INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT AKURDI, PUNE

Documentation On

"Classification of Galaxies based on their Shape"

PG-DBDA SEP 2020

Submitted By:

Group No: G-25

Shweta Waskar-1544

Snehal Waskar -1556

Mr. Prashant Karhale	**Mr. Akshay Tilekar**
Centre Coordinator	**Project Guide**

Acknowledgement

First and Foremost, we thank to almighty God for giving us the support, strength, positive spirit and talent to do this project.

“The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible”.

“This work is the result of inspiration, motivation, knowledge, interest, support, guidance,

cooperation and efforts by many people at different levels. We are indebted to all of them”.

We would also like to take this opportunity to acknowledge the valuable contributions made

by our family members by supporting and motivating us in every walk of life.

We are thankful to **Mr. Prashant Karhale**, Centre Co-ordinator IACSD CDAC, Akurdi Pune for providing the opportunity, infrastructure and facilities for entire work.

We would like to express our great appreciation to our project Guide **Mr. Akshay Tilekar**, internal Project Guides **Mr. Rahul Pund**, and **Mr. Manish Bendale** for their valuable and constructive suggestions during the planning and development of this project.

We also thank all staff members from the IACSD who in some way or other helped us in completion of this project.

We cannot conclude our acknowledgement without expressing our thanks to our friends who helped us directly or indirectly during the course of this project.

Feedback for improving the contents of the report would be more than welcome.

Abstract

There are billions of galaxies discovered in the universe which are of different shapes. These galaxies can be classified into different groups based on their shape. We aim for the algorithm using machine learning to classify the astronomical data of galaxies to detect their shape. We have used astronomical data which is either made public by the observatories or the data from virtual observatories. Our technic will classify the huge dataset of galaxies into the respective type with accuracy.

INDEX

Chapter 1

1.1 Introduction

Image classification is a significant and recurring theme in pattern recognition and digital image processing, with many applications in a number of domains related to images, including computer vision, medical image analysis, biology, etc. In the field of astronomy, galaxy classification, using digital images collected from large-scale sky surveys to determine the galaxy morphological classes, has long been of great interest to astronomy researchers.

For well over a century astronomers have differentiated or classified galaxies based on their visual appearance. The science of galaxy "morphology" matured with the work of Hubble (1926 & 1936), whose famous "Hubble Sequence" remains today a standard technique by which to study galaxies. This sequence of normal spiral and elliptical galaxies was further expanded and delineated into subclasses by de Vaucouleurs in his classic 1959 paper, also described in the *Hubble Atlas* (Sandage 1961; see also Sandage 1975). More recent reviews are given in Buta (1992), Roberts & Haynes (1994) and Van Den Bergh (1998). Even though the morphology/classification scheme has grown into a rather large and complex set of designations, the basic observables are simple: relative size and brightness of the nucleus with respect to the disk or outer envelope (quantified as the bulge-to-disk ratio), the shape of the disk (e.g., arms, bars, rings, or asymmetries), and other factors, such as the surface brightness, colour, and the presence of dust.

The Hubble sequence, at first glance, appears to be simply a qualitative "roadmap" giving convenient names to the wide variety of galaxies observed in the Universe. But as Sandage (1986) points out, the sequence does in fact separate galaxies into physically related classes, representing an "evolutionary" sequence. For example, the mean disk age of spirals progresses along the sequence (early to late-type), as does the present star-formation rate (e.g., late-type galaxies are forming stars at a much higher rate than early-type galaxies; see also Ferrini & Galli, 1988). Hubble & Humason (1931) understood that the morphological sequence was correlated to the galaxy density environment: galaxy *clusters* contain more lenticular and elliptical galaxies than the typical field density.

Morphological segregation appears to be a property of large clusters, including those of Virgo and Coma. Further, the luminosity functions of spirals is different from that of elliptical-type galaxies (cf. Binggeli et al. 1988). Although the origin and formation of elliptical galaxies is still largely not understood, more recent work has pointed to a scenario in which elliptical galaxies may be the creation of spiral galaxy mergers in the dense cores of galaxy clusters (e.g., Dressler 1980; Postman & Geller 1984; but also see Mamon 1992; Whitmore et al. 1993; Julian et al. 1997). There is evidence that widely separated morphological types (in the Hubble Sequence) have systematically different mean colors (e.g., B-V), presumably from different populations of stars, both old and newly formed, that dominate the light (e.g., Holmberg 1958; de Vaucouleurs 1977; Giovanelli & Haynes 1983; see the review by Roberts & Haynes 1994; Buta et al. 1994; Odewahn et al. 1996). Another class of galaxy, the dwarf spheroidal, also appears to be associated with the dense environments of clusters, although their origin, evolution, age and sustained existence (from the disruptive gravitational tidal forces) remains a mystery (cf. Sandage 1990; also discussed by Impey & Bothun 1997).

1.2 Purpose and Scope of Project

1.2.1 Classification of Galaxies Based on Their Shape

There are three main types of galaxies: Elliptical, Spiral, and Irregular. Two of these three types are further divided and classified into a system that is now known the tuning fork diagram. When Hubble first created this diagram, he believed that this was an evolutionary sequence as well as a classification.

