S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 9

Aim: Design and implement a neural network architecture to accurately classify handwritten digits, optimize parameters, and evaluate performance metrics.

Name of Student: Shrutika Pradeep Bagdi

Roll No.: CS22130

Semester/Year: VII/IV

Academic Session: 2025-26

Date of Performance:

Date of Submission:

AIM: Design and implement a neural network architecture to accurately classify handwritten digits, optimize parameters, and evaluate performance metrics.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- To be able to explain the structure and function of neural networks.
- To be able to select appropriate activation functions and loss functions for handwritten digit classification.
- To be able to implement different neural network architectures, such as multilayer perceptrons (MLPs) and convolutional neural networks (CNNs), for handwritten digit classification.

THEORY:

The term "Artificial Neural Network" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.The architecture of an artificial neural network:
To understand the concept of the architecture of an artificial neural network, we have to understand what a neural network consists of. To define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Lets us look at various types of layers available in an artificial neural network.
Artificial Neural Network primarily consists of three layers:

Input Layer:

As the name suggests, it accepts inputs in several different formats provided by the programmer.

Hidden Layer:

The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.

Output Layer:

The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

Artificial neural networks (ANNs) have become a powerful tool for computer vision tasks, such as image classification, object detection, and image segmentation. ANNs are inspired by the structure and function of the human brain, and they can learn to extract complex patterns from data.

- **Basic Structure of an ANN**

An ANN consists of a collection of interconnected nodes, called neurons. Each neuron receives inputs from other neurons or from external data sources. The neuron then processes these inputs and produces an output. The outputs of the neurons can be fed as inputs to other neurons, and this process can continue until the final output layer is reached.

- **Learning in ANNs**

ANNs learn by adjusting the weights of the connections between the neurons. These weights represent the strength of the signal that is passed from one neuron to another. The weights are adjusted through a process called backpropagation.
Backpropagation is an algorithm that calculates the error between the network's output and the desired output. The algorithm then propagates this error back through the network, adjusting the weights of the connections as it goes. This process is repeated until the network's output is close to the desired output

- **Types of ANNs for Computer Vision**

There are many different types of ANNs, but some of the most common types used for computer vision tasks include:

1. Feedforward neural networks: These networks have a simple structure in which the information flows in one direction, from the input layer to the output layer.
2. Convolutional neural networks (CNNs): These networks are specifically designed for image processing tasks. They use a special type of layer called a convolution layer, which is able to extract features from images.
3. Recurrent neural networks (RNNs): These networks are able to process sequential data, such as text or time series data. They are often used for tasks such as image captioning and video captioning.

- **Applications of ANNs in Computer Vision**

1. ANNs have been used to achieve state-of-the-art results on a wide variety of computer vision tasks. Some of the most notable applications of ANNs in computer vision include:
2. Image classification: ANNs can be used to classify images into different categories, such as cats, dogs, and cars.
3. Object detection: ANNs can be used to detect objects in images, and to localize them within the image.
4. Image segmentation: ANNs can be used to segment images into different regions, such as the foreground and background.
5. Image captioning: ANNs can be used to generate captions for images.
6. Video captioning: ANNs can be used to generate captions for videos.

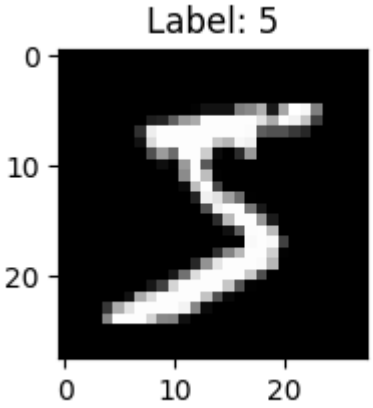
Code:
Here is a step-by-step explanation of how to build an artificial neural network (ANN) for classifying handwritten digits using the MNIST dataset:

Step 1: Import necessary libraries

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
import matplotlib.pyplot as plt
```

```
# Step 2: Load the MNIST dataset
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
print("Training data shape:", x_train.shape)
print("Testing data shape:", x_test.shape)
OUTPUT:
Training data shape: (60000, 28, 28)
Testing data shape: (10000, 28, 28)

# Step 3: Preprocess the data
# Normalize the pixel values from [0, 255] to [0, 1]
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
# Display an example image from the training set
plt.figure(figsize=(2, 2))
plt.imshow(x_train[0], cmap='gray')
plt.title(f"Label: {y_train[0]}")
plt.show()
```




```
# Step 4: Define the ANN model
model = keras.Sequential([
    # Flatten layer to convert the 28x28 image matrix into a 1D vector (784 pixels)
    layers.Flatten(input_shape=(28, 28)),

    # First hidden Dense layer with 512 neurons and a ReLU activation function
    layers.Dense(512, activation='relu'),

    # Second hidden Dense layer (optional, but improves performance)
    layers.Dense(128, activation='relu'),

    # Output layer with 10 neurons (one for each digit) and a Softmax activation
    layers.Dense(10, activation='softmax')
])

