กลุ่มไข่เจียวพิเศษใส่ไข่

ใช้ฟังชั่นสร้าง noise

```
def add_gaussian_noise(img, mean=0, std=1, noise_factor=0.5):
    noise = np.random.normal(mean, std, img.shape)
    noisy_img = img + noise_factor * noise
    noisy_img = np.clip(noisy_img, 0, 1)  # Ensure pixel values are in [0, 1] range
    return noisy_img
```

สร้างโมเดลออโตเอนโคเดอร์ที่ประกอบไปด้วยส่วน Encoder และส่วน Decoder โดยใช้ Keras โค้ดข้างบนสร้างโมเดลที่มีรูปแบบตามนี้:

ส่วน Encoder: ประกอบด้วย Conv2D และ MaxPooling2D layers เพื่อลดขนาดของข้อมูลของภาพในขณะที่เพิ่มความเชื่อมโยงในส่วนสูงขึ้นของภาพ

ส่วน Decoder: ประกอบด้วย Conv2D และ UpSampling2D layers เพื่อปรับคืนขนาดของข้อมูลและลดสัญญาณรบกวน โมเดลนี้มีฟังก์ชันสูญเสีย (loss function) เป็น 'binary_crossentropy' และใช้ตัวหน่วยการอัปเดทออปติไมซ์เป็นออปติไมซ์ Adam กำหนดโดยพารามิเตอร์ learning_rate

```
def create_autoencoder_model(optimizer, learning_rate):
    input_img = Input(shape=(28, 28, 1))

    # Encoder
    x = Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
    x = MaxPooling2D((2, 2), padding='same')(x)
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
    encoded = MaxPooling2D((2, 2), padding='same')(x)

    # Decoder
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(encoded)
    x = UpSampling2D((2, 2))(x)
    x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
    x = UpSampling2D((2, 2))(x)
    decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)

    autoencoder = Model(input_img, decoded)
    autoencoder.compile(optimizer=optimizer(learning_rate=learning_rate), loss='binary_crossentropy')

    return autoencoder
```

กำหนดพารามิเตอร์เช่น epochs , batch_size, และ learning_rate

```
# Set hyperparameters
eps = 10
batch_size = 32
learning_rate = 0.001
```

ใช้ ImageDataGenerator เพื่อทำการปรับเปลี่ยนข้อมูลฝึกเพื่อสร้างข้อมูลปรับปรุงที่หลากหลาย เช่น การหมุนภาพ การเยื้องภาพ การยืดหยุ่นและอื่น ๆ โดยที่ในขณะเดียวกันมีการเพิ่มสัญญาณรบกวนในภาพด้วยฟังก์ชัน add_gaussian_noise

```
# Create an instance of the data generator
datagen = ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest',
    preprocessing_function=add_gaussian_noise
)
```

กำหนด callback ที่จะหยุดการฝึกแบบอัตโนมัติหากไม่มี loss บนชุดฝึกเป็นเวลา 10 รอบ

```
# Define an Early Stopping callback
callback = EarlyStopping(monitor='loss', patience=10)
```

ใช้ fit_generator เพื่อฝึกโมเดลออโตเอนโคเดอร์ โดยการใช้ชุดข้อมูลและการปรับปรุงข้อมูลฝึก ฝึกโมเดลเป็นจำนวนรอบที่กำหนดด้วย epochs และใช้ขนาดชุดฝึก batch_size รวมถึงใช้ชุดตรวจสอบเพื่อตรวจสอบความแก้ไขของโมเดล