1. **Spiral**

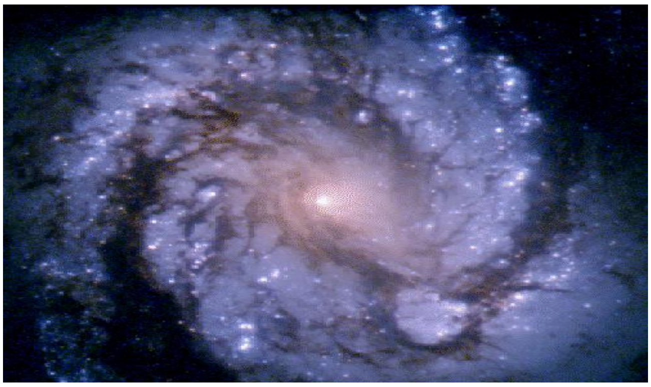


Fig: Spiral Galaxy

Spiral galaxies have three main components: a bulge, disk, and halo (see right). The bulge is a spherical structure found in the centre of the galaxy. This feature mostly contains older stars. The disk is made up of dust, gas, and younger stars. The disk forms arm structures. Our Sun is located in an arm of our galaxy, the Milky Way. The halo of a galaxy is a loose, spherical structure located around the bulge and some of the disk

2. Elliptical

Fig: Elliptical Galaxy

Elliptical galaxies are shaped like a spheroid, or elongated sphere. In the sky, where we can only see two of their three dimensions, these galaxies look like elliptical, or oval, shaped disks. The light is smooth, with the surface brightness decreasing as you go farther out from the centre. Elliptical galaxies are given a classification that corresponds to their elongation from a perfect circle, otherwise known as their ellipticity. The larger the number, the more elliptical the galaxy is. So, for example a galaxy of classification of E0 appears to be perfectly circular, while a classification of E7 is very flattened. The elliptical scale varies from E0 to E7. Elliptical galaxies have no particular axis of rotation.

3. Irregular

Irregular galaxies have no regular or symmetrical structure. They are divided into two groups, Irr I and IrrII. Irr I type galaxies have HII regions, which are regions of elemental hydrogen gas, and many Population I stars, which are young hot stars. Irr II galaxies simply seem to have large amounts of dust that block most of the light from the stars. All this dust makes is almost impossible to see distinct stars in the galaxy.

Fig: Irregular Galaxy

Chapter 2

2.1 Software Life Cycle Model

A process followed in software projects is SDLC. Each phase of SDLC produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced according to the design. Testing should be done on a developed product based on the requirement. The deployment should be done once the testing was completed. It aims to produce a high-quality system that meets or exceeds customer expectations, works effectively and efficiently in the current and planned information technology infrastructure, and is inexpensive to maintain and cost-effective to enhance.

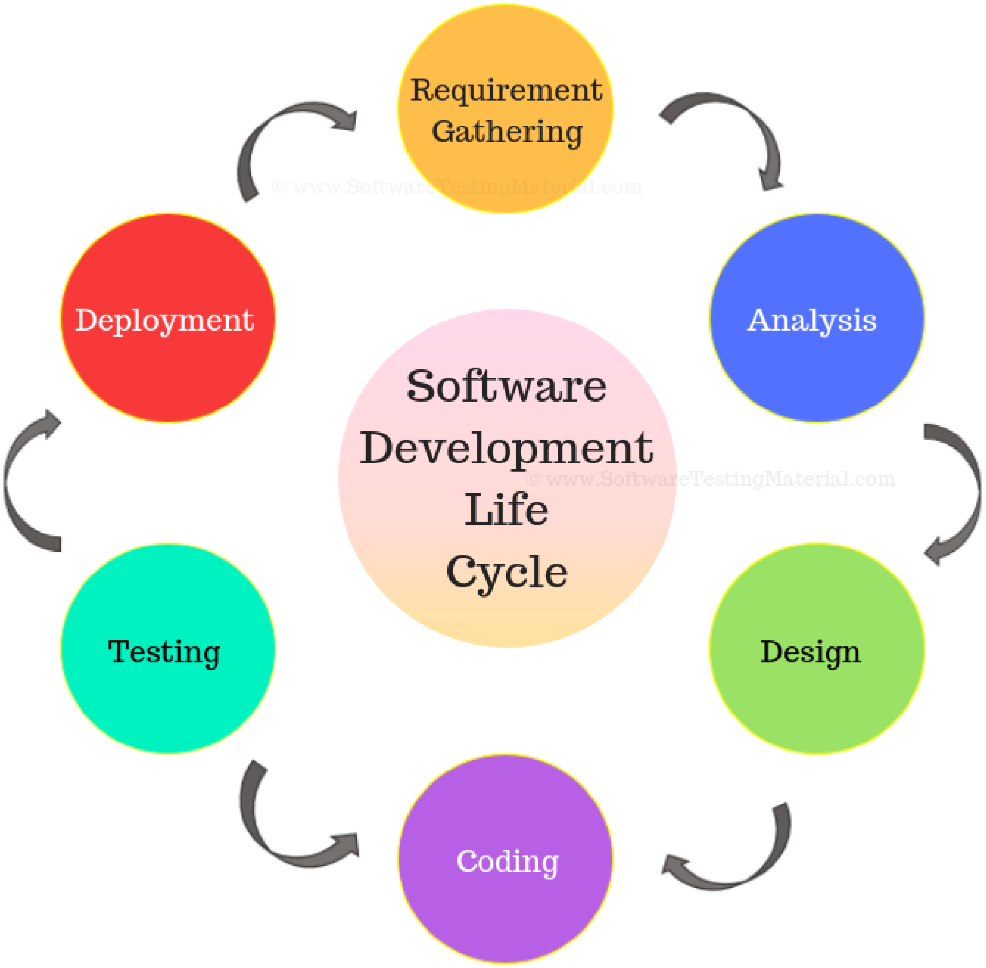

Requirement Gathering
Analysis
Design
Coding
Testing
Deployment
Software Development Life Cycle
© www.SoftwareTestingMaterial.com
© www.SoftwareTestingMaterial.com

