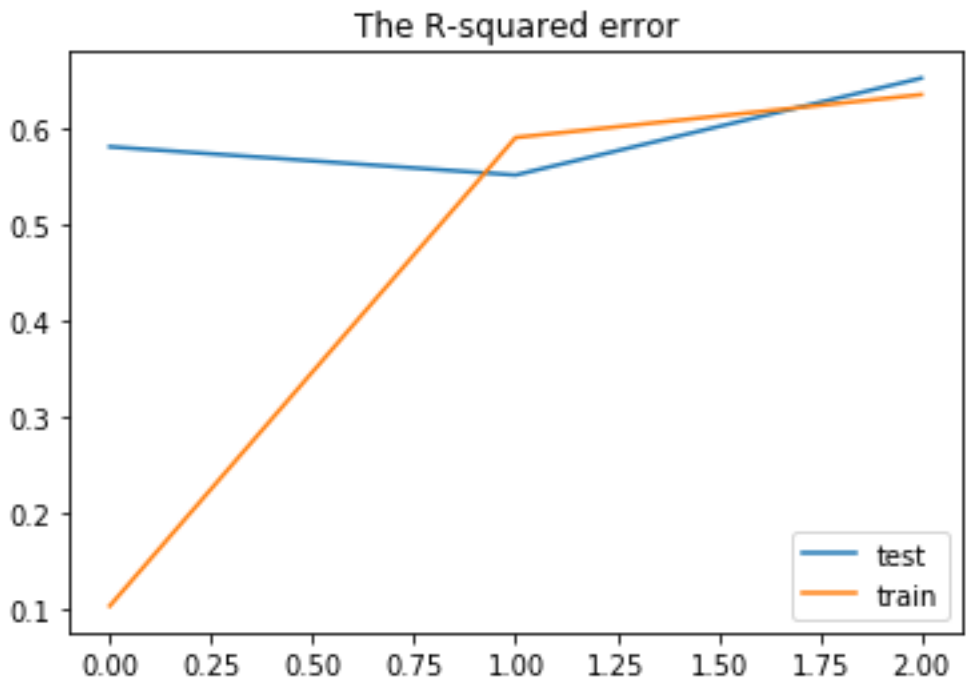
In [196]:

```
plt.plot(rsmodel.history.history['val_loss'],label='test')
plt.plot(rsmodel.history.history['loss'],label='train')
plt.legend()
plt.title('The squared loss')
plt.show()
```



In [195]:

```
plt.plot(rsmodel.history.history['val_R_sq'],label='test')
plt.plot(rsmodel.history.history['R_sq'],label='train')
plt.legend()
plt.title('The R-squared error')
plt.show()
```



b)

jittering

In [217]:

```
train_x1=[]
for i in range(train_x.shape[0]):
    k=np.random.choice(64, size=62, replace=True)
    train_x1.append(train_x[i,k][:,k])
```

In [291]:

```
test_x1=[]
for i in range(test_x.shape[0]):
    k=np.random.choice(64, size=62, replace=True)
    test_x1.append(test_x[i,k][:,k])
```

In [297]:

```
train_x1=np.array(train_x1)
test_x1=np.array(test_x1)
```

In [306]:

```
rsmodel = model(train_x1.shape[1:])
rsmodel.compile(loss='mean_squared_error', optimizer=ad,metrics=[R_sq]) # SGD, A
dam.....
```

In [307]:

```
rsmodel.fit(train_x1, train_y, epochs=3, batch_size=256,validation_data=(test_x1
, test_y))
## change batch_size to 64,128,256,512, 1024 see what could happened.
```

