Galaxy Morphology Classification based on Convolutional Neural Networks

A project submitted in the partial fullfillment of the requirements for the award of the degree of bachelor of science in Physics.

BY

Name	*Register Number*
Anjitha S M	**2210520**
Lekshmi S	**2210522**
Aadithyn A S	**2210523**

DEPARTMENT OF PHYSICS
MAR IVANIOS COLLEGE(AUTONOMOUS)
THIRUVANANTHAPURAM

(2021-2024)

DEPARTMENT OF PHYSICS
MAR IVANIOS COLLEGE(AUTONOMOUS)
THIRUVANANTHAPURAM

B.Sc. Physics Project Report

Certified that this is record of project work of Anjitha S M (2210520), Lekshmi S (2210522), Aadithyn A S (2210523) during the final year B.Sc. course in Department of Physics.

Head of the Department

Signatures of Examiners

1.

2.

MAR IVANIOS COLLEGE
Autonomous | Committed to Excellence in Higher Education since 1949
Re-accredited (Fourth Cycle) with A+ Grade by NAAC | DST-FIST College | NIRF 2023 - 45th Rank
UGC College with Potential for Excellence | UGC PARAMARSH Mentor College
(*Affiliated to the University of Kerala*)

CERTIFICATE

This is to certify that the project work entitled "**Galaxy Morphology Classification based on Convolutional Neural Networks",** submitted to the Controller of Examination, Mar Ivanios College (Autonomous), Thiruvananthapuram, under University of Kerala in partial requirement for the award of the degree of Bachelor of Science in Physics, is a record of work done by **Anjitha S M, Lekshmi S, Aadithyn A S,** students of Mar Ivanios College(Autonomous), Thiruvananthapuram, during the academic year 2021-2024, under my supervision and guidance.

Ms. Rhea Mary Josi
Assistant Professor
Department of Physics
(Machine Learning)

Place: Thiruvananthapuram
Date:

DECLARATION

We, hereby declare that project work entitled **"Galaxy Morphology Classification based on Convolutional Neural Networks"** is a record of work carried out by us independently, under the supervision and guidance of **Ms. Rhea Mary Josi** , Assistant Professor, Mar Ivanios College(Autonomous), Thiruvananthapuram, during the academic years 2021-2024, in the partial fullfilment of the requirement for the award of the degree of **Bachelor of Science in Physics** and the work has not been submitted for any other degree.

ANJITHA S M

LEKSHMI S

AADITHYN A S

Place: Thiruvananthapuram
Date:

ACKNOWLEDGEMENT

It is with great pleasure that we present our project entitled **"Galaxy Morphology Classification based on Convolutional Neural Networks"**. For this we owe much to many and it is a pleasure to record our gratitude.

We owe our deepest gratitude to our guide **Ms. Rhea Mary Josi**, Assistant Professor, Department of Physics(Machine Learning), Mar Ivanios College (Autonomous), for providing untiring help and relentless support throughout the technical work of this project.

We owe our profound thanks to our Co-Guide **Dr. Sreeja R**, Assistant Professor, Department of Physics, Mar Ivanios College(Autonomous), for her inspiring guidance from the initial to the final level of this work.

We owe our sincere thanks to **Dr. John Jacob**, Head of the Department of Physics, Mar Ivanios College(Autonomous) for his valuable advice throughout the course of this work.

We extent our profound thanks to our Principal Rev. **Fr. Vincy Varghese**, Mar Ivanios College(Autonomous), for the facilities that enabled us to complete this project.

We would also like to record the support and cooperation given to us by our family, friends, teaching and non-teaching staffs of Mar Ivanios College(Autonomous),who helped us directly and indirectly at various levels of this work.

ANJITHA S M

LEKSHMI S

AADITHYN A S

CONTENTS

LIST OF TABLES, FIGURES AND GRAPHS

Chapter 1
AN INTRODUCTION TO GALAXIES

1.1. Introduction

The vast expanse of the universe has intrigued humanity for thousands of years with its enigmatic mysteries. Galaxies, as splendid cosmic islands, are among its remarkable celestial marvels, each encompassing billions or even trillions of stars, planets, and other celestial entities.

Studying the types and properties of galaxies is crucial as it provides vital insights into the origin and evolution of the universe. Galaxy classification plays a key role in understanding the formation of galaxies and assessing our universe. The morphological classification of galaxies involves categorizing them based on their visual features. There are various classification schemes used to group galaxies according to their morphologies. Galaxy classification aids astrophysicists in tackling this challenge by utilizing extensive databases to test theories and draw new conclusions about the physical processes governing galaxies, star formation, and the evolution of the universe.

Traditionally, classifying galaxies involved visually examining two-dimensional galaxy images and labelling them based on their appearance. While human experts are fairly reliable at this, the process is too time-consuming to handle the massive amounts of recent astronomical data, as advancements in telescope and CCD camera technology have led to extremely large image datasets, such as those from the Sloan Digital Sky Survey (SDSS). These datasets are too vast for manual analysis to be practical. Galaxy classification depends on both images and spectra, and achieving accurate classification has been a longstanding goal for astrophysicists. However, the intricate nature of galaxies and the quality of the images have presented significant challenges, making the classification process less accurate.

The galaxy classification system aids astronomers in grouping galaxies based on their visual shapes, with the Hubble sequence being the most well-known. This sequence, developed by Edwin Hubble in 1926, is widely used for galaxy morphological classification. In recent years, advancements in computational tools and algorithms have enabled the automatic analysis of galaxy morphology. Several machine learning methods are being utilized to enhance the classification of galaxy images

Convolutional Neural Networks (CNNs) are a type of deep learning algorithm designed for processing structured grid-like data. They excel at tasks like image recognition, object detection, and natural language processing. CNNs use convolutional layers to automatically and adaptively learn spatial hierarchies of features from input data. This hierarchical feature learning enables CNNs to efficiently capture patterns and relationships within complex data, making them a cornerstone of modern artificial intelligence applications.

The convolution operation is widely used in the computer vision and signal processing community. It plays a significant role in conventional computer vision, particularly in noise reduction and edge detection. The concept of a Convolutional Neural Network (CNN) is not new. In 1998, CNN yielded impressive results in handwritten digit recognition. However, their performance declined significantly due to limitations in memory and hardware, as well as the lack of extensive training data. They struggled to accommodate much larger images. With the substantial advancements in processing power, memory capacity, and the availability of powerful GPUs and extensive datasets, it became feasible to train deeper, larger, and more intricate models. The machine learning Researchers had been working on learning models which included learning and extracting features from images.

1.2. Decrypting the Galaxies

Galaxies, the majestic islands of stars, gas, and dust, dot the vast expanse of the universe, captivating our imagination and pushing the boundaries of our

understanding. These cosmic structures, ranging in size from dwarfs to giants, hold billions to trillions of stars, bound together by gravity.

The Milky Way, our celestial home, is a barred spiral galaxy, adorned with arms of gas, dust, and stars swirling around a central bulge. Within its spiral arms, stellar nurseries give birth to new stars, while older stars twinkle with the wisdom of ages. In its core lies a supermassive black hole, shrouded in mystery and awe.

Beyond our own galaxy lie countless others, each with its own unique characteristics. Elliptical galaxies, like M87, are shaped like a flattened ball, with stars orbiting in random directions. Irregular galaxies, such as the Large Magellanic Cloud, lack a defined shape, often the result of gravitational interactions with other galaxies.

Galaxies come together in clusters, forming cosmic cities where galaxies interact, merge, and dance in gravitational choreography. Some galaxies, like the Andromeda Galaxy, are on a collision course with the Milky Way, destined to merge in a cosmic spectacle billions of years from now.

The study of galaxies, known as galactic astronomy, unlocks the secrets of the universe's past, present, and future. By observing the light emitted by galaxies across the electromagnetic spectrum, astronomers unravel the story of cosmic evolution, from the Big Bang to the present day.

Galaxies serve as cosmic laboratories, providing insights into the nature of dark matter, dark energy, and the fundamental forces that govern the universe. They are the building blocks of the cosmos, shaping the landscape of the universe and seeding the cosmos with the ingredients necessary for life.

1.3. Types of Galaxies[9]

At times, researchers sort galaxies according to their forms and characteristics. Alternatively, they categorize these celestial bodies by the actions occurring in their

central zones, driven by a massive black hole, and the perspectives from which we observe them.

1.3.a. Spiral galaxies

Our home galaxy, the Milky Way, is one type of celestial body called a spiral galaxy. These galaxies are shaped like enormous, spinning pinwheels with a flat disk of stars and a dense cluster of stars at the center.

The spiral arms can be tightly coiled or loosely arranged. Some galaxies' arms are not visible from Earth because we observe them from the side, or edge-on.

Surrounding spiral galaxies are halos, which consist of old stars, star clusters, and dark matter- an invisible substance that does not emit or reflect light but still exerts a gravitational force on other matter. New stars form in the gas-rich arms, while older stars are scattered throughout the disk, bulge, and halo.

The Milky Way and the Andromeda galaxies are both classified as barred spirals, which constitute two-thirds of this category. Barred spirals possess elongated bands of stars, gas, and dust that traverse their centers. Scientists think the presence of a bar indicates that a galaxy has reached full maturity.

Fig 1.1. NGC 2082, Bar spiral galaxy, Constellation swordfish image.

1.3.b. Elliptical Galaxies

The shape of elliptical galaxies varies from round to elliptical. They are much smaller than spiral galaxies. These stars orbit the nucleus in random directions and are generally older than stars in spiral galaxies because there is not enough gas to form new stars. Scientists believe that elliptical galaxies result from collisions and mergers with spiral galaxies.

Fig 1.2. NASA/ESA Hubble Space Telescope image shows an elliptical galaxy called NGC 2865.

1.3.c. Irregular Galaxies

Irregular galaxies have unusual shapes such as toothpicks, rings, and even small stars. They range from irregular dwarf galaxies with 100 million solar masses to giant galaxies with 10 billion solar masses. For example, a spiral galaxy above another galaxy with strong gravity will lose some material, become distorted, and transform into a new shape. Some galaxies, such as gas-rich dwarf galaxies, may be new, formed from material obtained from encounters. Or maybe when galaxies collide they can create a large, oddly shaped mixture. Some scientists suggest that some large regular galaxies may represent an intermediate stage between spiral and elliptical galaxies. Depending on the properties of the star and the composition of the initial galaxy. Regular galaxies also contain large amounts of gas and dust, which are the main ingredients for the formation of new stars.

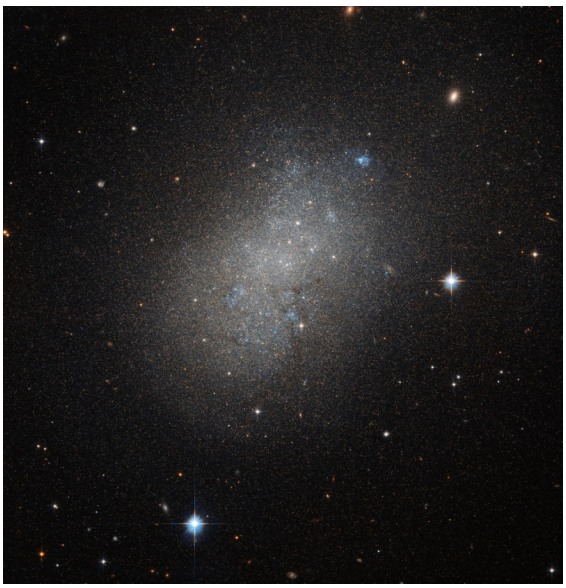


Fig 1.3. This image, courtesy of the NASA/ESA Hubble Space Telescope's Advanced Camera for Surveys (ACS), captures the glow of distant stars within NGC 5264, a dwarf galaxy located just over 15 million light-years away in the constellation of Hydra (The Sea Serpent).

1.3.d. Lenticular Galaxies

Lenticular galaxies are a mixture of spiral and elliptical galaxies. They have the central bulge and disk seen in spiral galaxies, but do not have spiral arms. However, like elliptical galaxies, lenticular galaxies also have old stars and regular stars. According to one view, these galaxies are old spiral galaxies whose spiral arms have disappeared. Another idea is that lens-like structures were formed by the merger of spiral galaxies.

Fig 1.4. Lenticular galaxy NGC 4886, imaged here by the Hubble Space Telescope, ESA/Hubble & NASA. Acknowledgement: Gilles Chapdelaine

1.3.e. Active galaxies

About 10% of galaxies are known to be active; This means that their centers are 100 times brighter than the total light of the stars. They can be spiral, oval or irregular. The Milky Way is not currently an active galaxy, but it may have experienced a burst of activity in the last few million years. Black holes have masses ranging from hundreds of thousands to billions of times that of the Sun. The black hole's gravity compresses and heats the accretion disk, causing the object to emit light of various wavelengths, from infrared to X-rays. The ring of blocks, called a torus, is several light-years in diameter. Near the black hole, a small portion of the lost gas may be pushed out of the disk, creating a jet of material travelling at close to the speed of light. Properties and behaviour of active galaxies. Scientists now believe that observing the centers of galaxies from different angles (for example, looking directly at the torus instead of looking from the side) gives rise to many drawings. Distribution. Radio noisy galaxies are often emitted by accretion disks and jets. Radio-quiet galaxies tend to emit little or no jets. Visible brightness is also considered as another part of our imagination. The aircraft is oriented more towards our line of sight and appears brighter and more varied when viewed from an angle than when viewed “down the barrel”,appear brighter and more variable than those viewed at wider angles.

