1)Feed Forward

import tensorflow as tf

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense, Flatten

from tensorflow.keras.datasets import mnist

from tensorflow.keras.utils import to\_categorical

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

x\_train = x\_train.astype('float32') / 255.0

x\_test = x\_test.astype('float32') / 255.0

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

model = Sequential([

Flatten(input\_shape=(28, 28)),

Dense(128, activation='relu'),

Dense(64, activation='relu'),

Dense(10, activation='softmax')

])

model.compile(

optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy']

)

model.fit(

x\_train, y\_train,

epochs=10,

batch\_size=32,

validation\_data=(x\_test, y\_test)

)

loss, accuracy = model.evaluate(x\_test, y\_test)

print(f"Test Loss: {loss:.4f}")

print(f"Test Accuracy: {accuracy:.4f}")

2)Multiclass Classification Using Deep Neural Networks

import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Flatten

from tensorflow.keras.datasets import mnist

from sklearn.preprocessing import LabelEncoder

from tensorflow.keras.utils import to\_categorical

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

X\_train, X\_test = X\_train / 255.0, X\_test / 255.0

X\_train = X\_train.reshape(-1, 28\*28)

X\_test = X\_test.reshape(-1, 28\*28)

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

model = Sequential([

Dense(128, activation='relu', input\_shape=(28\*28,)),

Dense(64, activation='relu'),

Dense(10, activation='softmax')

])

model.compile(optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=10, batch\_size=32, validation\_split=0.2)

test\_loss, test\_accuracy = model.evaluate(X\_test, y\_test)

print(f"Test Accuracy: {test\_accuracy\*100:.2f}%")

3)Binary Classification Using Deep Neural Networks

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Embedding, GlobalAveragePooling1D

from tensorflow.keras.datasets import imdb

from tensorflow.keras.preprocessing.sequence import pad\_sequences

max\_words = 10000

max\_len = 200

(X\_train, y\_train), (X\_test, y\_test) = imdb.load\_data(num\_words=max\_words)

X\_train = pad\_sequences(X\_train, maxlen=max\_len)

X\_test = pad\_sequences(X\_test, maxlen=max\_len)

model = Sequential([

Embedding(input\_dim=max\_words, output\_dim=128, input\_length=max\_len),

GlobalAveragePooling1D(),

Dense(64, activation='relu'),

Dense(1, activation='sigmoid')

])

model.compile(optimizer='adam',

loss='binary\_crossentropy',

metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=5, batch\_size=64, validation\_data=(X\_test, y\_test))

test\_loss, test\_accuracy = model.evaluate(X\_test, y\_test)

print(f"Test Accuracy: {test\_accuracy\*100:.2f}%")

4)Digit Recognition Using Convolutional Neural Networks

import tensorflow as tf

from tensorflow.keras import layers, models

from tensorflow.keras.datasets import mnist

from tensorflow.keras.utils import to\_categorical

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

X\_train = X\_train / 255.0

X\_test = X\_test / 255.0

X\_train = X\_train.reshape(-1, 28, 28, 1)

X\_test = X\_test.reshape(-1, 28, 28, 1)

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

model = models.Sequential([

layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, (3, 3), activation='relu'),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, (3, 3), activation='relu'),

layers.Flatten(),

layers.Dense(64, activation='relu'),

layers.Dense(10, activation='softmax')

])

model.compile(optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=5, batch\_size=64, validation\_data=(X\_test, y\_test))

test\_loss, test\_accuracy = model.evaluate(X\_test, y\_test)

print(f"Test Accuracy: {test\_accuracy\*100:.2f}%")

5) RNN Sentiment Analysis

import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Embedding, GlobalAveragePooling1D

from tensorflow.keras.datasets import imdb

from tensorflow.keras.preprocessing.sequence import pad\_sequences

max\_words = 10000

max\_len = 200

(X\_train, y\_train), (X\_test, y\_test) = imdb.load\_data(num\_words=max\_words)

X\_train = pad\_sequences(X\_train, maxlen=max\_len)

X\_test = pad\_sequences(X\_test, maxlen=max\_len)

model = Sequential([

Embedding(input\_dim=max\_words, output\_dim=128, input\_length=max\_len),

GlobalAveragePooling1D(),

Dense(64, activation='relu'),

Dense(1, activation='sigmoid')

])

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

print("Training the model...")

model.fit(X\_train, y\_train, epochs=10, batch\_size=64, validation\_data=(X\_test, y\_test))

print("\nEvaluating the model...")

test\_loss, test\_accuracy = model.evaluate(X\_test, y\_test)

print(f"\nTest Accuracy: {test\_accuracy \* 100:.2f}%")

word\_index = imdb.get\_word\_index()

reverse\_word\_index = {value: key for key, value in word\_index.items()}

def decode\_review(encoded\_review):

return " ".join([reverse\_word\_index.get(i - 3, "?") for i in encoded\_review])

print("\nPredicting sentiments for sample reviews...")

predictions = model.predict(X\_test[:5])

for i, pred in enumerate(predictions):

decoded\_review = decode\_review(X\_test[i])

sentiment = "Positive" if pred >= 0.5 else "Negative"

actual\_sentiment = "Positive" if y\_test[i] == 1 else "Negative"

print(f"\nReview {i+1}:")

print(f"Text: {decoded\_review}")

print(f"Predicted Sentiment: {sentiment}, Actual Label: {actual\_sentiment}")

6) VGG/Resnet

import tensorflow as tf

from tensorflow.keras.applications import VGG16

from tensorflow.keras import layers, models

from tensorflow.keras.datasets import cifar10

from tensorflow.keras.utils import to\_categorical

(x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data()

x\_train, x\_test = x\_train / 255.0, x\_test / 255.0

y\_train, y\_test = to\_categorical(y\_train, 10), to\_categorical(y\_test, 10)

base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=(32, 32, 3))

base\_model.trainable = False

model = models.Sequential([

base\_model,

layers.Flatten(),

layers.Dense(256, activation='relu'),

layers.Dense(10, activation='softmax')

])

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

model.fit(x\_train, y\_train, validation\_data=(x\_test, y\_test), epochs=5, batch\_size=64)

loss, acc = model.evaluate(x\_test, y\_test)

print(f"Test Accuracy: {acc:.2f}")

7) LSTM

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

data = np.sin(np.arange(100) \* 0.1)

X, y = [], []

for i in range(len(data) - 10):

X.append(data[i:i+10])

y.append(data[i+10])

X, y = np.array(X), np.array(y)

X = X.reshape(-1, 10, 1)

model = tf.keras.Sequential([

tf.keras.layers.LSTM(20, input\_shape=(10, 1)),

tf.keras.layers.Dense(1)

])

model.compile(optimizer="adam", loss="mse")

model.fit(X, y, epochs=10, verbose=0)

# Predict next 10 values

preds = []

seq = X[-1].flatten()

for \_ in range(10):

pred = model.predict(seq.reshape(1, 10, 1), verbose=0)[0, 0]

preds.append(pred)

seq = np.roll(seq, -1)

seq[-1] = pred

plt.plot(data, label="Actual")

plt.plot(range(90, 100), preds, 'ro-', label="Predicted")

plt.legend()

plt.show()