ICP6 REPORT

+ Code + Text

```
[3] 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

```
[3] # 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

```
[3] # 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 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - loss: 0.4281 - val_loss: 0.2468
Epoch 2/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.2368 - val_loss: 0.2001
Epoch 3/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1950 - val_loss: 0.1788
Epoch 4/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1780 - val_loss: 0.1716
Epoch 5/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1711 - val_loss: 0.1641
Epoch 6/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1642 - val_loss: 0.1586
Epoch 7/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1593 - val_loss: 0.1545
Epoch 8/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1554 - val_loss: 0.1518
Epoch 9/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1529 - val_loss: 0.1501
```

+ Code + Text

```
[3] Epoch 10/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1517 - val_loss: 0.1487
Epoch 11/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1498 - val_loss: 0.1475
Epoch 12/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1489 - val_loss: 0.1466
Epoch 13/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1477 - val_loss: 0.1457
Epoch 14/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1472 - val_loss: 0.1451
Epoch 15/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1462 - val_loss: 0.1443
Epoch 16/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1459 - val_loss: 0.1438
Epoch 17/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1450 - val_loss: 0.1433
Epoch 18/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1445 - val_loss: 0.1428
Epoch 19/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1446 - val_loss: 0.1423
Epoch 20/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1438 - val_loss: 0.1422
Epoch 21/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1433 - val_loss: 0.1415
Epoch 22/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1427 - val_loss: 0.1412
```

```
+ Code + Text

[3]
Epoch 23/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1429 - val_loss: 0.1410
Epoch 24/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1422 - val_loss: 0.1407
Epoch 25/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1419 - val_loss: 0.1403
Epoch 26/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1418 - val_loss: 0.1402
Epoch 27/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1416 - val_loss: 0.1399
Epoch 28/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1415 - val_loss: 0.1397
Epoch 29/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1412 - val_loss: 0.1395
Epoch 30/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1415 - val_loss: 0.1394
Epoch 31/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1405 - val_loss: 0.1393
Epoch 32/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1408 - val_loss: 0.1392
Epoch 33/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1405 - val_loss: 0.1391
Epoch 34/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1406 - val_loss: 0.1391
Epoch 35/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1403 - val_loss: 0.1389
```

```
+ Code + Text

[3]
Epoch 36/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1402 - val_loss: 0.1389
Epoch 37/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1404 - val_loss: 0.1388
Epoch 38/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1402 - val_loss: 0.1388
Epoch 39/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1401 - val_loss: 0.1387
Epoch 40/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1398 - val_loss: 0.1385
Epoch 41/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1401 - val_loss: 0.1385
Epoch 42/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1394 - val_loss: 0.1386
Epoch 43/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1397 - val_loss: 0.1384
Epoch 44/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1392 - val_loss: 0.1383
Epoch 45/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1400 - val_loss: 0.1385
Epoch 46/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1394 - val_loss: 0.1383
Epoch 47/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1390 - val_loss: 0.1382
Epoch 48/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1396 - val_loss: 0.1382
```

+ Code + Text

```
[3]  Epoch 49/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1395 - val_loss: 0.1382
     Epoch 50/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1392 - val_loss: 0.1382
     Epoch 51/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1393 - val_loss: 0.1381
     Epoch 52/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1392 - val_loss: 0.1381
     Epoch 53/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1397 - val_loss: 0.1380
     Epoch 54/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.1396 - val_loss: 0.1379
     Epoch 55/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1392 - val_loss: 0.1382
     Epoch 56/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1389 - val_loss: 0.1381
     Epoch 57/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1390 - val_loss: 0.1380
     Epoch 58/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1390 - val_loss: 0.1379
     Epoch 59/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1391 - val_loss: 0.1378
     Epoch 60/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1393 - val_loss: 0.1377
     Epoch 61/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1391 - val_loss: 0.1378
```