Fig. Software development Life Cycle

2.2 Overall Description

2.2.1 Data

Name	No. of classes	Class name
Milky Way	**01**	**Spiral**
M87	**01**	**Elliptical**
Rose Galaxy	**01**	**Irregular**

2.2.2 Imports

- Matplotlib:

  Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

- NumPy:

  NumPy **is** an open-source numerical Python library. NumPy contains a multi-dimensional array and matrix data structures. It can be utilised to perform a number of mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.

- TensorFlow:

  TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

- Keras:

  Keras is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow. It was developed to make implementing deep learning models as fast and easy as possible for research and development.

- Seaborn:

  Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data.

2.3 Requirement Specification

2.3.1 Initial non-functional requirement will be:

- Getting large datasets which can provide developer enough data to train the model.
- Maintain the minimum variance and bias so the model is successfully work.
- Avoid the under-fitting and over-fitting

2.3.2 Initial function requirement will be:

- Selecting the appropriate algorithms.
- Determining the appropriate input format to algorithm.
- Train the model.
- Test the model.

2.3.3 Hardware Requirement:

- Processor: Intel(R) Core(TM)i3 and above
- RAM: Minimum 4GB
- System Type: 6 4-bit Operating System, x64 based processor
- OS: Windows

2.3.4 Software Requirement:

- Jupyter NoteBook
- Colab

Chapter 3

3.1 Plan of Project Execution

3.1.1 Feasible Study Phase

- Study of different distributed algorithms.
- Different types of galaxy images are studied and corresponding.
  After detail study, labelling in done by segregating the images and with different shapes

3.1.2 Requirement Analysis

- The database is pre-processed such as Image reshaping, resizing and conversion to an array form.
- There is a huge database so basically the images with better resolution and angle are selected. After selection of images we should have deep knowledge about the different shapes and the ages they have. Huge research is done from galaxy classification.

3.1.3 Design Phase

- Designing algorithm and graphs for getting more accuracy and satisfaction.
- Adjusting the Images in categories so that our model identifies the shapes.

3.1.4 Implementation Phase

- CNN has different layers that are Dense, Dropout, Activation, Flatten, Convolution2D and MaxPooling2D these all will implement.

3.1.5 Testing Phase

- After the model is trained successfully the software can identify the shape if the galaxy species is contained in the database.
- After successful training and pre-processing, comparison of the test image and trained model takes place to predict the shapes.

