PERIYAR MANIAMMAI
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University)
Established Under Sec. 3 of UGC Act, 1956 • NAAC Accredited

think • innovate • transform

Name :S.Hareni
S.Priyadharshini
A.Oviya

Reg no:123011019013
123011019023
123011019034

Course:Introduction To Artificial Intelligence

Code :XCSHA1

Title :Handwriting Dedication

Introduction:

Handwriting detection, also known as handwriting recognition (HWR) or handwritten text recognition (HTR), is the process of converting handwritten text into a format that a computer can understand. HWR can be used to interpret handwritten text from a variety of sources, including paper documents, photographs, and touch-screens.

HWR is an active area of artificial intelligence research, and its accuracy is constantly improving with advances in machine learning. HWR can be used in a variety of industries, including enterprise, field services, and healthcare.

Here are some ways that HWR works:

Optical scanning: Also known as optical character recognition (OCR), this method senses the image of the written text from a piece of paper.

Intelligent word recognition: This method senses the image of the written text from a piece of paper.

Pen-based computer screen surface: This method senses the movements of the pen tip on the screen

Source code:

```
Import numpy as np

Import matplotlib.pyplot as plt

From tensorflow.keras.datasets import mnist

From tensorflow.keras.models import Sequential

From tensorflow.keras.layers import Dense, Flatten, Dropout

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


Y_train = to_categorical(y_train, 10)
Y_test = to_categorical(y_test, 10)


Model = Sequential()
Model.add(Flatten(input_shape=(28, 28)))
Model.add(Dense(128, activation='relu'))
Model.add(Dropout(0.2))
Model.add(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_split=0.2)
```

Test_loss, test_acc = model.evaluate(x_test, y_test)

Print(f"Test accuracy: {test_acc:.4f}")

Predictions = model.predict(x_test)

Plt.imshow(x_test[0], cmap='gray')

Plt.title(f"Predicted Label: {np.argmax(predictions[0])}")

Plt.show()

Output:

Epoch 1/5

750/750 ━━━━━━━━━━━━━━━━━━━━ 7s 7ms/step – accuracy: 0.8207 – loss: 0.6166 – val_accuracy: 0.9499 – val_loss: 0.1817

Epoch 2/5

750/750 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step – accuracy: 0.9435 – loss: 0.1918 – val_accuracy: 0.9616 – val_loss: 0.1332

Epoch 3/5

750/750 ━━━━━━━━━━━━━━━━━━━━ 5s 6ms/step – accuracy: 0.9602 – loss: 0.1373 – val_accuracy: 0.9677 – val_loss: 0.1099

Epoch 4/5

750/750 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step – accuracy: 0.9671 – loss: 0.1129 – val_accuracy: 0.9718 – val_loss: 0.0964

Epoch 5/5

750/750 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step – accuracy: 0.9717 – loss: 0.0961 – val_accuracy: 0.9721 – val_loss: 0.0891

313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step – accuracy: 0.9701 – loss: 0.0971

Test accuracy: 0.9743

313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step