Fig 1.5. Resembling a swirling witch's cauldron of glowing vapours, the black hole-powered core of a nearby active galaxy appears in this colourful NASA/ESA Hubble Space Telescope image.

1.3.f. Seyfert Galaxies

First discovered by American astronomer Carl Seyfert in 1943, Seyfert galaxies are the most active galaxies and also have the lowest energy. All Seyfert galaxies resemble normal galaxies in visible light, but they emit more infrared radiation. When seen in the infrared, some show bright emission from the ring-shaped donut. Some also emit X-rays. Seyfert galaxies tend to have low radio frequencies, although some form radio jets. Type I Seyfert galaxies exhibit unusual features in the visible spectrum, indicating that they move quickly near the bounty. Type II Seyferts show features reflecting slow motion. However, this difference may be due to differences in the positions of the observed galaxies.

Fig 1.6. The Hubble Space Telescope captured this image of spiral galaxy NCG 5728, which is also a Seyfert-type active galaxy. ESA/Hubble, A. Riess et al., J. Greene

1.3.g. Quasars

Quasars are the brightest type of active galaxies. They emit light across the electromagnetic spectrum, creating powerful particle jets that emit thousands of times the energy emitted by galaxies such as the Milky Way. The nearest quasar, called Markarian 231, is about 600 million light-years away, but the farther we look, the more quasars we see. The most distant quasars currently known are 13 billion light-

years away. Because light takes time to travel, scientists can use light from galaxies to look back in time to study the growth of black holes and the evolution of galaxies. The merging of galaxies in the young universe would have resulted in the enormous energy of quasars, but once the feeding frenzy was over, the black hole could not sustain it. It is thought that quasar activity may be interrupted and the entire phase will only last about 10 million years.

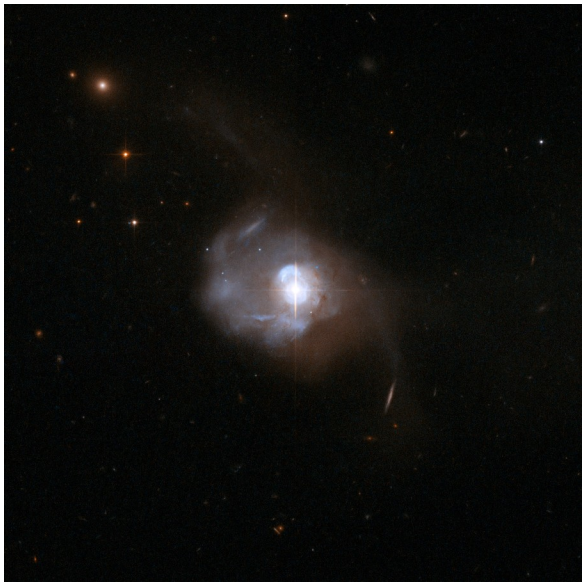


Fig 1.7. This Hubble Space Telescope image reveals a bright starlike glow in the center of active galaxy Markarian 231, the nearest quasar to Earth. NASA, ESA, the Hubble Heritage Team (STScI/ AURA)-ESA/Hubble Collaboration, and A. Evans (University of Virginia,Charlottesville/NRAO/Stony Brook University)

1.3.h. Blazars

Blazars produce light in the electromagnetic spectrum. They appear brighter than other active galaxies because their powerful jets are directed almost directly toward Earth. Observatories on Earth can capture high-energy particles (such as neutrinos) formed in space and follow them back to their home galaxies. This information allows scientists to see the environment around the Blazar supermassive black hole.

Fig 1.8. Markarian 421, imaged here by the National Science Foundation's Sloan Digital Sky Survey, is one of the closest blazars to Earth. Sloan Digital Sky Survey

1.4. Galaxy Classification Parameters

Galaxy classification is a fascinating field of astronomy that allows astronomers to classify and understand the massive galaxies in the universe. Various parameters are important for this classification, including shape, brightness, colour and spectral properties.

Morphology, or the shape and structure of a galaxy, is one of the most important classifications. Created by Edwin Hubble in 1926, the Hubble sequence is the most commonly used sequence to classify galaxies based on their morphology. He divided galaxies into three main types: elliptical, spiral and irregular. Elliptical galaxies are smooth and featureless and can come in many sizes and shapes, from nearly spherical to very elongated. Spiral galaxies have a central bulge surrounded by spiral arms filled with stars, gas, and dust, while irregular galaxies are undifferentiated and often appear chaotic.

Luminosity, or the total amount of light emitted by a galaxy, is another important factor for classification. Galaxies vary in brightness from faint dwarf galaxies to

bright giant galaxies. Luminosity is usually measured in magnitudes that represent the brightness of a galaxy at a standard distance from Earth.

Colour is important to understand the stars in the galaxy. The colour of the galaxy is affected by the type of stars it contains and the amount of dust and gas. For example, galaxies dominated by young, hot stars tend to appear blue, while galaxies dominated by old, cooler stars appear redder. The image matches colours to the brightness of the galaxy and can provide insight into the galaxy's evolutionary history and star formation.

Spectral features, including emission and absorption lines in the galaxy's spectrum, provide important information about the galaxy's composition, temperature, and motion. By analysing the spectra of galaxies, astronomers can determine things like chemical composition, star formation rates, and the presence of active galactic nuclei.

In addition to these, galaxies can also be classified based on their surroundings, such as whether they are part of a galaxy cluster or a separate group. Understanding galaxy distribution is not necessary to reveal the complex processes that form the universe and the evolution of galaxies over cosmic time.

1.5. Galaxy Morphologies

Galaxies are classified into three main morphological types: elliptical, spiral, and irregular.

1. Elliptical galaxies: These galaxies have a smooth and nearly featureless appearance. They are classified according to their elongation, ranging from E0 (nearly spherical) to E7 (highly elongated).

2. Spiral galaxies: These galaxies have a disk-like structure with spiral arms. They are further classified into two subtypes:

a. Normal spirals (S): These galaxies have well-defined spiral arms and a central bulge. They are further divided into Sa, Sb, and Sc based on the tightness of their spiral arms and the size of their central bulge.

b. Barred spirals (SB): These galaxies have a central bar-shaped structure, around which spiral arms wrap. They are classified similarly to normal spirals, with subtypes SBa, SBb, and SBc.

3. Irregular galaxies: These galaxies lack a distinct shape and do not fit into the categories of elliptical or spiral galaxies. They can be further divided into two types:

a. Irr I: These are irregular galaxies with some structure.

b. Irr II: These are irregular galaxies that appear more chaotic and lack any apparent structure.

This classification system, known as the Hubble sequence, is based on the work of Edwin Hubble and is still widely used today.

1.6. Why do we classify Galaxies?

Galaxies are the building blocks of our universe and offer a surprising array of shapes, sizes and structures. To understand this cosmic beauty, astronomers use the powerful tool of galaxy classification. This project report explores the main reasons why we classify galaxies and the information this provides about our universe.

First of all, galaxy classification provides the basic framework for organizing the many galactic forms found in the universe. By classifying galaxies according to morphological properties such as shape, symmetry and composition, astronomers can explore the complex landscape of cosmic diversity. This classification system,

developed by Edwin Hubble in the early 20th century, provides the basis for understanding evolutionary paths and relationships between different types of galaxies.

Galaxy classification is an important tool for studying galaxy evolution. By studying the distribution and properties of galaxies in different cosmic epochs, astronomers can trace the history of cosmic structure and reveal the processes that drove galaxy evolution. For example, the expansion of elliptical galaxies in dense space suggests that interactions such as mergers and tidal interactions play an important role in shaping galaxy morphology over time.

Galaxy classification helps identify and study specific groups of galaxies that have specific properties and behaviours. For example, active galaxies formed by the presence of heavy electrons emitted by supermassive black holes at their centers represent special objects that may affect our understanding of the processes of black holes and galaxy-black hole mergers. By classifying active galaxies according to their spectral signatures and emission strengths, astronomers can reveal the underlying processes that drive these cosmic giants.

Galaxy classification forms the basis of observational astronomy and guides the creation and interpretation of astronomical research and observations. By targeting specific galaxies in different regions of the cosmic field, astronomers can model the various galactic environments and conditions that exist on Earth. This mission allows scientists to answer important questions about galaxy formation, stellar evolution and cosmology with precision and accuracy.

In addition, galaxy classification facilitates collaboration and communication within the astronomy community, providing a common language and framework for sharing observations, theories, and discoveries. By following a standardized classification system, astronomers can improve our understanding of the world by sharing their

findings and insights internationally and across disciplines.

Galaxy classification in general plays an important role in our exploration of the mysteries of the universe. By arranging the kaleidoscope of galaxy images into unified groups, classification allows us to clearly and precisely explore the evolutionary history, physical structure and cosmic significance of galaxies. As we continue to explore the depths of space and time, galaxy classification remains an important tool for unraveling the cosmic fabric and uncovering the secrets of our celestial environment.

Chapter 2
METHODS OF CLASSIFICATION

Classification techniques in machine learning are simple techniques used to divide co ntent into different classes or groups based on their characteristics. This technique pla ys an important role in many applications such as spam detection, medical diagnosis, sentiment analysis and image recognition.

One of the main goals of classification algorithms is to learn how to map from input devices to corresponding lists. This process involves training a model on a data set where each data point is associated with a known class name. After training, the mod el can predict the list of invisible objects based on the learning model.

There are several classification methods in machine learning, each with its own advantages, disadvantages, and specific applications:

2.1. Support Vector Machines

Support vector machines (SVMs) classify data by finding hyperplanes that allow clas ses to be separated. They are efficient at high speed and control lines and are not sepa rated from the core process. The goal of SVM is to minimize classification error whil e maximizing margin, be robust to overprocessing, and optimize for missing data.
The Regularization parameter C measures generalization and training error.
SVM is widely used in many binary and multi-class operations, including text classifi cation, image recognition, and bioinformatics. SVMs are still popular due to their hig h efficiency, high performance, and ability to solve complex decision problems.

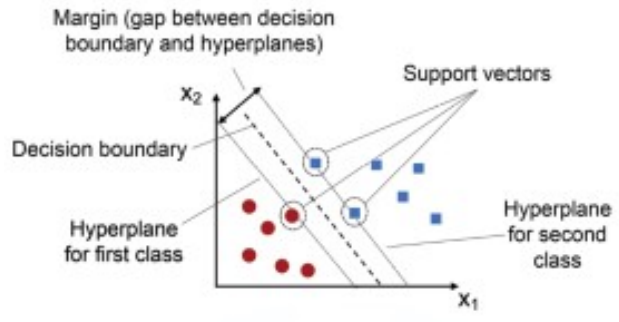


Fig 2.1. SVM

2.2. K Nearest Neighbors

The K-nearest neighbor (KNN) algorithm is a simple and effective machine learning technique used for classification and regression. It works by finding K similar events (nearest neighbors) for new ideas and then using their texts or values to make predicti ons. The main idea is that a sample will be similar to its neighbors and therefore the labels or values of the neighbors can be used to make predictions. KNN is a non para metric lazy learning algorithm; This means it requires no training or assumptions abo ut data classification and only needs to store training data. This makes it a flexible and powerful algorithm widely used in applications such as classification of images and texts by recommendations and time estimation.

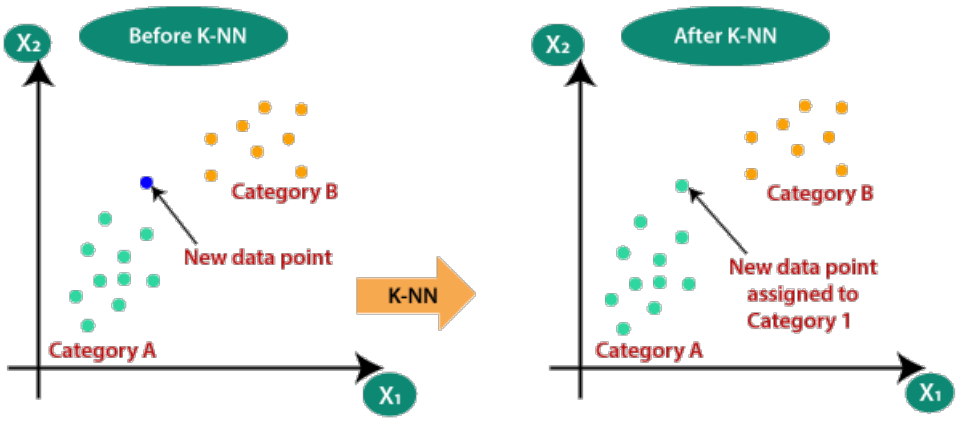


Fig 2.2. K-NN

2.3. Decision Tree

A decision tree is one of the maximum powerful tools of supervised gaining knowledge of algorithms used for both category and regression obligations. It builds a flowchart-like tree structure where every inner node denotes a take a look at on an characteristic, every branch represents an final results of the take a look at, and each leaf node (terminal node) holds a class label. It is built via recursively splitting the training data into subsets primarily based at the values of the attributes till a preventing criterion is met, together with the most intensity of the tree or the minimum quantity of samples required to break up a node.

