ICP6 REPORT

+ Code + Text

[2]
```
import numpy as np
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.datasets import mnist
from tensorflow.keras.callbacks import EarlyStopping

# Load the MNIST dataset
(x_train, _), (x_test, _) = mnist.load_data()

# Normalize pixel values to the range [0, 1]
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.

# Flatten the images for the autoencoder
x_train = x_train.reshape((len(x_train), -1))  # -1 infers the remaining dimension
x_test = x_test.reshape((len(x_test), -1))  # -1 infers the remain

# Define the dimensions of the input and the encoded representation
input_dim = x_train.shape[1]
encoding_dim = 16  # Compress to 16 features
```

+ Code + Text

T4 RAM Disk

[2]
```
# Define the input layer
input_layer = Input(shape=(input_dim,))

# Define the encoder
encoded = Dense(encoding_dim, activation='relu')(input_layer)
# Adding a layer
encoded1 = Dense(encoding_dim, activation='relu')(encoded)

# Adding a layer
decoded1 = Dense(encoding_dim, activation='relu')(encoded1)
# Define the decoder
decoded = Dense(input_dim, activation='sigmoid')(decoded1)

# Combine the encoder and decoder into an autoencoder model
autoencoder = Model(input_layer, decoded)

# Define EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss',
                               patience=5,  # Number of epochs with no improvement after which training will be stopped
                               restore_best_weights=True)  # Restores model to best weights with the lowest validation loss

# Compile the autoencoder model
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

+ Code + Text

```
[2]
# Train the autoencoder
autoencoder.fit(x_train, x_train,  # For autoencoders, input and output are the same
                epochs=100,  # Set a high number of epochs
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test),
                callbacks=[early_stopping])  # Add the early stopping callback
```

```
Epoch 1/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 0.4415 - val_loss: 0.2438
Epoch 2/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2357 - val_loss: 0.2080
Epoch 3/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2038 - val_loss: 0.1911
Epoch 4/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1880 - val_loss: 0.1782
Epoch 5/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1780 - val_loss: 0.1711
Epoch 6/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1710 - val_loss: 0.1648
Epoch 7/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1652 - val_loss: 0.1610
Epoch 8/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1621 - val_loss: 0.1592
Epoch 9/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1601 - val_loss: 0.1562
```

+ Code + Text

```
[2]
Epoch 10/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1568 - val_loss: 0.1533
Epoch 11/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1543 - val_loss: 0.1521
Epoch 12/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1531 - val_loss: 0.1513
Epoch 13/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1528 - val_loss: 0.1506
Epoch 14/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1517 - val_loss: 0.1500
Epoch 15/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1512 - val_loss: 0.1497
Epoch 16/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1506 - val_loss: 0.1484
Epoch 17/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1495 - val_loss: 0.1462
Epoch 18/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1476 - val_loss: 0.1445
Epoch 19/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1462 - val_loss: 0.1435
Epoch 20/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1448 - val_loss: 0.1428
Epoch 21/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1442 - val_loss: 0.1423
Epoch 22/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1439 - val_loss: 0.1420
```

+ Code + Text

[2]

```
Epoch 23/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1434 - val_loss: 0.1416
Epoch 24/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1430 - val_loss: 0.1413
Epoch 25/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1427 - val_loss: 0.1409
Epoch 26/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1424 - val_loss: 0.1407
Epoch 27/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1421 - val_loss: 0.1403
Epoch 28/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1419 - val_loss: 0.1400
Epoch 29/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1414 - val_loss: 0.1398
Epoch 30/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1413 - val_loss: 0.1397
Epoch 31/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1412 - val_loss: 0.1395
Epoch 32/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1411 - val_loss: 0.1393
Epoch 33/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1409 - val_loss: 0.1391
Epoch 34/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1404 - val_loss: 0.1389
Epoch 35/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1404 - val_loss: 0.1390
```

+ Code + Text

[2]

