AI-Based Detection of Planetary Nebulae

Bibal Sobeaux Pierre Gabriel

08/01/2024

1 Introduction

This document provides an overview of an AI project aimed at detecting planetary nebulae. Utilizing machine learning and image processing techniques, this project automates the identification and analysis of astronomical images to recognize planetary nebulae.

2 Project Overview

The project employs a series of Python scripts to scrape data, preprocess images, train a neural network model, and evaluate its performance. The objective is to classify images based on the presence of planetary nebulae.

3 File Descriptions

3.1 data_retrieval.py

3.1.1 Purpose:

This script automates the collection of planetary nebula data from specified web sources, structuring it in a CSV format for subsequent analysis. It is pivotal for building the dataset that the AI model will use for training and validation.

3.1.2 Data Structure:

The script extracts the following fields for each planetary nebula:

- **Name:** The designation of the nebula (e.g., Ou 1, Ou 2).
- **Galactic Coordinates (Gal. Coord.):** Galactic longitude and latitude of the nebula.
- **Right Ascension (RA) and Declination (DEC):** Celestial coordinates specifying the nebula's location in the sky.
- **Size:** Apparent size of the nebula in the sky, typically in arc minutes.
- **Status (Statut):** Classification or status of the nebula (e.g., 'True PN', 'Likely PN', 'New candidate').

3.1.3 CSV File Sample

The following table displays a sample extract from the CSV file containing planetary nebula data. This sample illustrates the data structure and format that ‘data_retrieval.py‘ script outputs for use in subsequent processing steps.

Name	**Gal. Coord.**	**RA**	**DEC**	**Size**	**Status**
Ou 1	G001.0+01.0	17h45m	-27d10m	15’	True PN
Ou 2	G002.0+02.0	17h50m	-26d30m	12’	Likely PN
Ou 3	G003.0+03.0	17h55m	-26d00m	18’	New candidate

Table 1: Sample data from the planetary nebula dataset.

Note: The RA and DEC values are given in hours, minutes, and degrees, minutes format respectively, while the size is listed in arc minutes. The status column indicates whether the nebula is a confirmed planetary nebula (True PN), a likely planetary nebula (Likely PN), or a new candidate yet to be confirmed (New candidate).

3.1.4 Functions and Implementation:

- `scrape_data`:
  - **Description:** Extracts data from a given URL. It parses the HTML content to locate and retrieve planetary nebula information, which is then structured into a CSV file.
  - **Process:** The function initiates a request to the URL, parses the HTML response, and identifies the table containing the nebula data. It iterates through the table rows, extracting the required fields, and writes them into a CSV file with headers matching the data structure described above.
- `retrieve_dataset`:
  - **Description:** Manages the scraping of data from multiple predefined URLs, organizing the collected data into respective CSV files.
  - **Process:** It iterates over a list of URL and filename pairs, invoking `scrape_data` for each pair to collect and save the nebula data.

Usage:

- The structured data output serves as a critical input for subsequent stages of the project, such as image processing and machine learning model training, facilitating the identification and classification of planetary nebulae.

3.2 data_exporter.py

3.2.1 Purpose:

This script is crucial for organizing and preparing the image dataset used in the machine learning pipeline. It handles the creation of directories for storing processed images and manages the distribution of images into training and testing sets.

3.2.2 Key Functions:

- `clear_directory`:
  - **Description:** Clears all files and subdirectories within a specified directory.
  - **Functionality:** Iterates through all items in the directory and removes each one, handling both files and subdirectories.
- `create_directories`:
  - **Description:** Sets up the necessary directory structure for image storage, ensuring that each required directory exists and is empty.
  - **Functionality:** Creates a base directory and a series of subdirectories for different categories of images, clearing them if they already exist.
- `split_dataset`:
  - **Description:** Divides a set of images into training and testing datasets based on a specified ratio.
  - **Functionality:** Splits both 'images' and 'not_images' into separate training and testing sets, ensuring a balanced distribution for model training and evaluation.
- `save_images`:
  - **Description:** Saves a collection of images into a specified directory.
  - **Functionality:** Iterates over an array of images, saving each one to the directory with a unique filename prefix.
- `export_dataset`:
  - **Description:** Manages the overall process of exporting the image dataset into organized training and testing sets.
  - **Functionality:** Invokes the above functions to create directory structures, split the dataset, and save images accordingly.

3.2.3 Directory Structure Created by the Script:

```
image_dataset/
|-- train/
|   |-- images/
|   `-- not_images/
`-- test/
    |-- images/
    `-- not_images/
```

3.2.4 Usage:

- This script is integral to the preprocessing phase of the project, preparing the image data for efficient and organized access during the training and testing of the machine learning model.
- The structured and segregated dataset allows for a streamlined workflow in subsequent stages, facilitating the model's ability to learn from and evaluate the data effectively.