During training, the decision tree algorithm selects the best features to classify the

data based on metrics such as entropy or Gini impurity, which measures the level of impurity or randomness in a subset. The goal is to find the attribute that maximizes the information gain or the reduction in impurity after the split.

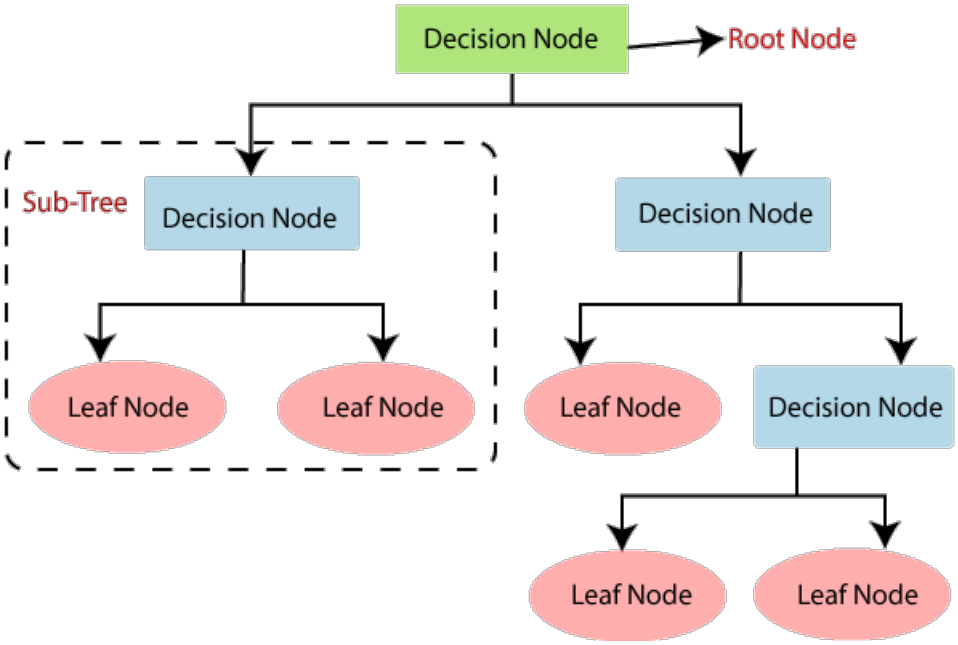


*Fig 2.3. **Decision Tree***

2.4. Random Forest

Random forest is an ensemble learning algorithm that combines multiple decision trees to increase the accuracy and robustness of predictions. It works by creating a collection of decision trees, each training on different data and features. The algorithm then combines the predictions from each tree to make a final prediction. Random forest reduces overfitting and improves generalization by averaging the errors and biases of individual trees. They also effectively handle high dimensional data and missing values. Random forests are widely used in applications such as image classification, natural language processing, and model prediction, and are often preferred to decision trees due to the efficiency and effectiveness of translation. They provide feature important sources, which help in feature selection and understanding the data.

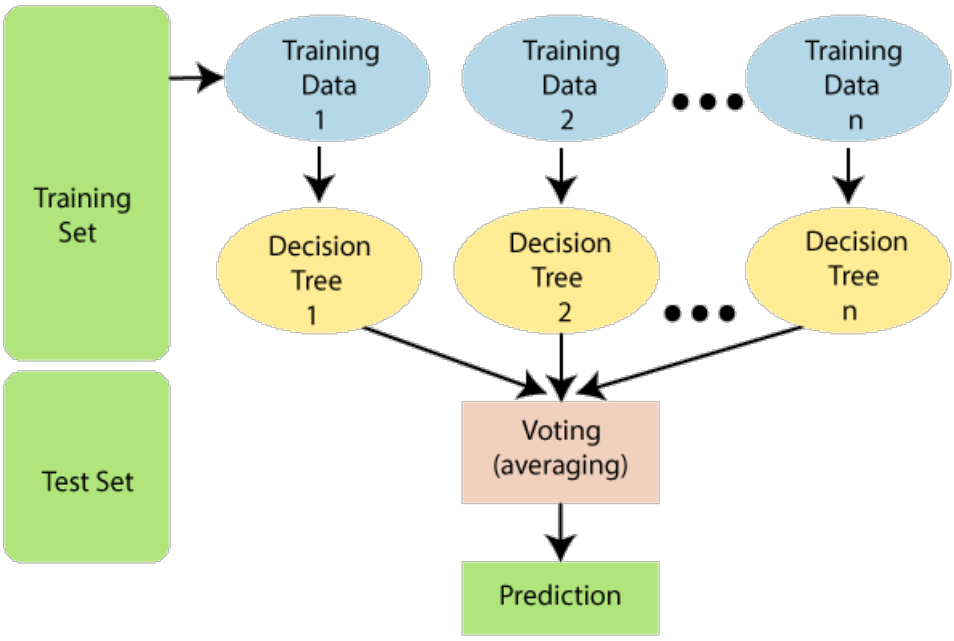


Fig 2.4. Random Forest

2.5. Logistic Regression

Logistic Regression is a statistical method used for the distribution of binary function s.The goal here is to estimate the probability that an input will fall into one of the two classes. It models the relationship between a variable (binary value) and one or more individual variables (features) by estimating the probability using logistic or sigmoid function. This function limits the output to the range 0 and 1, making it suitable for re presenting the result.Unlike linear regression, Logistic Regression applies a transfor mation to the linear combination of input features, ensuring prediction lie within the valid probability range.

The model's coefficients are optimized through techniques such as gradient descent, which evaluate the difference between predictions and class maps. Logistic regressio n is interpretable and can provide insight into the impact of individual characteristics on outcomes. It has applications in many fields, including medicine, finance, and bus iness.

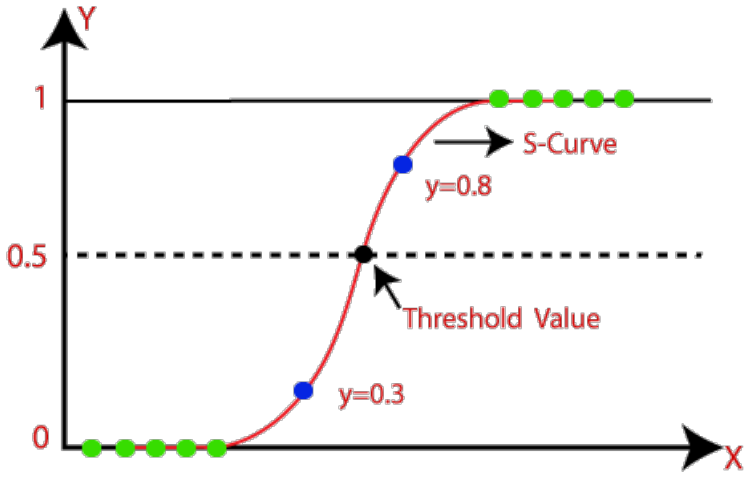


Fig 2.5. Logistic Regression

2.6. Naive Bayes

Naive Bayes stands as a straightforward yet potent machine learning algorithm groun ded in Bayes' theorem, estimating the likelihood of a hypothesis given prior informat ion. Its efficiency and simplicity stem from the assumption of feature independence. This model finds extensive application in text classification, sentiment analysis, and spam detection, demonstrating notable effectiveness despite its basic nature. Its advan tages include adeptness in managing high-dimensional data, robustness to noise and absent values, and the provision of probabilistic forecasts. While it may not surpass the performance of more intricate algorithms, Naive Bayes remains a favoured option across various domains due to its user-friendliness, interpretability, and consistency.

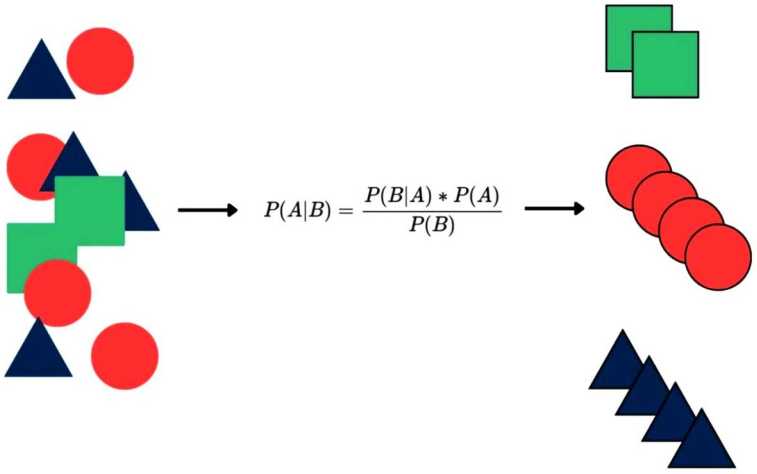


Fig 2.6. Naive Bayes

2.7. Neural Networks

[3] Neural networks are the combination of artificial intelligence and brain-inspired design that is reshaping modern computing. These networks are highly effect ive in machine learning by simulating the complex functions of the human brain thro ugh a complex set of neural networks. There are many types of neural networks, from feedforward to recurrent to convolutional, and each has its own tasks.

Following are the types of Neural Networks:

2.7.a. Artificial Neural Networks (ANN)

Artificial Neural Networks (ANN) are computational models inspired by the structure and function of the human brain. They consist of interconnected nodes arranged in layers, with information flowing through the network and undergoing transformation via weighted connections and activation functions. ANNs learn complex patterns from data through iterative training, with deep learning, a subset of ANNs, employing multiple hidden layers for hierarchical feature learning. ANNs are highly effective in tasks such as image recognition, natural language processing, and speech recognition, where they excel by capturing intricate data relationships. Despite their computational demands and data requirements, ANNs continue to drive significant advancements in artificial intelligence.

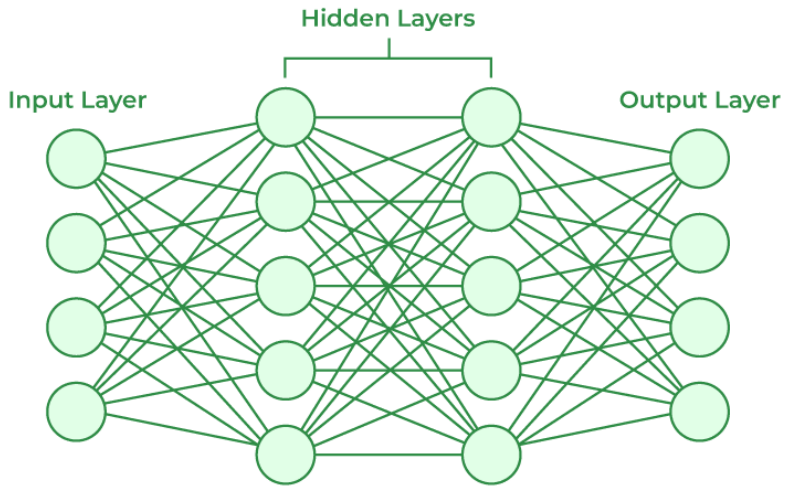


Fig 2.7. ANN

2.7.b. Convolutional Neural Network(CNN)

CNN is mainly used for image data. It is used in computer vision. Some real-life appl ications are sensing in autonomous vehicles. It has a combination of convolutional tec hniques and neurons. It is stronger than ANN and RNN.

Convolutional Neural Networks (CNNs) are specialized artificial neural networks tailored for processing structured grid-like data, notably images. Drawing inspiration from the organization of the visual cortex in animals, CNNs have significantly advanced image recognition tasks.

These networks comprise multiple layers, including convolutional layers, pooling layers, and fully connected layers. Convolutional layers use small filters or kernels to extract features by convolving across the input image. Subsequently, pooling layers downsample feature maps, reducing computational complexity while preserving crucial information. Finally, fully connected layers aggregate these features for classification or regression.

CNNs excel due to their capacity to autonomously learn hierarchical feature representations from raw data. Through extensive training on large datasets, CNNs discern patterns at varying levels of abstraction, from basic edges and textures to intricate object shapes.

Beyond image recognition, CNNs find application in natural language processing, analyzing text sequences through methods like word embeddings and attention mechanisms. They also play pivotal roles in medical imaging for disease diagnosis, object detection in autonomous vehicles, and object manipulation in robotics.

