ex:-10 Perform compression on mnist dataset using autoencoder

Aim:- to design and implement an Autoencoder network for dimensionality reduction of the mnist hand written digits dataset and to reconstruct the original images from

objectives.

1. to understand the concept of autoencoders and their application in data compression,
2. To train an autoencoder on the MNIST dataset for encoding and decoding digit images
3. To visualize and compare the original and reconstructed images
4. To observe the compression ration and reconstration quality
5. To analyze the models performance using mean squared Error (MSE).

Pseudocode :-

1. Import necessary libraries.
   - numpy, tensorflow (keras), matplotlib.
2. Load MNIST dataset
   - split into train and test set.
   - Normalize pixel values blw 0 and 1

3. Define Autoencoder architecture.

Encoder:-

Input layer: 784 (28 x 28 flattened image)

Dense (128, activation = 'relu')

Dense (64, activation = 'relu')

Dense (32, activation = 'relu') → compressed representation

Decoder:-

Dense (64, activation = 'relu')

Dense (128, activation = 'relu')

Dense (784, activation = Sigmoid) → reconstructed image.

4. Compile Autoencoder

- loss: mean squared Error (MSE).

- optimizer: Adam

5. Train Autoencoder

- Encode test images → get compressed representation.

→ Input = output = MNIST images

- epochs = 50 ; batch_size = 256

6. visualize.

- Display few original and reconstructed images side by side.

7. calculate reconstruction loss (MSE)

- compare reconstruction quality.

observation:

During training

- the training loss and validation loss gradually decreased with epochs.

- Encoded layer (32 units) effectively

Result :- Training data shape : (60000, 784)
Testing data shape : (10000, 784)

Epoch 1/50 step - loss : 0.0976
val - loss : 0.0379

5/50 step_loss : 0.0194
val_loss : - 0.0180

10/50 step_loss : - 0.0137
val - loss : - 0.0131

20/50 step_loss : - 0.0108
val - loss : - 0.0105

30/50 step_loss : - 0.0095
val_loss : - 0.0093

40/50 step_loss : - 0.0089
val_loss : - 0.0088

50/50 step_loss : - 0.0084
val_loss : - 0.0084

Reconstruction MSE loss on test data : 0.00844
compression Ration : 24.50 : 1

original.

7 7 1 0 4

Recontructed 2 2 1 0 4

1. The Autoencoder successfully learned to compress and reconstruct MNIST digits.

2. The compression ration achieved:-

$$\text{Compression Ratio} = \frac{784}{32} \approx 24.5:1$$

3. The Mean squared Error (MSE) on the test data set $\approx 0.01 \rightarrow 0.02$

4. The reconstructed digits closely resembled the original images, proving the efficiency of auto encoders in unsupervised feature learning and data compression.

Compression on MNIST dataset | Welcome to Colab - Colab

colab.research.google.com/#scrollTo=Omd8LwiP4czT

Gmail Maps News Translate Disney+ Hotstar -... YouTube All Bookmarks

Welcome to Colab | Cannot save changes

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text Run all Copy to Drive

RAM Disk

Table of contents

- Welcome to Colab!
  - Getting started
  - Data science
  - Machine learning
- More resources
  - Featured examples
- + Section

```
# --------------------------
# 1. Import Libraries
# --------------------------
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.optimizers import Adam

# --------------------------
# 2. Load and Preprocess Data
# --------------------------
(x_train, _), (x_test, _) = mnist.load_data()

# Normalize pixel values (0–255 → 0–1)
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.

# Flatten the 28x28 images → 784 features
x_train = x_train.reshape((len(x_train), 784))
x_test = x_test.reshape((len(x_test), 784))

print("Training data shape:", x_train.shape)
print("Testing data shape:", x_test.shape)

# --------------------------
# 3. Build Autoencoder Model
# --------------------------

# Encoder
```