# Step 5: Compile the ANN model
model.compile(optimizer='adam',
```

```
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])
model.summary() # Print a summary of the model's architecture
```

OUTPUT:

```
Model: "sequential_2"
```

Layer (type)	Output Shape	Param #
flatten_2 (Flatten)	(None, 784)	0
dense_5 (Dense)	(None, 512)	401,920
dense_6 (Dense)	(None, 128)	65,664
dense_7 (Dense)	(None, 10)	1,290

```
Total params: 468,874 (1.79 MB)
Trainable params: 468,874 (1.79 MB)
Non-trainable params: 0 (0.00 B)
```

```
# Step 6: Train the ANN model
print("\nStarting model training...")
# The 'fit' method trains the model for 10 epochs
model.fit(x_train, y_train, epochs=10, validation_split=0.1)
print("Model training complete.")
```

```
Starting model training...
Epoch 1/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16s 9ms/step - accuracy: 0.8955 -
loss: 0.3403 - val_accuracy: 0.9723 - val_loss: 0.0895
Epoch 2/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20s 9ms/step - accuracy: 0.9744 -
loss: 0.0843 - val_accuracy: 0.9763 - val_loss: 0.0770
Epoch 3/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15s 9ms/step - accuracy: 0.9834 -
loss: 0.0514 - val_accuracy: 0.9812 - val_loss: 0.0677
Epoch 4/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14s 9ms/step - accuracy: 0.9881 -
loss: 0.0377 - val_accuracy: 0.9800 - val_loss: 0.0719
Epoch 5/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14s 8ms/step - accuracy: 0.9907 -
loss: 0.0299 - val_accuracy: 0.9778 - val_loss: 0.0825
Epoch 6/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14s 8ms/step - accuracy: 0.9901 -
loss: 0.0286 - val_accuracy: 0.9822 - val_loss: 0.0795
Epoch 7/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21s 8ms/step - accuracy: 0.9925 -
loss: 0.0203 - val_accuracy: 0.9775 - val_loss: 0.0928
Epoch 8/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20s 8ms/step - accuracy: 0.9932 -
loss: 0.0204 - val_accuracy: 0.9827 - val_loss: 0.0790
Epoch 9/10
```

```
1688/1688 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14s 8ms/step - accuracy: 0.9948 -
loss: 0.0159 - val_accuracy: 0.9822 - val_loss: 0.0847
Epoch 10/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21s 9ms/step - accuracy: 0.9958 -
loss: 0.0124 - val_accuracy: 0.9798 - val_loss: 0.0892
Model training complete.
```

---

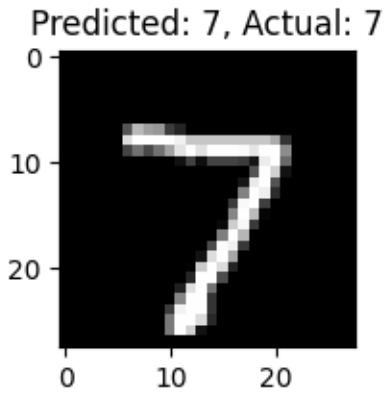
[16]
2s

 # Step 7: Evaluate the ANN model
print("\nEvaluating the model on the test data...")
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f"Test accuracy: {test_acc:.4f}")