```
Epoch 36/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1404 - val_loss: 0.1388
Epoch 37/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1403 - val_loss: 0.1387
Epoch 38/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1397 - val_loss: 0.1385
Epoch 39/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1401 - val_loss: 0.1384
Epoch 40/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1398 - val_loss: 0.1383
Epoch 41/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1400 - val_loss: 0.1382
Epoch 42/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1398 - val_loss: 0.1381
Epoch 43/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1398 - val_loss: 0.1381
Epoch 44/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1394 - val_loss: 0.1380
Epoch 45/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1395 - val_loss: 0.1380
Epoch 46/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1394 - val_loss: 0.1378
Epoch 47/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1393 - val_loss: 0.1377
Epoch 48/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1393 - val_loss: 0.1377
```

+ Code + Text

```
Epoch 49/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1392 - val_loss: 0.1378
Epoch 50/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1395 - val_loss: 0.1376
Epoch 51/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1390 - val_loss: 0.1376
Epoch 52/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1391 - val_loss: 0.1374
Epoch 53/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1389 - val_loss: 0.1375
Epoch 54/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1389 - val_loss: 0.1373
Epoch 55/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1387 - val_loss: 0.1373
Epoch 56/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1387 - val_loss: 0.1373
Epoch 57/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1389 - val_loss: 0.1372
Epoch 58/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1391 - val_loss: 0.1373
Epoch 59/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1388 - val_loss: 0.1372
Epoch 60/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1385 - val_loss: 0.1371
Epoch 61/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1388 - val_loss: 0.1370
```

+ Code + Text

```
Epoch 62/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1390 - val_loss: 0.1371
Epoch 63/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1387 - val_loss: 0.1369
Epoch 64/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1387 - val_loss: 0.1369
Epoch 65/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1385 - val_loss: 0.1370
Epoch 66/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1385 - val_loss: 0.1369
Epoch 67/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1384 - val_loss: 0.1369
Epoch 68/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1384 - val_loss: 0.1368
Epoch 69/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1384 - val_loss: 0.1368
Epoch 70/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1382 - val_loss: 0.1368
Epoch 71/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1380 - val_loss: 0.1367
Epoch 72/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1381 - val_loss: 0.1366
Epoch 73/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1382 - val_loss: 0.1367
Epoch 74/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1380 - val_loss: 0.1367
```

+ Code + Text

```
[2]
Epoch 75/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1384 - val_loss: 0.1366
Epoch 76/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1378 - val_loss: 0.1366
Epoch 77/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1378 - val_loss: 0.1364
Epoch 78/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1382 - val_loss: 0.1364
Epoch 79/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1378 - val_loss: 0.1365
Epoch 80/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1382 - val_loss: 0.1364
Epoch 81/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1378 - val_loss: 0.1364
Epoch 82/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1377 - val_loss: 0.1362
Epoch 83/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1374 - val_loss: 0.1364
Epoch 84/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1378 - val_loss: 0.1363
Epoch 85/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1375 - val_loss: 0.1362
Epoch 86/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1376 - val_loss: 0.1361
Epoch 87/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1378 - val_loss: 0.1361
```

+ Code + Text

```
[2]
Epoch 88/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1376 - val_loss: 0.1361
Epoch 89/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1377 - val_loss: 0.1360
Epoch 90/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1380 - val_loss: 0.1359
Epoch 91/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1374 - val_loss: 0.1360
Epoch 92/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1372 - val_loss: 0.1359
Epoch 93/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1377 - val_loss: 0.1358
Epoch 94/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1374 - val_loss: 0.1360
Epoch 95/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1375 - val_loss: 0.1359
Epoch 96/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1374 - val_loss: 0.1357
Epoch 97/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1373 - val_loss: 0.1357
Epoch 98/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1372 - val_loss: 0.1358
Epoch 99/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1371 - val_loss: 0.1356
Epoch 100/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1372 - val_loss: 0.1357
```

+ Code + Text

```
<keras.src.callbacks.history.History at 0x78c82f6e2530>
```

[3]
28s

```
import numpy as np
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.datasets import mnist
from tensorflow.keras.callbacks import TerminateOnNaN

# Define the TerminateOnNaN callback
terminate_on_nan = TerminateOnNaN()

# Load the MNIST dataset
(x_train, _), (x_test, _) = mnist.load_data()

# Normalize pixel values to the range [0, 1]
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.

# Flatten the images for the autoencoder
x_train = x_train.reshape((len(x_train), -1))  # -1 infers the remaining dimension
x_test = x_test.reshape((len(x_test), -1))  # -1 infers the remain
```

+ Code + Text

[3]
28s

