ICP6 Report:

ICP6 MAK.ipynb

File Edit View Insert Runtime Tools Help All changes saved

\+ Code + Text

[1]

```python
# 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')

# 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
```

ICP6 MAK.ipynb

File Edit View Insert Runtime Tools Help All changes saved

\+ Code + Text

[1]

```
Epoch 94/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1363 - val_loss: 0.1347
Epoch 95/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1367 - val_loss: 0.1346
Epoch 96/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1364 - val_loss: 0.1347
Epoch 97/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1362 - val_loss: 0.1346
Epoch 98/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1364 - val_loss: 0.1346
Epoch 99/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1360 - val_loss: 0.1347
Epoch 100/100
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1361 - val_loss: 0.1344
<keras.src.callbacks.history.History at 0x7b755e6323e0>
```

ICP6 MAK.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[2]

```python
# 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
```

ICP6 MAK.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[2]

```
Epoch 22/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1491 - val_loss: 0.1477
Epoch 23/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1491 - val_loss: 0.1474
Epoch 24/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1486 - val_loss: 0.1472
Epoch 25/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1484 - val_loss: 0.1470
Epoch 26/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1482 - val_loss: 0.1468
Epoch 27/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1481 - val_loss: 0.1466
Epoch 28/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1478 - val_loss: 0.1464
Epoch 29/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1476 - val_loss: 0.1463
Epoch 30/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1475 - val_loss: 0.1462
<keras.src.callbacks.history.History at 0x7b755b5a5150>
```

ICP6 MAK.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[3]

```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
```

ICP6 MAK.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[3]

```python
# 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,  # Number of epochs
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test),  # Validation data
                callbacks=[checkpoint])  # Add the ModelCheckpoint callback
```

ICP6 MAK.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[3] 30s

```
Epoch 26: val_loss improved from 0.14696 to 0.14650, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1485 - val_loss: 0.1465
Epoch 27/30
220/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1481
Epoch 27: val_loss improved from 0.14650 to 0.14605, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 0.1481 - val_loss: 0.1460
Epoch 28/30
224/235 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.1474
Epoch 28: val_loss improved from 0.14605 to 0.14552, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1474 - val_loss: 0.1455
Epoch 29/30
224/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1470
Epoch 29: val_loss improved from 0.14552 to 0.14507, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1470 - val_loss: 0.1451
Epoch 30/30
231/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1465
Epoch 30: val_loss improved from 0.14507 to 0.14465, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1465 - val_loss: 0.1447
<keras.src.callbacks.history.History at 0x7b755b4592d0>
```

ICP6 MAK.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

27s

```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.

# 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
```

ICP6 MAK.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[4] 27s

```python
# 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,  # Number of epochs
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test),  # Validation data
                callbacks=[reduce_lr])  # Add the ReduceLROnPlateau callback
```

ICP6 MAK.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

27s

```python
# 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,  # Number of epochs
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test),  # Validation data
                callbacks=[reduce_lr])  # Add the ReduceLROnPlateau callback
```

ICP6 MAK.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
Epoch 23/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.1419 - val_loss: 0.1400 - learning_rate: 0.0010
Epoch 24/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1412 - val_loss: 0.1396 - learning_rate: 0.0010
Epoch 25/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1408 - val_loss: 0.1394 - learning_rate: 0.0010
Epoch 26/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1406 - val_loss: 0.1388 - learning_rate: 0.0010
Epoch 27/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1401 - val_loss: 0.1386 - learning_rate: 0.0010
Epoch 28/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1400 - val_loss: 0.1382 - learning_rate: 0.0010
Epoch 29/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1394 - val_loss: 0.1379 - learning_rate: 0.0010
Epoch 30/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1394 - val_loss: 0.1375 - learning_rate: 0.0010
<keras.src.callbacks.history.History at 0x7b7559940940>
```

ICP6 MAK.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[5]

```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.

# 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
```

ICP6 MAK.ipynb

```
223/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1512
Epoch 26: val_loss improved from 0.15077 to 0.14976, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1512 - val_loss: 0.1498 - learning_rate: 0.0010
Epoch 27/30
211/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1507
Epoch 27: val_loss improved from 0.14976 to 0.14880, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1507 - val_loss: 0.1488 - learning_rate: 0.0010
Epoch 28/30
233/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1501
Epoch 28: val_loss improved from 0.14880 to 0.14808, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1501 - val_loss: 0.1481 - learning_rate: 0.0010
Epoch 29/30
235/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1495
Epoch 29: val_loss improved from 0.14808 to 0.14748, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 0.1495 - val_loss: 0.1475 - learning_rate: 0.0010
Epoch 30/30
225/235 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1489
Epoch 30: val_loss improved from 0.14748 to 0.14719, saving model to autoencoder_best.keras
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.1489 - val_loss: 0.1472 - learning_rate: 0.0010
<keras.src.callbacks.history.History at 0x7b755b119660>
```

ICP6 MAK.ipynb

```
[6] 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
[[2.0163922e-10 2.3680084e-11 1.8523325e-11 ... 3.0008762e-10
  1.0986750e-11 7.8811038e-12]
 [2.4881977e-14 4.5391537e-16 8.5657681e-15 ... 1.8063024e-15
  1.9584410e-13 4.0643737e-15]
 [1.3492281e-10 8.9158497e-10 8.2469781e-10 ... 4.5694579e-10
  5.5068056e-10 3.1767151e-11]
 ...
 [5.5762673e-18 2.6038686e-20 3.9066058e-19 ... 6.3080664e-19
  3.6704483e-19 1.7425277e-21]
 [3.4576025e-12 2.2963966e-14 2.4052919e-13 ... 1.8486422e-13
  1.2937895e-12 7.1462211e-14]
 [2.3631704e-23 1.4255839e-27 5.5586620e-25 ... 1.0181905e-25
  1.6021953e-22 1.4855783e-25]]
(10000, 784)
```

MY Github link: https://github.com/amanmushnam/BDA.git