Lab 5.3 กลุ่มบาบูนตึงตึง

1. อ่านภาพจากชุดข้อมูล Fashion MNIST:

```
# Read image from fashion_mnist dataset
(x_train, _), (x_test, _) = fashion_mnist.load_data()  # load images from dataset
x_train = x_train / 255.0
x_test = x_test / 255.0  # normalized image intensity
```

ใช้ fashion_mnist.load_data() จาก TensorFlow Keras เพื่อโหลดชุดข้อมูล Fashion MNIST และแบ่งเป็นชุดฝึกและชุดทดสอบ

ทำการปรับสเกลค่าความเข้มของพิกเซลในรูปภาพให้อยู่ในช่วง 0-1 ด้วยการหารด้วย 255

2. เพิ่มมิติของภาพ:

```
# Add a channel dimension (rank-4 shape) for compatibility with Conv2D layers
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
```

เพิ่มมิติใหม่ให้กับข้อมูลภาพเพื่อให้เข้ากับเครื่องมือ Conv2D ที่ต้องการ input ในรูปแบบ (batch_size, width, height, channels)

3. แบ่งชุดข้อมูล:

```
x_train, x_test = train_test_split(x_train, test_size=0.2, random_state=42)  # create train data, test data
x_train, x_val = train_test_split(x_train, test_size=0.2, random_state=42)  # create train data, validation data
```

แบ่งชุดข้อมูลฝึก (x_train) และชุดข้อมูลทดสอบ (x_test) ด้วย train_test_split โดยใช้ test_size เพื่อกำหนดส่วนของข้อมูลที่จะใช้สำหรับทดสอบ

นำชุดข้อมูลฝึกต่อมาแบ่งเป็นชุดข้อมูลฝึก (x_train) และชุดข้อมูลตรวจสอบความถูกต้อง (x_val) เพื่อใช้ในการปรับแต่งและควบคุมการฝึกโมเดล

4. เตรียมฟังก์ชันสำหรับการเพิ่มเสียงรบกวนแบบ Gaussian Noise:

```python
# Prepare Gaussian Noise Function
def add_gaussian_noise(image, noise_factor=0.3, noise_mean=0, noise_std=0.5):
    noise = noise_factor * np.random.normal(loc=noise_mean, scale=noise_std, size=image.shape)
    img_noisy = image + noise
    img_noisy = np.clip(img_noisy, 0, 1)  # Clip values to the range [0, 1]
    return img_noisy
```

add_gaussian_noise เป็นฟังก์ชันที่เพิ่ม noise แบบ Gaussian Noise ลงในรูปภาพ

นำ noise มาบวกกับรูปภาพและทำการควบคุมค่าพิกเซลให้อยู่ในช่วง 0-1 โดยใช้ np.clip

5. Create Autoencoder Model:

```python
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 with Dropout
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(encoded)
    x = Dropout(0.5)(x)  # Add Dropout layer
    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
```

สร้างโมเดลแบบออโตเอนโค้ดด้วย create_autoencoder_model

โมเดลนี้มีสองส่วนหลักคือส่วน Encoder และส่วน Decoder โดยมีการใช้ Dropout ในส่วน Decoder

โมเดลถูกคอมไพล์ด้วยค่า optimizer, learning rate และ loss function ที่ใช้ในการฝึก

6. ImageDataGenerator for Data Augmentation:

```
# Define ImageDataGenerator with parameters
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'
```

```
# Define ImageDataGenerator with parameters
datagen = ImageDataGenerator(
    rotation_range=rotation_range,
    width_shift_range=width_shift_range,
    height_shift_range=height_shift_range,
    shear_range=shear_range,
    zoom_range=zoom_range,
    horizontal_flip=horizontal_flip,
    preprocessing_function=add_gaussian_noise,
    fill_mode=fill_mode)
```

ใช้ ImageDataGenerator สร้างข้อมูลสำหรับการฝึกโมเดล โดยใช้การปรับแต่งข้อมูล (data augmentation) เช่น การหมุน, การเลื่อน, การเพิ่มเข้าไป, การเบียด, การซูม, และการพลิกภาพ และยังเรียกใช้ฟังก์ชัน add_gaussian_noise เพื่อเพิ่มสัญญาณรบกวน.