Although CNNs deliver impressive results, their training demands substantial computational resources. Designing optimal architectures often involves a blend of empirical experimentation and domain expertise. Nonetheless, their remarkable

performance and adaptability continue to propel advancements across diverse realms of artificial intelligence.

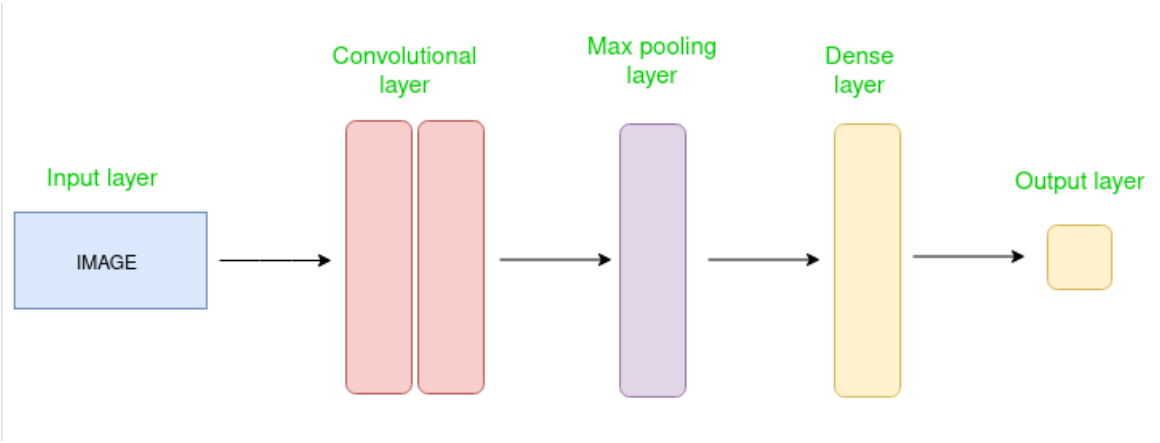


Fig 2.8.CNN

2.8. Why CNN over others?

Convolutional Neural Networks (CNNs) excel in image recognition tasks due to several advantages:

- Hierarchical feature learning: CNN automatically learns hierarchical representations of features by capturing low-level features such as edges and high-level features such as shapes and objects. This layered representation allows them to understand complex patterns in images.

- Spatial Invariance: Thanks to convolution and pooling layers, CNNs can capture patterns regardless of their location in the image. These properties make them resistant to translation and distortion, which is important for tasks such as object detection and image classification.

- Parameter Sharing: CNNs use parameter sharing to apply weighting techniques to different parts of the input. This reduces the number of errors compared to the full network, making CNNs more efficient and powerful, especially for large images.

- Local connectivity: CNNs exploit the location of features in the image by connecting each neuron to a local area of the container. This local connection allows CNN to take advantage of location and structure.

- Pre-learning models and transfer learning: CNNs trained on large datasets such as ImageNet can be used as custom objects for a variety of tasks. By fine-tuning models on small datasets before training them, CNNs can achieve high performance using less data, making them especially useful in areas where models are less trained.

- Data augmentation: CNNs can be enhanced by methods such as rotating, rotating, and scaling to optimize the size of the training data. This helps prevent overfitting and improve generalization to unobserved data.

- Availability of frameworks and hardware acceleration: There are many deep learning programs (e.g. TensorFlow, PyTorch) that provide the performance of CNNs. Additionally, hardware acceleration technologies such as GPU and TPU can accelerate the training and inference of CNN, making it suitable for practical applications.

Chapter 3
DATASET AND MODEL IMPLEMENTATION

3.1. Literature Survey

- [i] **Deep Galaxy: Classification of Galaxies based on Deep Convolutional Neural Networks**

Nour Eldeen M. Khalifa, Mohamed Hamed N. Taha, Aboul Ella Hassanien, I. M. Selim

The paper "Deep Galaxy: Classification of Galaxies based on Deep Convolutional Neural Networks" introduces a deep convolutional neural network architecture for classifying galaxies into three main categories: Elliptical, Spiral, and Irregular. The study addresses the significance of galaxy classification in understanding the formation and evolution of the universe, particularly in the context of the Hubble sequence developed by Edwin Hubble in 1926. It discusses the limitations of prior classification methods and highlights the potential of deep learning techniques in automating and improving the accuracy of galaxy classification. The proposed architecture, consisting of 8 layers with a main convolutional layer for features extraction and two fully connected layers for classification, achieved a testing accuracy of 97.27%, outperforming previous works in the field. The dataset used in the research comprises over 13,000 images of galaxies from the EFIGI catalog, with a focus on Elliptical, Spiral, and Irregular galaxy types. The paper provides a detailed explanation of the proposed deep galaxies CNN architecture, the experimental results, and comparative analysis with other related works, demonstrating the effectiveness of the approach

The paper highlights the historical challenges of visually inspecting and categorizing galaxies due to the time-consuming nature of the process, especially with the increasing size of astronomical databases. It discusses the evolution of convolutional neural networks (CNNs) and the significant impact of deep learning in visual detection and recognition, emphasizing the ability to utilize raw data images without the need for expert knowledge in optimization or feature design. The proposed architecture's experimental results demonstrate its high accuracy in classifying

galaxies, with a median accuracy of 97.27% over five different runs. The study concludes by emphasizing the increasing interest in galaxy classification and the potential of deep convolutional neural networks in addressing this research area, paving the way for future advancements in the field.

This contains a comprehensive overview of the challenges and significance of galaxy classification, introducing a novel deep convolutional neural network architecture for automating and improving the accuracy of classifying galaxies. It provides detailed insights into the dataset characteristics, the proposed architecture, experimental results, and comparative analysis with prior works, establishing the effectiveness of the approach in achieving high testing accuracy. The study contributes to the growing body of research leveraging deep learning techniques in the field of astrophysics, particularly in the automated analysis of galaxy morphology, and sets the stage for further advancements and applications in this domain.

- [ii] **Morphological Galaxy Classification Using Machine Learning**

Siddhartha Kasivajhula, Naren Raghavan, and Hemal Shah

The paper "Morphological Galaxy Classification Using Machine Learning" compares the performance of three machine learning (ML) algorithms - Support Vector Machines (SVM), Random Forests (RF), and Naïve Bayes (NB) - in classifying galaxies based on their morphology. The motivation behind this research stems from the manual classification method being slow, error-prone, and inadequate for handling the explosion of data in astronomy, particularly from programs like the Sloan Digital Sky Survey. The study explores the effectiveness of morphic features derived from image analysis and PCA features compressed from direct image pixel data. The authors detail the system architecture, including image preprocessing, feature extraction, and classification stages, and present experimental results, concluding that RF outperformed SVM and NB, and that morphic features were more effective than PCA features. The paper also outlines future work, suggesting areas for

improvement in classification accuracy, such as integrating photometric features and implementing a staged classification approach.

This begins by highlighting the challenges in manual galaxy classification, which is slow, error-prone, and unable to cope with the increasing volume of astronomical data. The authors identify the need for machine learning algorithms to automate the classification process, given the substantial data generated by programs like the Sloan Digital Sky Survey. The study aims to assess the performance of different ML algorithms - SVM, RF, and NB - using morphic and PCA features extracted from galaxy images. The paper provides a detailed overview of the system architecture, which involves image preprocessing, feature extraction, and classification stages, thus setting the stage for the experiments and results presented.

The experimental results demonstrate that RF outperformed SVM and NB in classifying galaxies based on morphic and PCA features. The study found that morphic features were more effective than PCA features, and the optimal number of PCA features was determined to be between 8 and 24. The conclusion also suggests future work to enhance classification accuracy, such as incorporating more training data, integrating photometric features with morphic features, and implementing a staged classification approach. The paper provides a comprehensive overview of the experimental results, comparing the mean accuracies and standard deviations of the classifiers, and discusses the implications of the findings. Additionally, the study references previous research and acknowledges the input from astronomers directly involved in the Sloan Digital Sky Survey, emphasizing the relevance and significance of their work in the field.

The paper offers a thorough investigation into the use of machine learning algorithms for morphological galaxy classification, highlighting the superiority of RF over SVM and NB, and the effectiveness of morphic features compared to PCA features. The authors provide valuable insights into the challenges and opportunities in automating galaxy classification, paving the way for future research directions to further enhance classification accuracy.

3.2. Dataset Characteristics

This project is based on morphological classfication using Convolutional Neural Network. It shows how computational cosmology could help to make hard classificaiton easy. The galaxy can be classified in various ways, they can be in three classes namely - Elliptical, Spiral, Irregular or they could be classified more deeper as Disk, Smooth, Bulge, Spiral etc.

This project uses **astroNN** dataset, This dataset is generated from **Sloan Digital Sky Survey** , and the labels comes from **Galaxy Zoo**. We can even preprocess Galaxy Zoo and can convert it into three main labels - Elliptical, Spiral, Irregular. Galaxy10 SDSS is a dataset contains 21785, 69x69 pixels colored galaxy images (g, r and i band) separated in 10 classes. AstroNN dataset contains images for 10 different labels namely - Galaxy10 dataset (21785 images).

├── Class 0 (3461 images): Disk, Face-on, No Spiral

├── Class 1 (6997 images): Smooth, Completely round

├── Class 2 (6292 images): Smooth, in-between round

├── Class 3 (394 images): Smooth, Cigar shaped

├── Class 4 (1534 images): Disk, Edge-on, Rounded Bulge

├── Class 5 (17 images): Disk, Edge-on, Boxy Bulge

├── Class 6 (589 images): Disk, Edge-on, No Bulge

├── Class 7 (1121 images): Disk, Face-on, Tight Spiral

├── Class 8 (906 images): Disk, Face-on, Medium Spiral

├── Class 9 (519 images): Disk, Face-on, Loose Spiral

These classes are mutually exclusive, but Galaxy Zoo relies on human volunteers to classify galaxy images and the volunteers do not agree on all images. For this reason, Galaxy10 only contains images for which more than 55% of the votes agree on the

class. That is, more than 55% of the votes among 10 classes are for a single class for that particular image. If none of the classes get more than 55%, the image will not be included in Galaxy10 as no agreement was reached. As a result, 21785 images after the cut.

The proposed ConvNet galaxy arcitecture consists of one input layer having 16 filters, followed by 4 hidden layers, 1 penultimate dense layer and an Output Softmax layer. Also included data augmentation such as shear, zoom, rotation, rescaling, and flip. ‘tanh’ activation function is used here ‘relu’ is an alternative. The dataset could also be generated manually using Hubblesite collection and other similar digital surveys and then after preprocessing can be used for training.

3.3. Implementation Plan

- Collecting the dataset: The dataset is obtained from astroNN generated from [2]Sloan Digital Sky Survey , and the labels comes from Galaxy Zoo. Source: https://astronn.readthedocs.io/en/stable/galaxy10sdss.html
- Data Preprocessing: Preprocess the images by resizing them to a consistant size, normalizing pixel values and augmenting the dataset to increase variability.The original images are 424x424, but were cropped to 207x207 centered at the images and then downscaled 3 times via bilinear interpolation to 69x69 in order to make them manageable on most computer and graphics card memory.
- Building a model: The model is a CNN model that classifies the galaxies.
- Training the model: Split the dataset into training, validation and test sets. Train the CNN model on the training set.
- Testing the model: The model was tested where the model classified the galaxies in the testing set.

  Visualize the confusion matrix to analyse the models classification performance across different morphological classes.