```
# Define the dimensions of the input and the encoded representation
input_dim = x_train.shape[1]
encoding_dim = 16  # Compress to 16 features

# Define the input layer
input_layer = Input(shape=(input_dim,))

# Define the encoder
encoded = Dense(encoding_dim, activation='relu')(input_layer)
# Adding a layer
encoded1 = Dense(encoding_dim, activation='relu')(encoded)

# Adding a layer
decoded1 = Dense(encoding_dim, activation='relu')(encoded1)
# Define the decoder
decoded = Dense(input_dim, activation='sigmoid')(decoded1)

# Combine the encoder and decoder into an autoencoder model
autoencoder = Model(input_layer, decoded)

# Compile the autoencoder model
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

+ Code + Text

[3]
```
# Train the autoencoder
# Assuming x_train and x_test are your training and validation datasets
autoencoder.fit(x_train, x_train,  # For autoencoders, input and output are the same
                epochs=30,  # Set the number of epochs
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test),
                callbacks=[terminate_on_nan])  # Add the TerminateOnNaN callback
```

```
Epoch 1/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 6ms/step - loss: 0.4305 - val_loss: 0.2453
Epoch 2/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.2361 - val_loss: 0.2102
Epoch 3/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.2075 - val_loss: 0.1912
Epoch 4/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1868 - val_loss: 0.1729
Epoch 5/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1719 - val_loss: 0.1654
Epoch 6/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1656 - val_loss: 0.1612
Epoch 7/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1620 - val_loss: 0.1582
Epoch 8/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1593 - val_loss: 0.1562
```

+ Code + Text

[3]
```
Epoch 9/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1574 - val_loss: 0.1544
Epoch 10/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1552 - val_loss: 0.1530
Epoch 11/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1539 - val_loss: 0.1516
Epoch 12/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1528 - val_loss: 0.1504
Epoch 13/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1515 - val_loss: 0.1490
Epoch 14/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1504 - val_loss: 0.1478
Epoch 15/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1486 - val_loss: 0.1468
Epoch 16/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1480 - val_loss: 0.1449
Epoch 17/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1458 - val_loss: 0.1439
Epoch 18/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1452 - val_loss: 0.1434
Epoch 19/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1443 - val_loss: 0.1428
Epoch 20/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1439 - val_loss: 0.1424
Epoch 21/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1437 - val_loss: 0.1421
```

+ Code + Text

[3]
```
Epoch 22/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1437 - val_loss: 0.1418
Epoch 23/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1432 - val_loss: 0.1414
Epoch 24/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1426 - val_loss: 0.1413
Epoch 25/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1427 - val_loss: 0.1410
Epoch 26/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1422 - val_loss: 0.1406
Epoch 27/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1415 - val_loss: 0.1399
Epoch 28/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1412 - val_loss: 0.1392
Epoch 29/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1403 - val_loss: 0.1388
Epoch 30/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1401 - val_loss: 0.1385
<keras.src.callbacks.history.History at 0x78c83281dc60>
```

+ Code + Text

[4]
```python
import numpy as np
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.datasets import mnist
from tensorflow.keras.callbacks import ModelCheckpoint

# Define the ModelCheckpoint callback
checkpoint = ModelCheckpoint(filepath='autoencoder_best.keras',  # File path to save the model
                             monitor='val_loss',  # Metric to monitor
                             save_best_only=True,  # Save only the best model (based on the monitored metric)
                             mode='min',  # Minimize the monitored metric (e.g., validation loss)
                             save_weights_only=False,  # Save the entire model (set to True to save only weights)
                             verbose=1)  # Print a message when saving the model


# Load the MNIST dataset
(x_train, _), (x_test, _) = mnist.load_data()

# Normalize pixel values to the range [0, 1]
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.

# Flatten the images for the autoencoder
x_train = x_train.reshape((len(x_train), -1))  # -1 infers the remaining dimension
```