```
Evaluating the model on the test data...
313/313 - 1s - 3ms/step - accuracy: 0.9796 - loss: 0.0904
Test accuracy: 0.9796
```

 # Step 8: Make a prediction on a sample image
print("\nMaking a prediction on the first test image...")
sample_image = x_test[0]
sample_label = y_test[0]
prediction = model.predict(np.expand_dims(sample_image, axis=0))
predicted_digit = np.argmax(prediction[0])
print(f"The actual digit is: {sample_label}")
print(f"The model predicted: {predicted_digit}")
 # Plot the sample image and the prediction
plt.figure(figsize=(2, 2))
plt.imshow(sample_image, cmap='gray')
plt.title(f"Predicted: {predicted_digit}, Actual: {sample_label}")
plt.show()

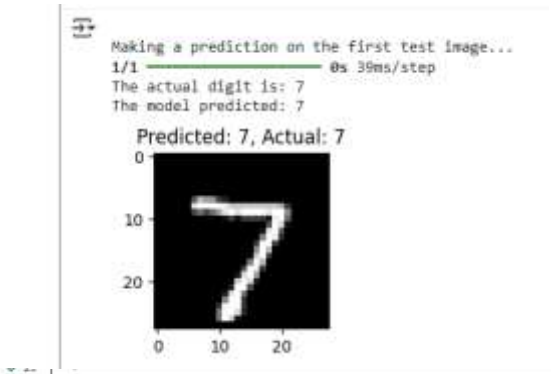
```
Making a prediction on the first test image...
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
The actual digit is: 7
The model predicted: 7
```

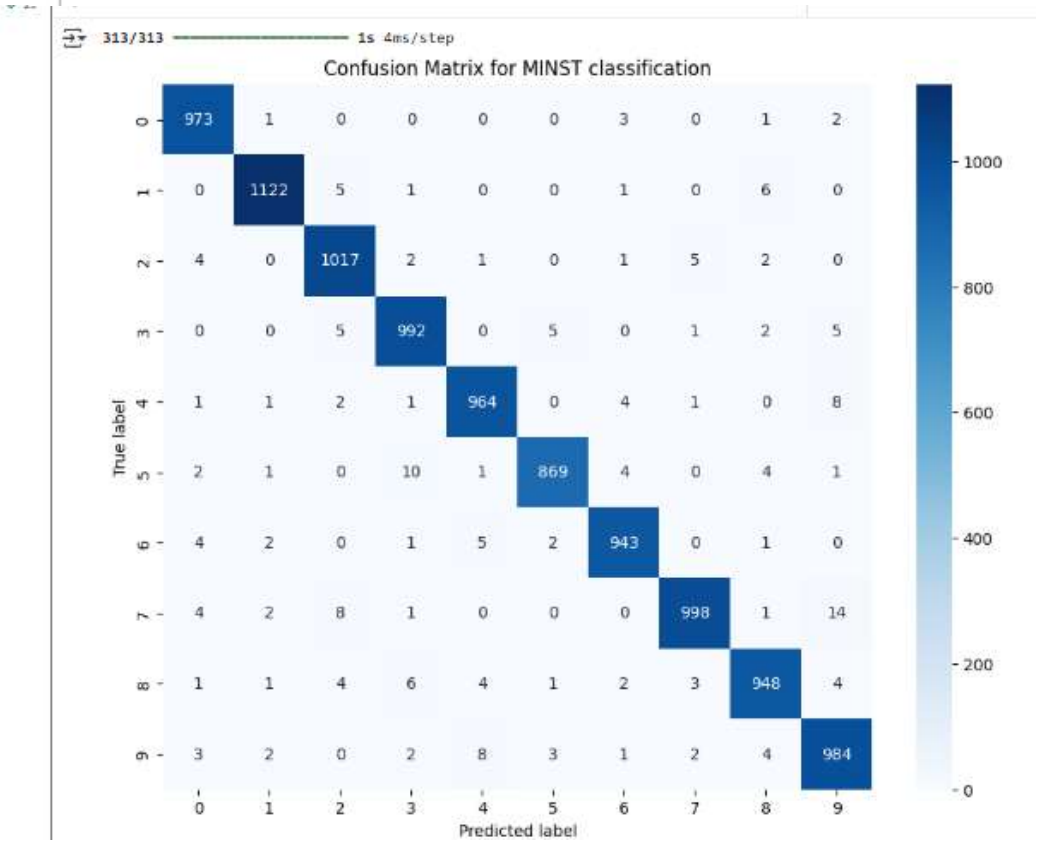


from sklearn.metrics import confusion_matrix, classification_report

```
import seaborn as sns
y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)
#create confusion matrix
cm = confusion_matrix(y_test, y_pred_classes)
#plot confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=range(10),
yticklabels=range(10))
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix for MINST classification')
plt.show()
```

INPUT & OUTPUT:





LINK COLAB->
https://colab.research.google.com/drive/1S8GqDFiPmzIj4T-9_6wSKAQLyFCbZSQq#scrollTo=PIIIpUGKFyLv

CONCLUSION:

DISCUSSION QUESTIONS:

1) What is an artificial neural network (ANN)?
2) Explain the components of your neural network and their functions.
3) How does an ANN learn?
4) What are the different types of ANNs?
5) What are the activation functions used in ANNs?

REFERENCES:

- Computer Vision: Algorithms and Applications, Richard Szeliski, 2010, Springer.
- Computer Vision - A modern Approach, D. Forsyth, J. Ponce, 2nd Edition 2011, Pearson India.
- OpenCV Computer Vision with Python, Joseph Howse, 2013, Packt Publishing.
- Dictionary of Computer Vision and Image Processing, R. B. Fisher, T. P. Breckon, K. Dawson-Howe, A. Fitzgibbon, C. Robertson, E. Trucco, C. K. I. Williams, 2nd Edition, 2014, Wiley.
- OpenCV Computer Vision with Python, Joseph Howse, 2013, Packt Publishing.
- https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network
- https://ssagesproject.github.io/docs/Artificial%20Neural%20Network%20Sampling.html
- https://www.geeksforgeeks.org/digital-image-processing-basics/
- https://www.simplilearn.com/image-processing-article
- https://www.mygreatlearning.com/blog/digital-image-processing-explained/