BLOCK DIAGRAM

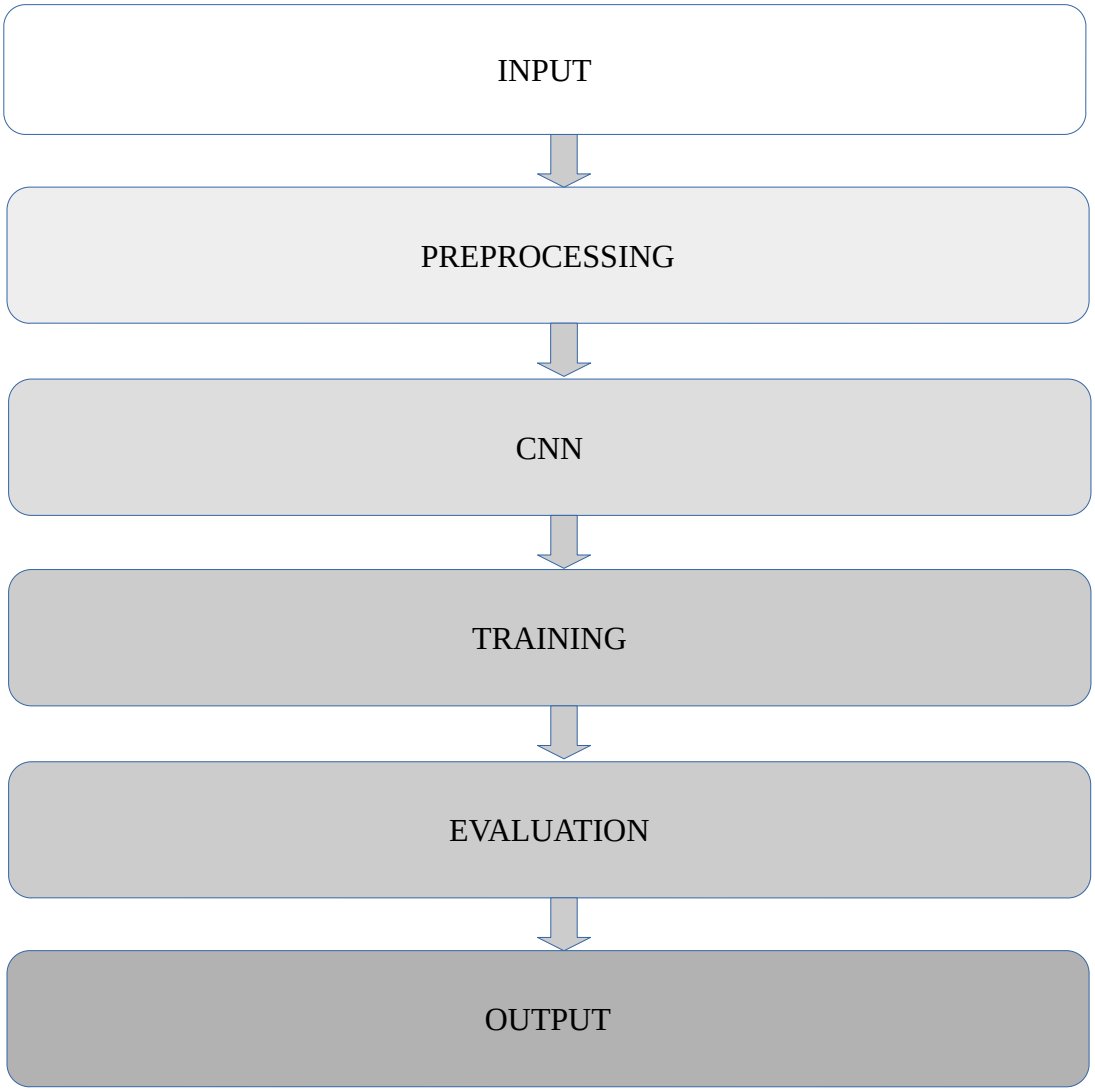


Fig 3.1. Block Diagram of Implementation plan

3.4. Libraries, Packages, Modules and Functions used

- CV2

The OpenCV library, often abbreviated as cv2, is a powerful tool for computer vision tasks in Python. It provides a wide range of functions for image and video processing, including object detection, image transformation, and feature extraction. With its extensive documentation and active community support, OpenCV is a valuable resource for both beginners and experts in the field of computer vision.

- MATH

The math module in Python provides essential mathematical functions. It includes

functions for basic arithmetic operations, trigonometric calculations, logarithms, and more. By leveraging the math module, Python programmers can perform a wide range of mathematical tasks efficiently and accurately.

- NUMPY

The NumPy module is a fundamental package for scientific computing in Python. It provides support for multi-dimensional arrays and matrices, along with a wide range of mathematical functions to operate on these arrays efficiently. With its powerful array manipulation capabilities, NumPy is essential for tasks such as data manipulation, linear algebra, statistical operations, and more in the Python ecosystem.

- MATPLOTLIB

Matplotlib is a versatile Python library for creating static, interactive, and animated visualizations. With its intuitive interface and extensive functionality, it's widely used in scientific computing, data analysis, and machine learning. Whether you're plotting simple line charts or complex 3D visualizations, Matplotlib provides the tools to bring your data to life.

- TENSOR FLOW

TensorFlow is a powerful open-source library for machine learning and deep learning. It provides a flexible and efficient framework for building and training various types of neural network models. With its extensive documentation, rich ecosystem, and support for both CPU and GPU computation, TensorFlow is widely used by researchers and practitioners in the field of artificial intelligence.

- UTILS

The utils module in scikit-learn provides a collection of utility functions to support various machine learning tasks. It includes functions for data manipulation, preprocessing, and model evaluation. By leveraging the functionalities within the utils module, users can streamline their workflow and enhance the efficiency of their machine learning pipelines.

- SK LEARN

Scikit-learn is a powerful Python library for machine learning. It offers simple and

efficient tools for data mining and data analysis. With its extensive documentation and user-friendly interface, it's an excellent choice for both beginners and experienced machine learning practitioners.

- SK LEARN METRICS

Scikit-learn's metrics module offers a comprehensive set of tools for evaluating the performance of machine learning models. It includes functions for calculating various metrics such as accuracy, precision, recall, F1-score, and more. These metrics enable users to assess the effectiveness and reliability of their models across different tasks and datasets.

- CONFUSION MATRIX DISPLAY

Scikit-learn provides functions to generate and display confusion matrices, which are helpful for evaluating the performance of classification models. By using the confusion_matrix function, users can calculate the matrix, and then visualize it using tools like matplotlib. This visualization helps to understand the distribution of true positive, true negative, false positive, and false negative predictions made by the model.

- CONFUSION MATRIX

The confusion matrix in scikit-learn provides a detailed summary of the performance of a classification model. It tabulates the counts of true positive, true negative, false positive, and false negative predictions made by the model. This matrix serves as a valuable tool for understanding the strengths and weaknesses of the classifier across different classes.

- SKLEARN MODEL SELECTION

The sklearn.model_selection module in scikit-learn provides tools for model selection and evaluation. It includes functions for cross-validation, train-test splitting, and parameter tuning using techniques like grid search and randomized search. By utilizing this module, practitioners can efficiently assess the performance of machine learning models and choose the best ones for their tasks.

- TRAIN TEST SPLIT

The train_test_split module in scikit-learn is essential for splitting datasets into

training and testing sets. By randomly dividing the data, it enables practitioners to assess the performance of machine learning models effectively. This module simplifies the process of creating robust and reliable models by providing a standardized approach to data partitioning.

- KERAS

Keras is a high-level neural networks API written in Python. It provides a user-friendly interface for building, training, and deploying deep learning models. With its simplicity and flexibility, Keras has become a popular choice among researchers and practitioners for rapid prototyping and experimentation in the field of artificial intelligence.

- KERAS LAYERS

Keras layers are the building blocks of neural networks in Keras, a high-level deep learning library. These layers enable users to construct complex neural network architectures by stacking them together. With a wide range of pre-defined layers for different purposes, users can easily customize and design models tailored to their specific tasks and datasets.

- KERAS MODULES

The Keras models module is a fundamental component of the Keras deep learning library. It provides a high-level interface for building and training deep learning models, allowing users to easily create neural networks using simple, intuitive APIs. With models, users can construct various types of models, including sequential models for simple architectures and functional models for more complex networks with shared layers or multiple inputs/outputs.

- KERAS CALL BACKS

The Keras Callback module is a powerful tool for customizing and monitoring the training process of neural networks. It allows users to define and implement functions that can be called at various stages during training, such as at the start or end of an epoch, before or after a batch, or when a certain performance threshold is reached. With Callbacks, users can implement techniques like early stopping, learning rate scheduling, and model checkpointing to improve training efficiency and performance.

- TENSORFLOW. KERAS. PREPROCESSING. IMAGE

The TensorFlow.keras.preprocessing.image module is a powerful tool for working with image data in deep learning projects. It provides utilities for loading, preprocessing, and augmenting image datasets seamlessly within the Keras API. With functions for tasks like resizing, rescaling, and data augmentation, this module streamlines the process of preparing image data for training convolutional neural networks.

- IMAGE DATA GENERATOR

The ImageDataGenerator module in Keras offers a powerful way to augment image data for deep learning tasks. By applying various transformations such as rotation, flipping, and shifting, it helps increase the diversity of training data, leading to more robust models. This module streamlines the process of preprocessing image data on-the-fly, saving time and memory during model training.

- CONV2D

In Keras, Conv2D (Convolutional 2D) is a layer that performs a convolutional operation on an input image, extracting features by scanning small regions of the image with a set of learnable filters. It is a crucial component in building Convolutional Neural Networks (CNNs) for image processing and computer vision tasks. The Conv2D layer takes three main arguments: the number of filters, the kernel size, and the activation function. It applies the filters to the input image, generating feature maps that capture local patterns and spatial hierarchies, ultimately enabling the network to learn rich and robust representations of the input data.

- MAXPOOL 2D

In Keras, MaxPooling2D is a layer that performs max pooling, a downsampling operation, on an input image or feature map. It reduces spatial dimensions to reduce the number of parameters and computation, and to help prevent overfitting. MaxPooling2D takes two main arguments: pool_size and strides, which define the size of the pooling window and the step size for sliding the window, respectively. It scans the input image, selecting the maximum value within each window, and outputs a feature map with reduced spatial dimensions, retaining only the most prominent

features. This helps to retain important information while reducing the dimensionality of the data.

- FLATTEN

The flatten module in Keras is a convenient way to convert multi-dimensional input data into a one-dimensional array. It serves as a preprocessing step before feeding the data into a fully connected neural network layer. By flattening the input, it ensures compatibility with the subsequent layers in the neural network architecture.

- DENSE

With its dense module, users can easily create fully connected neural networks by stacking layers of neurons, enabling them to tackle a wide range of tasks, from image classification to natural language processing.

- PROPOUT

The propout module in Keras is a regularization technique that aims to improve the generalization performance of neural networks. It randomly sets a fraction of input units to zero during training, similar to dropout, but with a twist: it only applies to the input layer. By selectively dropping input units, propout helps prevent overfitting and encourages the network to learn more robust features from the data.

- SEQUENTIAL

The Sequential module in Keras provides a simple way to build and train neural networks. It allows users to add layers to the model sequentially, one after the other. This intuitive approach makes it easy to create feedforward networks where the output of each layer serves as the input to the next layer

- LOAD MODEL

The Keras library provides a convenient module for loading pre-trained models. Using the load_model function, you can easily load a model saved in the Hierarchical Data Format (HDF5) format. This allows you to quickly access and use pre-trained models for tasks such as prediction, fine-tuning, or transfer learning in your own projects.

- EARLY STOPPING

The early stopping module in Keras is a powerful tool for preventing overfitting

during model training. By monitoring a specified validation metric, such as validation loss or accuracy, the training process can be halted when the metric stops improving, thus preventing the model from learning noise from the training data. This technique helps to save time and computational resources while ensuring that the model generalizes well to unseen data.

3.5. Model Design and Implementation

Designing a CNN model for galaxy classification involves several key steps:

1. Data Collection and Preprocessing: Gather a large dataset of galaxy images with labeled classes (e.g., spiral, elliptical, irregular). Preprocess the images by resizing them to a standard size and normalizing pixel values.

2. Model Architecture Selection: Choose a suitable CNN architecture for the task. Common choices include VGG, ResNet, or custom architectures. Consider the complexity of the features present in galaxy images when selecting the model.

3. Input Layer: The input layer should match the dimensions of the preprocessed galaxy images.

4. Convolutional Layers: Stack multiple convolutional layers to extract hierarchical features from the input images. Start with fewer filters in the initial layers and gradually increase the number of filters.

5. Pooling Layers: Intersperse pooling layers (e.g., max pooling) between convolutional layers to reduce spatial dimensions and extract dominant features.

6. Flattening Layer: Flatten the output of the last convolutional layer to prepare it for input into the fully connected layers.

7. Fully Connected Layers: Add one or more fully connected layers to learn high-level features and perform classification. Include dropout layers to prevent overfitting.

8. Output Layer: The output layer should have neurons equal to the number of classes in the dataset, with softmax activation to produce class probabilities.

9. Loss Function and Optimizer: Choose an appropriate loss function such as categorical cross-entropy for multi-class classification. Use optimizers like Adam or RMSprop to minimize the loss function during training.

10. Training: Split the dataset into training, validation, and test sets. Train the model on the training data, validating its performance on the validation set. Fine-tune hyperparameters based on validation performance.

11. Evaluation: Evaluate the trained model on the test set to assess its performance metrics such as accuracy, precision, recall, and F1-score.

12. Fine-tuning and Optimization: Experiment with different architectures, hyperparameters, and data augmentation techniques to improve model performance.

13. Deployment: Once satisfied with the model's performance, deploy it to classify galaxies in new unseen data.

CNN Model :Code Implementation

We can start by importing necessary libraries needed for the model such as Tensorflow, Keras etc.

- Model Initialization

```
model = Sequential()
```

It initializes a sequential model. Sequential models are a linear stack of layers.

- Adding Convolutional Layers

```
model.add(Conv2D(16, (3, 3), activation='tanh', strides=(1, 1),
padding='same', input_shape=input_shape))
```

```
model.add(Conv2D(32, (3,3), activation='tanh', strides=(1, 1),
padding='same'))
```

```
model.add(Conv2D(32, (3,3), activation='tanh', strides=(1, 1),
padding='same'))
```

```
model.add(Conv2D(32, (3,3), activation='tanh', strides=(1, 1),
padding='same'))
```

These lines add four convolutional layers to the model. Each `Conv2D` layer applies 2D convolutions with a specified number of filters, kernel size, activation function, stride, and padding.