+ Code + Text

```
[3]  Epoch 62/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1389 - val_loss: 0.1378
     Epoch 63/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1391 - val_loss: 0.1379
     Epoch 64/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1391 - val_loss: 0.1377
     Epoch 65/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.1387 - val_loss: 0.1378
     Epoch 66/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.1391 - val_loss: 0.1379
     Epoch 67/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1388 - val_loss: 0.1377
     Epoch 68/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1389 - val_loss: 0.1377
     Epoch 69/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1385 - val_loss: 0.1378
     Epoch 70/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1386 - val_loss: 0.1376
     Epoch 71/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1388 - val_loss: 0.1377
     Epoch 72/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1389 - val_loss: 0.1377
     Epoch 73/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1388 - val_loss: 0.1376
     Epoch 74/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1389 - val_loss: 0.1376
```

```
+ Code + Text

[3]
Epoch 75/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1387 - val_loss: 0.1376
Epoch 76/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1390 - val_loss: 0.1377
Epoch 77/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1389 - val_loss: 0.1376
Epoch 78/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1388 - val_loss: 0.1376
Epoch 79/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1388 - val_loss: 0.1376
Epoch 80/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1386 - val_loss: 0.1376
Epoch 81/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1388 - val_loss: 0.1376
Epoch 82/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1392 - val_loss: 0.1375
Epoch 83/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1385 - val_loss: 0.1375
Epoch 84/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1388 - val_loss: 0.1375
Epoch 85/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1387 - val_loss: 0.1375
Epoch 86/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1387 - val_loss: 0.1375
Epoch 87/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1388 - val_loss: 0.1374
```

```
+ Code + Text

[3]
Epoch 88/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1384 - val_loss: 0.1375
Epoch 89/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1388 - val_loss: 0.1374
Epoch 90/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1390 - val_loss: 0.1375
Epoch 91/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1388 - val_loss: 0.1375
Epoch 92/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1383 - val_loss: 0.1374
Epoch 93/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1382 - val_loss: 0.1375
Epoch 94/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1386 - val_loss: 0.1374
Epoch 95/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1384 - val_loss: 0.1374
Epoch 96/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1388 - val_loss: 0.1373
Epoch 97/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1387 - val_loss: 0.1373
Epoch 98/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1389 - val_loss: 0.1373
Epoch 99/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1385 - val_loss: 0.1374
Epoch 100/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1387 - val_loss: 0.1373
<keras.src.callbacks.history.History at 0x7d494ce92f80>
```

+ Code + Text

```
[4]  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
```

+ Code + Text

```
[4]  # 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')

     # 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,
```

+ Code + Text

```
[4]                 shuffle=True,
                    validation_data=(x_test, x_test),
                    callbacks=[terminate_on_nan])  # Add the TerminateOnNaN callback
```

```
Epoch 1/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step - loss: 0.4253 - val_loss: 0.2428
Epoch 2/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.2291 - val_loss: 0.1982
Epoch 3/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1940 - val_loss: 0.1826
Epoch 4/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1815 - val_loss: 0.1745
Epoch 5/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1740 - val_loss: 0.1682
Epoch 6/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1684 - val_loss: 0.1635
Epoch 7/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1641 - val_loss: 0.1604
Epoch 8/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1613 - val_loss: 0.1582
Epoch 9/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1587 - val_loss: 0.1566
Epoch 10/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1576 - val_loss: 0.1548
Epoch 11/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1556 - val_loss: 0.1537
```

+ Code + Text

```
[4]
Epoch 12/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1553 - val_loss: 0.1531
Epoch 13/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1537 - val_loss: 0.1523
Epoch 14/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1535 - val_loss: 0.1516
Epoch 15/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1527 - val_loss: 0.1512
Epoch 16/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1527 - val_loss: 0.1507
Epoch 17/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1517 - val_loss: 0.1503
Epoch 18/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1514 - val_loss: 0.1499
Epoch 19/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1512 - val_loss: 0.1495
Epoch 20/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1510 - val_loss: 0.1493
Epoch 21/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1507 - val_loss: 0.1489
Epoch 22/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1501 - val_loss: 0.1486
Epoch 23/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1498 - val_loss: 0.1483
Epoch 24/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.1496 - val_loss: 0.1478
```