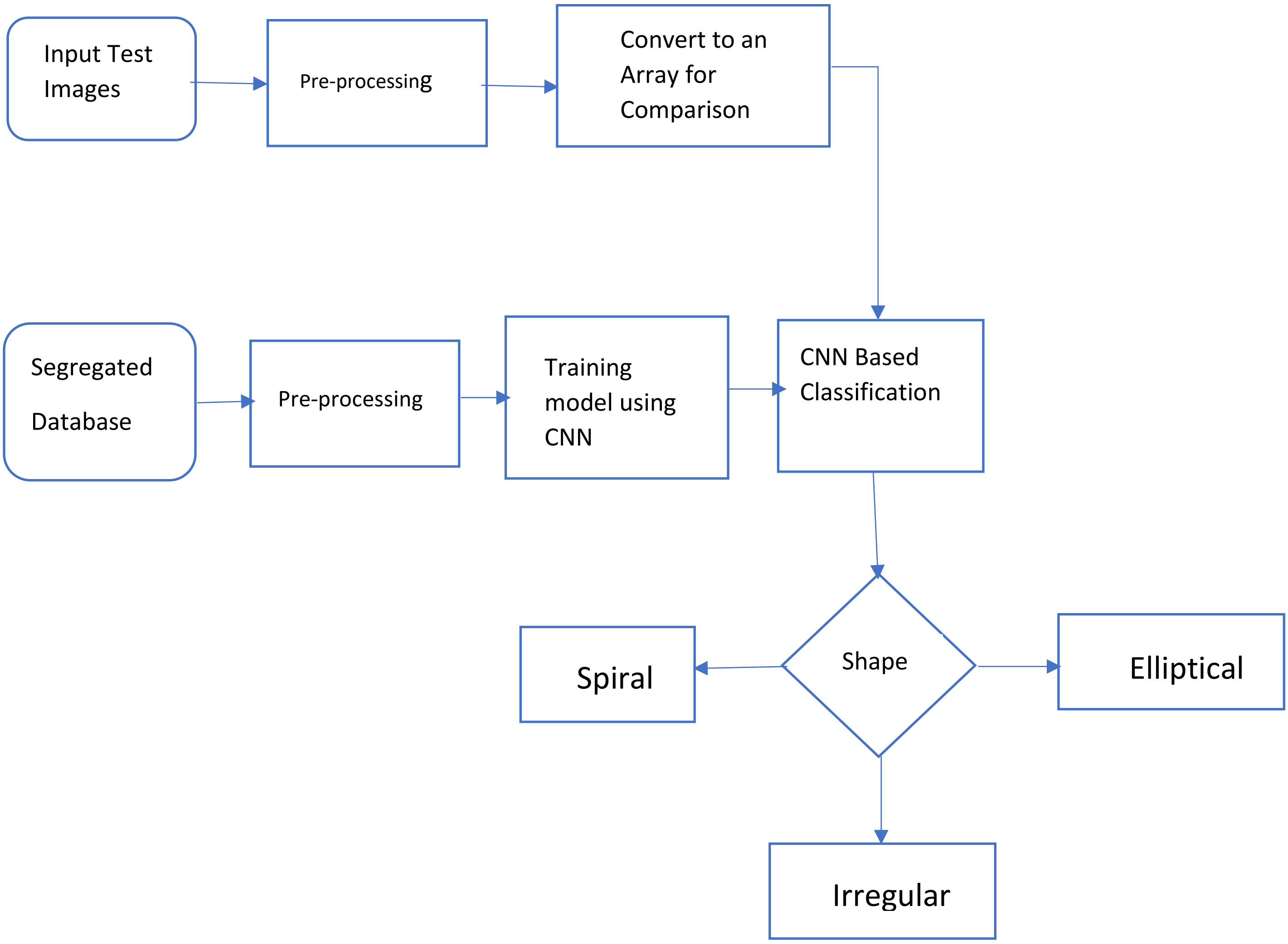
3.1.6 Deployment Phase

- After follow all above steps then deploy the model on the Internet so that it will be useful for the astronomer for their further studies.

3.2 System Design

3.2.1 Flowchart of the System:

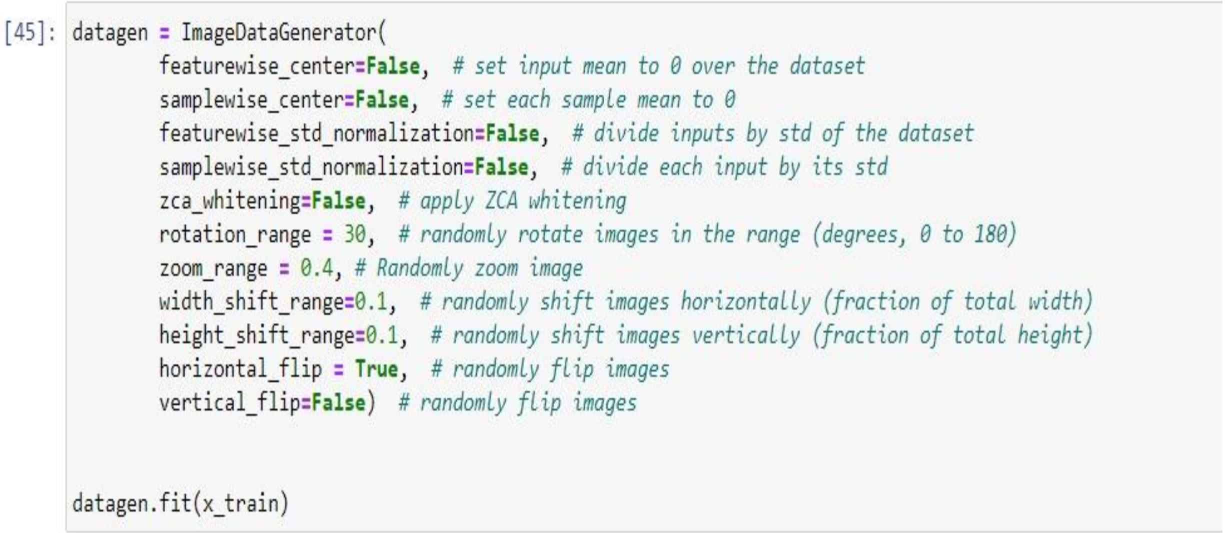
The flowchart of the algorithm is represented in Figure



The above flowchart describes the working flow of the project. First pre-processed the data and selected the model. Then algorithm was later implemented in python and deployed on web. The model was later tested against raw images.

3.3 Data Pre-processing

- The dataset is divided into 80% for training and 20% for validation.
- First, augmentation settings are applied to the training data.
- Set height and width of input image.
- The settings applied include flipping (horizontal), shearing of range (0.2) and zoom (0.2). Rescaling image values between (0 –1) called normalization. All these parameters are stored in the variable “train_datagen” and “test_datagen”.



```
[45]: datagen = ImageDataGenerator(
              featurewise_center=False,  # set input mean to 0 over the dataset
              samplewise_center=False,  # set each sample mean to 0
              featurewise_std_normalization=False,  # divide inputs by std of the dataset
              samplewise_std_normalization=False,  # divide each input by its std
              zca_whitening=False,  # apply ZCA whitening
              rotation_range = 30,  # randomly rotate images in the range (degrees, 0 to 180)
              zoom_range = 0.4, # Randomly zoom image
              width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
              height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
              horizontal_flip = True,  # randomly flip images
              vertical_flip=False)  # randomly flip images


      datagen.fit(x_train)
```