7. กำหนดพารามิเตอร์และสร้างโมเดล:

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

สร้างโมเดลออโตเอนโค้ด (autoencoder) โดยใช้ฟังก์ชัน create_autoencoder_model และใช้ optimizer และ learning rate ที่กำหนด

8. กำหนด Callback:

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

กำหนด Early Stopping callback สำหรับการหยุดการฝึกโมเดลเมื่อค่า loss ของการฝึกไม่ดีขึ้น (monitor='loss') เกินระยะเวลาที่กำหนด (patience=10)

9. Train the Autoencoder Model:

```
# Set hyperparameters
eps = 50
batch_size = 32
learning_rate = 0.01
```

กำหนดพารามิเตอร์แบบเซ็ตเอง เช่น eps สำหรับจำนวนรอบการฝึก, batch_size สำหรับขนาดแบตช์ในการฝึก, และ learning_rate สำหรับอัตราการเรียนรู้ของโมเดล

```
# 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
)
```

ใช้ fit_generator เพื่อฝึกโมเดลโดยใช้ชุดข้อมูลที่ถูกปรับแต่งจาก ImageDataGenerator

กำหนดจำนวนรอบการฝึกด้วย eps, จำนวนข้อมูลในแต่ละ batch ด้วย batch_size, และข้อมูลตรวจสอบความถูกต้องของโมเดลด้วย validation_data

ใช้ Early Stopping callback เพื่อหยุดการฝึกเมื่อค่า loss ไม่ดีขึ้นเป็นเวลานาน

10. บันทึกโมเดลออโตเอนโค้ดและประวัติการฝึก:

```
# Save the trained autoencoder model to a file
autoencoder.save("autoencoder_model2.h5")
```

โมเดล Autoencoder ที่ถูกฝึกจะถูกบันทึกลงในไฟล์ 'autoencoder_model2.h5' เพื่อใช้ในการให้บริการในอนาคต.

```
# Save the training history to a file using pickle
with open("training_history2.pkl", "wb") as history_file:
    pickle.dump(history.history, history_file)
```

บันทึกประวัติการฝึกลงในไฟล์ "training_history2.pkl" โดยใช้ pickle.dump

11. โหลดโมเดลออโตเอนโค้ด:

```
# Load the trained autoencoder model from the saved file
autoencoder = load_model("autoencoder_model2.h5")
```

โหลดโมเดลออโตเอนโค้ดที่ถูกฝึกมาจากไฟล์ "autoencoder_model2.h5" ด้วย load_model จาก TensorFlow Keras

12. สร้างฟังก์ชันสำหรับการเพิ่มเสียงรบกวนแบบ Gaussian Noise:

```
# Function to add Gaussian noise to an image
def add_gaussian_noise(image, noise_factor=0.3, noise_mean=0, noise_std=0.5):
    noise = noise_factor * np.random.normal(loc=noise_mean, scale=noise_std, size=image.shape)
    img_noisy = image + noise
    img_noisy = np.clip(img_noisy, 0, 1)  # Clip values to the range [0, 1]
    return img_noisy
```

สร้างฟังก์ชัน add_gaussian_noise แบบเดียวกับในส่วนการเตรียมข้อมูลในการฝึกโมเดล

13. โหลดชุดข้อมูล Fashion MNIST:

```
# Load the Fashion MNIST dataset
(x_train, _), (x_test, _) = fashion_mnist.load_data()  # Load images from the dataset
x_train = x_train / 255.0
x_test = x_test / 255.0  # Normalize image intensity
```

โหลดชุดข้อมูล Fashion MNIST และทำการปรับสเกลค่าความเข้มของพิกเซลในรูปภาพให้อยู่ในช่วง 0-1 ด้วยการหารด้วย 255

14. เพิ่มมิติของภาพ:

```
# Add a channel dimension (rank-4 shape) for compatibility with Conv2D layers
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
```

เพิ่มมิติใหม่ให้กับข้อมูลภาพเพื่อให้เข้ากับโมเดลออโตเอนโค้ด

15. สร้างชุดข้อมูลทดสอบที่มี noise:

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

สร้างชุดข้อมูลทดสอบที่มีการเพิ่ม noise แบบ Gaussian Noise เข้าไปในรูปภาพต้นฉบับ x_test โดยใช้ฟังก์ชัน add_gaussian_noise

16. ทำนายภาพที่ผ่านการลบ noise:

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

ใช้โมเดลออโตเอนโค้ดที่ถูกโหลดมาก่อนหน้านี้ (autoencoder) เพื่อทำนายภาพที่ผ่านการลบ noise จากชุดข้อมูลทดสอบที่มี noise (x_test_noisy)

17โหลดประวัติการฝึกจากไฟล์:

```
# Load the training history from the saved pickle file
with open("training_history2.pkl", "rb") as history_file:
    loaded_history = pickle.load(history_file)
```

