Lab#4

Image Enhancement with AutoEncoder

Agenda

- 4.1 Data Preparation
- 4.2 Autoencoder
- 4.3 Hyperparameter Tuning

Libraries

```
#Array, image processing
import cv2
import numpy as np
import matplotlib.pyplot as plt
#Model Operation
from keras import Model, Input
import keras.utils as image
From keras.wrappers.scikit_learn import KerasRegressor
from tensorflow.keras.layers import Dense, Conv2D, MaxPool2D, UpSampling2D
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
# io
import glob
from tqdm import tqdm
import warnings;
warnings.filterwarnings('ignore')
```

4.1 DATA PREPARATION

Data Preparation

1. อ่านไฟล์ภาพทั้งหมดเก็บในในรูป array (จำนวนภาพไม่น้อยกว่า 100 ภาพ)

   glob.glob ()

   load_img(fname, target_size, interpolation="nearest") # target_size ไม่ควรเกิน (100,100)

   img_to_array(img)

2. Normalzied ภาพ (เพื่อให้ค่า pixel intensity = [0, 1])

   img/255

3. Append images to an array

4. แบ่งชุดข้อมูลเป็น Training_data, Testing_data (70 : 30)

   train_x, test_x = train_test_split(imgs, random_state=k, test_size=0.3)

5. แบ่งชุดข้อมูล Training_data เป็น Training_data, Validation_data (80:20)

   train_x, val_x = train_test_split(train_x, random_state=k, test_size=0.2)

6. กำหนด noise parameters

   noise distribution: normal, noise_mean = 0, noise_std = scalar, noise_factor = scalar

7. สร้าง noise บวกเพิ่มเข้าในภาพ train_x, val_x, test_x (เลือกอย่างน้อย 2 ชุดพารามิเตอร์เพื่อให้เห็นประมาณ noise ที่แตกต่าง)

   train_x_noise = train_x + (noise_factor * np.random.normal(loc=noise_mean, scale=noise_std, size=imgs.shape))

   val_x_noise = val_x + (noise_factor * np.random.normal(loc=noise_mean, scale=noise_std, size=imgs.shape))

   test_x_noise = test_x + (noise_factor * np.random.normal(loc=noise_mean, scale=noise_std, size=imgs.shape))