+ Code + Text

```
[4]  Epoch 25/30
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1491 - val_loss: 0.1475
     Epoch 26/30
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1489 - val_loss: 0.1472
     Epoch 27/30
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1483 - val_loss: 0.1466
     Epoch 28/30
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1476 - val_loss: 0.1465
     Epoch 29/30
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1475 - val_loss: 0.1461
     Epoch 30/30
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1474 - val_loss: 0.1460
     <keras.src.callbacks.history.History at 0x7d494b362620>
```

+ Code + Text

T4 RAM Disk

```
[5]  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.
```

+ 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=[checkpoint])  # Add the ModelCheckpoint callback
```

```
Epoch 1/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.4141
Epoch 1: val_loss improved from inf to 0.24290, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 0.4136 - val_loss: 0.2429
Epoch 2/30
224/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.2340
Epoch 2: val_loss improved from 0.24290 to 0.20782, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.2335 - val_loss: 0.2078
Epoch 3/30
218/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2032
Epoch 3: val_loss improved from 0.20782 to 0.18541, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.2027 - val_loss: 0.1854
```

```
+ Code + Text

[5]
Epoch 4/30
220/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1839
Epoch 4: val_loss improved from 0.18541 to 0.17501, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1838 - val_loss: 0.1750
Epoch 5/30
227/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1740
Epoch 5: val_loss improved from 0.17501 to 0.16726, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1740 - val_loss: 0.1673
Epoch 6/30
217/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1670
Epoch 6: val_loss improved from 0.16726 to 0.15976, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1669 - val_loss: 0.1598
Epoch 7/30
231/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1600
Epoch 7: val_loss improved from 0.15976 to 0.15482, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1599 - val_loss: 0.1548
Epoch 8/30
223/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1554
Epoch 8: val_loss improved from 0.15482 to 0.15130, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1553 - val_loss: 0.1513
Epoch 9/30
222/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1520
Epoch 9: val_loss improved from 0.15130 to 0.14905, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1520 - val_loss: 0.1491
Epoch 10/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1502
```

```
+ Code + Text

[5]
Epoch 10: val_loss improved from 0.14905 to 0.14721, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1502 - val_loss: 0.1472
Epoch 11/30
222/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1479
Epoch 11: val_loss improved from 0.14721 to 0.14489, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1478 - val_loss: 0.1449
Epoch 12/30
224/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1462
Epoch 12: val_loss improved from 0.14489 to 0.14356, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1461 - val_loss: 0.1436
Epoch 13/30
226/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1449
Epoch 13: val_loss improved from 0.14356 to 0.14306, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1449 - val_loss: 0.1431
Epoch 14/30
222/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1443
Epoch 14: val_loss improved from 0.14306 to 0.14247, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1443 - val_loss: 0.1425
Epoch 15/30
228/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1438
Epoch 15: val_loss improved from 0.14247 to 0.14222, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1438 - val_loss: 0.1422
Epoch 16/30
229/235 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.1436
Epoch 16: val_loss improved from 0.14222 to 0.14186, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.1436 - val_loss: 0.1419
```

```
+ Code + Text

[5]
Epoch 17/30
214/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1437
Epoch 17: val_loss improved from 0.14186 to 0.14139, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1436 - val_loss: 0.1414
Epoch 18/30
220/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1432
Epoch 18: val_loss improved from 0.14139 to 0.14115, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1431 - val_loss: 0.1411
Epoch 19/30
215/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1427
Epoch 19: val_loss improved from 0.14115 to 0.14086, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1426 - val_loss: 0.1409
Epoch 20/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1423
Epoch 20: val_loss improved from 0.14086 to 0.14069, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1423 - val_loss: 0.1407
Epoch 21/30
213/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1424
Epoch 21: val_loss improved from 0.14069 to 0.14040, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1423 - val_loss: 0.1404
Epoch 22/30
227/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1419
Epoch 22: val_loss improved from 0.14040 to 0.14014, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1419 - val_loss: 0.1401
Epoch 23/30
215/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1418
```

