ICP6 REPORT

+ Code + Text

1m

```
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

# 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

```
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1450 - val_loss: 0.1429
Epoch 13/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1440 - val_loss: 0.1416
Epoch 14/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1428 - val_loss: 0.1404
Epoch 15/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1415 - val_loss: 0.1395
Epoch 16/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1409 - val_loss: 0.1390
Epoch 17/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1401 - val_loss: 0.1383
Epoch 18/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1398 - val_loss: 0.1382
Epoch 19/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1395 - val_loss: 0.1378
Epoch 20/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1392 - val_loss: 0.1377
Epoch 21/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1388 - val_loss: 0.1374
Epoch 22/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1385 - val_loss: 0.1372
Epoch 23/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1386 - val_loss: 0.1371
Epoch 24/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1384 - val_loss: 0.1368
Epoch 25/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1381 - val_loss: 0.1366
Epoch 26/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1383 - val_loss: 0.1365
Epoch 27/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1379 - val_loss: 0.1363
Epoch 28/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1377 - val_loss: 0.1362
Epoch 29/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1375 - val_loss: 0.1360
Epoch 30/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1374 - val_loss: 0.1359
Epoch 31/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1371 - val_loss: 0.1356
Epoch 32/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1368 - val_loss: 0.1353
Epoch 33/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1367 - val_loss: 0.1352
```

+ Code + Text

```
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1450 - val_loss: 0.1429
Epoch 13/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1440 - val_loss: 0.1416
Epoch 14/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1428 - val_loss: 0.1404
Epoch 15/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1415 - val_loss: 0.1395
Epoch 16/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1409 - val_loss: 0.1390
Epoch 17/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1401 - val_loss: 0.1383
Epoch 18/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1398 - val_loss: 0.1382
Epoch 19/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1395 - val_loss: 0.1378
Epoch 20/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1392 - val_loss: 0.1377
Epoch 21/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1388 - val_loss: 0.1374
Epoch 22/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1385 - val_loss: 0.1372
Epoch 23/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1386 - val_loss: 0.1371
Epoch 24/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1384 - val_loss: 0.1368
Epoch 25/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1381 - val_loss: 0.1366
Epoch 26/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1383 - val_loss: 0.1365
Epoch 27/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1379 - val_loss: 0.1363
Epoch 28/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1377 - val_loss: 0.1362
Epoch 29/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1375 - val_loss: 0.1360
Epoch 30/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1374 - val_loss: 0.1359
Epoch 31/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1371 - val_loss: 0.1356
Epoch 32/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1368 - val_loss: 0.1353
Epoch 33/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1367 - val_loss: 0.1352
```

+ Code + Text

1m

```
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1367 - val_loss: 0.1352
Epoch 34/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1366 - val_loss: 0.1349
Epoch 35/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1363 - val_loss: 0.1346
Epoch 36/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1360 - val_loss: 0.1344
Epoch 37/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1357 - val_loss: 0.1340
Epoch 38/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1354 - val_loss: 0.1338
Epoch 39/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1353 - val_loss: 0.1336
Epoch 40/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1349 - val_loss: 0.1332
Epoch 41/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1347 - val_loss: 0.1329
Epoch 42/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1341 - val_loss: 0.1329
Epoch 43/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1345 - val_loss: 0.1324
Epoch 44/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1339 - val_loss: 0.1324
Epoch 45/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1333 - val_loss: 0.1322
Epoch 46/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1337 - val_loss: 0.1318
Epoch 47/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1334 - val_loss: 0.1316
Epoch 48/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1334 - val_loss: 0.1316
Epoch 49/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1331 - val_loss: 0.1312
Epoch 50/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1326 - val_loss: 0.1310
Epoch 51/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1326 - val_loss: 0.1309
Epoch 52/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1325 - val_loss: 0.1308
Epoch 53/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1324 - val_loss: 0.1305
Epoch 54/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1324 - val_loss: 0.1306
```

+ Code + Text

1m