8. แสดงภาพเปรียบเทียบ ภาพที่เพิ่ม noise และภาพ ต้นฉบับ

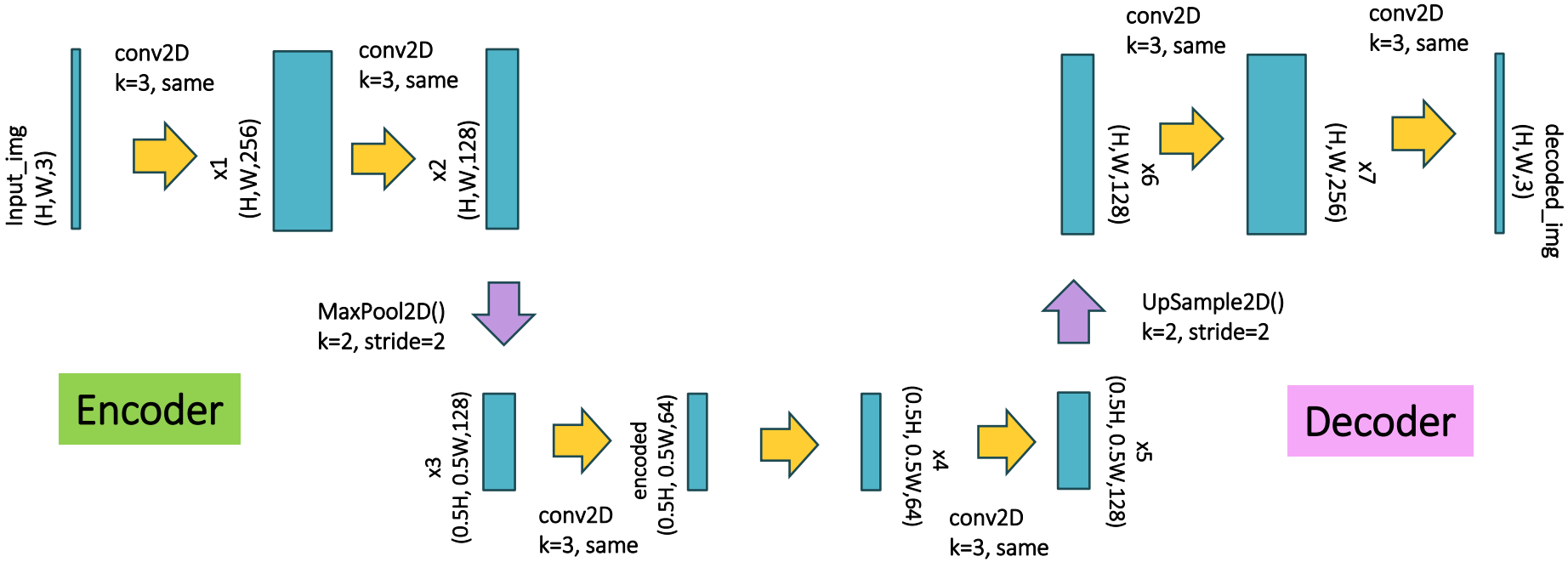
4.1

Adding Noise

Choose to plot at least 3 images

4.2 AUTOENCODER MODEL

Step#1: Create Layers of Encoder Model



1. **# กำหนด Object แต่ละเลเยอร์ของ Encoder Architecture**

   Input_img = Input(shape=(height, width, ch))

   #encoding architecture

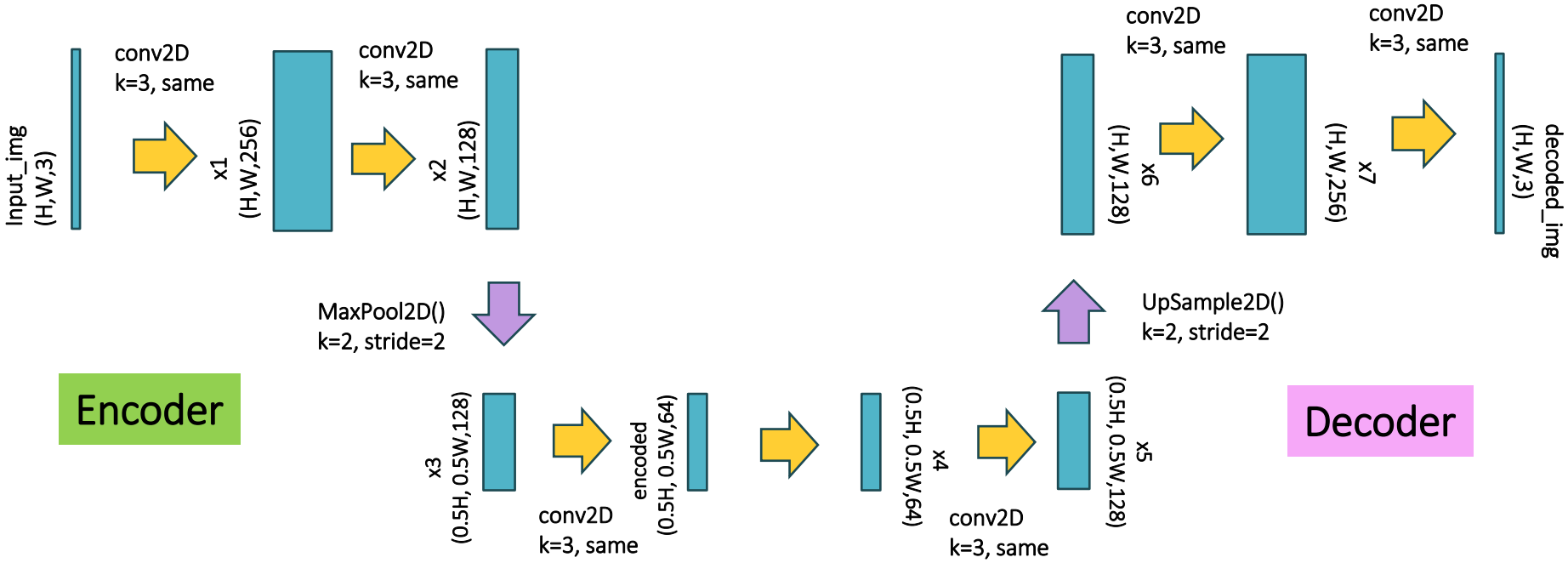
   Example Layer#1:

