**ASSIGNMENT HELP**

**MANUAL**



SUBMITTED

TO

VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE

FOR THE SKILL AND COMPETENCY EVALUATION OF

DEEP LEARNING [ CAUA31202]

IN

**CSE AI DEPARTMENT**

BY

**Vedant Rakesh Mukhekar**

**Class: T.Y. BTech Division: A Batch: A2**

**Batch Teacher**

**Dr. ANURADHA YENKIKAR.**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SR. NO.** | **CONTENTS** | **PAGE NO.** |
| **1** | **PROBLEM STATEMENT** | **4-5** |
| **2** | **LIBRARY USED** | **5-6** |
| **3** | **THEORY** | **6-7** |
| **4** | **METHODOLOGY** | **7-14** |
| **5** | **ADVANTAGES & DISADVANTAGES** | **14-15** |
| **6** | **WORKING** | **16-17** |
| **7** | **DIAGRAM** | **17-18** |
| **8** | **CONCLUSION** | **18-19** |

### Problem Statement

The objective of this project is to implement **Feedforward Neural Networks (FNN)** using Python with the Keras library, built on top of TensorFlow. Feedforward Neural Networks are a fundamental type of artificial neural network where connections between nodes do not form cycles. The aim is to demonstrate how to create, train, and evaluate a feedforward neural network model to solve a classification problem using a standard dataset, such as the MNIST dataset of handwritten digits.

### Libraries Used

* **Python Libraries**:
  + Keras: A high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
  + TensorFlow: An open-source machine learning framework that provides a robust set of tools for building and training neural networks.
  + NumPy: A library for numerical operations in Python.
  + Matplotlib: A library for creating static, animated, and interactive visualizations in Python.

### Theory

Feedforward Neural Networks are among the simplest forms of artificial neural networks and consist of an input layer, one or more hidden layers, and an output layer. Each layer consists of nodes (neurons) that process input data and pass the output to the next layer. The network learns to map input features to outputs through a process called training, where weights are adjusted based on the loss (error) calculated from the predicted output versus the actual output.

#### Key Concepts

* **Neurons and Layers**: Each neuron receives input from the previous layer, applies an activation function, and passes its output to the next layer. The layers in the network can be defined as:
  + **Input Layer**: The layer that receives the input data.
  + **Hidden Layers**: Intermediate layers where computation and feature extraction occur.
  + **Output Layer**: The layer that produces the final output (predictions).
* **Activation Functions**: Functions that introduce non-linearity into the network, allowing it to learn complex patterns. Common activation functions include:
  + **ReLU (Rectified Linear Unit)**: f(x)=max⁡(0,x)f(x) = \max(0, x)f(x)=max(0,x)
  + **Sigmoid**: f(x)=11+e−xf(x) = \frac{1}{1 + e^{-x}}f(x)=1+e−x1​
  + **Softmax**: Used in the output layer for multi-class classification, providing probabilities for each class.
* **Loss Function**: A function that measures the difference between the predicted output and the actual output. Common loss functions for classification problems include:
  + **Categorical Crossentropy**: Used for multi-class classification.
  + **Binary Crossentropy**: Used for binary classification.
* **Optimizer**: An algorithm used to update the weights of the network based on the gradients of the loss function. Common optimizers include:
  + **Stochastic Gradient Descent (SGD)**
  + **Adam**
  + **RMSprop**
* **Training Process**: Involves the following steps:
  + **Forward Propagation**: The input data passes through the network to generate predictions.
  + **Loss Calculation**: The loss function computes the error between the predictions and the actual labels.
  + **Backpropagation**: The optimizer adjusts the weights based on the gradients calculated from the loss.
  + **Iteration**: The above steps are repeated for a specified number of epochs or until convergence.

#### Applications of Feedforward Neural Networks

* **Image Classification**: Identifying objects within images.
* **Speech Recognition**: Transcribing spoken words into text.
* **Medical Diagnosis**: Assisting in diagnosing diseases based on patient data.
* **Financial Predictions**: Forecasting stock prices and market trends.