```
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1324 - val_loss: 0.1306
Epoch 55/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1323 - val_loss: 0.1305
Epoch 56/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1321 - val_loss: 0.1302
Epoch 57/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1318 - val_loss: 0.1304
Epoch 58/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1318 - val_loss: 0.1302
Epoch 59/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1320 - val_loss: 0.1300
Epoch 60/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1320 - val_loss: 0.1300
Epoch 61/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1320 - val_loss: 0.1299
Epoch 62/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1316 - val_loss: 0.1300
Epoch 63/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1315 - val_loss: 0.1298
Epoch 64/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1311 - val_loss: 0.1297
Epoch 65/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1314 - val_loss: 0.1298
Epoch 66/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1312 - val_loss: 0.1297
Epoch 67/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1311 - val_loss: 0.1296
Epoch 68/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1314 - val_loss: 0.1296
Epoch 69/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1310 - val_loss: 0.1297
Epoch 70/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1314 - val_loss: 0.1294
Epoch 71/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1312 - val_loss: 0.1295
Epoch 72/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1309 - val_loss: 0.1295
Epoch 73/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1315 - val_loss: 0.1295
Epoch 74/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1308 - val_loss: 0.1294
Epoch 75/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1309 - val_loss: 0.1293
```

+ Code + Text

```
Epoch 76/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1310 - val_loss: 0.1294
Epoch 77/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1309 - val_loss: 0.1293
Epoch 78/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1310 - val_loss: 0.1292
Epoch 79/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1310 - val_loss: 0.1293
Epoch 80/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1307 - val_loss: 0.1292
Epoch 81/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1311 - val_loss: 0.1294
Epoch 82/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1309 - val_loss: 0.1292
Epoch 83/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1310 - val_loss: 0.1292
Epoch 84/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1309 - val_loss: 0.1291
Epoch 85/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1306 - val_loss: 0.1292
Epoch 86/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1307 - val_loss: 0.1292
Epoch 87/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1303 - val_loss: 0.1291
Epoch 88/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1308 - val_loss: 0.1291
Epoch 89/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1306 - val_loss: 0.1292
Epoch 90/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1305 - val_loss: 0.1291
Epoch 91/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1306 - val_loss: 0.1290
Epoch 92/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1308 - val_loss: 0.1290
Epoch 93/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1307 - val_loss: 0.1291
Epoch 94/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1308 - val_loss: 0.1291
Epoch 95/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1306 - val_loss: 0.1290
Epoch 96/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1308 - val_loss: 0.1290
```

+ Code + Text

```
Epoch 97/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1305 - val_loss: 0.1291
Epoch 98/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1306 - val_loss: 0.1289
Epoch 99/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1305 - val_loss: 0.1291
Epoch 100/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1307 - val_loss: 0.1291
<keras.src.callbacks.history.History at 0x7bac6015db40>
```

[2]

```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 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

# 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)
```

```
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')

# 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
```

```
235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.2271 - val_loss: 0.1967
Epoch 3/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1944 - val_loss: 0.1833
Epoch 4/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1815 - val_loss: 0.1729
Epoch 5/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1731 - val_loss: 0.1669
Epoch 6/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1660 - val_loss: 0.1596
Epoch 7/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1601 - val_loss: 0.1566
Epoch 8/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1574 - val_loss: 0.1546
Epoch 9/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1554 - val_loss: 0.1528
Epoch 10/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1544 - val_loss: 0.1517
```

+ Code + Text

```
Epoch 11/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1529 - val_loss: 0.1508
Epoch 12/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1524 - val_loss: 0.1502
Epoch 13/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1516 - val_loss: 0.1497
Epoch 14/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1509 - val_loss: 0.1493
Epoch 15/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1507 - val_loss: 0.1490
Epoch 16/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1503 - val_loss: 0.1485
Epoch 17/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1496 - val_loss: 0.1480
Epoch 18/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1495 - val_loss: 0.1476
Epoch 19/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1490 - val_loss: 0.1473
Epoch 20/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1488 - val_loss: 0.1468
Epoch 21/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1482 - val_loss: 0.1465
Epoch 22/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1479 - val_loss: 0.1457
Epoch 23/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1469 - val_loss: 0.1452
Epoch 24/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1464 - val_loss: 0.1441
Epoch 25/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1453 - val_loss: 0.1437
Epoch 26/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1448 - val_loss: 0.1433
Epoch 27/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1447 - val_loss: 0.1429
Epoch 28/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1444 - val_loss: 0.1425
Epoch 29/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1440 - val_loss: 0.1423
Epoch 30/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1436 - val_loss: 0.1422
<keras.src.callbacks.history.History at 0x7bac403a08e0>
```

+ Code + Text

T4 RAM Disk

```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
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
```

+ Code + Text

```python
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')