   x1 = Conv2D(256, (3, 3), activation='relu', padding='same')(Input_img)

   เขียน code เชื่อมต่อ Layer#2 – 5 จนได้ผลลัพธ์ จาก encoded (x4)

   ตามรูปภาพโครงข่าย

Step#2: Create Layers of Decoder Model



2. # กำหนด Object แต่ละเลเยอร์ของ Decoder Architecture

   # decoding architecture

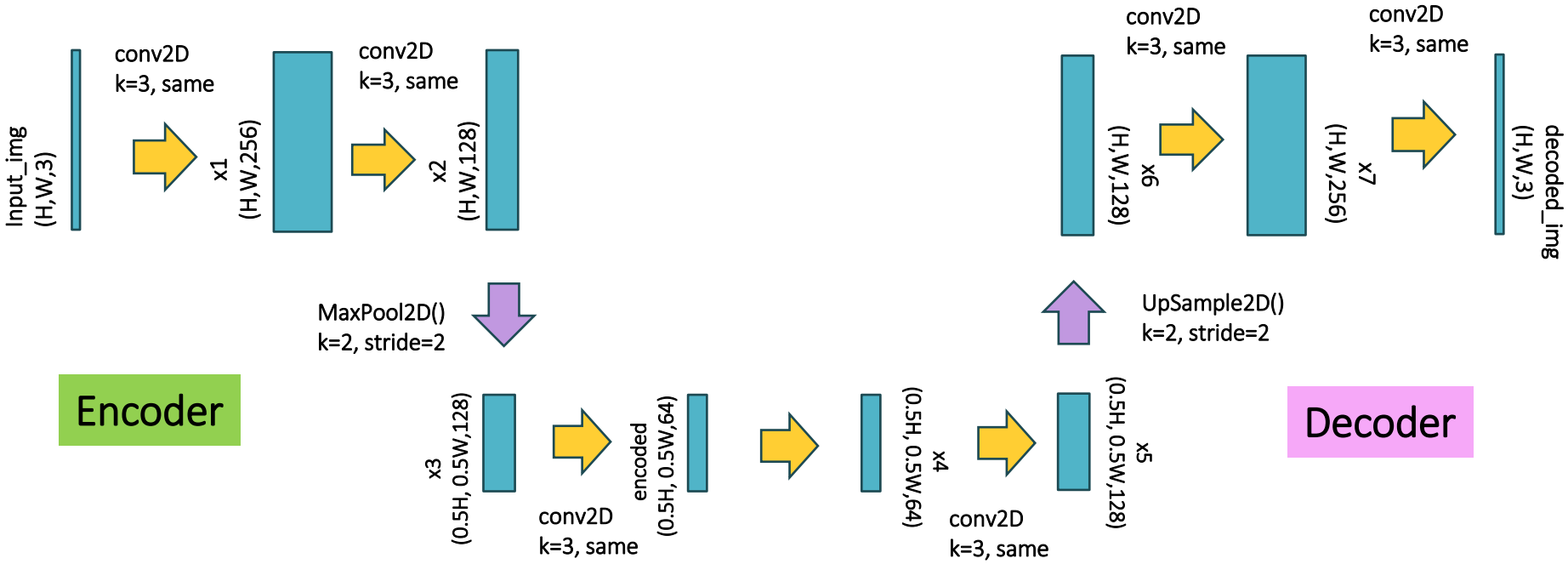
   Example Layer#6

   x4 = Conv2D(64, (3, 3), activation='relu', padding='same')(encoded)