### Methodology

1. **Set Up the Environment**: Install the necessary libraries, including TensorFlow and Keras.
2. **Load the Dataset**: Use a standard dataset such as MNIST, which contains images of handwritten digits and their corresponding labels.
3. **Preprocess the Data**: Normalize the image data and convert labels to a one-hot encoded format.
4. **Build the Feedforward Neural Network Model**:
   * Define the architecture of the model, specifying the number of layers and neurons in each layer.
   * Choose appropriate activation functions, loss function, and optimizer.
5. **Train the Model**: Fit the model to the training data while monitoring performance on validation data.
6. **Evaluate the Model**: Assess the model's performance on a separate test dataset.
7. **Visualize Results**: Plot the training and validation loss/accuracy to analyze model performance.

### Advantages & Disadvantages

* **Advantages**:
  + **Simplicity**: Feedforward Neural Networks are relatively easy to implement and understand.
  + **Effectiveness**: They can learn complex relationships in data and achieve high accuracy in various applications.
  + **Scalability**: Can be scaled by increasing the number of hidden layers or neurons to improve performance.
* **Disadvantages**:
  + **Limited to Fixed Input Size**: Requires a fixed input size, making it less flexible for varying input data.
  + **Overfitting**: May overfit the training data, especially with a large number of parameters and limited data.
  + **Training Time**: Training can be computationally expensive, particularly for large datasets.

### Working Example (Python Code)

Here's a simple implementation of a Feedforward Neural Network using Keras and TensorFlow:

python

Copy 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

from tensorflow.keras.utils import to\_categorical

# Load the MNIST dataset

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

# Normalize the data

x\_train = x\_train.astype('float32') / 255

x\_test = x\_test.astype('float32') / 255

# One-hot encode the labels

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

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

# Build the Feedforward Neural Network model

model = Sequential()

model.add(Flatten(input\_shape=(28, 28))) # Flatten the input

model.add(Dense(128, activation='relu')) # Hidden layer with 128 neurons

model.add(Dense(64, activation='relu')) # Hidden layer with 64 neurons

model.add(Dense(10, activation='softmax')) # Output layer for 10 classes

# Compile the model

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

# Train the model

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

# Evaluate the model

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

print(f'Test accuracy: {test\_accuracy:.4f}')

# Visualize the training history

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Model Accuracy')

plt.ylabel('Accuracy')

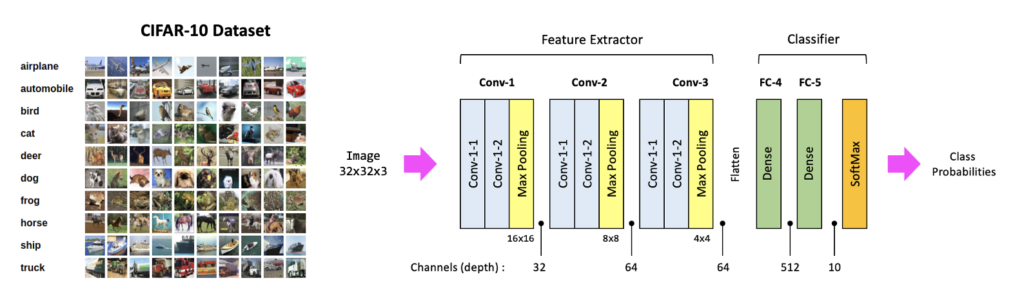
plt.xlabel('Epoch')

plt.legend()

plt.show()

### Diagram

### Understanding Feedforward Neural Networks | LearnOpenCV



### Conclusion

The implementation of a **Feedforward Neural Network** using Keras and TensorFlow demonstrates the fundamental principles of neural network architecture and training. This project successfully applied these concepts to the MNIST dataset for digit classification, achieving a high level of accuracy. The combination of Keras and TensorFlow allows for efficient model building, training, and evaluation, making them powerful tools for developing neural network applications. As the field of deep learning continues to evolve, feedforward neural networks serve as a foundational technology for more complex architectures, enabling advancements in various domains such as image recognition, natural language processing, and medical diagnostics.