+ Code + Text

```
[4] x_test = x_test.reshape((len(x_test), -1))  # -1 infers the remain

    # Define the dimensions of the input and the encoded representation
    input_dim = x_train.shape[1]
    encoding_dim = 16  # Compress to 16 features

    # Define the input layer
    input_layer = Input(shape=(input_dim,))

    # Define the encoder
    encoded = Dense(encoding_dim, activation='relu')(input_layer)
    # Adding a layer
    encoded1 = Dense(encoding_dim, activation='relu')(encoded)

    # Adding a layer
    decoded1 = Dense(encoding_dim, activation='relu')(encoded1)
    # Define the decoder
    decoded = Dense(input_dim, activation='sigmoid')(decoded1)

    # Combine the encoder and decoder into an autoencoder model
    autoencoder = Model(input_layer, decoded)

    # Compile the autoencoder model
    autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

+ Code + Text

```
[4] # Train the autoencoder
    # Assuming x_train and x_test are your training and validation datasets
    autoencoder.fit(x_train, x_train,  # For autoencoders, input and output are the same
                    epochs=30,  # Number of epochs
                    batch_size=256,
                    shuffle=True,
                    validation_data=(x_test, x_test),  # Validation data
                    callbacks=[checkpoint])  # Add the ModelCheckpoint callback
```

```
Epoch 1/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.4222
Epoch 1: val_loss improved from inf to 0.23446, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 6ms/step - loss: 0.4218 - val_loss: 0.2345
Epoch 2/30
219/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2285
Epoch 2: val_loss improved from 0.23446 to 0.20068, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.2280 - val_loss: 0.2007
Epoch 3/30
231/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1969
Epoch 3: val_loss improved from 0.20068 to 0.17941, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1968 - val_loss: 0.1794
Epoch 4/30
228/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1783
Epoch 4: val_loss improved from 0.17941 to 0.16985, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1782 - val_loss: 0.1698
```

+ Code + Text

```
[4]
Epoch 5/30
226/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1697
Epoch 5: val_loss improved from 0.16985 to 0.16166, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1695 - val_loss: 0.1617
Epoch 6/30
225/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1617
Epoch 6: val_loss improved from 0.16166 to 0.15679, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1617 - val_loss: 0.1568
Epoch 7/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1576
Epoch 7: val_loss improved from 0.15679 to 0.15442, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1576 - val_loss: 0.1544
Epoch 8/30
224/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1553
Epoch 8: val_loss improved from 0.15442 to 0.15208, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1553 - val_loss: 0.1521
Epoch 9/30
227/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1531
Epoch 9: val_loss improved from 0.15208 to 0.15004, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1531 - val_loss: 0.1500
Epoch 10/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1509
Epoch 10: val_loss improved from 0.15004 to 0.14886, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1509 - val_loss: 0.1489
Epoch 11/30
229/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1501
```

+ Code + Text

```
[4]
Epoch 11: val_loss improved from 0.14886 to 0.14798, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1501 - val_loss: 0.1480
Epoch 12/30
224/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1489
Epoch 12: val_loss improved from 0.14798 to 0.14720, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1489 - val_loss: 0.1472
Epoch 13/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1488
Epoch 13: val_loss improved from 0.14720 to 0.14644, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1487 - val_loss: 0.1464
Epoch 14/30
224/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1476
Epoch 14: val_loss improved from 0.14644 to 0.14560, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1476 - val_loss: 0.1456
Epoch 15/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1469
Epoch 15: val_loss improved from 0.14560 to 0.14470, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1469 - val_loss: 0.1447
Epoch 16/30
224/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1459
Epoch 16: val_loss improved from 0.14470 to 0.14373, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1459 - val_loss: 0.1437
Epoch 17/30
221/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1450
Epoch 17: val_loss improved from 0.14373 to 0.14277, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1450 - val_loss: 0.1428
```

+ Code + Text

```
Epoch 18/30
226/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1438
Epoch 18: val_loss improved from 0.14277 to 0.14199, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1438 - val_loss: 0.1420
Epoch 19/30
221/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1435
Epoch 19: val_loss improved from 0.14199 to 0.14158, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1435 - val_loss: 0.1416
Epoch 20/30
225/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1429
Epoch 20: val_loss improved from 0.14158 to 0.14137, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1429 - val_loss: 0.1414
Epoch 21/30
226/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1427
Epoch 21: val_loss improved from 0.14137 to 0.14091, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1427 - val_loss: 0.1409
Epoch 22/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1427
Epoch 22: val_loss improved from 0.14091 to 0.14069, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1427 - val_loss: 0.1407
Epoch 23/30
229/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1419
Epoch 23: val_loss improved from 0.14069 to 0.14033, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1419 - val_loss: 0.1403
Epoch 24/30
217/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1420
```

+ Code + Text T4