# 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
```

```
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1528 - val_loss: 0.1507
Epoch 17/30
216/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1516
Epoch 17: val_loss improved from 0.15074 to 0.15013, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1516 - val_loss: 0.1501
Epoch 18/30
221/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1507
Epoch 18: val_loss improved from 0.15013 to 0.14951, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1507 - val_loss: 0.1495
Epoch 19/30
209/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1505
Epoch 19: val_loss improved from 0.14951 to 0.14899, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1505 - val_loss: 0.1490
Epoch 20/30
220/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1497
Epoch 20: val_loss improved from 0.14899 to 0.14828, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1497 - val_loss: 0.1483
Epoch 21/30
211/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1489
Epoch 21: val_loss improved from 0.14828 to 0.14792, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1490 - val_loss: 0.1479
Epoch 22/30
225/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1490
```

+ Code + Text

[3]
```
Epoch 23/30
222/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1485
Epoch 23: val_loss improved from 0.14755 to 0.14718, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1485 - val_loss: 0.1472
Epoch 24/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1481
Epoch 24: val_loss improved from 0.14718 to 0.14687, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1481 - val_loss: 0.1469
Epoch 25/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1483
Epoch 25: val_loss improved from 0.14687 to 0.14654, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1483 - val_loss: 0.1465
Epoch 26/30
212/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1472
Epoch 26: val_loss improved from 0.14654 to 0.14638, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1473 - val_loss: 0.1464
Epoch 27/30
231/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1473
Epoch 27: val_loss improved from 0.14638 to 0.14606, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1473 - val_loss: 0.1461
Epoch 28/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1475
Epoch 28: val_loss improved from 0.14606 to 0.14602, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1475 - val_loss: 0.1460
Epoch 29/30
229/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1474
Epoch 29: val_loss improved from 0.14602 to 0.14578, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1474 - val_loss: 0.1458
Epoch 30/30
211/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1473
Epoch 30: val_loss improved from 0.14578 to 0.14575, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1472 - val_loss: 0.1458
<keras.src.callbacks.history.History at 0x7bac5c5d8670>
```

[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 ReduceLROnPlateau

# Define the ReduceLROnPlateau callback
```

+ Code + Text

[4] 27s

```
# 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.

# 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

[4]
```python
# 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 2/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2320 - val_loss: 0.1968 - learning_rate: 0.0010
Epoch 3/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1932 - val_loss: 0.1819 - learning_rate: 0.0010
Epoch 4/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1810 - val_loss: 0.1701 - learning_rate: 0.0010
Epoch 5/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1696 - val_loss: 0.1628 - learning_rate: 0.0010
Epoch 6/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1629 - val_loss: 0.1586 - learning_rate: 0.0010
Epoch 7/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1602 - val_loss: 0.1569 - learning_rate: 0.0010
Epoch 8/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1586 - val_loss: 0.1558 - learning_rate: 0.0010
Epoch 9/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1570 - val_loss: 0.1549 - learning_rate: 0.0010
Epoch 10/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1564 - val_loss: 0.1541 - learning_rate: 0.0010
Epoch 11/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1551 - val_loss: 0.1534 - learning_rate: 0.0010
Epoch 12/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1544 - val_loss: 0.1528 - learning_rate: 0.0010
Epoch 13/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1539 - val_loss: 0.1523 - learning_rate: 0.0010
Epoch 14/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1538 - val_loss: 0.1518 - learning_rate: 0.0010
Epoch 15/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1532 - val_loss: 0.1514 - learning_rate: 0.0010
Epoch 16/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1527 - val_loss: 0.1509 - learning_rate: 0.0010
```

+ Code + Text

[4]

```
Epoch 17/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1523 - val_loss: 0.1506 - learning_rate: 0.0010
Epoch 18/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1521 - val_loss: 0.1503 - learning_rate: 0.0010
Epoch 19/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1517 - val_loss: 0.1500 - learning_rate: 0.0010
Epoch 20/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1511 - val_loss: 0.1499 - learning_rate: 0.0010
Epoch 21/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1509 - val_loss: 0.1494 - learning_rate: 0.0010
Epoch 22/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1507 - val_loss: 0.1491 - learning_rate: 0.0010
Epoch 23/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1506 - val_loss: 0.1488 - learning_rate: 0.0010
Epoch 24/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1502 - val_loss: 0.1485 - learning_rate: 0.0010
Epoch 25/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1497 - val_loss: 0.1481 - learning_rate: 0.0010
Epoch 26/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1495 - val_loss: 0.1478 - learning_rate: 0.0010
Epoch 27/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1489 - val_loss: 0.1476 - learning_rate: 0.0010
Epoch 28/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1490 - val_loss: 0.1473 - learning_rate: 0.0010
Epoch 29/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1487 - val_loss: 0.1471 - learning_rate: 0.0010
Epoch 30/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1485 - val_loss: 0.1470 - learning_rate: 0.0010
<keras.src.callbacks.history.History at 0x7bac5c284a60>
```