- Adding MaxPooling Layers

```
model.add(MaxPool2D((2, 2)))
```

```
model.add(MaxPool2D((2, 2)))
```

```
model.add(MaxPool2D((2, 2)))
```

```
model.add(MaxPool2D((2, 2)))
```

These lines add four max-pooling layers to the model. Max-pooling reduces the spatial dimensions of the input, aiding in reducing computational complexity and controlling overfitting.

- Adding Dropout Layers

```
model.add(Dropout(0.25))
```

```
model.add(Dropout(0.5))
```

These lines add dropout layers to the model. Dropout layers randomly set a fraction of input units to zero during training, which helps prevent overfitting.

- Flatten Layer

```
model.add(Flatten())
```

This layer flattens the input data into a one-dimensional array. It prepares the data for input into the fully connected layers.

- Adding Fully Connected Layers

```
model.add(Dense(32, activation='tanh'))
```

```
model.add(Dense(len(imageLabel),activation='softmax'))
```

These lines add two fully connected dense layers. The first dense layer has 32 units with a hyperbolic tangent activation function, and the second dense layer has units equal to the number of classes in the '`imageLabel`' with a softmax activation function.

- Model Summary

```
model.summary()
```

This line prints a summary of the model architecture, including the type and shape of each layer and the total number of parameters.

Chapter 4

RESULT, ANALYSIS AND LIMITATIONS

4.1. Results of Classification

4.1.a. Model Accuracy

Model accuracy, in classification tasks, refers to the proportion of correctly predicted instances out of the total instances in the dataset. It is a fundamental metric used to evaluate the performance of a classification model.

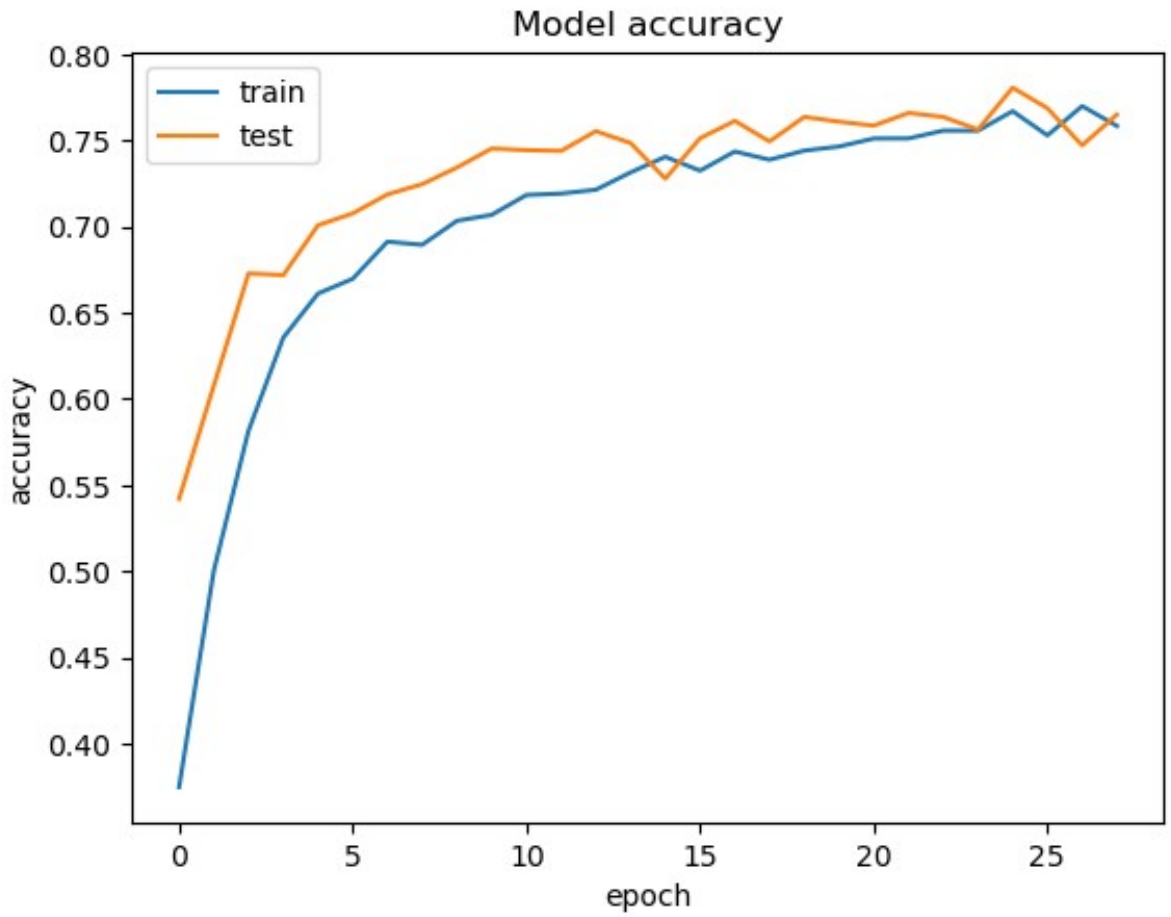


Fig 4.1. Model Accuracy

4.1.b. Model Loss

Model loss, also known as training loss or objective function, represents a measure of how well the model is performing during the training phase. It quantifies the

difference between the predicted output of the model and the actual target labels in the training data.

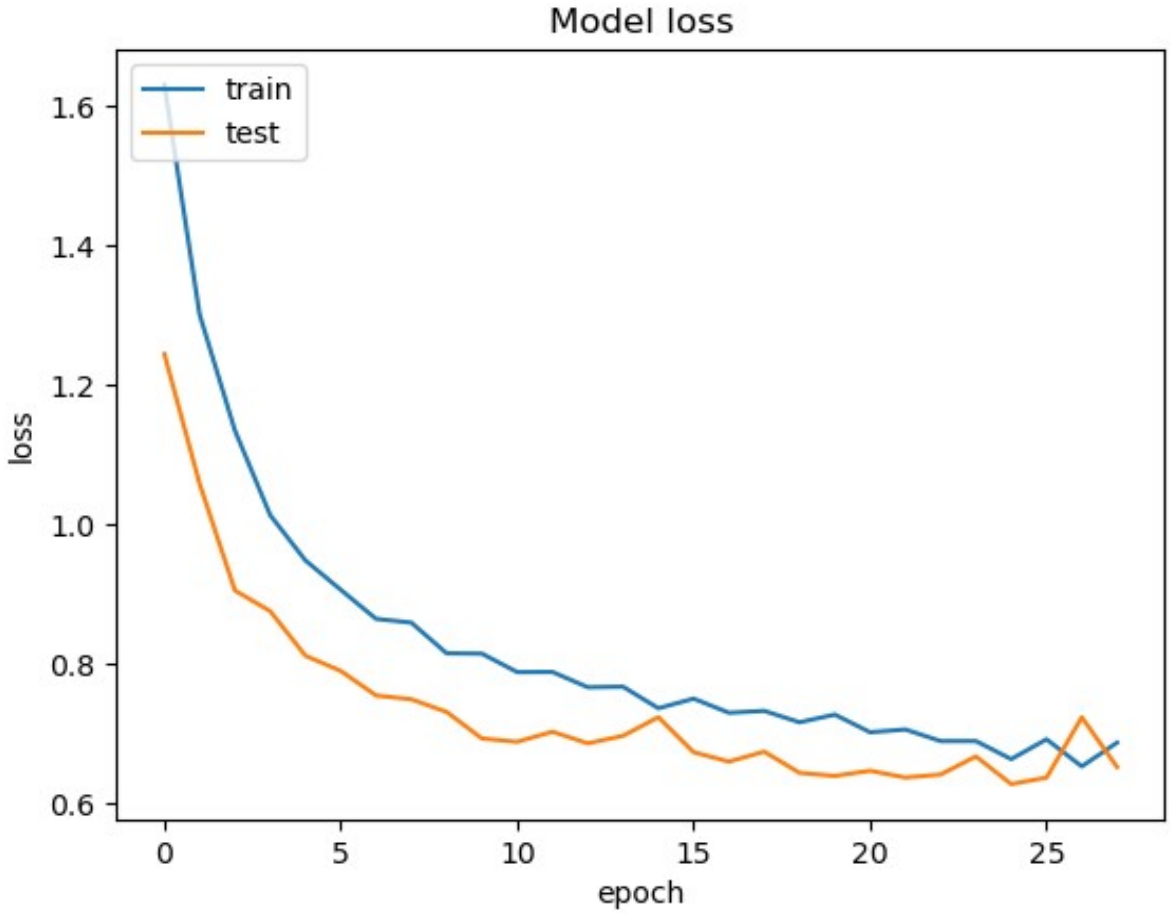


Fig 4.2. Model Loss

4.1.c. Confusion Matrix

A confusion matrix is a table used to evaluate the performance of a classification model. It allows visualization of the performance of an algorithm by presenting the counts of true positive, true negative, false positive, and false negative predictions made by the model on a specific dataset.

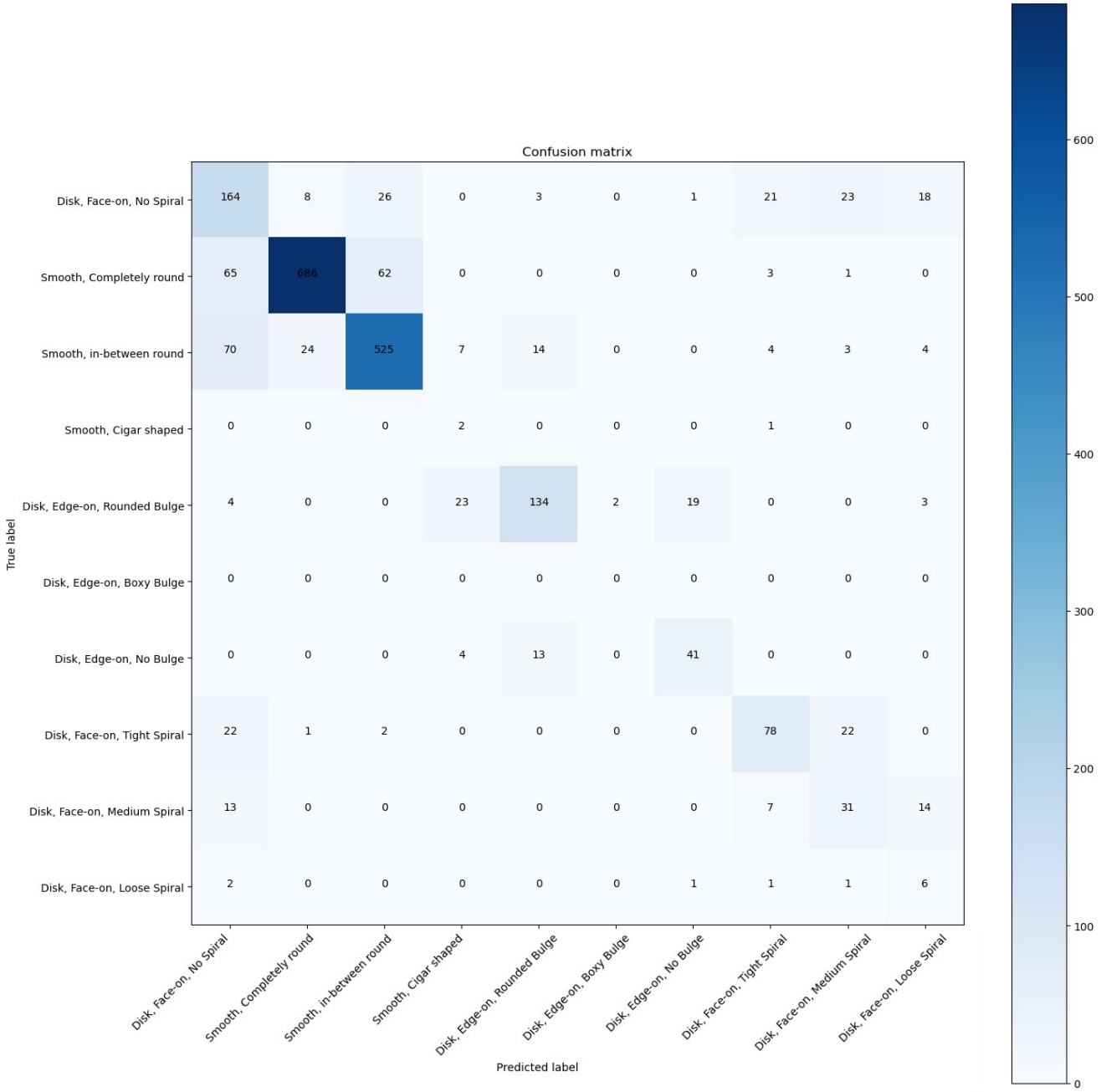


Fig 4.3. Confusion Matrix of the Prediction Model

4.2. Analysis

• Model Accuracy : The primary metric for evaluating a classifier's performance is its accuracy on a test dataset . Here this Classification Model delivers an average accuracy of 75.6%.