   เขียน code เชื่อม encoded มายัง decoder Layer#6 – 10

   (ตามรูปภาพโครงข่าย)

Step#3: Connect Encoder and Decoder Model



3. # สร้างโมเดล autoencoder และกำหนด optimizer setting

```
autoencoder = Model(Input_img, decoded)
autoencoder.compile(optimizer='adam', loss='mse')
                #loss ใช้ Mean Square Error
autoencoder.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 80, 80, 3)]       0

 conv2d (Conv2D)             (None, 80, 80, 256)       7168

 conv2d_1 (Conv2D)           (None, 80, 80, 128)       295040

 max_pooling2d (MaxPooling2D  (None, 40, 40, 128)      0
 )

 conv2d_2 (Conv2D)           (None, 40, 40, 64)        73792

 conv2d_3 (Conv2D)           (None, 40, 40, 64)        36928

 up_sampling2d (UpSampling2D  (None, 80, 80, 64)       0
 )

 conv2d_4 (Conv2D)           (None, 80, 80, 128)       73856

 conv2d_5 (Conv2D)           (None, 80, 80, 256)       295168

 conv2d_6 (Conv2D)           (None, 80, 80, 3)         6915

=================================================================
Total params: 788,867
Trainable params: 788,867
Non-trainable params: 0
_________________________________________________________________
```

4.2

Autoencoder Model

Step#2: Training Autoencoder Model

1. # กำหนด Training parameter

   e = จำนวน epoch โดย กำหนดให้ เลือกทดลอง epoch = [2, 4, 8, 16] อย่างน้อย 1 ค่า

   b = batch_size = [16, 32, 64, 128] ทดลองอย่างน้อย 2 ค่า

   จำนวนภาพทั้งหมด ไม่น้อกว่า 1,000 ภาพ

2. # เริ่มการ training

```
callback = EarlyStopping(monitor='loss', patience=3)
history = autoencoder.fit(train_x_noisy, train_x,
                          epochs=eps,
                          batch_size=bs,
                          shuffle=True,
                          validation_data=(val_x_noisy, val_x),
                          callbacks=[callback],verbose=1)
```

3. # ทดสอบ autoencoder model ด้วย test data

```
predictions_test = autoencoder.predict(test_x_noisy)
```

4. #View Loss from history

```
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()
```

5. # Display Result image

Training autoencoder เก็บ Loss แต่ละรอบออกมา

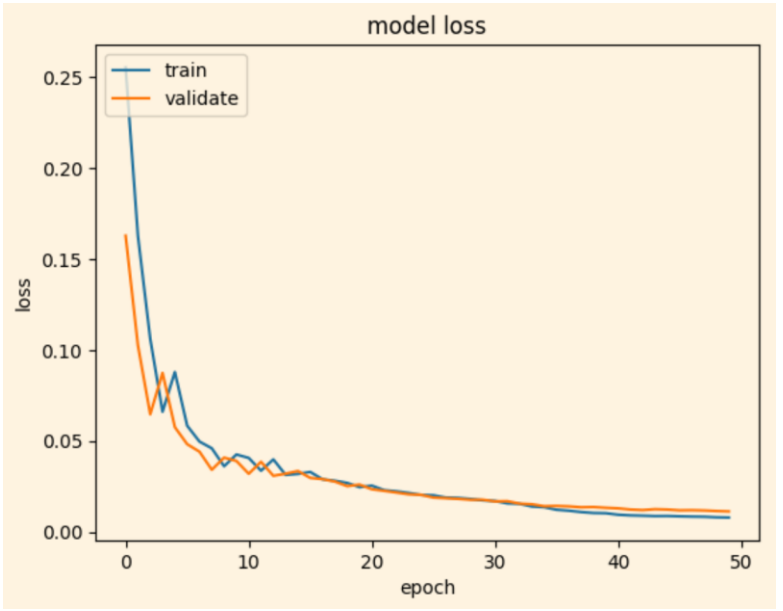
Predict results เมื่อผ่านภาพ validate, testing