Variables Terminal

00:31 Python 3

Compression on MNIST dataset | Welcome to Colab - Colab

colab.research.google.com/#scrollTo=zRCglOuiqd3N

Gmail | Maps | News | Translate | Disney+ Hotstar -... | YouTube | All Bookmarks

Welcome to Colab | Cannot save changes

File Edit View Insert Runtime Tools Help

Share

Commands | + Code | + Text | Run all | Copy to Drive

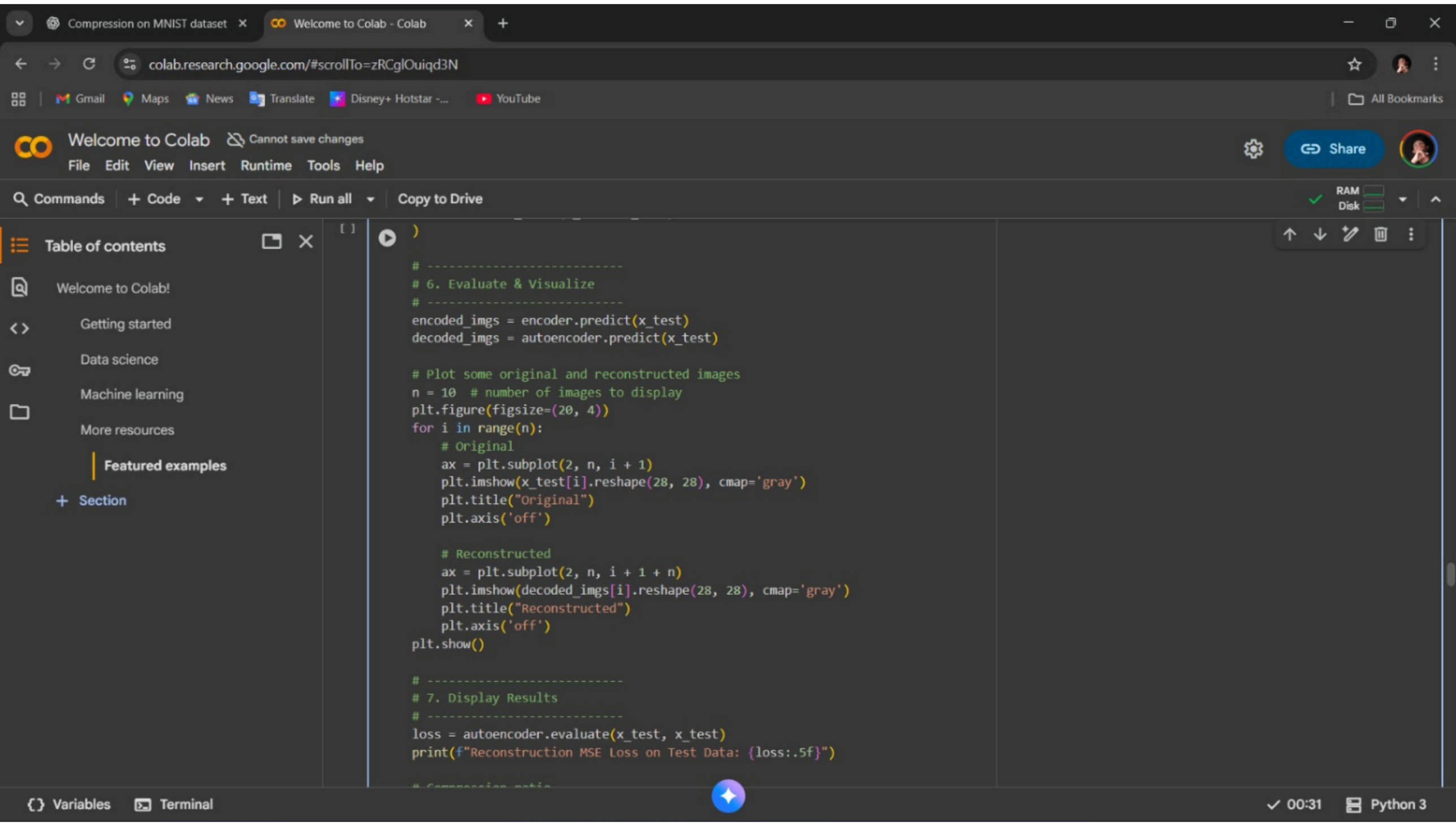
RAM Disk

Table of contents

- Welcome to Colab!
  - Getting started
  - Data science
  - Machine learning
  - More resources
    - Featured examples
- + Section

```
input_img = Input(shape=(784,))
encoded = Dense(128, activation='relu')(input_img)
encoded = Dense(64, activation='relu')(encoded)
encoded = Dense(32, activation='relu')(encoded)   # compressed representation

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

# Combine encoder + decoder
autoencoder = Model(input_img, decoded)

# Encoder model (for compression)
encoder = Model(input_img, encoded)

# --------------------------
# 4. Compile Model
# --------------------------
autoencoder.compile(optimizer=Adam(learning_rate=0.001), loss='mse')

# --------------------------
# 5. Train Autoencoder
# --------------------------
history = autoencoder.fit(
    x_train, x_train,
    epochs=50,
    batch_size=256,
    shuffle=True,
    validation_data=(x_test, x_test)
)
```

Variables | Terminal | 00:31 | Python 3


Compression on MNIST dataset
Welcome to Colab - Colab
colab.research.google.com/#scrollTo=zRCglOuiqd3N
Gmail
Maps
News
Translate
Disney+ Hotstar -...
YouTube
All Bookmarks
Welcome to Colab
Cannot save changes
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
Run all
Copy to Drive
RAM
Disk
Table of contents
Welcome to Colab!
Getting started
Data science
Machine learning
More resources
Featured examples
+ Section
[ ]
)
# ---------------------------
# 6. Evaluate & Visualize
# ---------------------------
encoded_imgs = encoder.predict(x_test)
decoded_imgs = autoencoder.predict(x_test)
# Plot some original and reconstructed images
n = 10 # number of images to display
plt.figure(figsize=(20, 4))
for i in range(n):
# Original
ax = plt.subplot(2, n, i + 1)
plt.imshow(x_test[i].reshape(28, 28), cmap='gray')
plt.title("Original")
plt.axis('off')
# Reconstructed
ax = plt.subplot(2, n, i + 1 + n)
plt.imshow(decoded_imgs[i].reshape(28, 28), cmap='gray')
plt.title("Reconstructed")
plt.axis('off')
plt.show()
# ---------------------------
# 7. Display Results
# ---------------------------
loss = autoencoder.evaluate(x_test, x_test)
print(f"Reconstruction MSE Loss on Test Data: {loss:.5f}")
Variables
Terminal
00:31
Python 3