• Training Time: It's essential to consider the time it takes to train the CNN, especially for large datasets and complex architectures. Here the model earlystopped at 28 epochs with an average of 15 sec per epoch or 48ms/step.

• Resource Utilization: CNNs can be computationally intensive, so it's important to consider the hardware resources required for training and inference, such as GPU utilization and memory usage. While running the model in Google Colaboratory provided with an Intel Xeon CPU with 2 vCPUs (virtual CPUs) and 12.7GB of RAM, and it utilised 5.4/12.7 GB as its peak usage.

• Confusion Matrix: This confusion matrix represents the performance of a multi-class classification model across 10 different classes.
Class Labels: The rows and columns in the confusion matrix represent the actual and predicted class labels, respectively.
Diagonal Elements (True Positives): The numbers along the diagonal (from top-left to bottom-right) represent the number of instances that were correctly classified.

True Positives of each class in this Confusion Matrix are:

Class 0: 164 instances were correctly classified, with some confusion with classes 1, 2, 7, 8, and 9.

Class 1: 686 instances were correctly classified, with some confusion with classes 0, 2, 7, 8, and 9.

Class 2: 525 instances were correctly classified, with some confusion with classes 0, 1, 3, 4, 7, 8, and 9.

Class 3: Only 2 instances were correctly classified, with no confusion with other classes.

Class 4: 134 instances were correctly classified, with some confusion with classes 0, 3, 6, and 9.

Class 5: No instances were correctly classified.

Class 6: 41 instances were correctly classified, with some confusion with classes 3, 4, 7, and 9.

Class 7: 78 instances were correctly classified, with some confusion with classes 0, 8, and 9.

Class 8: 31 instances were correctly classified, with some confusion with classes 0, 1, 7, and 9.

Class 9: 6 instances were correctly classified, with some confusion with classes 0, 1, 6, 7, and 8.

4.3. Limitations of the Classification Model

An average validation accuracy of 75.6% indicates decent performance, but there are still limitations to consider based on the provided confusion matrix:

1. Confusion between similar classes: The model struggles to distinguish between certain classes, as evidenced by off-diagonal elements in the confusion matrix. This suggests limitations in capturing subtle differences between classes.

Eg: The misclassification in the Confusion Matrix are given below:

Class 0:
- Misclassified as:
  - Class 1: 8 instances
  - Class 2: 26 instances
  - Class 3: 1 instance
  - Class 7: 21 instances
  - Class 8: 23 instances
  - Class 9: 18 instances

Class 1:
- Misclassified as:
  - Class 0: 65 instances
  - Class 2: 62 instances
  - Class 7: 3 instances
  - Class 8: 1 instance

Class 2:
- Misclassified as:
  - Class 0: 70 instances
  - Class 1: 24 instances

- Class 3: 7 instances
- Class 4: 14 instances
- Class 7: 4 instances
- Class 8: 3 instances
- Class 9: 4 instances

Class 3:
- Misclassified as:
  - Class 0: 4 instances

Class 4:
- Misclassified as:
  - Class 0: 4 instances
  - Class 3: 23 instances
  - Class 6: 19 instances
  - Class 9: 3 instances

Class 6:
- Misclassified as:
  - Class 3: 4 instances
  - Class 4: 13 instances
  - Class 7: 41 instances

Class 7:
- Misclassified as:
  - Class 0: 22 instances
  - Class 1: 1 instance
  - Class 2: 2 instances
  - Class 8: 22 instances

Class 8:

- Misclassified as:
  - Class 0: 13 instances
  - Class 7: 7 instances
  - Class 9: 14 instances

Class 9:

- Misclassified as:
  - Class 0: 2 instances
  - Class 6: 1 instance
  - Class 7: 1 instance
  - Class 8: 1 instance
  - Class 9: 6 instances

These misclassifications provide insights into where the model struggles to distinguish between certain classes, highlighting areas for further analysis and potential model improvement.

2. Class imbalance: Some classes have fewer instances, leading to lower accuracy for those classes. This imbalance can bias the model's learning and affect its ability to generalize well.

Eg: The class ‘Disc,Edge-on,Boxy Bulge’only has 17 instances/images leading to a row of all zeros depicting no classificaiton at all.

3. Misclassification of rare classes: Classes with fewer instances may be prone to higher misclassification rates, impacting the overall accuracy and reliability of the classifier.

4. Model complexity: Depending on the complexity of the CNN architecture, there may be limitations in computational resources required for training and inference, as well as potential overfitting or underfitting issues.

5. Dataset Acuracy: There is no guarantee on the accuracy of the labels. Moreover, Galaxy10 is not a balanced dataset and it should only be used for educational or experimental purpose.

CHAPTER 5

CONCLUSION

5.1. Conclusion

The CNN galaxy classification model trained on the Galaxy10 SDSS dataset achieved an accuracy of 75.6%. While this accuracy demonstrates the model's capability to effectively classify galaxies into their respective categories, several factors must be considered. The model's performance is commendable, considering the complexity of galaxy images and the challenges associated with classifying them accurately, also due to device specifications and other technical difficulties.

However, further improvements may be possible through fine-tuning the model architecture, optimizing hyperparameters, and augmenting the training data. Despite its limitations, the model represents a promising advancement in the field of astronomy, providing astronomers with a valuable tool for automating the classification of galaxies and facilitating scientific discoveries. Continued research and refinement of CNN models or Classfication models using other machine learning methodologies are promising a strong and reliable branch for galaxy classification which holds the potential to enhance our understanding of the universe's vast and diverse celestial objects.

5.2. Future Scope

Achieving 75.6% accuracy in morphological classification of galaxies is a good starting point. Some future directions to enhance this CNN algorithm:

1.Data Augmentation:Increase the diversity and quantity of this training data through techniques like rotation, flipping, and scaling to improve generalization .

2.Transfer Learning: Utilize pre-trained models on larger image datasets like ImageNet and fine-tune them on the galaxy dataset to leverage their learned features.

3.Ensemble Methods: Combine predictions from multiple CNN models trained with different initializations or architectures to boost performance.

4.Data Quality: Ensure high-quality annotations and clean data to reduce noise and improve model performance.

5.Incorporate Contextual Information: Include additional data sources or features, such as spectroscopic data or multi-wavelength images, to provide more contextual information to the model.

6.Attention Mechanisms: Incorporate attention mechanisms to allow the model to focus on relevant parts of the image, potentially improving classification accuracy.

7.Domain Adaptation: Explore techniques for domain adaptation to improve the generalization of the model across different telescopes or survey datasets.

The utilization of CNN algorithms for morphological classification of galaxies in astrophysics holds considerable future promise. With the expansion of datasets and advancements in computational power, CNNs can potentially achieve greater accuracy in classifying and comprehending the morphology of galaxies. This advancement could contribute to enhancing our knowledge of galaxy formation, evolution, and the identification of rare or unusual galactic structures. Moreover, as telescopes and observational techniques continue to improve, CNNs could serve as essential tools for efficiently analyzing vast amounts of data, thereby supporting astronomers in their research pursuits.

REFERENCES AND CITATIONS

1.Literature survey references:

i. Deep Galaxy: Classification of Galaxies based on Deep Convolutional Neural Networks

(*Nour Eldeen M. Khalifa, Mohamed Hamed N. Taha, Aboul Ella Hassanien, I. M. Selim)*
https://arxiv.org/abs/1709.02245

ii. Morphological Galaxy Classification Using Machine Learning

(*Siddhartha Kasivajhula, Naren Raghavan, and Hemal Shah*)

https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://cs229.stanford.edu/proj2007/KasivajhulaRaghavanShah-MorphologicalGalaxyClassification.pdf&ved=2ahUKEwi6-vOf--WFAxU1e2wGHVNaCZcQFnoECBsQAQ&usg=AOvVaw2cW7g5nsNmRuLDc13F30Zj

2.Galaxy10 dataset images come from Sloan Digital Sky Survey (SDSS)

astroNN : https://github.com/henrysky/astroNN

3. Neural Networks Classfication

https://www.analyticsvidhya.com/blog/2021/11/neural-network-for-classification-with-tensorflow/

https://academic.oup.com/mnras/article/464/4/4463/2417400?login=false

4. Classification Images

https://vitalflux.com/classification-model-svm-classifier-python-example/

https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

https://databasecamp.de/en/ml/naive-bayes-algorithm

https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/

https://www.geeksforgeeks.org/introduction-convolution-neural-network/

https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

https://www.javatpoint.com/machine-learning-random-forest-algorithm

https://www.javatpoint.com/logistic-regression-in-machine-learning

5. Galaxy Images
https://science.nasa.gov/universe/galaxies/types/

6. OpenAI ChatGPT
https://openai.com/

7. Meta AI
https://ai.meta.com/meta-ai/

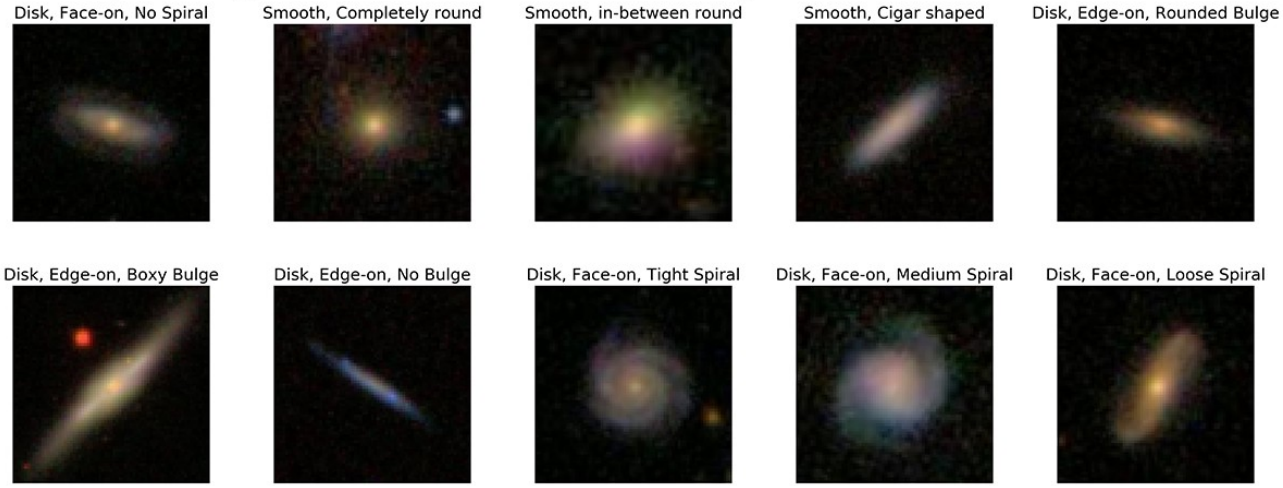
8. Galaxy Classification
https://github.com/iamsh4shank/gal-classifier

9. Galaxy Classfication
https://www.google.com/url?q=https://phas.ubc.ca/~hickson/astr505/astr505_2016-2.pdf&usg=AOvVaw247bT6w5rLosINQxJC0aMJ&cs=1&hl=en-US