โหลดประวัติการฝึกของโมเดลจากไฟล์ "training_history2.pkl" โดยใช้ pickle.load

18 พล็อตกราฟค่าความสูญเสียของการฝึกและการตรวจสอบความถูกต้อง:

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

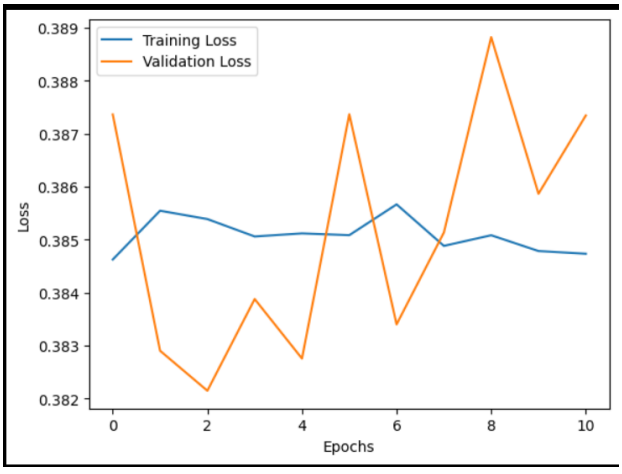
พล็อตค่าความสูญเสียของการฝึกและการตรวจสอบความถูกต้องจากประวัติการฝึกที่ถูกโหลดมา

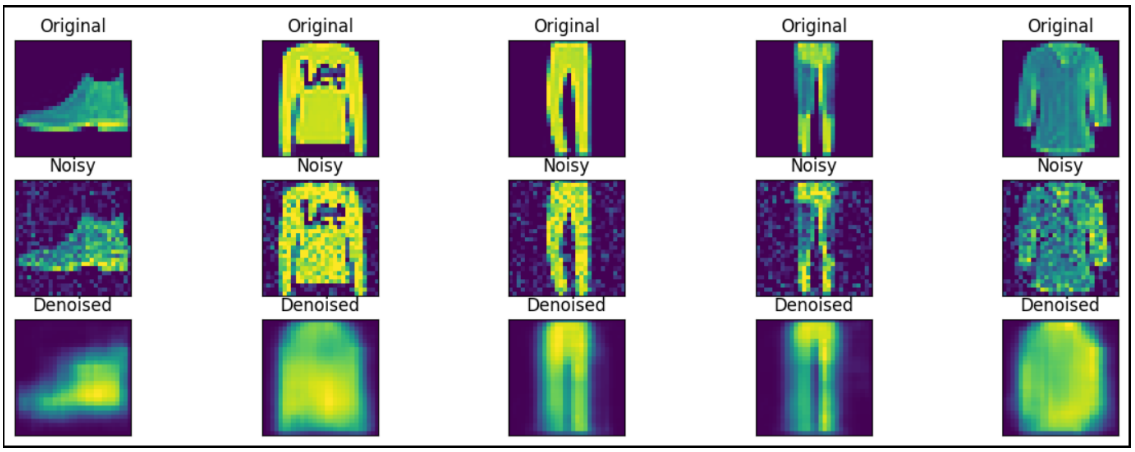
19. พล็อตรูปภาพ:

```
# 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)

plt.figure(figsize=(15, 5))
for i in range(5):
    # Original image
    ax = plt.subplot(3, 5, 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, 5, i + 1 + 5)
    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, 5, i + 1 + 2 * 5)
    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()
```

พล็อตรูปภาพที่ผ่านการลบเสียงรบกวน เพื่อเปรียบเทียบรูปภาพต้นฉบับ (Original) กับรูปภาพที่มีเสียงรบกวน (Noisy) และรูปภาพที่ผ่านการลบเสียงรบกวน (Denoised) โดยเลือกพล็อตเพียง 5 รูปภาพตัวอย่าง

ผลลัพธ์





แบบที่ 2

```python
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
```

```python
# Define ImageDataGenerator with parameters
rotation_range = 20
width_shift_range = 0.2
height_shift_range = 0.2
shear_range = 0.2
zoom_range = 0.2
horizontal_flip = True
fill_mode='nearest'
```

```python
# Set hyperparameters
eps = 100
batch_size = 8
learning_rate = 0.001
```

เปลี่ยน create_autoencoder_model โดยมีการเอา Dropout ในส่วน Decoder ออก

เปลี่ยนข้อมูลที่ใช้ใน ImageDataGenerator

เปลี่ยน eps , batch_size และlearning rate

ผลลัพธ์