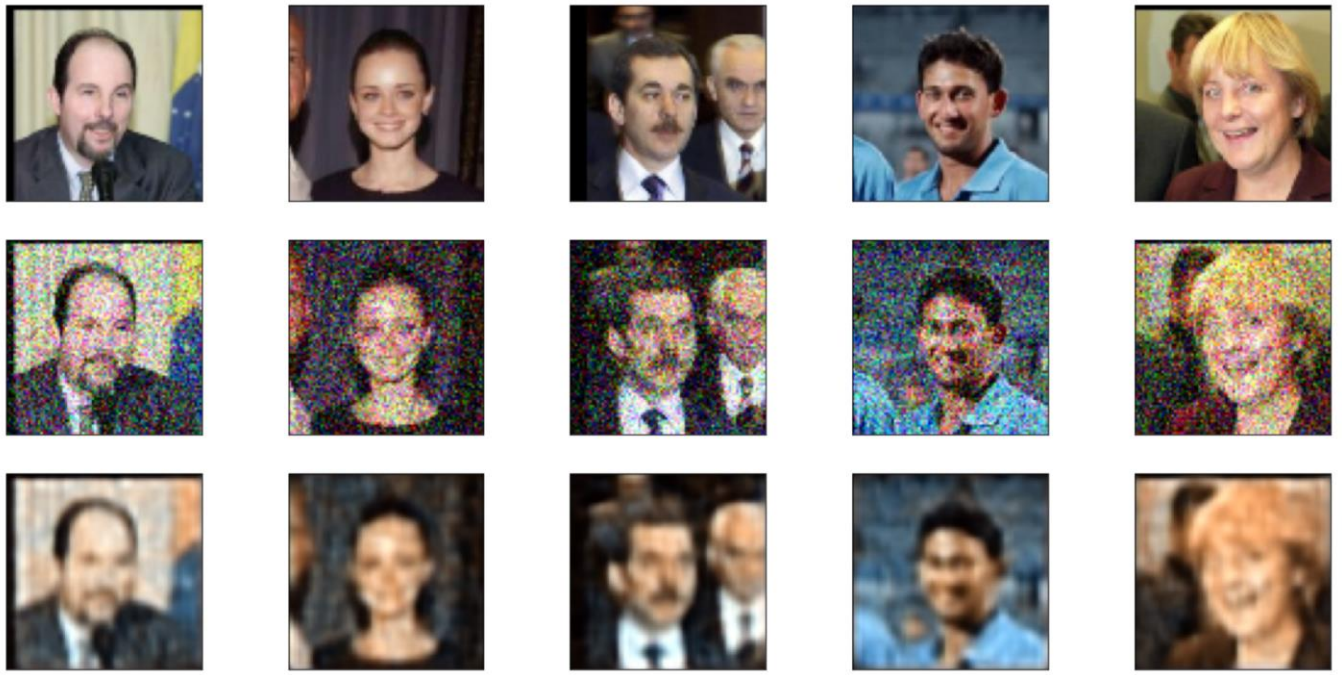
แสดงกราฟเปรียบเทียบผลจาก training และ validation Loss

แสดงรูปภาพ Input, noise, และ ผลลัพธ์จาก Testing images

4.2

Autoencoder Model





4.3 HYPERPARAMETER TUNING

Hyperparameter Tuning

1. สร้าง function create_autoencoder() โดยภายในกำหนดชั้นของ Convolutional layers ตาม 4.2 step 1
2. สร้าง KerasRegressor Model เพื่อทำหน้าที่เชื่อมพารามิเตอร์จาก GridSearchCV ไปยัง ฟังก์ชั่นสร้าง autoencoder

   model = KerasRegressor(build_fn=create_autoencoder, epochs=2, batch_size=16, verbose=0)
3. สร้าง Dict ของชุดพารามิเตอร์ เงื่อนไขพารามิเตอร์ตามตารางหน้าท้ายเอกสาร