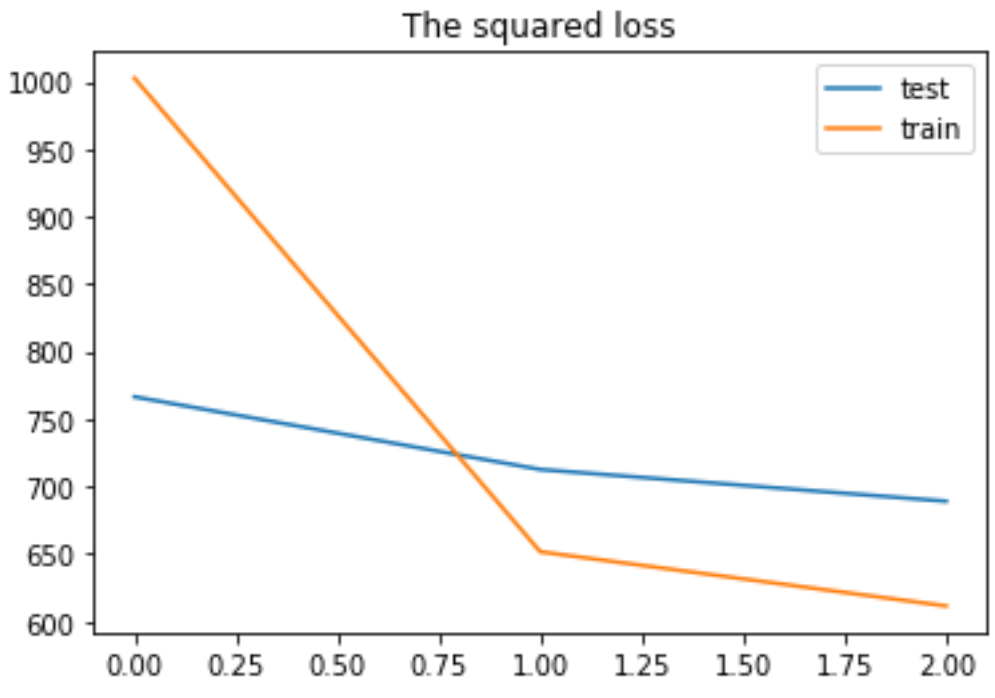
```
Train on 18059 samples, validate on 2261 samples
Epoch 1/3
18059/18059 [==============================] - 167s 9ms/step - loss:
1002.5260 - R_sq: -0.5924 - val_loss: 766.4923 - val_R_sq: -0.0180
Epoch 2/3
18059/18059 [==============================] - 156s 9ms/step - loss:
651.6875 - R_sq: -0.0348 - val_loss: 712.6505 - val_R_sq: 0.0516
Epoch 3/3
18059/18059 [==============================] - 173s 10ms/step - loss
: 611.5601 - R_sq: 0.0279 - val_loss: 689.1409 - val_R_sq: 0.0807
```

Out[307]:

```
<keras.callbacks.History at 0x19df8fe10>
```

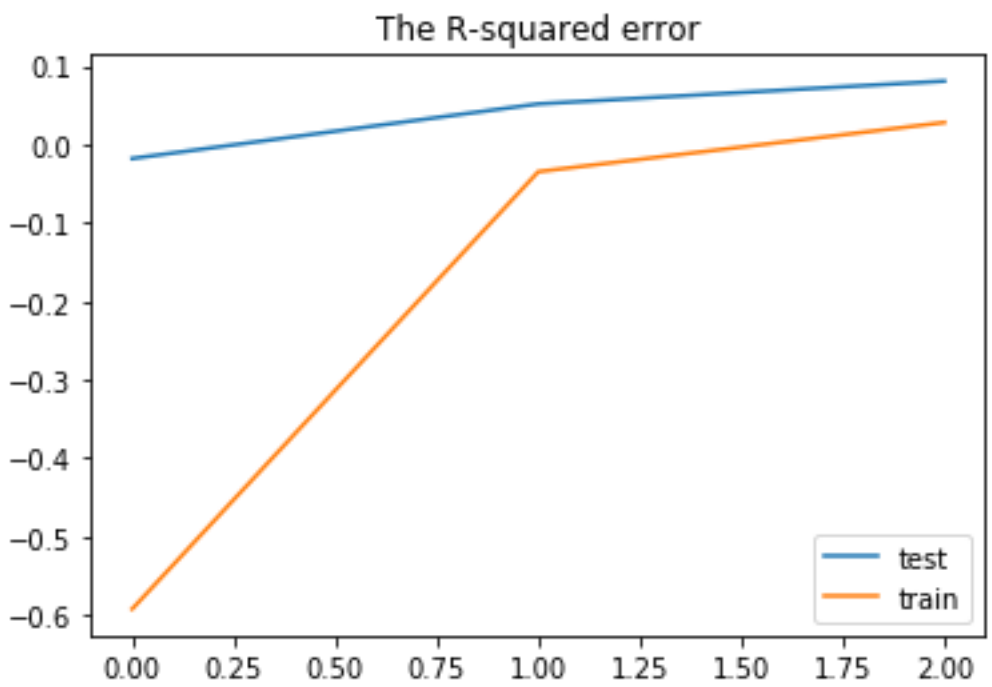
In [308]:

```
plt.plot(rsmodel.history.history['val_loss'],label='test')
plt.plot(rsmodel.history.history['loss'],label='train')
plt.legend()
plt.title('The squared loss')
plt.show()
```