3.3 model_training.py

3.3.1 Purpose:

This script is central to the AI component of the project, encompassing the development, training, and optimization of a neural network model designed for classifying images of planetary nebulae.

3.3.2 Key Functions and Their Implementation:

- `preprocess_input`:
  - **Description:** Sets up data augmentation strategies and normalization layers to prepare the dataset for training.
  - **Implementation:** Uses a series of Keras layers for random flips, rotations, zooms, and resizing, along with a normalization layer.
- `create_data_generators`:
  - **Description:** Generates training and validation datasets from the image data stored in specified directories.
  - **Implementation:** Utilizes TensorFlow's utility function to create image data batches from directories, labeling them categorically.
- `build_model`:
  - **Description:** Constructs the neural network architecture, optionally using hyperparameters for tuning.
  - **Implementation:** Builds a model based on the VGG16 architecture with additional dense and dropout layers, and configures its compilation settings.
- `train_model`:
  - **Description:** Executes the training process of the model using the training and validation datasets.
  - **Implementation:** Trains the model for a specified number of epochs with early stopping based on validation loss.
- `hyperparameter_tuning`:
  - **Description:** Searches for the best hyperparameters to improve model performance.
  - **Implementation:** Uses Keras Tuner's Hyperband algorithm, tuning parameters like dropout rate and learning rate.
- `run_training`:
  - **Description:** Orchestrates the entire training process, including data preparation, model building, and training.
  - **Implementation:** Integrates all the above functions, optionally performs hyperparameter tuning, and saves the trained model.