```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 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)
```

+ Code + Text

✓ 27s

[5]
```
# 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.
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)

# 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

[5]
```
# 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
```

```
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1594 - val_loss: 0.1581 - learning_rate: 0.0010
Epoch 17/30
222/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1593
Epoch 17: val_loss improved from 0.15806 to 0.15775, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1593 - val_loss: 0.1578 - learning_rate: 0.0010
Epoch 18/30
217/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1590
Epoch 18: val_loss improved from 0.15775 to 0.15729, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1590 - val_loss: 0.1573 - learning_rate: 0.0010
Epoch 19/30
231/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1585
Epoch 19: val_loss improved from 0.15729 to 0.15685, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1585 - val_loss: 0.1569 - learning_rate: 0.0010
Epoch 20/30
211/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1583
Epoch 20: val_loss improved from 0.15685 to 0.15666, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1583 - val_loss: 0.1567 - learning_rate: 0.0010
Epoch 21/30
226/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1582
Epoch 21: val_loss improved from 0.15666 to 0.15624, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1582 - val_loss: 0.1562 - learning_rate: 0.0010
Epoch 22/30
224/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1575
Epoch 22: val_loss improved from 0.15624 to 0.15600, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1575 - val_loss: 0.1560 - learning_rate: 0.0010
Epoch 23/30
214/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1574
Epoch 23: val_loss improved from 0.15600 to 0.15568, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1574 - val_loss: 0.1557 - learning_rate: 0.0010
Epoch 24/30
227/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1571
Epoch 24: val_loss improved from 0.15568 to 0.15534, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1571 - val_loss: 0.1553 - learning_rate: 0.0010
Epoch 25/30
```

```
+ Code + Text

[ ] 232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1565
Epoch 26: val_loss improved from 0.15513 to 0.15493, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1565 - val_loss: 0.1549 - learning_rate: 0.0010
Epoch 27/30
224/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1559
Epoch 27: val_loss improved from 0.15493 to 0.15481, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1560 - val_loss: 0.1548 - learning_rate: 0.0010
Epoch 28/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1560
Epoch 28: val_loss improved from 0.15481 to 0.15439, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1560 - val_loss: 0.1544 - learning_rate: 0.0010
Epoch 29/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1553
Epoch 29: val_loss improved from 0.15439 to 0.15410, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1553 - val_loss: 0.1541 - learning_rate: 0.0010
Epoch 30/30
226/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1555
Epoch 30: val_loss improved from 0.15410 to 0.15390, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1555 - val_loss: 0.1539 - learning_rate: 0.0010
<keras.src.callbacks.history.History at 0x7bac5fcb9930>

[ ] 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
[[3.04569947e-09 8.48431370e-09 6.95855320e-08 ... 4.42325021e-09
  6.57742802e-08 2.67733657e-09]
 [3.80927765e-13 1.47661321e-12 1.31823146e-11 ... 3.93063403e-12
  1.89269564e-12 2.19412921e-12]
 [1.33664594e-14 9.45564265e-13 3.16072580e-13 ... 1.58756315e-15
  1.10404741e-13 1.39074936e-13]
 ...
 [1.53665069e-12 3.36058763e-12 3.26837203e-11 ... 3.21588893e-12
```

```
+ Code + Text

✓ 2s [6]
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
[[3.04569947e-09 8.48431370e-09 6.95855320e-08 ... 4.42325021e-09
  6.57742802e-08 2.67733657e-09]
 [3.80927765e-13 1.47661321e-12 1.31823146e-11 ... 3.93063403e-12
  1.89269564e-12 2.19412921e-12]
 [1.33664594e-14 9.45564265e-13 3.16072580e-13 ... 1.58756315e-15
  1.10404741e-13 1.39074936e-13]
 ...
 [1.53665069e-12 3.36058763e-12 3.26837203e-11 ... 3.21588893e-12
  1.96218198e-11 1.05144079e-11]
 [6.90370983e-10 6.96731228e-10 2.61842059e-09 ... 1.12616871e-09
  2.41975018e-09 6.43706421e-10]
 [8.72439816e-13 2.37235036e-13 1.15436285e-12 ... 9.12576004e-12
  5.95949451e-13 1.08045550e-12]]
(10000, 784)
```

My github link:https://github.com/akshaykumarpathem/bda.git