```
+ Code + Text

[5]
Epoch 23: val_loss improved from 0.14014 to 0.14000, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1418 - val_loss: 0.1400
Epoch 24/30
234/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1416
Epoch 24: val_loss improved from 0.14000 to 0.13985, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1416 - val_loss: 0.1398
Epoch 25/30
214/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1413
Epoch 25: val_loss improved from 0.13985 to 0.13959, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1413 - val_loss: 0.1396
Epoch 26/30
230/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1409
Epoch 26: val_loss improved from 0.13959 to 0.13939, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1409 - val_loss: 0.1394
Epoch 27/30
226/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1405
Epoch 27: val_loss improved from 0.13939 to 0.13906, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1405 - val_loss: 0.1391
Epoch 28/30
230/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1408
Epoch 28: val_loss improved from 0.13906 to 0.13885, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1408 - val_loss: 0.1389
Epoch 29/30
216/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1403
Epoch 29: val_loss improved from 0.13885 to 0.13849, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1403 - val_loss: 0.1385
```

+ Code + Text

T4 RAM Disk

```
Epoch 30/30
[5] 212/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1400
Epoch 30: val_loss improved from 0.13849 to 0.13818, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1399 - val_loss: 0.1382
<keras.src.callbacks.history.History at 0x7d494c348460>
```

```
[6] 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()
```

+ Code + Text

```
[6] # Normalize pixel values to the range [0, 1]
31s 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)
```

+ Code + Text

[6] 31s

```
# 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=[reduce_lr])  # Add the ReduceLROnPlateau callback
```

```
Epoch 1/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - loss: 0.4440 - val_loss: 0.2502 - learning_rate: 0.0010
Epoch 2/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.2436 - val_loss: 0.2274 - learning_rate: 0.0010
Epoch 3/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.2260 - val_loss: 0.2078 - learning_rate: 0.0010
Epoch 4/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.2036 - val_loss: 0.1901 - learning_rate: 0.0010
Epoch 5/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1883 - val_loss: 0.1777 - learning_rate: 0.0010
```

+ Code + Text

[6] 31s

```
Epoch 6/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1771 - val_loss: 0.1708 - learning_rate: 0.0010
Epoch 7/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1712 - val_loss: 0.1674 - learning_rate: 0.0010
Epoch 8/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1679 - val_loss: 0.1654 - learning_rate: 0.0010
Epoch 9/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1665 - val_loss: 0.1641 - learning_rate: 0.0010
Epoch 10/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1652 - val_loss: 0.1629 - learning_rate: 0.0010
Epoch 11/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1640 - val_loss: 0.1623 - learning_rate: 0.0010
Epoch 12/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1634 - val_loss: 0.1615 - learning_rate: 0.0010
Epoch 13/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1627 - val_loss: 0.1610 - learning_rate: 0.0010
Epoch 14/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1622 - val_loss: 0.1607 - learning_rate: 0.0010
Epoch 15/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1618 - val_loss: 0.1603 - learning_rate: 0.0010
Epoch 16/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1616 - val_loss: 0.1596 - learning_rate: 0.0010
Epoch 17/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1608 - val_loss: 0.1590 - learning_rate: 0.0010
Epoch 18/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1601 - val_loss: 0.1587 - learning_rate: 0.0010
```

+ Code + Text

T4 RAM Disk

[6]
```
Epoch 19/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1597 - val_loss: 0.1583 - learning_rate: 0.0010
Epoch 20/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1597 - val_loss: 0.1582 - learning_rate: 0.0010
Epoch 21/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1592 - val_loss: 0.1580 - learning_rate: 0.0010
Epoch 22/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1594 - val_loss: 0.1577 - learning_rate: 0.0010
Epoch 23/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1589 - val_loss: 0.1575 - learning_rate: 0.0010
Epoch 24/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1591 - val_loss: 0.1574 - learning_rate: 0.0010
Epoch 25/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1586 - val_loss: 0.1572 - learning_rate: 0.0010
Epoch 26/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1589 - val_loss: 0.1571 - learning_rate: 0.0010
Epoch 27/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1581 - val_loss: 0.1571 - learning_rate: 0.0010
Epoch 28/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1579 - val_loss: 0.1570 - learning_rate: 0.0010
Epoch 29/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1580 - val_loss: 0.1569 - learning_rate: 0.0010
Epoch 30/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1581 - val_loss: 0.1568 - learning_rate: 0.0010
<keras.src.callbacks.history.History at 0x7d494cd07d90>
```

+ Code + Text

T4

[7]
```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)