3.3.3 Neural Network Model Schematic:

The neural network model employed in this project is based on the VGG16 architecture, a popular model in the field of deep learning. Below is a schematic representation of the model's architecture:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 16, 16, 512)       14714688
_________________________________________________________________
global_average_pooling2d     (None, 512)               0
(GlobalAveragePooling2D)
_________________________________________________________________
dense (Dense)                (None, 256)               131328
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 258
=================================================================
Total params: 14,879,170
Trainable params: 164,482
Non-trainable params: 14,714,688
```

3.3.4 Usage:

- This script is integral for the machine learning aspect of the project, enabling the automated classification of images into categories indicative of the presence or absence of planetary nebulae.
- The architecture and training regimen of the model are pivotal in determining its accuracy and effectiveness in real-world applications.

3.3.5 Architectural Improvements:

Enhancing the neural network architecture and incorporating cutting-edge deep learning methodologies present significant opportunities for advancing the model's performance:

- **Advanced Architectures:** Exploring architectures beyond VGG16, such as Transformer-based models (like Vision Transformers), Capsule Networks, or U-Net for image segmentation, could provide better feature extraction capabilities and improve classification accuracy, especially in complex astronomical images.
- **Attention Mechanisms:** Implementing attention mechanisms, which have shown great success in natural language processing, could allow the model to focus on the most relevant parts of an image, potentially enhancing its ability to identify subtle features of planetary nebulae.

- **Depth and Width of the Network:** Experimenting with the depth (number of layers) and width (number of units in each layer) of the network could lead to a more optimized model. Deeper networks can capture more complex patterns, but they also require more data and computational power.
- **Custom Layer Development:** Developing custom layers tailored to specific characteristics of astronomical images, such as layers that better handle the varying brightness and contrast inherent in such images, could improve model performance.
- **Neural Architecture Search (NAS):** Employing NAS techniques to automatically discover the best-performing model architecture for this specific task could result in a more efficient and effective model than manually designed architectures.
- **Generative Adversarial Networks (GANs):** Leveraging GANs for data augmentation or for generating synthetic images of nebulae could enhance the training process, especially in cases of limited real data.

These architectural improvements and advanced techniques represent the forefront of deep learning research. By integrating these innovations, the model can potentially achieve greater accuracy and efficiency, pushing the boundaries of automated astronomical classification and analysis.

3.4 nebula_image_downloader.py

3.4.1 Purpose:

This script is designed for downloading and preprocessing images of planetary nebulae, facilitating their use in machine learning models for classification and analysis.

3.4.2 Key Functions and Their Implementation:

- `load_and_preprocess_images_parallel`:
  - **Description:** Concurrently loads and preprocesses images based on celestial coordinates and field of view (FOV) data from a CSV file.
  - **Implementation:** Reads celestial coordinates (RA, DEC) and FOV from the CSV file and constructs URLs to download images using the hips2fits service. It employs a ThreadPoolExecutor for parallel downloading and normalizes the images for consistency.
- `construct_hips2fits_url`:
  - **Description:** Constructs the URL for the hips2fits service to download images based on specified astronomical parameters.
  - **Implementation:** Generates a URL with parameters including right ascension, declination, field of view, and image size, tailored for accessing the desired astronomical images.
- `download_and_normalize_image`:
  - **Description:** Downloads an image from a given URL and normalizes it for use in data analysis.

  - **Implementation:** Retrieves the image using the FITS format and normalizes the pixel values to a range of 0 to 1, enhancing the image's suitability for machine learning applications.

- `display_images`:
  - **Description:** Visualizes a set of preprocessed images, both in a grid layout and individually.
  - **Implementation:** Uses matplotlib to display a specified number of images in a grid format and highlights one image individually, providing a visual inspection of the preprocessing results.

- `clear_astropy_cache`:
  - **Description:** Clears the download cache used by Astropy, a common astronomical computing package.
  - **Implementation:** Invokes Astropy's utility to clear cached data, ensuring that storage does not become cluttered with temporary files.

3.4.3 Usage:

- This script streamlines the process of acquiring and preparing astronomical images for analysis, an essential step in the pipeline of detecting and studying planetary nebulae.
- By automating the download and preprocessing stages, the script significantly enhances the efficiency of data preparation, allowing for more focus on model development and analysis.

