Task 1

Overview

This module provides functionality to train and evaluate handwritten image data from MNIST dataset. This also offers graphs for training evaluation.

Dependencies

- `numpy`: A fundamental package for scientific computing in Python.
- `tensorflow`: An end-to-end open-source platform for machine learning.
- `matplotlib`: A collection of functions that make matplotlib work like MATLAB, used for creating static, interactive, and animated visualizations in Python.
- `seaborn`: A Python data visualization library based on matplotlib, providing a high-level interface for drawing attractive and informative statistical graphics.
- `scikit-learn`: Tools for performance evaluation metrics in machine learning from the scikit-learn library.

Functions

`set_seed(seed)`

- **Purpose**: Sets the seed for random number generation to ensure reproducible results in experiments.
- **Parameters**:
  - `seed`: A int object for the seed.

`normalize(data)`

- **Purpose**: Normalizes the pixel values by dividing with 255.
- **Parameters**:
  - `data`: A numpy array of image data.
- **Returns**: A normalized numpy array of image data.

`Get_neural_network()`

- **Purpose**: Creates an artificial neural network.
- **Returns**: A keras model.

`Plot_accuracy(history)`

- **Purpose**: Plots the training and validation accuracy graph per 3 epochs.
- **Parameters**:
  - `history`: A keras history object.

Task 2

Overview

This module provides functionalities to interact with SQLite databases using Python. It contains functions to create a connection to a database and to create tables within it.

Dependencies

- `sqlite3`: A built-in Python library that provides an interface to the SQLite database.

Functions

`create_connection(db_file)`

- **Purpose**: Establishes a connection to the SQLite database specified by the `db_file` parameter.
- **Parameters**:
  - `db_file`: A string representing the path to the database file.
- **Returns**: A connection object to the SQLite database.
- **Exceptions**: Prints an error message if a connection to the database cannot be established.

`create_table(conn, create_table_sql)`

- **Purpose**: Creates a table using the `create_table_sql` statement passed to the function.
- **Parameters**:
  - `conn`: A connection object to the database where the table will be created.
  - `create_table_sql`: A string containing the SQL statement for creating a table.
- **Exceptions**: If the table cannot be created, the function does not explicitly handle exceptions but will fail silently since the exception is not caught or printed.

`create_entry(conn, entry)`

- **Purpose**: Creates a entry using the `entry` in the solution table.
- **Parameters**:
  - `conn`: A connection object to the database.
  - `entry`: A list or tuple containing the values for creating a entry.

`select_all_entries(conn, table)`

- **Purpose**: Returns a table .
- **Parameters**:
  - `conn`: A connection object to the database.
  - `table`: A string containing the table name.

`update_entry(conn, data)`

- **Purpose**: Updates a table row from id.
- **Parameters**:
  - `conn`: A connection object to the database.
  - `data`: A string containing the new values with id.

`delete_entry(conn, id)`

- **Purpose**: Deletes a table row from id.
- **Parameters**:
  - `conn`: A connection object to the database.
  - `id`: A integer containing index value.

Conclusion

The provided Python script is a basic example to demonstrate how to create a simple sqlite database. It also provides basic functions to add, update or retrieve data from a database.

Task 3

This documentation covers the Python script designed to interact with the Google Sheets API. The script enables users to read data from a Google Sheet.

Features

- Retrieve data from a specified column in https://docs.google.com/spreadsheets/d/1HSJarrTevcqeSblr61I_jv1Z14PAAK0yjdSwhFYhyMA/edit?usp=sharing
- Plot a retrieved data in a window.

Prerequisites

Google Cloud Platform Project

You must have a Google Cloud Platform project with the Google Sheets API enabled. To enable the API and create credentials:

1. Visit the **Google Cloud Console**.
2. Select or create a new project.
3. Navigate to "APIs & Services" > "Dashboard" and click "ENABLE APIS AND SERVICES" to search for the Google Sheets API and enable it.
4. Go to "Credentials" and click on "Create credentials". Choose "OAuth client ID" and set up your consent screen if prompted.
5. Once the OAuth client ID is created, download the JSON file containing your credentials. Rename this file to ***credentials.json*** and place it in your working directory.

Python Environment

Make sure Python 3.x is installed on your system. You can download Python from the **official site**.

Install Dependencies

Install the required libraries using pip:

```
pip install --upgrade google-api-python-client google-auth-httplib2 google-auth-oauthlib
```

Usage

Script Setup

Store your ***credentials.json*** file in the same directory as your script.

Running the Script

Execute the script in your Python environment. On the first run, it will prompt you to authorize access to your Google Sheets. Follow the instructions in the authentication flow.

Refresh Token

The script uses a file named ***`token.json`*** to store access and refresh tokens. This means you won't need to go through the authorization process on subsequent script executions, as long as the token remains valid.

Functions

`plot_bar_graph(valuedict)`

This function plots a bar graph.

Parameters:

- valuedict: Dictionary frequency of items

`main()`

This function handles Google API intrigation, reads data from the specified ***`spreadsheet_id`*** and. It stores the retrieved data and plots a bar graph in a window.

Limitations

- The script is read-only and will not modify data in Google Sheets.
- It requires ***`token.json`*** for storing OAuth tokens, which may need refreshing if the access token expires or is revoked.

Error Handling

- The script prints an error message if the connection to the database fails or if no data is found in the specified range.

Security Considerations

- Keep ***`credentials.json`*** and ***`token.json`*** secure and do not expose them in publicly accessible areas.
- Follow Google's best practices for OAuth to ensure your application's security.

Conclusion

The provided Python script is a basic example to demonstrate how to connect to and read data from the Google Sheets API. For more advanced operations like writing data or managing spreadsheets, you will need to modify the scopes and add appropriate functions. Always refer to the **Google Sheets API documentation** for detailed information on available features and capabilities.