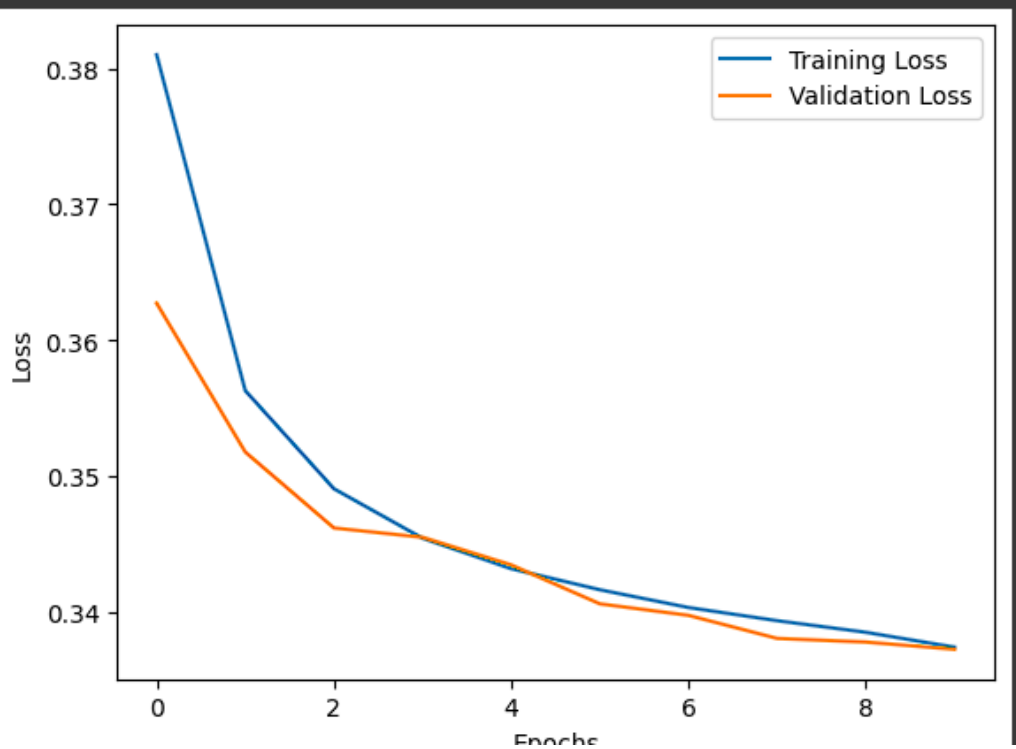
```
# Train the autoencoder using fit_generator
history = autoencoder.fit_generator(
    datagen.flow(x_train, x_train, batch_size=batch_size),
    epochs=eps,
    steps_per_epoch=x_train.shape[0] // batch_size,
    validation_data=datagen.flow(x_val, x_val, batch_size=batch_size),
    callbacks=[callback],
    verbose=1
)
```

พล็อตค่าสูญเสียในชุดฝึกและชุดตรวจสอบเพื่อดูกราฟการเรียนรู้ของโมเดล

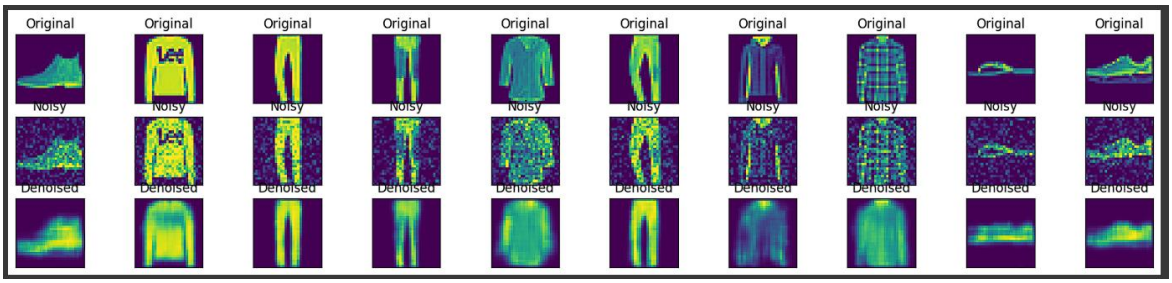
```
# Plot Training and Validation Loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Generate noisy test data
x_test_noisy = np.array([add_gaussian_noise(img) for img in x_test])

# Make predictions using the trained autoencoder
predict_test = autoencoder.predict(x_test_noisy)
```



ผลลัพธ์



ผลลัพธ์การทดสอบครั้งที่สองเลือกปรับเป็น

Epos = 10

Batch size = 16

Learning rate = 0.01

และนำไปรัน

```
# Define Gaussian Noise Function
def add_gaussian_noise(img, mean=0, std=0.6, noise_factor=0.3):
    noise = np.random.normal(mean, std, img.shape)
    noisy_img = img + noise_factor * noise
    noisy_img = np.clip(noisy_img, 0, 1)  # Ensure pixel values are in [0, 1] range
    return noisy_img
def create_model(optimizer, learning_rate):
    # Encoder
    input_img = Input(shape=(28, 28, 1))
    x = Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
    x = MaxPooling2D((2, 2), padding='same')(x)
    encoded = MaxPooling2D((2, 2), padding='same')(x)

    # Decoder
    x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
    x = UpSampling2D((2, 2))(x)
    decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)

    autoencoder = Model(input_img, decoded)
    autoencoder.compile(optimizer=optimizer(learning_rate=learning_rate), loss='mean_squared_error')
    return autoencoder

eps = 10
batch_size = 16
learning_rate = 0.01


# Create the autoencoder model
autoencoder = create_model(optimizer=Adam, learning_rate=learning_rate)

datagen = ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='constant',
    preprocessing_function=add_gaussian_noise
)

callback = EarlyStopping(monitor='loss', patience=10)
history = autoencoder.fit_generator(
    datagen.flow(x_train, x_train, batch_size=batch_size),
    epochs=eps,
    steps_per_epoch=x_train.shape[0] // batch_size,
    validation_data=datagen.flow(x_val, x_val, batch_size=batch_size),
    callbacks=[callback],
    verbose=1
)
```