# 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.
```

+ Code + Text

```
[7]  # 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

```
[7]  # 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: 0.4149
Epoch 1: val_loss improved from inf to 0.23882, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step - loss: 0.4145 - val_loss: 0.2388 - learning_rate: 0.0010
Epoch 2/30
233/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.2269
Epoch 2: val_loss improved from 0.23882 to 0.19598, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.2268 - val_loss: 0.1960 - learning_rate: 0.0010
Epoch 3/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1930
Epoch 3: val_loss improved from 0.19598 to 0.18051, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1930 - val_loss: 0.1805 - learning_rate: 0.0010
Epoch 4/30
229/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1794
```

```
+ Code  + Text

[7]  Epoch 4: val_loss improved from 0.18051 to 0.17072, saving model to autoencoder_best.keras
235s 235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1793 - val_loss: 0.1707 - learning_rate: 0.0010
     Epoch 5/30
     212/235 ━━━━━━━━━━━━━━━━━━── 0s 2ms/step - loss: 0.1712
     Epoch 5: val_loss improved from 0.17072 to 0.16554, saving model to autoencoder_best.keras
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1710 - val_loss: 0.1655 - learning_rate: 0.0010
     Epoch 6/30
     218/235 ━━━━━━━━━━━━━━━━━━── 0s 2ms/step - loss: 0.1660
     Epoch 6: val_loss improved from 0.16554 to 0.16209, saving model to autoencoder_best.keras
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1659 - val_loss: 0.1621 - learning_rate: 0.0010
     Epoch 7/30
     227/235 ━━━━━━━━━━━━━━━━━━━─ 0s 2ms/step - loss: 0.1625
     Epoch 7: val_loss improved from 0.16209 to 0.15862, saving model to autoencoder_best.keras
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1625 - val_loss: 0.1586 - learning_rate: 0.0010
     Epoch 8/30
     223/235 ━━━━━━━━━━━━━━━━━━── 0s 2ms/step - loss: 0.1596
     Epoch 8: val_loss improved from 0.15862 to 0.15637, saving model to autoencoder_best.keras
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1596 - val_loss: 0.1564 - learning_rate: 0.0010
     Epoch 9/30
     216/235 ━━━━━━━━━━━━━━━━━━── 0s 2ms/step - loss: 0.1578
     Epoch 9: val_loss improved from 0.15637 to 0.15509, saving model to autoencoder_best.keras
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1578 - val_loss: 0.1551 - learning_rate: 0.0010
     Epoch 10/30
     225/235 ━━━━━━━━━━━━━━━━━━━─ 0s 2ms/step - loss: 0.1561
     Epoch 10: val_loss improved from 0.15509 to 0.15389, saving model to autoencoder_best.keras
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1561 - val_loss: 0.1539 - learning_rate: 0.0010
```

```
+ Code  + Text

[7]  Epoch 11/30
     219/235 ━━━━━━━━━━━━━━━━━━── 0s 2ms/step - loss: 0.1549
     Epoch 11: val_loss improved from 0.15389 to 0.15318, saving model to autoencoder_best.keras
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1549 - val_loss: 0.1532 - learning_rate: 0.0010
     Epoch 12/30
     218/235 ━━━━━━━━━━━━━━━━━━── 0s 2ms/step - loss: 0.1545
     Epoch 12: val_loss improved from 0.15318 to 0.15259, saving model to autoencoder_best.keras
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1545 - val_loss: 0.1526 - learning_rate: 0.0010
     Epoch 13/30
     227/235 ━━━━━━━━━━━━━━━━━━━─ 0s 2ms/step - loss: 0.1533
     Epoch 13: val_loss improved from 0.15259 to 0.15075, saving model to autoencoder_best.keras
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1533 - val_loss: 0.1508 - learning_rate: 0.0010
     Epoch 14/30
     224/235 ━━━━━━━━━━━━━━━━━━━─ 0s 3ms/step - loss: 0.1519
     Epoch 14: val_loss improved from 0.15075 to 0.15007, saving model to autoencoder_best.keras
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1519 - val_loss: 0.1501 - learning_rate: 0.0010
     Epoch 15/30
     235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1513
     Epoch 15: val_loss improved from 0.15007 to 0.14924, saving model to autoencoder_best.keras
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1513 - val_loss: 0.1492 - learning_rate: 0.0010
     Epoch 16/30
     221/235 ━━━━━━━━━━━━━━━━━━── 0s 2ms/step - loss: 0.1505
     Epoch 16: val_loss improved from 0.14924 to 0.14890, saving model to autoencoder_best.keras
     235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1505 - val_loss: 0.1489 - learning_rate: 0.0010
     Epoch 17/30
     234/235 ━━━━━━━━━━━━━━━━━━━─ 0s 2ms/step - loss: 0.1500
```

