Task 1:

MNIST Digit Recognition with Deep Learing

This documentation gives a detailed breakdown of the included Python script for recognizing digits from the MNIST dataset. It contains the code's capabilities, how to install it, how to use it, and other critical information that users need to be aware of.

Bring in the required libraries: These deep learning packages, TensorFlow and Keras, are used to create and train neural networks.

Load the MNIST dataset: keras.datasets is used to load the MNIST dataset.60,000 training images and 10,000 test images of handwritten digits (0-9) with matching labels make up the mnist database.

Normalize pixel values: By dividing the image's pixel values by 255.0, the images' pixel values are normalized to fall between [0, 1]. It lowers the training process's computing expense.

Construct a sequential model: A sequential model is a stack of layers that is linear. Max-pooling layers are inserted after convolutional layers. The data is flattened after the convolutional and pooling layers. With ReLU activation functions, several dense, completely connected layers are added. For multiclass classification, the final dense layer comprises 10 units with softmax activation.

Compile the model: The sparse categorical cross-entropy loss function and Adam optimizer are used to create the Sequential model. Accuracy serves as the evaluation metric.

Restructure the dataset: The input data (Xtrain and Xtest) are restructured to have dimensions (number of samples, height, width, and channels) that correspond to the input shape anticipated by the model (28x28 pixels with one channel).

Train the model: Using the training data (Xtrain, ytrain), the model is trained. It operated with a batch size of 1000 throughout 20 epochs.

Evaluate the model: Using test data, the model is assessed, and test loss and test accuracy are computed.

Print the evaluation report: To assess how well the model works with obfuscated data, the script produces the test accuracy and test loss.

Installation and Execution

Follow these installation instructions to utilize this code:

1. Installing Python: Make sure Python is installed on your system by following this first step. If not, get it from the official Python website and install it: https://www.python.org/downloads/

2. Installing TensorFlow and Keras: *pip install tensorflow keras*

3. Installation of Matplotlib (for visualization): *pip install matplotlib*

4. Launch a command prompt or terminal.

5. Go to the directory where the Python script was saved.

6. To execute the script, use the following command: Task_1.py in Python

7. The code will preprocess the data after loading the MNIST dataset.

8. Using the training set of data, it will define, construct, and train a convolutional neural network (CNN) model.

9. It will be able to track progress throughout training by viewing updates for each epoch that include the training dataset's loss and accuracy.

10. The script will evaluate the model on the test dataset after training is complete and print the test accuracy and loss.

Task 2:

Task Manager Application

Users can conduct CRUD (Create, Read, Update, Delete) actions on tasks saved in a PostgreSQL database using this Python script, which creates a simple task manager application. The code's capabilities, installation, usage, and other crucial information are all covered in full in this manual.

Database Interaction: Users can store and manage tasks by connecting to a PostgreSQL database with this application.

The following CRUD actions can be executed by users on tasks:
Create: Add a new task.
Read: View all the created tasks.
Update: Change task descriptions and the state of completion.
Delete: Delete task.

User Interface: The script provides an interactive command-line interface that is user-friendly.

Error Handling: The code provides error handling to address probable problems like inaccurate user inputs or database connection issues.

Installation and Execution

Follow these steps to utilize this code:

1. Installing Python with psycopg2: Make sure Python is set up on your computer. If not, go to https://www.python.org/downloads/ on the official Python website to download and install it. Run the following command to install the psycopg2 library for PostgreSQL database interaction:
pip install psycopg2

2. Database Setup: If PostgreSQL isn't already installed on your computer, do so now. https://www.postgresql.org/download/ PostgreSQL downloads. Create the "task-manager" PostgreSQL database and create a user with the credentials "postgres" and "123456." The script allows you to change these credentials as necessary.

3. Launch a command prompt or terminal.

4. Go to the directory where the Python script was saved.

5. To execute the script, use the following command: Task_2.py in Python

6. The script will show a menu with the following choices:

1. Include a new task.
2. Finish all jobs
3. Make a task update
4. Remove a task
5. Quit

7. Select a choice by entering in the matching number and press Enter.

8. Depending on the option you choose, follow the instructions to add, view, change, or delete tasks.

9. Select option 5 to end the program.

Notes: Verify that the PostgreSQL database is operating and that the script's database connection information (host, database name, username, and password) corresponds to your database configuration.

Database table: The script expected a "tasks" database with the attributes "id" (auto-incremented), "task" (task description), and "completed" (boolean) as specified. The code includes the table creation query, however it is currently commented out. If you need to build the table, remove the comment.

Task 3

Find latitude and longitude with Google API

If entered a address as input, geocoding initiates a request to the Google Maps Geocoding API to get the address' latitude and longitude. It deals with potential issues that could arise during API requests and responses, such as situations in which the specified address might not be discovered. The script makes it simple to use by asking the user to provide an address.

Installation and Execution

You must have Python installed on your system in order to use this code. Additionally, you will require a Google Maps Platform API key. To install, take these actions:

1. Python Installation: Download and install Python from the official Python website at https://www.python.org/downloads/

2. Google Maps API Key: From the https://console.cloud.google.com, you must receive an API key. Make sure your project has the Google Maps Geocoding API enabled.

3. Code setup: Substitute your actual Google Maps API key for the api_key variable.

4. Launch a command prompt or terminal.

5. Go to the directory where the Python script was saved.

6. To execute the script, use the following command: Task_3.py in Python

7. You will be prompted by the script to input an address. Enter the address once you've typed it.

8. If the specified address is found, the script will then make a request to the Google Maps Geocoding API and display the latitude and longitude of the location.

9. The script will output an error message if the address cannot be located or if there are any problems with the request.

Note: Make sure your Google Maps API key is active and the Geocoding API is enabled. The script won't run properly without a key.

By following these instructions, you can easily retrieve the latitude and longitude of addresses using the provided Python script and your Google Maps API key.