Chapter 4

4.1 Model Building

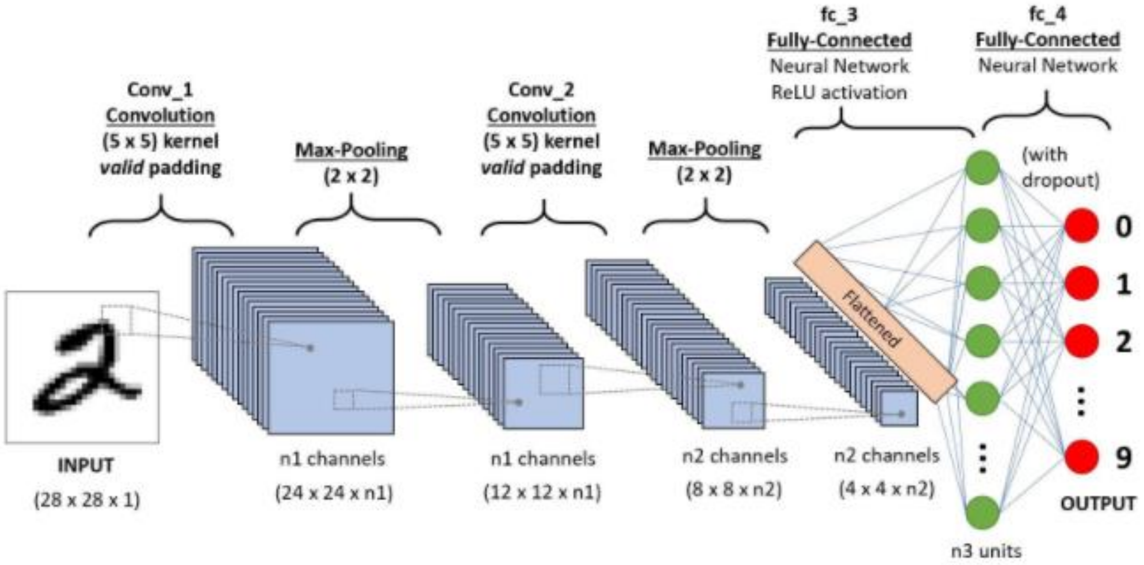
4.1.1 Algorithm Research and Selection:

For this classification problem, following deep learning model was used:

Convolution Neural Network

It is well known for its widely used in applications of image and video recognition and also in recommender systems and Natural Language Processing (NLP).

However, convolutional is more efficient because it reduces the number of parameters which makes different from other deep learning models.



Main Steps to build a CNN (or) Conv. net:

1. Convolution Operation
2. ReLU Layer (Rectified Linear Unit)
3. Pooling Layer (Max Pooling)
4. Flattening
5. Fully Connected Layer

1. Convolution Operation:

Convolution is the first layer to extract features from the input image and it learns the relationship between features using kernel or filters with input images.

2. ReLU Layer:

ReLU stands for the Rectified Linear Unit for a non-linear operation. The output is f(x) =max (0,x).we use this because to introduce the non-linearity to CNN.

3. Pooling Layer:

It is used to reduce the number of parameters by down sampling and retain only the valuable information to process further. There are types of Pooling:

- Max Pooling.
- Average and Sum pooling.

4. Flattening:

We flatten our entire matrix into a vector like a vertical one. So, that it will be passed to the input layer.

5. Fully Connected Layer:

We pass our flatten vector into input Layer. We combined these features to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the outputs.

4.2 Algorithm Implementation

To implement the idea of disease detection and to train the machine accordingly requires lot of steps which are mentioned below: -

1. Label Data for input like Training Data, Testing Data and Valid Data in different folders.
2. Import all libraries like NumPy, Pandas, Matplotlib, Tensorflow, keras Convolution2D, MaxPooling2D, Flatten, Dense, Sequential Model, ImageDataGenerator, image, random etc.
3. Assign paths of the folders where training and testing data are available.
4. Assign Dictionary to write all the shape name in a sequential way.
5. Add Convolutional Layer with 32 filters and 64 filters and apply Relu and softmax activation function.
6. Add Maxpooling layer for extracting the features from convolutional layer.
7. Repeat Step5 and Step6.