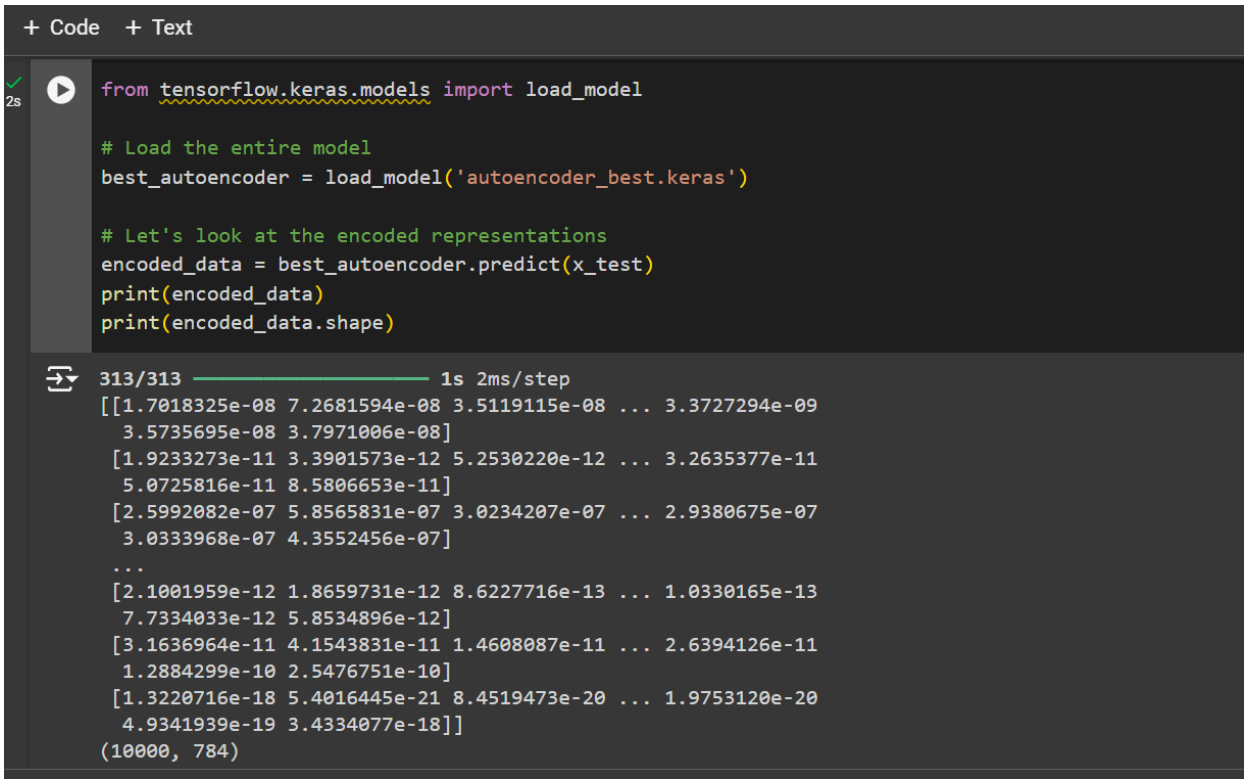
```
+ Code + Text

✓ 35s [7] Epoch 17: val_loss improved from 0.14890 to 0.14831, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1500 - val_loss: 0.1483 - learning_rate: 0.0010
Epoch 18/30
221/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1497
Epoch 18: val_loss improved from 0.14831 to 0.14787, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1497 - val_loss: 0.1479 - learning_rate: 0.0010
Epoch 19/30
213/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1490
Epoch 19: val_loss improved from 0.14787 to 0.14753, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1490 - val_loss: 0.1475 - learning_rate: 0.0010
Epoch 20/30
231/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1490
Epoch 20: val_loss improved from 0.14753 to 0.14727, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1490 - val_loss: 0.1473 - learning_rate: 0.0010
Epoch 21/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1482
Epoch 21: val_loss improved from 0.14727 to 0.14692, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1482 - val_loss: 0.1469 - learning_rate: 0.0010
Epoch 22/30
223/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1485
Epoch 22: val_loss improved from 0.14692 to 0.14667, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1485 - val_loss: 0.1467 - learning_rate: 0.0010
Epoch 23/30
219/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1483
Epoch 23: val_loss improved from 0.14667 to 0.14637, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1483 - val_loss: 0.1464 - learning_rate: 0.0010
```

```
+ Code + Text

✓ 35s [7] Epoch 24/30
220/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1477
Epoch 24: val_loss improved from 0.14637 to 0.14622, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1477 - val_loss: 0.1462 - learning_rate: 0.0010
Epoch 25/30
211/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1475
Epoch 25: val_loss improved from 0.14622 to 0.14576, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1474 - val_loss: 0.1458 - learning_rate: 0.0010