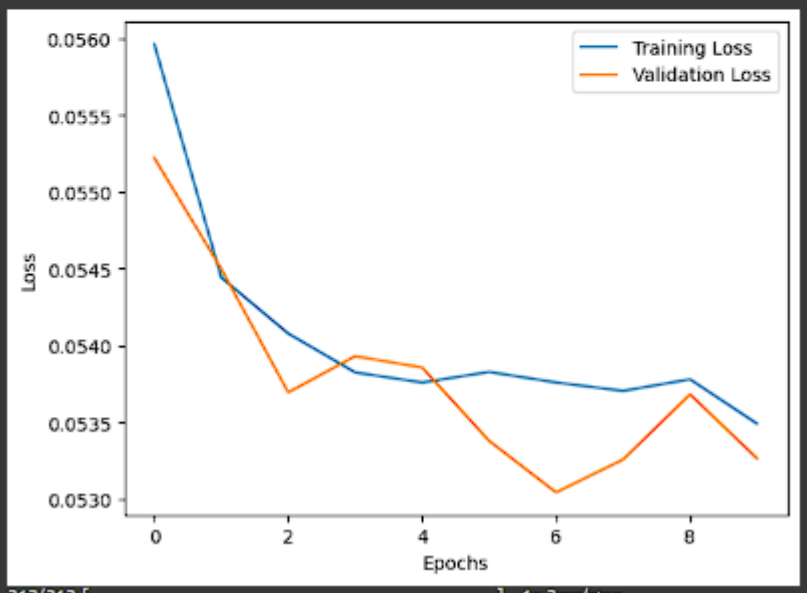
```
Epoch 1/10
<ipython-input-23-9ef0d73c872b>:43: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  history = autoencoder.fit_generator(
3000/3000 [==============================] - 42s 14ms/step - loss: 0.0560 - val_loss: 0.0552
Epoch 2/10
3000/3000 [==============================] - 42s 14ms/step - loss: 0.0544 - val_loss: 0.0545
Epoch 3/10
3000/3000 [==============================] - 41s 14ms/step - loss: 0.0541 - val_loss: 0.0537
Epoch 4/10
3000/3000 [==============================] - 42s 14ms/step - loss: 0.0538 - val_loss: 0.0539
Epoch 5/10
3000/3000 [==============================] - 40s 13ms/step - loss: 0.0538 - val_loss: 0.0539
Epoch 6/10
3000/3000 [==============================] - 43s 14ms/step - loss: 0.0538 - val_loss: 0.0534
Epoch 7/10
3000/3000 [==============================] - 40s 13ms/step - loss: 0.0538 - val_loss: 0.0530
Epoch 8/10
```

ผลลัพธ์จากเจนตาราง

```
[24]  # Plot Training and Validation Loss
      plt.plot(history.history['loss'], label='Training Loss')
      plt.plot(history.history['val_loss'], label='Validation Loss')
      plt.xlabel('Epochs')
      plt.ylabel('Loss')
      plt.legend()
      plt.show()

      # Generate noisy test data
      x_test_noisy = np.array([add_gaussian_noise(img) for img in x_test])

      # Make predictions using the trained autoencoder
      predict_test = autoencoder.predict(x_test_noisy)
```




```
313/313 [==============================] - 1s 2ms/step
```

ผลลัพธ์นำมาแสดงเป็นภาพ

```
# Visualize original, noisy, and denoised images
n = 10  # Number of examples to visualize

plt.figure(figsize=(20, 4))
for i in range(n):
    # Original image
    ax = plt.subplot(3, n, i + 1)
    plt.imshow(x_test[i])
    plt.title("Original")
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Noisy image
    ax = plt.subplot(3, n, i + 1 + n)
    plt.imshow(x_test_noisy[i])
    plt.title("Noisy")
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Denoised image
    ax = plt.subplot(3, n, i + 1 + 2 * n)
    plt.imshow(predict_test[i].reshape(28, 28))
    plt.title("Denoised")
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

plt.show()
```