Inferring Photometric Redshifts from Multichannel Images

Balázs Menkó

November 6, 2024

Contents

1 Theoretical Background

The Sloan Digital Sky Survey (SDSS) offers an extensive dataset for examining the redshift of galaxies, which gauges the stretching of light from distant objects due to the universe's expansion. Redshift is a crucial tool for understanding galaxy motion and distance, with higher redshifts correlating to greater distances and earlier stages of the universe. SDSS has recorded redshifts for millions of galaxies, aiding astronomers in mapping the universe's large-scale structure and determining the rate of cosmic expansion. For my semester-long project, I will utilize the SDSS database to train a Convolutional Neural Network (CNN) to predict galaxy redshifts based on their colored images.

2 SDSS Database

To access the SDSS database, the `astroquery` Python package is necessary. It enables querying the database using SQL with the following command:

```
astroquery.sdss.SDSS.query_sql("""
    SELECT *
    FROM TableName
""")
```

On the SDSS webpage, you can find a Schema Browser, which provides a description of all tables and their columns. For the dataset, I use the **SpecObj** view, which is *a view of Spectro objects that contains only the clean spectra.*

To obtain images, I discovered that this can be easily accomplished using a URL query string. All images can be downloaded using the following function:

```
def get_sdss_image(ra, dec, scale, size):
    url = "https://skyserver.sdss.org/dr16/" + \
          "SkyServerWS/ImgCutout/" + \
          f"getjpeg?ra={ra}&dec={dec}&scale={scale}" + \
          f"&width={size}&height={size}"
    if url is valid
        return flattened image
    else
        return Error message
```

By querying the **right ascension** (`ra`) and **declination** (`dec`) from the **SpecObj** table, images can be easily downloaded in a loop that iterates through astronomical coordinates. The `scale`[1] parameter controls the image scale, with smaller values providing a higher zoom level. The `size`[2] parameter sets the dimensions in pixels for both the height and width, ensuring the images are square.

[1]For this project, I used a value of 0.3.
[2]I downloaded images at 128x128 pixels.

2.1 Create Own Database

As the next step, I needed to query data from the SDSS Server. I identified that the required columns were the spherical coordinates, the redshift value (denoted by `z`) and its error, along with the velocity dispersion and its error. I filtered the data to include only galaxies with an ID smaller than the `BIGINT` value. This filter was applied because there was an error during the upload process due to the length of the ID value.

Next, I filtered out records that had a zero offset and no warnings.[3] Finally, I searched for records where the errors were positive, less than ten thousand, and less than ten times the original value.

```
astroquery.sdss.SDSS.query_sql("""
    SELECT specobjid, ra, dec, z, zErr, velDisp, velDispErr
    FROM SpecObj
    WHERE class='Galaxy'
      AND specobjid < 9223372036854775807 -- less than BIGINT
      AND zOffset=0
      AND zWarning=0
      AND zErr>0
      AND z>zErr*10000
      AND velDispErr > 0
      AND velDisp > velDispErr*10
""")
```

Since I had access to a PostgreSQL server, I began uploading the data along with the corresponding images. However, I found that the progress was slower than expected, so I decided to upload the data to the server in batches. After downloading 50 images and their associated data, I would upload them to the server using a loop.[4] Currently, I have uploaded 3,000 images in the database, but I plan to run the program until I reach 10,000–15,000 images, as having more data will improve the training of the networks.

3 Future Steps

Building a CNN with the `tensorflow.keras` Python package is straightforward, as the built-in functions simplify network creation. For example, `Conv2D` is used for 2D convolution, `ReLU` serves as the activation function, and `MaxPooling2D` is used for max pooling. Furthermore normalization is needed for the pictures because it is easier to calculate number in intervall of zero and one, than calculate huge number. Since I do not have access to a GPU on Kooplex, the training progress will be slower, but I plan to work efficiently and achieve acceptable results with smaller datasets. Eventually, I will train a network using the full dataset once it's available.

[3] `zWarning`: *Bitmask of warning values; 0 means all is well*

[4] I wrote all ObjectIDs to a *.csv* file to s

A Animation of Zooming and Codes

The animation of zooming and Python notebook can be found on Google Drive.

B Examples for Downloaded Images

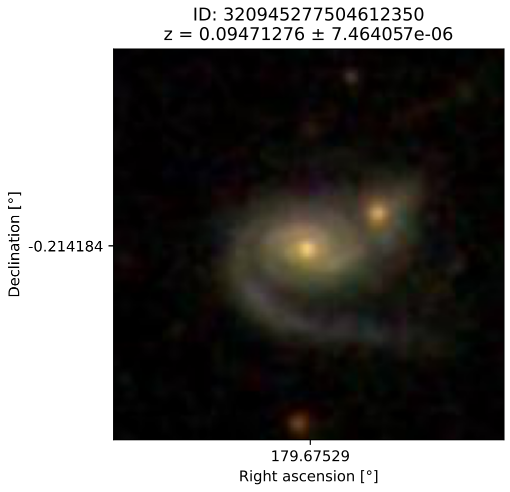

Figure 1: An example for galaxies close to each other.

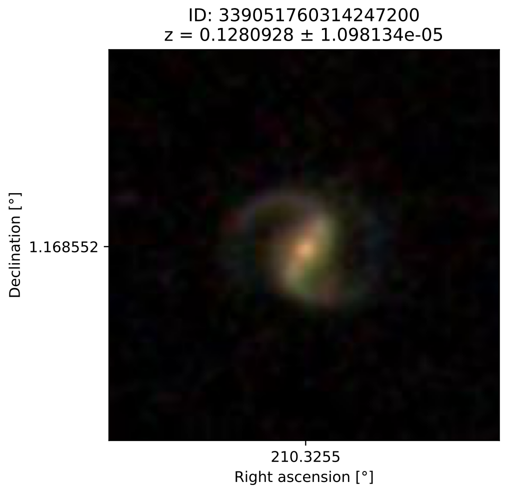

Figure 2: An example for a single barred spiral galaxy.