```
Epoch 24: val_loss improved from 0.14033 to 0.14009, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1420 - val_loss: 0.1401
Epoch 25/30
222/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1415
Epoch 25: val_loss improved from 0.14009 to 0.14001, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1415 - val_loss: 0.1400
Epoch 26/30
210/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1415
Epoch 26: val_loss improved from 0.14001 to 0.13965, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1415 - val_loss: 0.1397
Epoch 27/30
208/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1411
Epoch 27: val_loss improved from 0.13965 to 0.13940, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1411 - val_loss: 0.1394
Epoch 28/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1406
Epoch 28: val_loss improved from 0.13940 to 0.13911, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1406 - val_loss: 0.1391
Epoch 29/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1409
Epoch 29: val_loss improved from 0.13911 to 0.13891, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1409 - val_loss: 0.1389
Epoch 30/30
234/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1404
Epoch 30: val_loss improved from 0.13891 to 0.13853, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1404 - val_loss: 0.1385
```

+ Code + Text T4 RAM Disk

```
<keras.src.callbacks.history.History at 0x78c82f5c2a10>
```

```python
import numpy as np
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.datasets import mnist
from tensorflow.keras.callbacks import ReduceLROnPlateau

# Define the ReduceLROnPlateau callback
reduce_lr = ReduceLROnPlateau(monitor='val_loss',  # Metric to monitor
                              factor=0.5,  # Factor by which the learning rate will be reduced (new_lr = lr * factor)
                              patience=3,  # Number of epochs with no improvement after which learning rate will be reduced
                              min_lr=1e-6,  # Lower bound for the learning rate
                              verbose=1)  # Print message when the learning rate is reduced


# Load the MNIST dataset
(x_train, _), (x_test, _) = mnist.load_data()

# Normalize pixel values to the range [0, 1]
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
```

+ Code + Text

```
[5]
# Flatten the images for the autoencoder
x_train = x_train.reshape((len(x_train), -1))  # -1 infers the remaining dimension
x_test = x_test.reshape((len(x_test), -1))  # -1 infers the remain

# Define the dimensions of the input and the encoded representation
input_dim = x_train.shape[1]
encoding_dim = 16  # Compress to 16 features

# Define the input layer
input_layer = Input(shape=(input_dim,))

# Define the encoder
encoded = Dense(encoding_dim, activation='relu')(input_layer)
# Adding a layer
encoded1 = Dense(encoding_dim, activation='relu')(encoded)

# Adding a layer
decoded1 = Dense(encoding_dim, activation='relu')(encoded1)
# Define the decoder
decoded = Dense(input_dim, activation='sigmoid')(decoded1)

# Combine the encoder and decoder into an autoencoder model
autoencoder = Model(input_layer, decoded)
```

+ Code + Text

```
[5]
# Compile the autoencoder model
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

# Train the autoencoder
# Assuming x_train and x_test are your training and validation datasets
autoencoder.fit(x_train, x_train,  # For autoencoders, input and output are the same
                epochs=30,  # Number of epochs
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test),  # Validation data
                callbacks=[reduce_lr])  # Add the ReduceLROnPlateau callback
```

```
Epoch 1/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 6ms/step - loss: 0.4483 - val_loss: 0.2595 - learning_rate: 0.0010
Epoch 2/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2496 - val_loss: 0.2196 - learning_rate: 0.0010
Epoch 3/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.2133 - val_loss: 0.1976 - learning_rate: 0.0010
Epoch 4/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1948 - val_loss: 0.1830 - learning_rate: 0.0010
Epoch 5/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1819 - val_loss: 0.1732 - learning_rate: 0.0010
Epoch 6/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1724 - val_loss: 0.1658 - learning_rate: 0.0010
Epoch 7/30
```

+ Code + Text T4

