```
import os
from google.colab import drive
drive.mount('/content/drive')

path = "/content/drive/My Drive/暑期科研/"

os.chdir(path)
os.listdir(path)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
['chromosome_l_x_train.npy',
 'chromosome_l_x_test.npy',
 'chromosome_l_y_test.npy',
 'chromosome_l_y_train.npy',
 'chromosome_r_y_test.npy',
 'chromosome_r_y_train.npy',
 'chromosome_r_x_test.npy',
 'chromosome_r_x_train.npy']
```

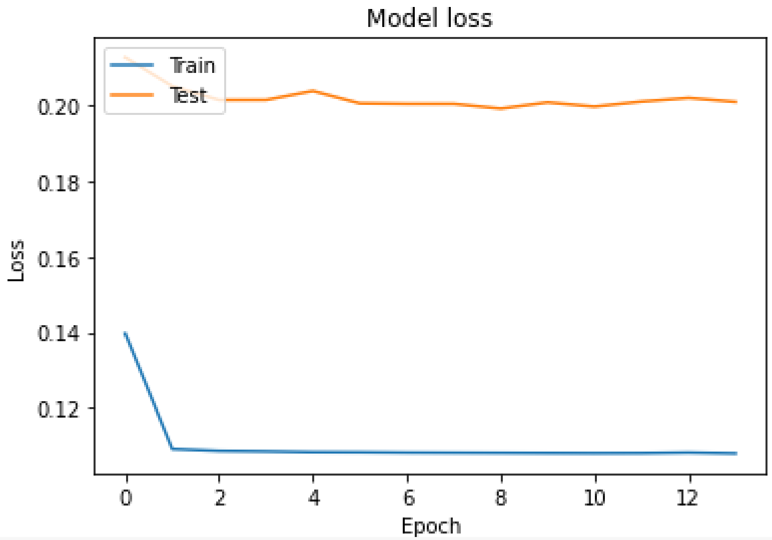
```
import glob

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


from keras.preprocessing import image
from keras.models import Model
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
from keras.layers import Input, Dense, Activation, BatchNormalization, Flatten, Conv2D
from keras.layers import MaxPooling2D, Dropout, UpSampling2D
import os

x_train_savepath = './chromosome_r_x_train.npy'
y_train_savepath = './chromosome_r_y_train.npy'
```

```
x_test_savepath = './chromosome_r_x_test.npy'
y_test_savepath = './chromosome_r_y_test.npy'
print('-------------Load Datasets-----------------')
x_train_save = np.load(x_train_savepath)
y_train = np.load(y_train_savepath)
x_test_save = np.load(x_test_savepath)
y_test = np.load(y_test_savepath)
x_train = np.reshape(x_train_save, (len(x_train_save), 150, 150,1))
x_test = np.reshape(x_test_save, (len(x_test_save), 150, 150,1))

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
# x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
# x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))
print(x_train.shape)
print(x_test.shape)
#

class Autoencoder():
    def __init__(self):

        self.img_shape = (150,150,1)

        optimizer = Adam(lr=0.001)

        self.autoencoder_model = self.build_model()
        self.autoencoder_model.compile(loss='binary_crossentropy', optimizer=optimizer)
        self.autoencoder_model.summary()

    def build_model(self):
        input_layer = Input(shape=self.img_shape)

        # encoder
        h = Conv2D(64, (3, 3), activation='relu', padding='same')(input_layer)
        h = MaxPooling2D((2, 2), padding='same')(h)
        h = Conv2D(64, (3, 3), activation='relu', padding='same')(h)
        h = MaxPooling2D((3, 3), padding='same')(h)
```

```
        # decoder
        h = Conv2D(64, (3, 3), activation='relu', padding='same')(h)
        h = UpSampling2D((3, 3))(h)
        h = Conv2D(64, (3, 3), activation='relu', padding='same')(h)
        h = UpSampling2D((2, 2))(h)

        output_layer = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(h)

        return Model(input_layer, output_layer)

    def train_model(self, x_train, y_train, x_test, y_test, epochs, batch_size):
        early_stopping = EarlyStopping(monitor='val_loss',
                                       min_delta=0,
                                       patience=5,
                                       verbose=1,
                                       mode='auto')
        history = self.autoencoder_model.fit(x_train, x_train,
                                             batch_size=batch_size,
                                             epochs=epochs,
                                             validation_data=(x_test, x_test),
                                             callbacks=[early_stopping])
        plt.plot(history.history['loss'])
        plt.plot(history.history['val_loss'])
        plt.title('Model loss')
        plt.ylabel('Loss')
        plt.xlabel('Epoch')
        plt.legend(['Train', 'Test'], loc='upper left')
        plt.show()

    def eval_model(self, x_test):
        preds = self.autoencoder_model.predict(x_test)
        return preds

ae = Autoencoder()
ae.train_model(x_train, x_train, x_test, x_test, epochs=20, batch_size=4)
```