8. Add Flatten Layer to convert 3D Array to 1D Array.

9. Add Hidden Layers or Dense Layers with 128 neurons and activation function is relu

10. Add Hidden layer or Dropout layer with value 0.4

11. Add Hidden Layers or Dense Layers with 128 neurons and activation function is relu.

12. Add Output Layer with 4 neurons and Activation function Softmax.

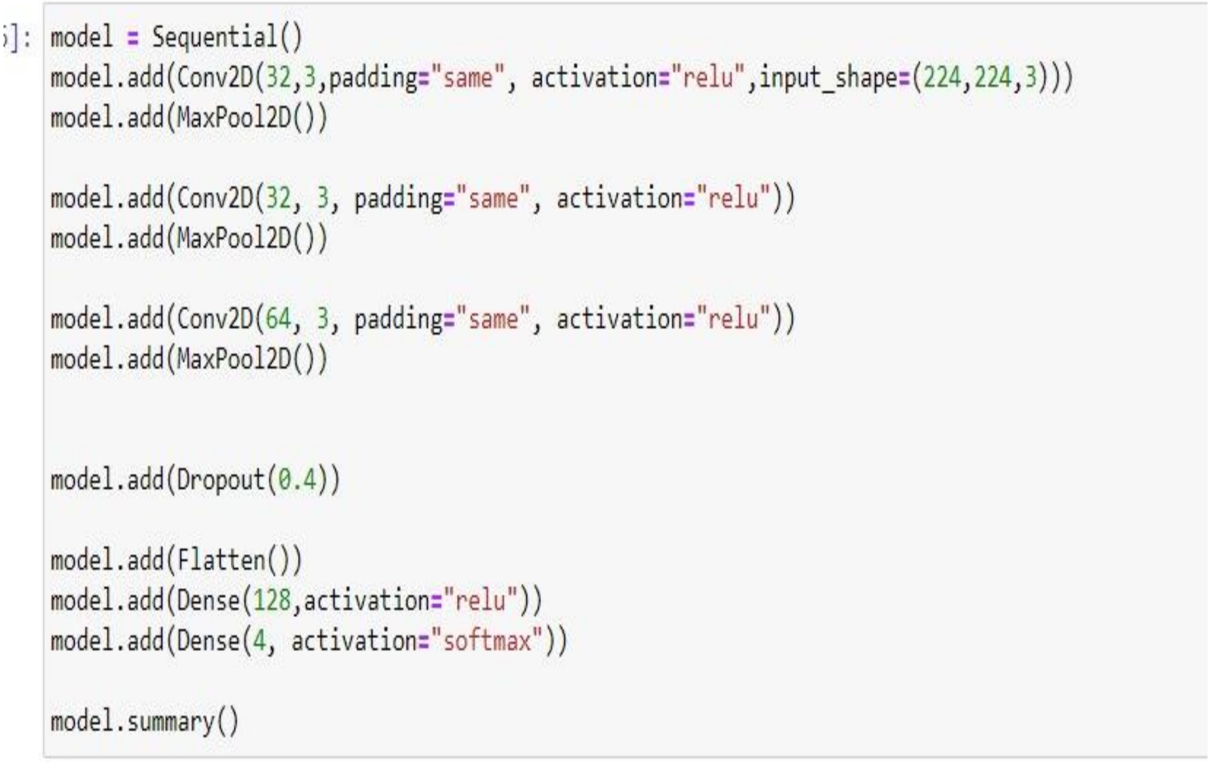
13. Apply Compile function to compile all the layers with loss function categorical_crossentropy and optimizer='Adam'.

14. Model Fit Function is used to fit all variables like training set, steps per epoch=516, epochs=20, validation data=test set etc.

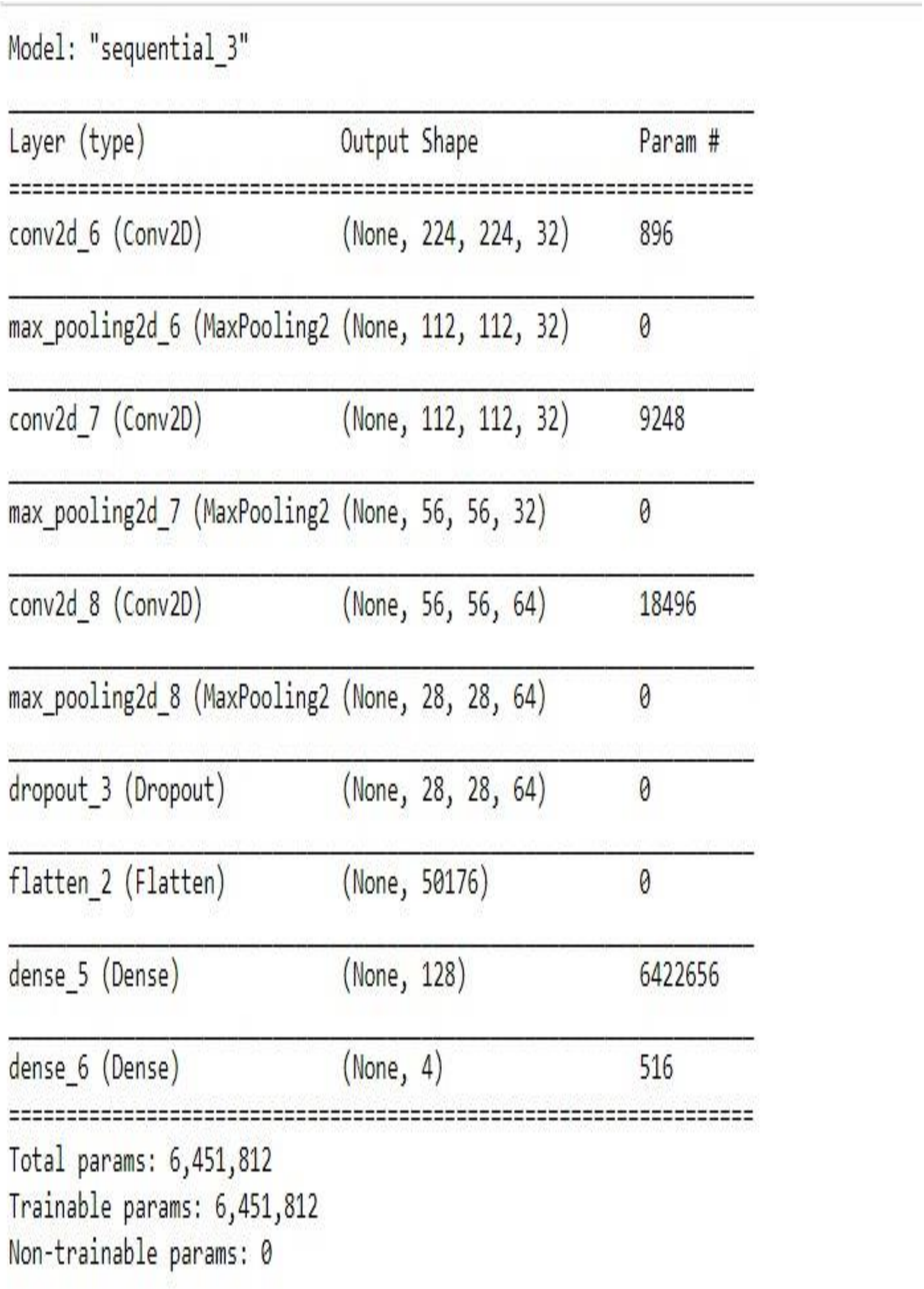
15. Start Training, minimum time taken to train data in this dataset will be 3 hours.