3.5 negative_image_downloader.py

3.5.1 Purpose:

This script focuses on downloading images that do not contain planetary nebulae, known as negative samples, to create a balanced dataset for training the machine learning model.

3.5.2 Key Functions and Their Implementation:

- `generate_random_coordinates`:
  - **Description:** Generates random celestial coordinates (Right Ascension and Declination) and Field of View (FOV) values.
  - **Implementation:** Produces random RA and Dec values over the entire celestial sphere, along with a range of FOV values, to ensure diverse image samples.

- `download_and_process_image`:
  - **Description:** Downloads and normalizes a single image based on given celestial coordinates and FOV.
  - **Implementation:** Constructs a URL for the hips2fits service to download the image in FITS format, then normalizes its pixel values to a 0-1 range, suitable for image processing and analysis.

- `download_negative_samples`:
  - **Description:** Downloads a specified number of negative sample images using parallel processing.
  - **Implementation:** Utilizes a thread pool to concurrently download and process images, enhancing efficiency in handling large volumes of data.
- `display_images`:
  - **Description:** Visually presents the downloaded images in a grid layout for inspection and verification.
  - **Implementation:** Employs matplotlib to display a specified number of images, providing a visual overview of the negative samples obtained.
- `clear_astropy_cache`:
  - **Description:** Clears the download cache used by the Astropy library.
  - **Implementation:** Ensures that temporary files generated during image downloads are removed, preventing unnecessary storage usage.

3.5.3 Usage:

- This script is crucial in creating a comprehensive dataset for training the machine learning model, as it provides a variety of images that the model should learn to classify as non-nebulae, thus aiding in improving the model's accuracy and generalization capability.
- The balance achieved by including both positive (nebulae images) and negative samples is vital for effective training and reducing bias in the model's predictions.

3.6 main_training.py

3.6.1 Purpose:

This script serves as the central orchestrator of the project, integrating various components from data retrieval and image processing to dataset preparation and model training.

3.6.2 Process Flow:

1. **Data Retrieval:** Initiates the data retrieval process (currently commented out) to collect planetary nebula information.
2. **Image Processing:** Loads and preprocesses images of planetary nebulae using the 'nebula_image_downloader' and clears the Astropy cache post-processing.
3. **Negative Sample Processing:** Downloads and processes negative samples (images without nebulae) in a quantity matching the number of nebula images.
4. **Dataset Export:** Exports the processed positive and negative samples into structured datasets ready for training.
5. **Model Training:** Executes the training process of the machine learning model, including optional hyperparameter tuning.

3.6.3 Usage:

- This script is the entry point for running the entire pipeline, encompassing end-to-end steps from raw data to a trained model, making it a crucial component of the project.
- It simplifies the execution of multiple complex tasks into a streamlined process, enhancing the efficiency and reproducibility of the model training phase.

3.7 test_model_accuracy.py

3.7.1 Purpose:

This script is dedicated to evaluating the trained model's accuracy and overall performance, providing critical feedback on the effectiveness of the model.

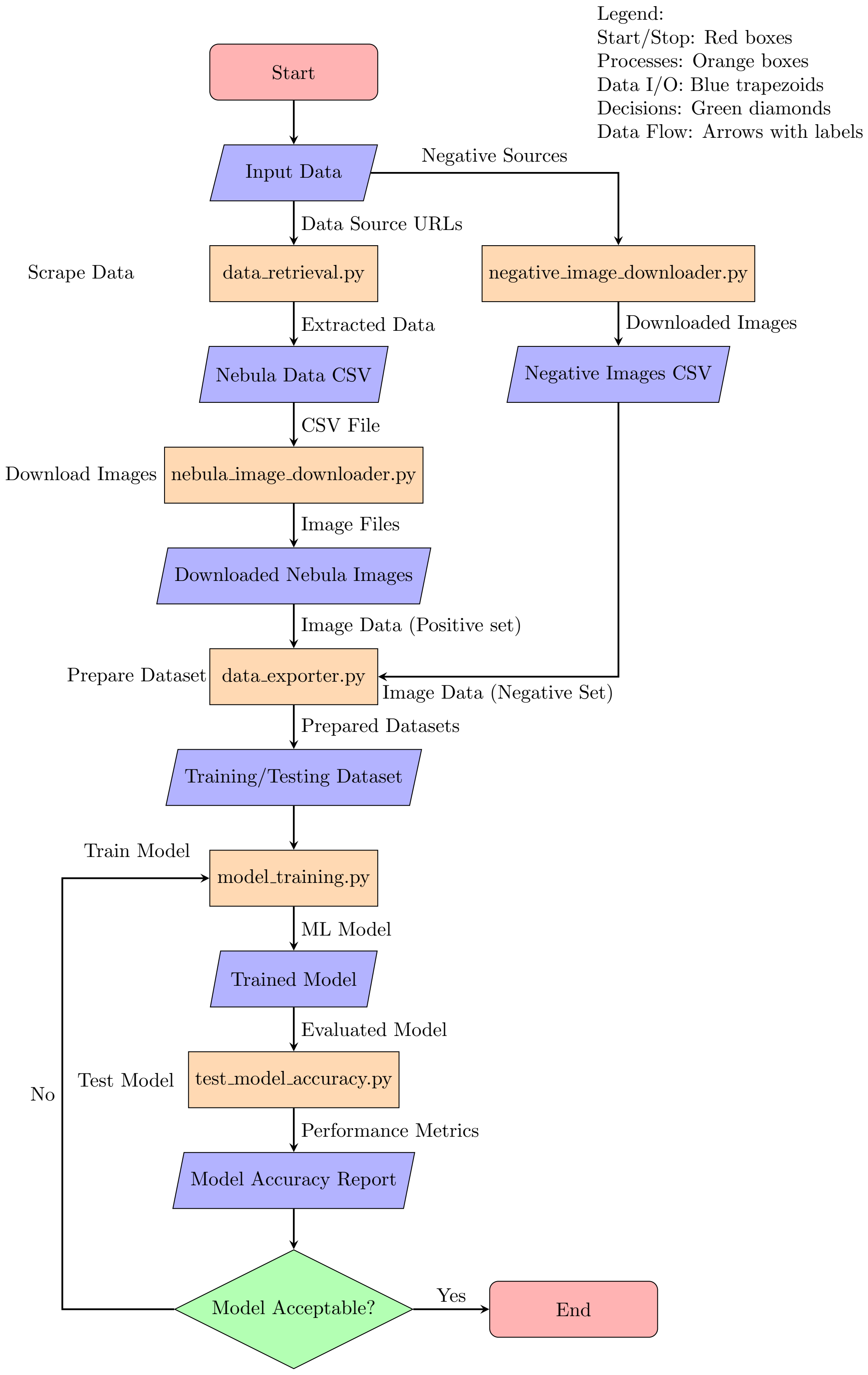
3.7.2 Key Functions:

- `preprocess_input`:
  - **Description:** Processes images to prepare them for testing, ensuring they are in the correct format for the model.
- `load_model`:
  - **Description:** Loads the trained neural network model, making it available for performance evaluation.
- `create_test_generator`:
  - **Description:** Creates a data generator for the test dataset, facilitating the evaluation of the model with unseen data.
- `evaluate_model`:
  - **Description:** Conducts the actual assessment of the model, computing its performance metrics on the test data.
- `print_evaluation_report`:
  - **Description:** Outputs an evaluation report, detailing the model's accuracy and other relevant performance metrics.

3.7.3 Usage:

- This script is pivotal in the model development lifecycle, providing insights into the model's ability to generalize and perform on new data.
- The evaluation metrics and reports generated are essential for understanding the strengths and limitations of the model, guiding further improvements.

4 Flowchart of File Interactions



This flowchart provides a visual representation of the interaction and sequence of operations among the various scripts in the project. It starts with data retrieval and ends with model evaluation, showcasing the step-by-step process followed in the project.

5 Conclusion

The AI-based Detection of Planetary Nebulae project represents a comprehensive approach to automating the identification of celestial bodies in astronomical images. Through a series of interconnected Python scripts, the project leverages machine learning techniques for data processing, neural network model training, and evaluation. This document offers a detailed overview of each component, aiding new programmers and scientists in understanding the workflow and methodologies employed in this cutting-edge research.