Spring 2025: CS5720

Neural Networks and Deep Learning - ICP-6

Gonaboyina Vijay Vardhan (700755141)

Github link: https://github.com/Vijayvardhan02/Neural-networks-deep-learning-ICP6
Video link :
https://drive.google.com/file/d/1ot9_Gfm81fO6XPJ1S4uxpxVbh9g1piV8/view?usp=drive_link

1. Add one more hidden layer to autoencoder
2. Do the prediction on the test data and then visualize one of the reconstructed version of that test data. Also, visualize the same test data before reconstruction using Matplotlib
3. Repeat the question 2 on the denoisening autoencoder
4. plot loss and accuracy using the history object

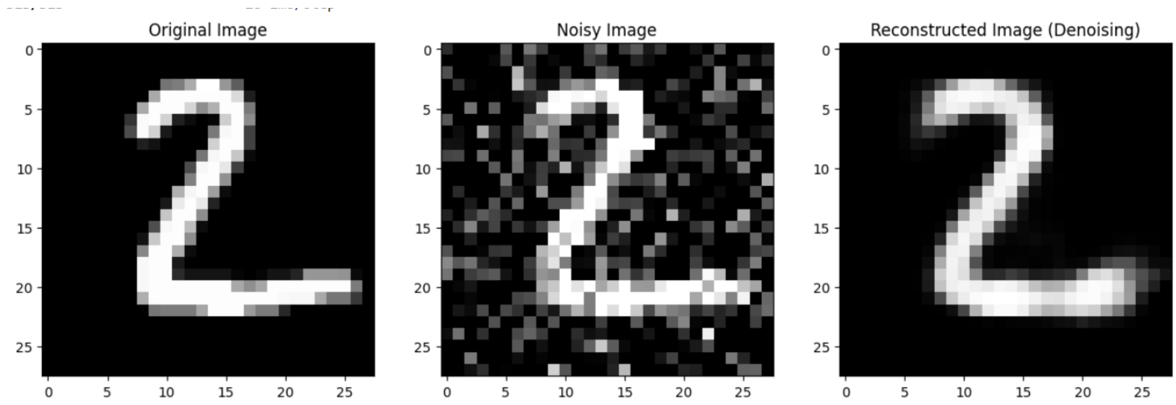
Code:

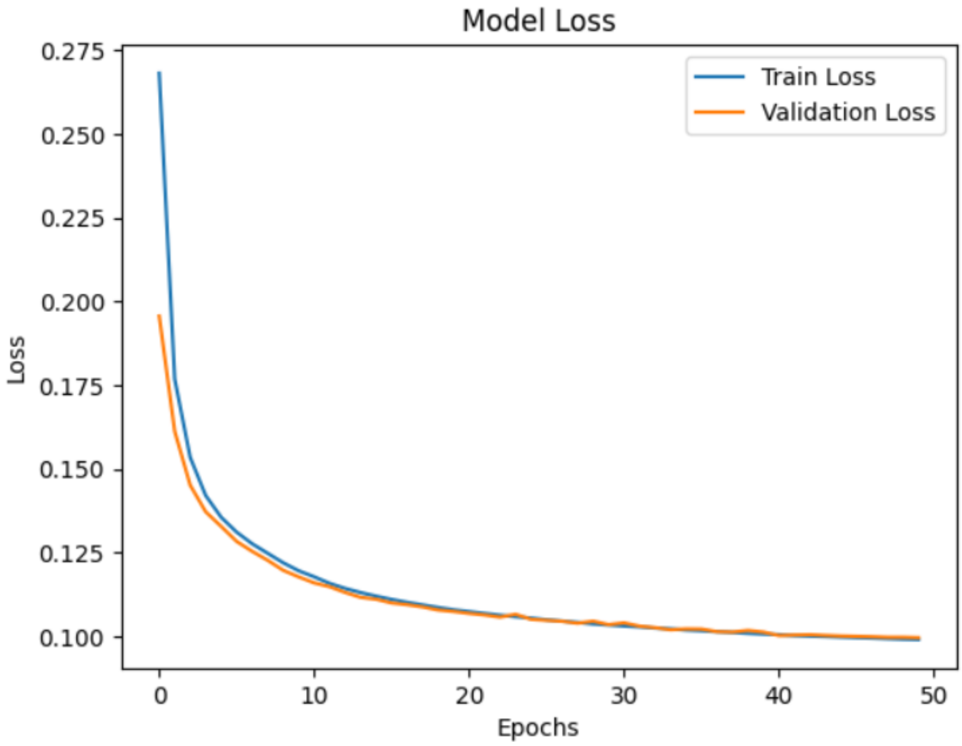
```
import numpy as np
import matplotlib.pyplot as plt
from keras.datasets import mnist
from keras.layers import Input, Dense
from keras.models import Model

# 1. Load data
(x_train, _), (x_test, _) = mnist.load_data()

# 2. Normalize and flatten the data
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
x_train = x_train.reshape((len(x_train), 784))
x_test = x_test.reshape((len(x_test), 784))

# 3. Add noise to the data
def add_noise(data, noise_factor=0.3):
    noise = np.random.normal(loc=0.0, scale=1.0, size=data.shape)
    noisy_data = data + noise_factor * noise
    noisy_data = np.clip(noisy_data, 0., 1.)
    return noisy_data

x_train_noisy = add_noise(x_train)
x_test_noisy = add_noise(x_test)

# 4. Build the Autoencoder model (with an extra hidden layer)
input_img = Input(shape=(784,))
```