```
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1656 - val_loss: 0.1595 - learning_rate: 0.0010
Epoch 8/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1598 - val_loss: 0.1564 - learning_rate: 0.0010
Epoch 9/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1572 - val_loss: 0.1546 - learning_rate: 0.0010
Epoch 10/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1559 - val_loss: 0.1534 - learning_rate: 0.0010
Epoch 11/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1546 - val_loss: 0.1525 - learning_rate: 0.0010
Epoch 12/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1535 - val_loss: 0.1514 - learning_rate: 0.0010
Epoch 13/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1523 - val_loss: 0.1504 - learning_rate: 0.0010
Epoch 14/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1514 - val_loss: 0.1492 - learning_rate: 0.0010
Epoch 15/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1504 - val_loss: 0.1479 - learning_rate: 0.0010
Epoch 16/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1492 - val_loss: 0.1466 - learning_rate: 0.0010
Epoch 17/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1480 - val_loss: 0.1458 - learning_rate: 0.0010
Epoch 18/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1473 - val_loss: 0.1456 - learning_rate: 0.0010
Epoch 19/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1462 - val_loss: 0.1445 - learning_rate: 0.0010
```

+ Code + Text T4 RAM Disk Ger

```
Epoch 20/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1458 - val_loss: 0.1442 - learning_rate: 0.0010
Epoch 21/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1458 - val_loss: 0.1438 - learning_rate: 0.0010
Epoch 22/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1452 - val_loss: 0.1434 - learning_rate: 0.0010
Epoch 23/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1446 - val_loss: 0.1430 - learning_rate: 0.0010
Epoch 24/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1444 - val_loss: 0.1424 - learning_rate: 0.0010
Epoch 25/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1439 - val_loss: 0.1419 - learning_rate: 0.0010
Epoch 26/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1429 - val_loss: 0.1408 - learning_rate: 0.0010
Epoch 27/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1424 - val_loss: 0.1401 - learning_rate: 0.0010
Epoch 28/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1417 - val_loss: 0.1397 - learning_rate: 0.0010
Epoch 29/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1412 - val_loss: 0.1393 - learning_rate: 0.0010
Epoch 30/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1411 - val_loss: 0.1391 - learning_rate: 0.0010
<keras.src.callbacks.history.History at 0x78c8324c23e0>
```

+ Code + Text T4

```
import numpy as np
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.datasets import mnist
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, TerminateOnNaN, ReduceLROnPlateau

# EarlyStopping callback to stop training if validation loss stops improving
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# ModelCheckpoint callback to save the best model based on validation loss
checkpoint = ModelCheckpoint(filepath='autoencoder_best.keras', monitor='val_loss', save_best_only=True, verbose=1)

# TerminateOnNaN callback to stop training if the loss becomes NaN
terminate_on_nan = TerminateOnNaN()

# Define the ReduceLROnPlateau callback
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5,   patience=3, min_lr=1e-6, verbose=1)

# Load the MNIST dataset
(x_train, _), (x_test, _) = mnist.load_data()

# Normalize pixel values to the range [0, 1]
x_train = x_train.astype('float32') / 255.
```

+ Code + Text

```
[6]  x_test = x_test.astype('float32') / 255.

     # Flatten the images for the autoencoder
     x_train = x_train.reshape((len(x_train), -1))  # -1 infers the remaining dimension
     x_test = x_test.reshape((len(x_test), -1))  # -1 infers the remain

     # Define the dimensions of the input and the encoded representation
     input_dim = x_train.shape[1]
     encoding_dim = 16  # Compress to 16 features

     # Define the input layer
     input_layer = Input(shape=(input_dim,))

     # Define the encoder
     encoded = Dense(encoding_dim, activation='relu')(input_layer)
     # Adding a layer
     encoded1 = Dense(encoding_dim, activation='relu')(encoded)

     # Adding a layer
     decoded1 = Dense(encoding_dim, activation='relu')(encoded1)
     # Define the decoder
     decoded = Dense(input_dim, activation='sigmoid')(decoded1)
```

+ Code + Text

```
[6]  # Combine the encoder and decoder into an autoencoder model
     autoencoder = Model(input_layer, decoded)