   opts = [ ] , lnR = [ ], bs = [ ], eps = [ ]

   param_grid = dict(batch_size=bs, epochs=eps, optimizer=opts, learning=lnR)
4. สร้าง parameter set (ตามเงื่อนไขใน param_grid) และจัดการ Cross Validation ด้วย GridSearchCV

   grid = GridSearchCV(estimator=model, n_jobs=1, verbose= 10, cv=2, param_grid=param_grid)
5. รัน grid search เพื่อ train และ ค้นหาพารามิเตอร์ที่ดีที่สุด เพื่อลด noise ในภาพอินพุท เพื่อให้ได้ใกล้เคียง ภาพเป้าหมายที่ไม่มี noise

   grid_result = grid.fit(train_x_noise, train_x)
6. แสดงพารามิเตอร์และ score หรือ error ที่ให้ผลลัพธ์ validation ที่ดีที่สุด

   grid_result.best_params_ , grid_result.best_score_
7. แสดงสถิติค่าเฉลี่ย (mean) และค่าเบี่ยงเบนมาตรฐาน (stds) ของค่า score ในแต่ละค่าพารามิเตอร์

   means = grid_result.cv_results_['mean_test_score’]

   stds = grid_result.cv_results_['std_test_score’]

   params = grid_result.cv_results_['params’]

```
-0.035230 (0.000611) with: {'batch_size': 8, 'epochs': 2, 'optimizer': 'SGD'}
-0.027494 (0.006680) with: {'batch_size': 8, 'epochs': 2, 'optimizer': 'RMSprop'}
-0.197460 (0.010895) with: {'batch_size': 8, 'epochs': 2, 'optimizer': 'Adadelta'}
-0.010964 (0.001539) with: {'batch_size': 8, 'epochs': 2, 'optimizer': 'Adam'}
-0.024397 (0.001206) with: {'batch_size': 8, 'epochs': 4, 'optimizer': 'SGD'}
-0.073503 (0.041198) with: {'batch_size': 8, 'epochs': 4, 'optimizer': 'RMSprop'}
-0.175634 (0.021580) with: {'batch_size': 8, 'epochs': 4, 'optimizer': 'Adadelta'}
-0.006595 (0.000302) with: {'batch_size': 8, 'epochs': 4, 'optimizer': 'Adam'}
```

Getting Results from best parameters

1. กำหนด Training parameter ตามค่าที่ได้จาก best_params_

```
e = best_epoch,        b = best_batch_size ,    o = best_optimizer,    l = best_learning_rate
# กำหนด optimizer setting
autoencoder = Model(Input_img, decoded)
autoencoder.compile(optimizer=o, loss='mse’)
```

2. Train model ด้วยค่าพารามิเตอร์ที่ดีที่สุด

```
callback = EarlyStopping(monitor='loss', patience=3)
history = autoencoder.fit(train_x_noise, train_x,  epochs=e, batch_size=b, shuffle=True, v
            alidation_data=(val_x_noise, val_x), callbacks=[early_stopper])
```

3. ทดสอบ autoencoder model ด้วย validation_data และ test_data

```
test_predictions = autoencoder.predict(test_x_noise)
```