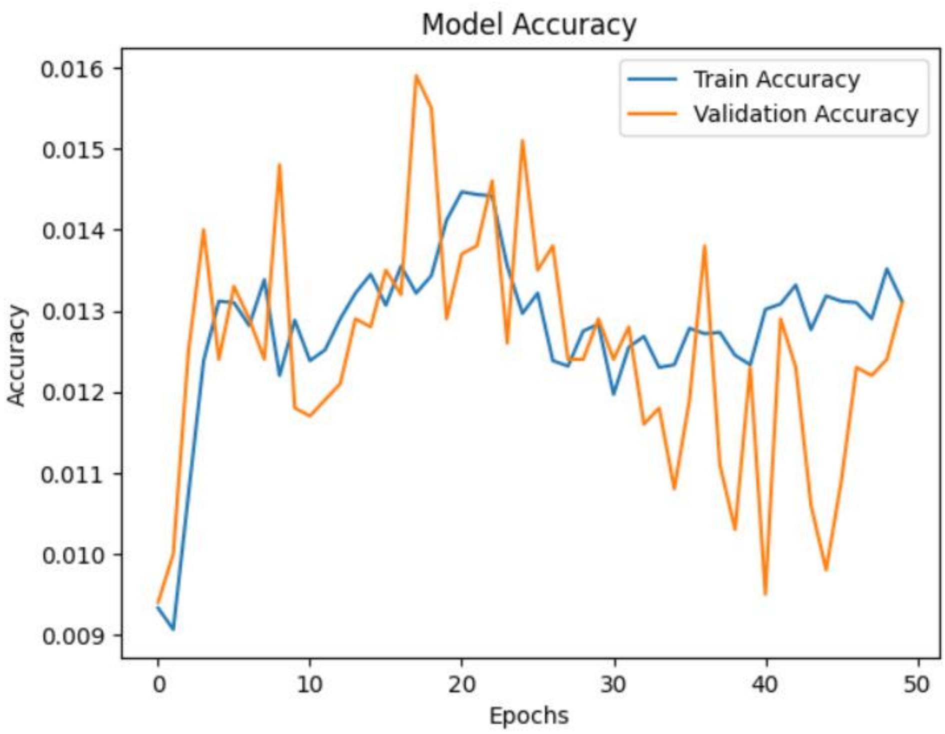
```python
# Encoder with additional hidden layer
encoded = Dense(128, activation='relu')(input_img)
encoded = Dense(64, activation='relu')(encoded)  # Additional hidden layer
encoded = Dense(32, activation='relu')(encoded)  # Bottleneck layer

# Decoder
decoded = Dense(64, activation='relu')(encoded)
decoded = Dense(128, activation='relu')(decoded)
decoded = Dense(784, activation='sigmoid')(decoded)

# Autoencoder model
autoencoder = Model(input_img, decoded)

# Compile the model
autoencoder.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# 5. Train the Autoencoder on noisy data
history = autoencoder.fit(x_train_noisy, x_train, epochs=50, batch_size=256, shuffle=True, validation_data=(x_test_noisy, x_test))

# 6. Visualize reconstructed images
decoded_imgs_noisy = autoencoder.predict(x_test_noisy)

# Visualize an example before and after reconstruction
n = 1  # Index of the test image to visualize
plt.figure(figsize=(15, 5))

# Original image
plt.subplot(1, 3, 1)
plt.imshow(x_test[n].reshape(28, 28), cmap='gray')
plt.title("Original Image")
```

```python
# Noisy image
plt.subplot(1, 3, 2)
plt.imshow(x_test_noisy[n].reshape(28, 28), cmap='gray')
plt.title("Noisy Image")

# Reconstructed image
plt.subplot(1, 3, 3)
plt.imshow(decoded_imgs_noisy[n].reshape(28, 28), cmap='gray')
plt.title("Reconstructed Image (Denoising)")

plt.show()

# 7. Plot loss
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss')
plt.legend()
plt.show()

# 8. Plot accuracy
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')
plt.legend()
plt.show()
```

Output:


Original Image
Noisy Image
Reconstructed Image (Denoising)


Model Loss
Train Loss
Validation Loss
Loss
Epochs


Model Accuracy
Train Accuracy
Validation Accuracy
Accuracy
Epochs