Epoch 26/30
222/235 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.1472
Epoch 26: val_loss improved from 0.14576 to 0.14561, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.1472 - val_loss: 0.1456 - learning_rate: 0.0010
Epoch 27/30
230/235 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.1469
Epoch 27: val_loss improved from 0.14561 to 0.14514, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.1469 - val_loss: 0.1451 - learning_rate: 0.0010
Epoch 28/30
226/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1465
Epoch 28: val_loss improved from 0.14514 to 0.14481, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1465 - val_loss: 0.1448 - learning_rate: 0.0010
Epoch 29/30
232/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1462
Epoch 29: val_loss improved from 0.14481 to 0.14464, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1462 - val_loss: 0.1446 - learning_rate: 0.0010
Epoch 30/30
227/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1460
```

```
Epoch 30: val_loss improved from 0.14464 to 0.14433, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1460 - val_loss: 0.1443 - learning_rate: 0.0010
<keras.src.callbacks.history.History at 0x7d494c99d000>
```



```
+ Code + Text

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
[[1.7018325e-08 7.2681594e-08 3.5119115e-08 ... 3.3727294e-09
  3.5735695e-08 3.7971006e-08]
 [1.9233273e-11 3.3901573e-12 5.2530220e-12 ... 3.2635377e-11
  5.0725816e-11 8.5806653e-11]
 [2.5992082e-07 5.8565831e-07 3.0234207e-07 ... 2.9380675e-07
  3.0333968e-07 4.3552456e-07]
 ...
 [2.1001959e-12 1.8659731e-12 8.6227716e-13 ... 1.0330165e-13
  7.7334033e-12 5.8534896e-12]
 [3.1636964e-11 4.1543831e-11 1.4608087e-11 ... 2.6394126e-11
  1.2884299e-10 2.5476751e-10]
 [1.3220716e-18 5.4016445e-21 8.4519473e-20 ... 1.9753120e-20
  4.9341939e-19 3.4334077e-18]]
(10000, 784)
```

My Github Repository Link:- https://github.com/niharika0912/BDA.git