APPENDIX

Sample images from the Galaxy10 dataset

Example image of each class



Galaxy10 Dataset: Henry Leung/Jo Bovy2018, Data Source: SDSS/Galaxy Zoo

Code of the Classfication Model:

https://github.com/Kezton/Final_Project/blob/main/FINAL%20PROJECT.ipynb

https://github.com/Kezton/Final_Project/blob/main/FINAL%20PROJECT.py

Code snippets are followed:

```
!pip install scikit-learn
!pip install --upgrade scikit-learn
from sklearn.metrics import ConfusionMatrixDisplay
```

```
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import utils
from sklearn.metrics import confusion_matrix
from keras.layers import Conv2D, MaxPool2D, Flatten, Dense, Dropout
from keras.models import Sequential, load_model
from keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
!pip install astroNN
```

```
from astroNN.datasets import load_galaxy10sdss
```

```
images, labels = load_galaxy10sdss()
label = utils.to_categorical(labels, num_classes=10)
```

```
print (labels)
print (label.shape[0])
```

```
train_x,test_x=train_test_split(np.arange(labels.shape[0]),test_size=0.1)
train_images,train_labels,test_images,test_labels=images[train_x],label[train_x],images[test_x],label[test_x]
```

```
print (len(train_x))
print (len(test_x))
print (len(train_labels))
```

```
imageLabel = ["Disk, Face-on, No Spiral", "Smooth, Completely round",
              "Smooth, in-between round","Smooth, Cigar shaped",
              "Disk, Edge-on, Rounded Bulge", "Disk, Edge-on, Boxy Bulge",
              "Disk, Edge-on, No Bulge","Disk, Face-on, Tight Spiral",
              "Disk, Face-on, Medium Spiral","Disk, Face-on, Loose Spiral"]
```

```
fig, axes = plt.subplots(ncols = 10, nrows = 10, figsize = (35,30))
index = 0
for i in range(10):
  for j in range(10):
    axes[i,j].set_title(imageLabel[labels[index]])
    axes[i,j].imshow(images[index].astype(np.uint8))
    axes[i,j].get_xaxis().set_visible(False)
    axes[i,j].get_yaxis().set_visible(False)
    index +=1
plt.show()
```

```
plt.imshow(train_images[0].astype(np.uint8))
print (labels[0])
print (label[0])
print (train_images.shape)
```

```
X_train = np.array([cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) for image in train_images])
X_test = np.array([cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) for image in test_images])
#grayscale conversion

# Value normalization
X_train  = X_train/255
X_test  = X_test/255
```

```
plt.imshow(X_train[0])
```

```
print(np.shape(X_train))

X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)

input_shape = (X_train.shape[1], X_train.shape[2], 1)

print(input_shape)
```

```
print (X_train.shape)
print (train_labels.shape)
print (train_labels)
```

```
datagen = ImageDataGenerator(rescale=1./255,
                             rotation_range=90,
                             zoom_range=0.2,
                             horizontal_flip=True,)

datagen.fit(X_train)

datagen.fit(X_test)
```

```
model = Sequential()
model.add(Conv2D(16, (3, 3), activation='tanh', strides=(1, 1),
    padding='same', input_shape=input_shape))
model.add(Conv2D(32, (3, 3), activation='tanh', strides=(1, 1),
    padding='same'))
model.add(MaxPool2D((2, 2)))
model.add(Conv2D(32, (3, 3), activation='tanh', strides=(1, 1),
    padding='same'))
model.add(MaxPool2D((2, 2)))
model.add(Conv2D(32, (3, 3), activation='tanh', strides=(1, 1),
    padding='same'))
model.add(MaxPool2D((2, 2)))
model.add(Conv2D(32, (3, 3), activation='tanh', strides=(1, 1),
    padding='same'))
model.add(MaxPool2D((2, 2)))
```

```
model.compile(loss='categorical_crossentropy',
     optimizer='adam',
     metrics=['acc'])

es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=3)
batch_size=64
history = model.fit(X_train, train_labels,
                    epochs=30,
                    steps_per_epoch = int(np.ceil(X_train.shape[0]/ float(64))) ,
                    batch_size=32, validation_data=(X_test, test_labels), callbacks=[es])
```

```
%matplotlib inline
fig = plt.gcf()
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
fig.savefig('Model_Accuracy.png')
```

```
fig = plt.gcf()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
fig.savefig('Model_Loss.png')
```

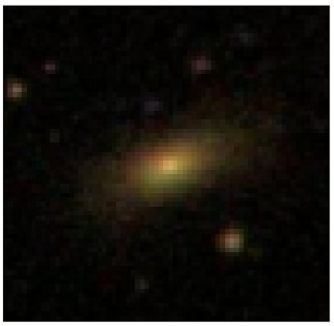
```
import itertools
def plot_confusionM(cm, class_names):
    figure = plt.figure(figsize=(15, 15))
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title("Confusion matrix")
    plt.colorbar()
    tick_marks = np.arange(len(class_names))
    plt.xticks(tick_marks, class_names, rotation=45)
    plt.yticks(tick_marks, class_names)

    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j], horizontalalignment="center")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

pred = model.predict(X_test)

pred_label = np.argmax(pred, axis=1)
actual_label = np.argmax(test_labels, axis=1)

cm = confusion_matrix(pred_label+1, actual_label+1)
print (cm)
plot_confusionM(cm, imageLabel)
```

```
fig, axes = plt.subplots(ncols=7, nrows=3, sharex=False,
    sharey=True, figsize=(27, 24))
index = 0
for i in range(3):
    for j in range(7):
        axes[i,j].set_title('actual:' + imageLabel[actual_label[index]] + '\n'
                            + 'predicted:' + imageLabel[pred_label[index]])
        axes[i,j].imshow(test_images[index].astype(np.uint8), cmap='gray')
        axes[i,j].get_xaxis().set_visible(False)
        axes[i,j].get_yaxis().set_visible(False)
        index += 1
plt.show()
```

Prediction Sample

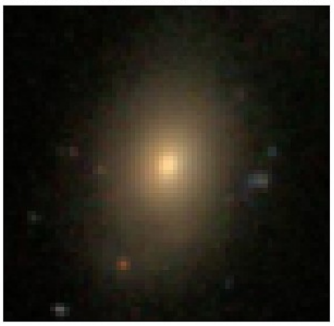
Actual:Disk, Face-on, No Spiral
Predicted:Disk, Face-on, No Spiral

Actual:Smooth, in-between round
Predicted:Smooth, in-between round

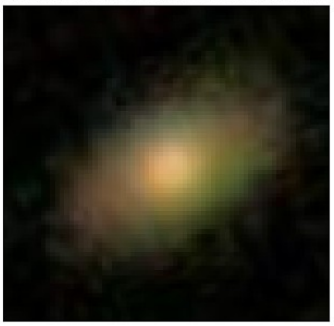


Actual:Smooth, Completely round
Predicted:Smooth, Completely round

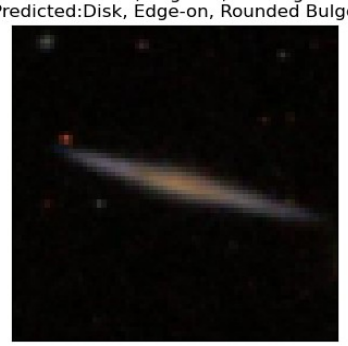
Actual:Smooth, in-between round
Predicted:Smooth, in-between round



Actual:Smooth, in-between round
Predicted:Smooth, in-between round



Actual:Disk, Edge-on, No Bulge
Predicted:Disk, Edge-on, Rounded Bulge



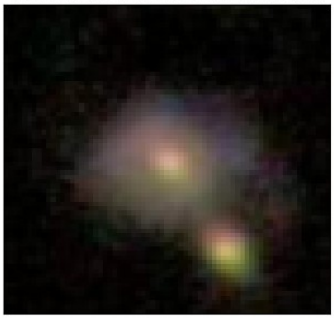
Actual:Disk, Face-on, No Spiral
Predicted:Smooth, in-between round

Actual:Disk, Face-on, No Spiral
Predicted:Smooth, Completely round

Actual:Smooth, Completely round
Predicted:Smooth, Completely round

Actual:Smooth, Completely round
Predicted:Smooth, Completely round

Actual:Disk, Face-on, No Spiral
Predicted:Disk, Face-on, No Spiral



Actual:Smooth, Completely round
Predicted:Smooth, Completely round

Actual:Smooth, in-between round
Predicted:Smooth, in-between round

Actual:Disk, Face-on, Tight Spiral
Predicted:Disk, Face-on, Tight Spiral

Actual:Smooth, in-between round
Predicted:Smooth, in-between round

Actual:Smooth, Completely round
Predicted:Smooth, Completely round

Actual:Smooth, in-between round
Predicted:Smooth, in-between round

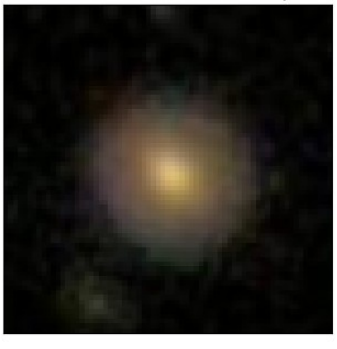
Actual:Smooth, Completely round
Predicted:Smooth, Completely round

Actual:Disk, Face-on, Medium Spiral
Predicted:Disk, Face-on, Medium Spiral

Actual:Smooth, Completely round
Predicted:Smooth, Completely round

Actual:Disk, Face-on, No Spiral
Predicted:Smooth, in-between round

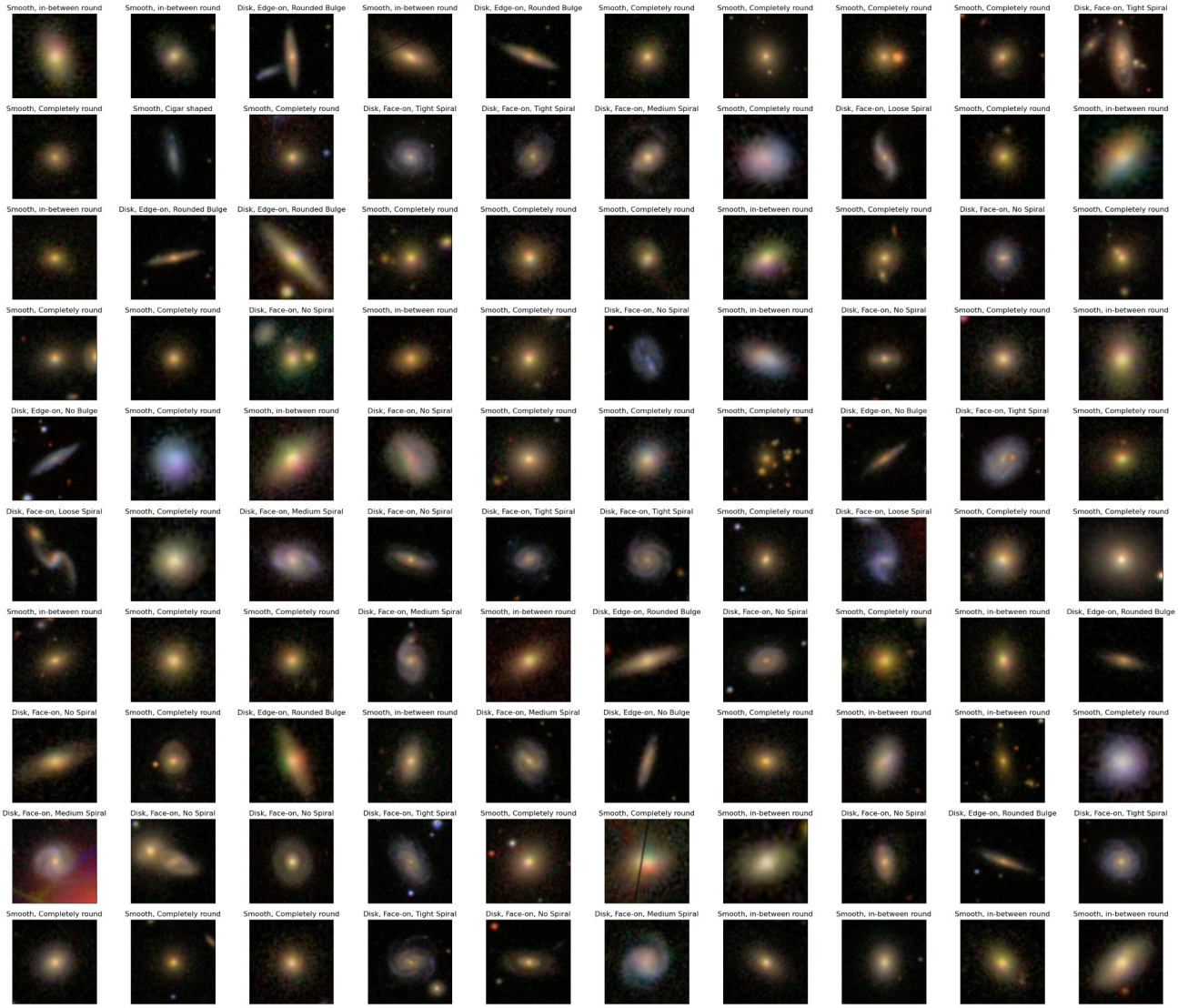
Actual:Disk, Face-on, No Spiral
Predicted:Disk, Face-on, No Spiral



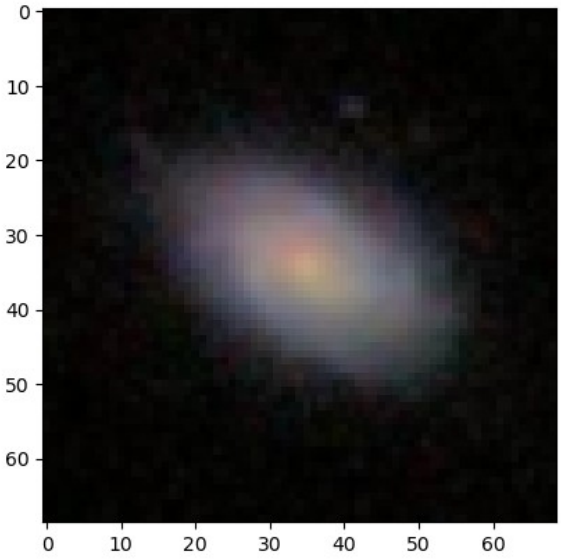
Actual:Smooth, Completely round
Predicted:Smooth, Completely round

Actual:Smooth, Completely round
Predicted:Smooth, Completely round

Datasample of Galaxy10 dataset



Resized Image sample



Grayscaled Image sample