16. After the model gets trained, apply path of the folder where valid images are available.

17. Apply Predict function to predict the valid image to get output.

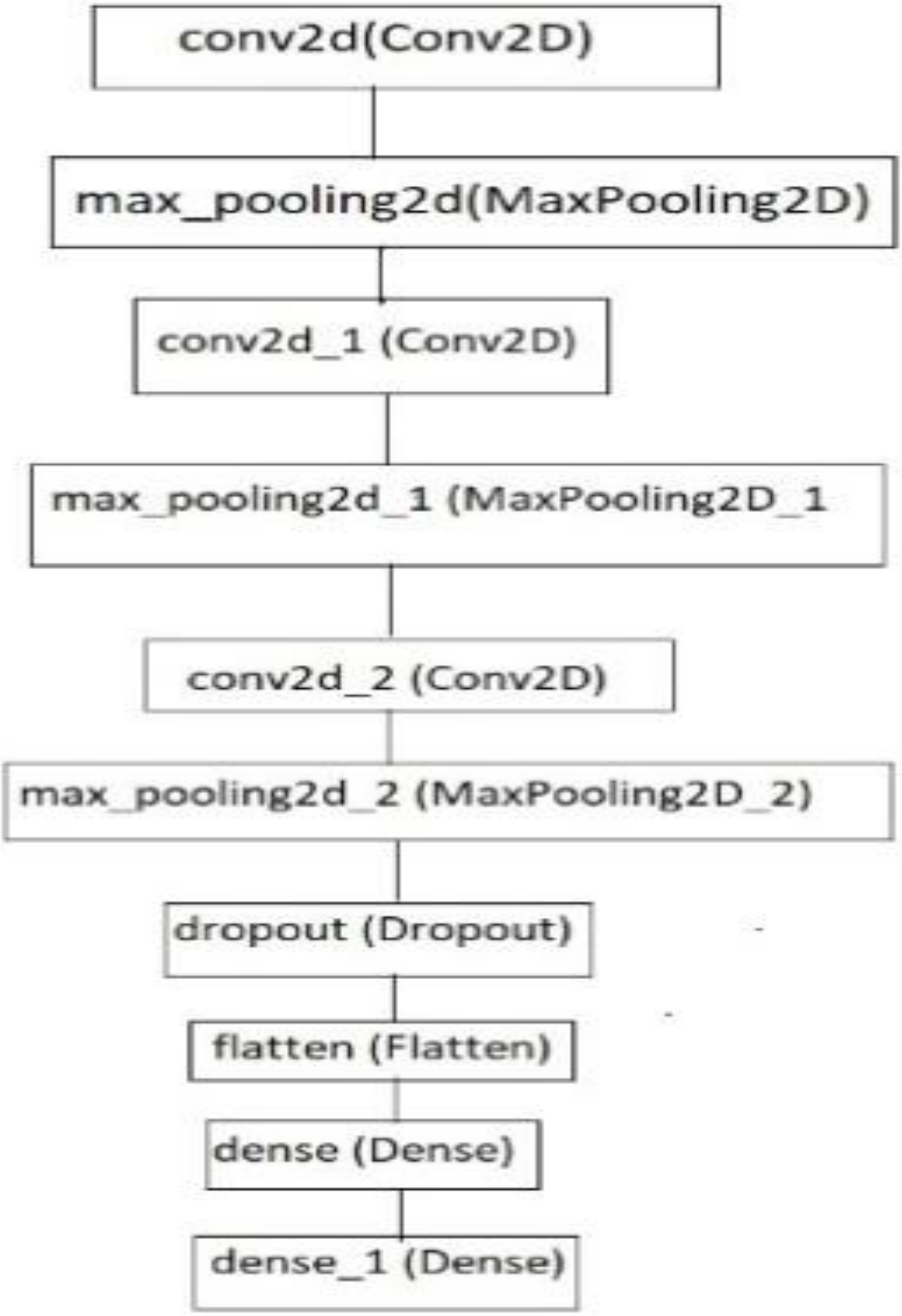


```
model = Sequential()
model.add(Conv2D(32,3,padding="same", activation="relu",input_shape=(224,224,3)))
model.add(MaxPool2D())

model.add(Conv2D(32, 3, padding="same", activation="relu"))
model.add(MaxPool2D())

model.add(Conv2D(64, 3, padding="same", activation="relu"))
model.add(MaxPool2D())


model.add(Dropout(0.4))

model.add(Flatten())
model.add(Dense(128,activation="relu"))
model.add(Dense(4, activation="softmax"))

model.summary()
```