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

     # Training with multiple callbacks
     autoencoder.fit(x_train, x_train,
                     epochs=30,  # You can set a high number of epochs
                     batch_size=256,
                     shuffle=True,
                     validation_data=(x_test, x_test),
                     callbacks=[reduce_lr, early_stopping, checkpoint, terminate_on_nan])  # Using multiple callbacks
```

```
Epoch 1/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.4443
Epoch 1: val_loss improved from inf to 0.23248, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 6ms/step - loss: 0.4438 - val_loss: 0.2325 - learning_rate: 0.0010
Epoch 2/30
223/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2223
Epoch 2: val_loss improved from 0.23248 to 0.19773, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.2218 - val_loss: 0.1977 - learning_rate: 0.0010
Epoch 3/30
218/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1955
```

+ Code + Text

```
[6]
Epoch 3: val_loss improved from 0.19773 to 0.18047, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1952 - val_loss: 0.1805 - learning_rate: 0.0010
Epoch 4/30
217/235 ━━━━━━━━━━━━━━━━━━── 0s 2ms/step - loss: 0.1793
Epoch 4: val_loss improved from 0.18047 to 0.17146, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1791 - val_loss: 0.1715 - learning_rate: 0.0010
Epoch 5/30
222/235 ━━━━━━━━━━━━━━━━━━── 0s 3ms/step - loss: 0.1713
Epoch 5: val_loss improved from 0.17146 to 0.16499, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1712 - val_loss: 0.1650 - learning_rate: 0.0010
Epoch 6/30
220/235 ━━━━━━━━━━━━━━━━━━── 0s 3ms/step - loss: 0.1653
Epoch 6: val_loss improved from 0.16499 to 0.15923, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1651 - val_loss: 0.1592 - learning_rate: 0.0010
Epoch 7/30
219/235 ━━━━━━━━━━━━━━━━━━── 0s 3ms/step - loss: 0.1603
Epoch 7: val_loss improved from 0.15923 to 0.15617, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1602 - val_loss: 0.1562 - learning_rate: 0.0010
Epoch 8/30
213/235 ━━━━━━━━━━━━━━━━━━── 0s 2ms/step - loss: 0.1573
Epoch 8: val_loss improved from 0.15617 to 0.15354, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1572 - val_loss: 0.1535 - learning_rate: 0.0010
Epoch 9/30
229/235 ━━━━━━━━━━━━━━━━━━━─ 0s 2ms/step - loss: 0.1542
Epoch 9: val_loss improved from 0.15354 to 0.15155, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1542 - val_loss: 0.1516 - learning_rate: 0.0010
```

+ Code + Text

```
[6]
Epoch 10/30
216/235 ━━━━━━━━━━━━━━━━━━── 0s 2ms/step - loss: 0.1524
Epoch 10: val_loss improved from 0.15155 to 0.15020, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1524 - val_loss: 0.1502 - learning_rate: 0.0010
Epoch 11/30
224/235 ━━━━━━━━━━━━━━━━━━━─ 0s 2ms/step - loss: 0.1514
Epoch 11: val_loss improved from 0.15020 to 0.14922, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1514 - val_loss: 0.1492 - learning_rate: 0.0010
Epoch 12/30
220/235 ━━━━━━━━━━━━━━━━━━── 0s 2ms/step - loss: 0.1504
Epoch 12: val_loss improved from 0.14922 to 0.14768, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1504 - val_loss: 0.1477 - learning_rate: 0.0010
Epoch 13/30
219/235 ━━━━━━━━━━━━━━━━━━── 0s 2ms/step - loss: 0.1484
Epoch 13: val_loss improved from 0.14768 to 0.14604, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1484 - val_loss: 0.1460 - learning_rate: 0.0010
Epoch 14/30
233/235 ━━━━━━━━━━━━━━━━━━━─ 0s 2ms/step - loss: 0.1475
Epoch 14: val_loss improved from 0.14604 to 0.14505, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1475 - val_loss: 0.1450 - learning_rate: 0.0010
Epoch 15/30
227/235 ━━━━━━━━━━━━━━━━━━━─ 0s 2ms/step - loss: 0.1460
Epoch 15: val_loss improved from 0.14505 to 0.14429, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1460 - val_loss: 0.1443 - learning_rate: 0.0010
Epoch 16/30
222/235 ━━━━━━━━━━━━━━━━━━── 0s 2ms/step - loss: 0.1457
```

+ Code + Text