Compression on MNIST dataset | Welcome to Colab - Colab

colab.research.google.com/#scrollTo=zRCglOuiqd3N

Gmail | Maps | News | Translate | Disney+ Hotstar -... | YouTube | All Bookmarks

Welcome to Colab | Cannot save changes

File Edit View Insert Runtime Tools Help

Share

Commands | + Code | + Text | Run all | Copy to Drive

RAM Disk

Table of contents

- Welcome to Colab!
- Getting started
- Data science
- Machine learning
- More resources
- Featured examples
- + Section

```
[ ]
# Compression ratio
compression_ratio = 784 / 32
print(f"Compression Ratio: {compression_ratio:.2f}:1")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Training data shape: (60000, 784)
Testing data shape: (10000, 784)
Epoch 1/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 12ms/step - loss: 0.0976 - val_loss: 0.0379
Epoch 2/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0349 - val_loss: 0.0257
Epoch 3/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0248 - val_loss: 0.0216
Epoch 4/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0214 - val_loss: 0.0194
Epoch 5/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0194 - val_loss: 0.0180
Epoch 6/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0177 - val_loss: 0.0162
Epoch 7/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0163 - val_loss: 0.0152
Epoch 8/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0151 - val_loss: 0.0142
Epoch 9/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0144 - val_loss: 0.0136
Epoch 10/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0137 - val_loss: 0.0131
Epoch 11/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0133 - val_loss: 0.0127
Epoch 12/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0129 - val_loss: 0.0126
Epoch 13/50
```

Variables | Terminal | 00:31 | Python 3

Compression on MNIST dataset | Welcome to Colab - Colab

colab.research.google.com/#scrollTo=zRCglOuiqd3N

Gmail | Maps | News | Translate | Disney+ Hotstar -... | YouTube | All Bookmarks

Welcome to Colab | Cannot save changes

File Edit View Insert Runtime Tools Help

Share

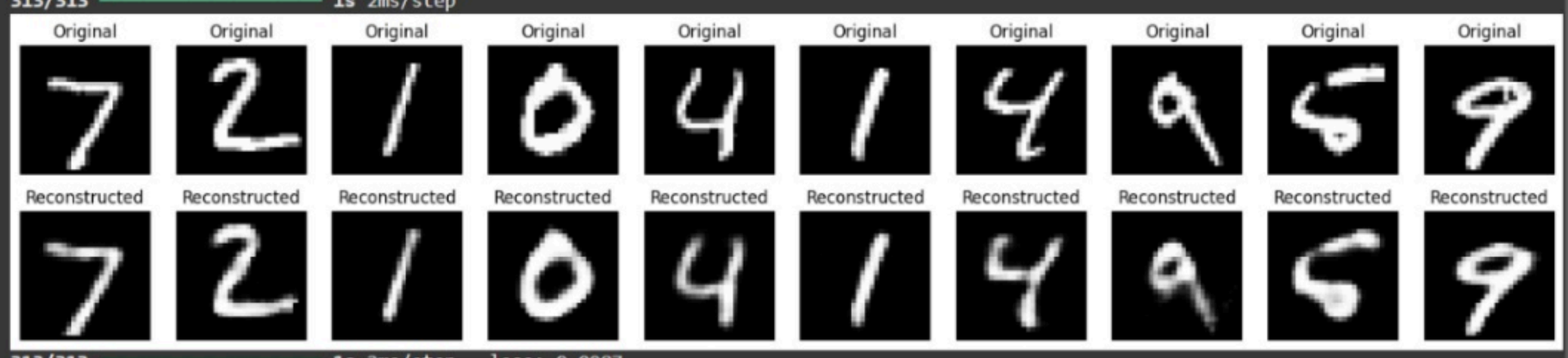
Commands | + Code | + Text | Run all | Copy to Drive

RAM Disk

Table of contents

- Welcome to Colab!
  - Getting started
  - Data science
  - Machine learning
  - More resources
    - Featured examples
- + Section

```
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0087 - val_loss: 0.0086
Epoch 44/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.0086 - val_loss: 0.0086
Epoch 45/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0087 - val_loss: 0.0086
Epoch 46/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0086 - val_loss: 0.0085
Epoch 47/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0085 - val_loss: 0.0087
Epoch 48/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0085 - val_loss: 0.0084
Epoch 49/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0085 - val_loss: 0.0084
Epoch 50/50
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0084 - val_loss: 0.0084
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```




```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.0087
Reconstruction MSE Loss on Test Data: 0.00844
Compression Ratio: 24.50:1
```

Variables | Terminal | 00:31 | Python 3