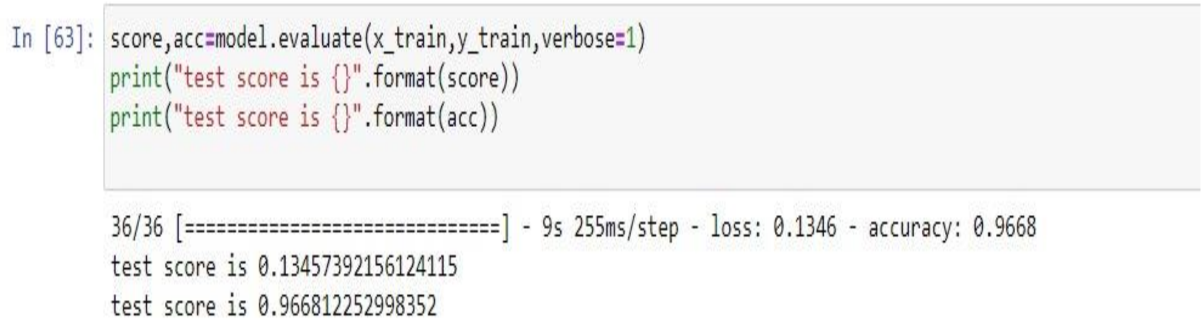
```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 224, 224, 32)      896
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 112, 112, 32)      0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 112, 112, 32)      9248
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 56, 56, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 56, 56, 64)        18496
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 28, 28, 64)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 28, 28, 64)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 50176)             0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               6422656
_________________________________________________________________
dense_6 (Dense)              (None, 4)                 516
=================================================================
Total params: 6,451,812
Trainable params: 6,451,812
Non-trainable params: 0
```

4.3 Model Flowchart



4.4 Result

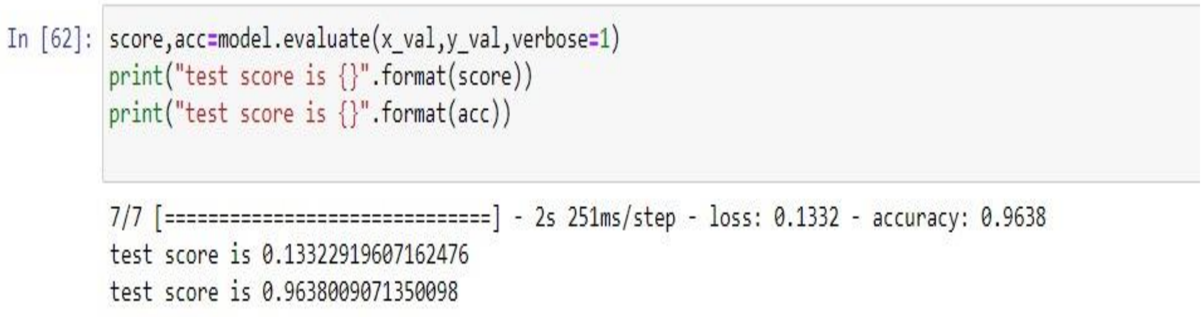
4.4.1 Training Accuracy



```
In [63]: score,acc=model.evaluate(x_train,y_train,verbose=1)
         print("test score is {}".format(score))
         print("test score is {}".format(acc))

         36/36 [==============================] - 9s 255ms/step - loss: 0.1346 - accuracy: 0.9668
         test score is 0.13457392156124115
         test score is 0.966812252998352
```

- ➢ Applied CNN model on Training data and got 96.68% accuracy for Train Data.

4.4.2 Testing Accuracy



```
In [62]: score,acc=model.evaluate(x_val,y_val,verbose=1)
         print("test score is {}".format(score))
         print("test score is {}".format(acc))

         7/7 [==============================] - 2s 251ms/step - loss: 0.1332 - accuracy: 0.9638
         test score is 0.13322919607162476
         test score is 0.9638009071350098
```

- ➢ Applied CNN model on testing data and got 96.38% accuracy for Test Data.

Training Accuracy	96.68%
Testing Accuracy	96.38%

Future Scope

Space telescopes generates huge amount of astronomical data. Anyone can use these data sets to study various objects in the universe. We have implemented CNN based method to classify the galaxies based on their shape with the accuracy of 96.67%. Our short term future scope is to increase the accuracy as much as possible. We will also work on detection of the shape of very faint objects accurately. We will further implement machine learning algorithm to detect shape and study the internal structure and objects of the galaxy

Conclusion

This system has utilized deep learning capabilities to achieve automatic classification of galaxy based on their shape. This system is based on a simple classification mechanism which exploits the feature extraction functionalities of CNN. For prediction finally, the model utilizes the fully connected layers. The system has achieved an overall 96% testing accuracy on publically accessible dataset,

It is concluded from accuracy that CNN is highly suitable for automatic detection and classification of galaxies.

We have successfully classified the galaxies based on their shape having training and testing accuracy of 96.68% and 96.38% respectively.

References

1. https://irsa.ipac.caltech.edu/data/2MASS/docs/gallery/galmorph/
2. https://www.britannica.com/science/galaxy#ref68104
3. https://imagine.gsfc.nasa.gov/science/objects/galaxies1.html#:~:text=Galaxies%20are%20classified%20by%20shape,and%20a%20central%20%22bulge%22.
4. https://esahubble.org/images/archive/category/galaxies/