In [309]:

```
plt.plot(rsmodel.history.history['val_R_sq'],label='test')
plt.plot(rsmodel.history.history['R_sq'],label='train')
plt.legend()
plt.title('The R-squared error')
plt.show()
```



c)

In [310]:

```
# self define lorenz

def lorenz(y, y_hat):
    return K.mean(K.log(K.square(y-y_hat)+1), axis=-1)
#/K.shape(val_y)
```

In [311]:

```
rsmodel.compile(loss=lorenz, optimizer=ad,metrics=[R_sq]) # SGD, Adam.....
rsmodel.fit(train_x1, train_y, epochs=3, batch_size=256,validation_data=(test_x1
, test_y))
```

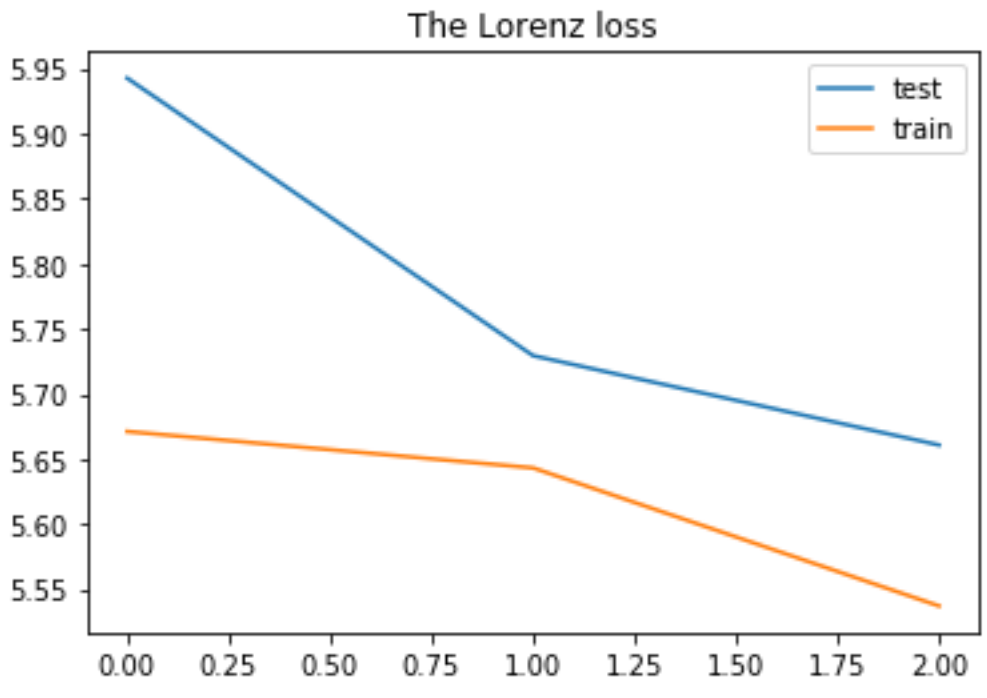
```
Train on 18059 samples, validate on 2261 samples
Epoch 1/3
18059/18059 [==============================] - 168s 9ms/step - loss:
5.6711 - R_sq: -0.3604 - val_loss: 5.9426 - val_R_sq: -0.4874
Epoch 2/3
18059/18059 [==============================] - 148s 8ms/step - loss:
5.6431 - R_sq: -0.3151 - val_loss: 5.7294 - val_R_sq: 0.0252
Epoch 3/3
18059/18059 [==============================] - 151s 8ms/step - loss:
5.5371 - R_sq: -0.1707 - val_loss: 5.6607 - val_R_sq: -0.0038
```

Out[311]:

```
<keras.callbacks.History at 0x1a1306128>
```

In [312]:

```
plt.plot(rsmodel.history.history['val_loss'],label='test')
plt.plot(rsmodel.history.history['loss'],label='train')
plt.legend()
plt.title('The Lorenz loss')
plt.show()
```



In [313]:

```
plt.plot(rsmodel.history.history['val_R_sq'],label='test')
plt.plot(rsmodel.history.history['R_sq'],label='train')
plt.legend()
plt.title('The R-squared error')
plt.show()
```