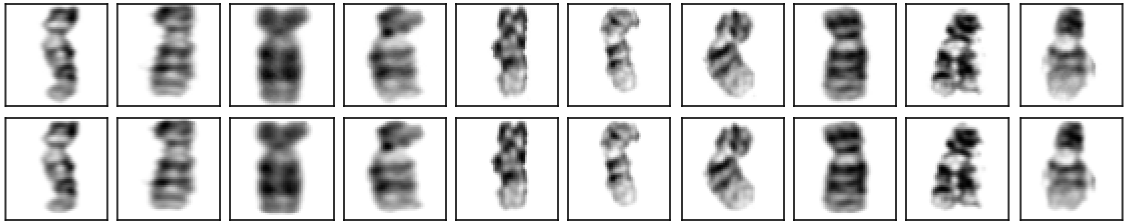
```
-------------Load Datasets----------------
(988, 150, 150, 1)
(188, 150, 150, 1)
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 150, 150, 1)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 150, 150, 64)      640
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 75, 75, 64)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 75, 75, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 25, 25, 64)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 25, 25, 64)        36928
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 75, 75, 64)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 75, 75, 64)        36928
_________________________________________________________________
up_sampling2d_4 (UpSampling2 (None, 150, 150, 64)      0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 150, 150, 1)       577
=================================================================
Total params: 112,001
Trainable params: 112,001
Non-trainable params: 0
_________________________________________________________________
Train on 988 samples, validate on 188 samples
Epoch 1/20
988/988 [==============================] - 7s 7ms/step - loss: 0.1397 - val_loss: 0.2127
Epoch 2/20
988/988 [==============================] - 6s 6ms/step - loss: 0.1092 - val_loss: 0.2051
Epoch 3/20
988/988 [==============================] - 6s 6ms/step - loss: 0.1087 - val_loss: 0.2014
Epoch 4/20
988/988 [==============================] - 6s 6ms/step - loss: 0.1086 - val_loss: 0.2015
Epoch 5/20
988/988 [==============================] - 6s 6ms/step - loss: 0.1084 - val_loss: 0.2038
```

```
Epoch 6/20
988/988 [==============================] - 6s 6ms/step - loss: 0.1084 - val_loss: 0.2005
Epoch 7/20
988/988 [==============================] - 6s 6ms/step - loss: 0.1083 - val_loss: 0.2004
Epoch 8/20
988/988 [==============================] - 6s 6ms/step - loss: 0.1082 - val_loss: 0.2004
Epoch 9/20
988/988 [==============================] - 6s 6ms/step - loss: 0.1082 - val_loss: 0.1992
Epoch 10/20
988/988 [==============================] - 6s 6ms/step - loss: 0.1082 - val_loss: 0.2008
Epoch 11/20
988/988 [==============================] - 6s 6ms/step - loss: 0.1081 - val_loss: 0.1997
Epoch 12/20
988/988 [==============================] - 6s 6ms/step - loss: 0.1081 - val_loss: 0.2010
Epoch 13/20
988/988 [==============================] - 6s 6ms/step - loss: 0.1083 - val_loss: 0.2020
Epoch 14/20
988/988 [==============================] - 6s 6ms/step - loss: 0.1081 - val_loss: 0.2009
Epoch 00014: early stopping
```




```
decoded_test = ae.eval_model(x_test)

import matplotlib.pyplot as plt
%matplotlib inline
shape = (150, 150)
fig, axes = plt.subplots(2,10
```

```
fig, axes = plt.subplots(2,10,
                         figsize=(10, 2),
                         subplot_kw={
                             'xticks': [],
                             'yticks': []
                         },
                         gridspec_kw=dict(hspace=0.1, wspace=0.1))
for i in range(10):
    axes[0][i].imshow(np.reshape(x_test[i], shape),cmap='gray')
    axes[1][i].imshow(np.reshape(decoded_test[i], shape),cmap='gray')
plt.show()
```



```
### get the error term of all testing dataset images
from sklearn.preprocessing import StandardScaler, MinMaxScaler
scaler = MinMaxScaler()
#scale it
scaled_input_test = scaler.fit_transform(x_test.reshape(-1,22500))
#scale it
scaled_output_test = scaler.fit_transform(decoded_test.reshape(-1,22500))
```

```
### get the error term of all training set images

# Firstly, we get the decoder image of training set.
decoded_train = ae.eval_model(x_train)

from sklearn.preprocessing import StandardScaler, MinMaxScaler
scaler = MinMaxScaler()
#scale it
scaled_input_train = scaler.fit_transform(x_train.reshape(-1,22500))
```

```
#scale it
scaled_output_train = scaler.fit_transform(decoded_train.reshape(-1,22500))
```

```
import pandas as pd
sequences = range(1,1177)
```

```
from keras import losses
x = losses.binary_crossentropy(scaled_input_train,scaled_output_train)
y = losses.binary_crossentropy(scaled_input_test,scaled_output_test)
```

```
seqs_ds = pd.DataFrame(sequences)
mse = np.append(x,y)
seqs_ds['MSE'] = mse
seqs_ds
```

```
mse_threshold  =  np.quantile(seqs_ds['MSE'],  0.84)
print(f'MSE  0.9  threshhold:{mse_threshold}')
```

```
MSE 0.9 threshhold:1.5023871660232544
```

2 3 1.418232

```
seqs_ds['MSE_Outlier']  =  0
seqs_ds.loc[seqs_ds['MSE']  >  mse_threshold,  'MSE_Outlier']  =  1
```

```
print(f"Num  of  MSE  outlier:{seqs_ds['MSE_Outlier'].sum()}")

seqs_ds[seqs_ds['MSE_Outlier']==1]
```

```
Num of MSE outlier:188
```

	0	MSE	MSE_Outlier
818	819	1.505959	1
988	989	2.506710	1
989	990	1.950874	1
990	991	1.784902	1
991	992	1.775405	1
...	...	...	...
1171	1172	1.845395	1
1172	1173	2.576621	1
1173	1174	2.489601	1
1174	1175	2.455830	1
1175	1176	2.434648	1

188 rows × 3 columns