```
Epoch 16: val_loss improved from 0.14429 to 0.14390, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1457 - val_loss: 0.1439 - learning_rate: 0.0010
Epoch 17/30
227/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1449
Epoch 17: val_loss improved from 0.14390 to 0.14340, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1449 - val_loss: 0.1434 - learning_rate: 0.0010
Epoch 18/30
233/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1448
Epoch 18: val_loss improved from 0.14340 to 0.14309, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1448 - val_loss: 0.1431 - learning_rate: 0.0010
Epoch 19/30
226/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1442
Epoch 19: val_loss improved from 0.14309 to 0.14289, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1442 - val_loss: 0.1429 - learning_rate: 0.0010
Epoch 20/30
233/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1444
Epoch 20: val_loss improved from 0.14289 to 0.14267, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1444 - val_loss: 0.1427 - learning_rate: 0.0010
Epoch 21/30
228/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1442
Epoch 21: val_loss improved from 0.14267 to 0.14262, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1442 - val_loss: 0.1426 - learning_rate: 0.0010
Epoch 22/30
224/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1442
Epoch 22: val_loss improved from 0.14262 to 0.14238, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1442 - val_loss: 0.1424 - learning_rate: 0.0010
```

+ Code + Text

```
Epoch 23/30
221/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1439
Epoch 23: val_loss improved from 0.14238 to 0.14217, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1439 - val_loss: 0.1422 - learning_rate: 0.0010
Epoch 24/30
213/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1437
Epoch 24: val_loss improved from 0.14217 to 0.14205, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1437 - val_loss: 0.1420 - learning_rate: 0.0010
Epoch 25/30
226/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1433
Epoch 25: val_loss improved from 0.14205 to 0.14192, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1434 - val_loss: 0.1419 - learning_rate: 0.0010
Epoch 26/30
218/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1436
Epoch 26: val_loss improved from 0.14192 to 0.14167, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1436 - val_loss: 0.1417 - learning_rate: 0.0010
Epoch 27/30
216/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1429
Epoch 27: val_loss improved from 0.14167 to 0.14154, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1429 - val_loss: 0.1415 - learning_rate: 0.0010
Epoch 28/30
220/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1431
Epoch 28: val_loss improved from 0.14154 to 0.14119, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1431 - val_loss: 0.1412 - learning_rate: 0.0010
Epoch 29/30
217/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1423
```

```
+ Code + Text

[6]
Epoch 29: val_loss improved from 0.14119 to 0.14118, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1424 - val_loss: 0.1412 - learning_rate: 0.0010
Epoch 30/30
229/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1423
Epoch 30: val_loss improved from 0.14118 to 0.14075, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1423 - val_loss: 0.1408 - learning_rate: 0.0010
<keras.src.callbacks.history.History at 0x78c8322cb460>

from tensorflow.keras.models import load_model

# Load the entire model
best_autoencoder = load_model('autoencoder_best.keras')

# Let's look at the encoded representations
encoded_data = best_autoencoder.predict(x_test)
print(encoded_data)
print(encoded_data.shape)

313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
[[3.59317298e-11 5.51484379e-11 3.78944209e-10 ... 2.84300472e-10
  1.03231493e-10 1.13847626e-10]
 [5.89395232e-13 3.05378075e-12 1.05212236e-12 ... 3.90656528e-13
  2.99522906e-12 3.54769426e-12]
 [1.31491551e-09 3.06862136e-09 2.98511638e-09 ... 1.91978522e-09
  1.97679295e-09 3.38054784e-09]
 ...
 [3.54246882e-15 1.93683841e-14 1.10577752e-13 ... 5.41742707e-15
  5.96521979e-15 3.26632913e-14]
 [1.61036698e-10 1.81973203e-09 2.44318410e-09 ... 1.38981071e-09
  1.17646861e-08 1.72258197e-09]
 [3.09676807e-16 1.30165578e-16 1.50808168e-16 ... 6.92074184e-18
  3.96147701e-16 4.46666731e-16]]
(10000, 784)
```

My github repository link:-
https://github.com/PraveenDondapati/bda.git