AstroNN and Stellar Classification Report

Seif Eldin hesham khashaba

202200973

Ahmed Mostafa

2022011143

12/21/2024

Draft 2

Table of Contents

1. Introduction

This report documents the methodologies and functions used in two deep learning notebooks focusing on galaxy and stellar classification. The datasets used are AstroNN and SDSS17, with objectives to classify astronomical objects into distinct categories.

---

2. Dataset Description

AstroNN Dataset:

- A pre-labeled dataset of galaxy images, available via the `astroNN.datasets` library.
- Contains 10 unique galaxy classes.

SDSS17 Dataset:

- Stellar classification dataset from the Sloan Digital Sky Survey (SDSS17).
- Includes 100,000 samples with 18 features, including spectral data, redshift, and classification labels.

---

3. Methodology

Data Loading and Preprocessing:

- **Galaxy Classification:**
  - Downloaded using `load_galaxy10` from `astroNN.datasets`.
  - Saved locally as `.npy` files to optimize memory usage.
  - Loaded with memory mapping to reduce RAM constraints.
- **Stellar Classification:**
  - Downloaded from Kaggle.
  - Features were scaled using `StandardScaler` for normalization.
  - Target labels were encoded using `LabelEncoder`.
  - Data split into training (80%) and testing (20%) sets.

Model Design and Training:

- **Galaxy Classification:**
    - Built using TensorFlow/Keras.
    - Consisted of convolutional layers for feature extraction and dense layers for classification.
- **Stellar Classification:**
    - A simple feedforward neural network with dropout layers for regularization.
    - Activation functions: ReLU for hidden layers and softmax for the output layer.

---

4. Problems in the Data

1. **Large Dataset Size:**
    - Galaxy10 dataset (~2.74GB) required optimized loading strategies to prevent memory overflow.
    - Stellar dataset (100,000 samples) posed computational challenges for training on limited hardware.
2. **Class Imbalances:**
    - Galaxy classification data displayed imbalances, impacting model performance on minority classes.
3. **Feature Variability:**
    - Stellar classification data included features with significantly different scales.

---

5. Optimizations and Constraints Management

Memory usage and large dataset handling:

- Used memory-mapped arrays (`mmap_mode`) for loading Galaxy10 images.
- Saved processed data locally to avoid repetitive downloads.

Comparison of Optimizers:

- Tested optimizers including SGD, Adam, and RMSprop.
    - **SGD:** Slower convergence but provided stable learning.
    - **Adam:** Achieved faster convergence and better accuracy.
    - **RMSprop:** Balanced performance but prone to overfitting.

---

6. Results and Insights

- **Galaxy Classification:**
    - Achieved ~90% accuracy with data augmentation and Adam optimizer.
- **Stellar Classification:**
    - Achieved ~85% accuracy, highlighting the importance of feature scaling and label encoding.

---

7. Additional Insights from Documentation

GalaxyDataGenerator Class:

- Handles the Galaxy dataset efficiently by managing data loading, preprocessing, and batching.
- Key features:
    - Uses memory-mapped mode (`mmap_mode='r'`) to avoid memory overflow.
    - Shuffles and normalizes images; converts labels to one-hot encoding.

Independent Functions:

- `plot_sample_images`: Visualizes dataset samples for validation.
- `visualize_predictions`: Displays model predictions alongside true labels.

Stages of Processing:

1. **Data Splitting:**
    - Ensures an 80-20 train-validation split for reproducibility.
2. **Model Training:**
    - Uses callbacks for early stopping and learning rate adjustment.
3. **Performance Visualization:**
    - Assesses accuracy and loss trends to identify overfitting or underfitting.
4. **Model Evaluation and Saving:**
    - Saves the trained model for further use.

8. Conclusion

Both notebooks successfully implemented deep learning techniques for galaxy and stellar classification. Optimizations like memory mapping and careful choice of optimizers helped overcome hardware constraints. Insights from the GalaxyDataGenerator and additional functions enhanced efficiency and usability. Future work could involve exploring additional architectures and addressing class imbalances through data augmentation or weighted losses.