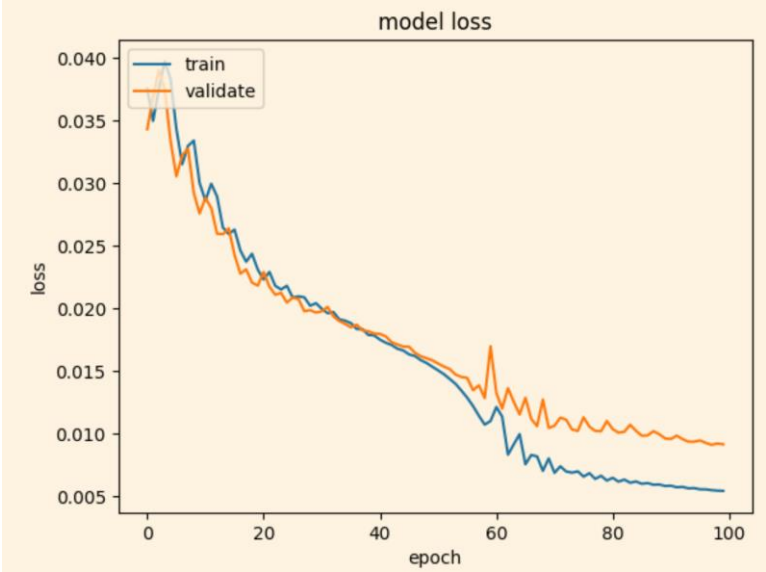
4. View Loss from history

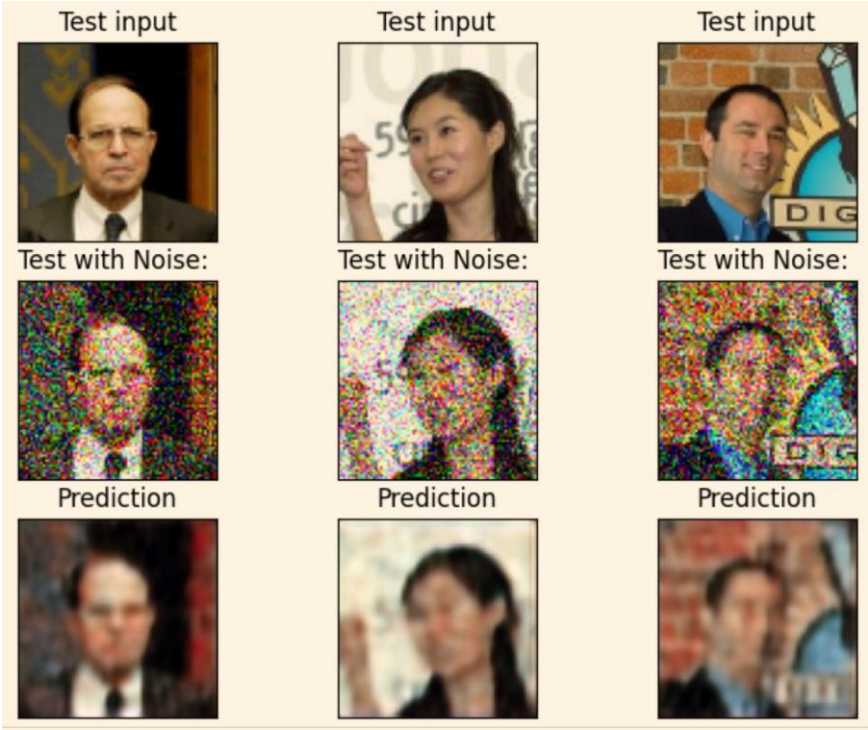
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss') plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

5. Display Result test image

4.2

Autoencoder Model





สรุปผลการปรับพารามิเตอร์

สรุปผลค่าความผิดพลาดในการประมาณค่าภาพจากโมเดล AutoEncoder

จากพารามิเตอร์ที่ได้รับตามตารางในลิงค์ด้านล่าง (สามารถทดสอบมากกว่าที่กำหนดได้)

https://docs.google.com/spreadsheets/d/1LNzKsoPVwL58mStpg16SDogO9cSEdz8Ll2zBxjZfTog/edit?usp=sharing

1. แสดงค่า Best parameter ที่ให้ Error ต่ำสุด 3 อันดับ สังเกตุค่าเฉลี่ยและค่า std ของ Error จากแต่ละ Cross Validation ว่ามีสัดส่วนใดของ training data ให้ผลแตกต่างจากส่วนอื่นหรือไม่ อย่างไร
2. ผลลัพธ์ของภาพทดสอบ (Test image) มีลักษณะเป็นอย่างไร คิดว่าโครงสร้าง autoencoder เพียงพอจะได้ผลลัพธ์ที่ดีแล้วหรือไม่ เพราะเหตุใด