2023

Classifying Galaxy Images Using Improved Residual Networks

Jaykumar Patel
University of Windsor

Classifying Galaxy Images Using Improved Residual Networks

By

Jaykumar Patel

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2023

Classifying Galaxy Images Using Improved Residual Networks

by

Jaykumar Patel

APPROVED BY:

D. Xiao
Department of Physics

S. Samet
School of Computer Science

D. Wu
School of Computer Science

May 29, 2023

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

The field of astronomy has made tremendous progress in recent years thanks to advancements in technology and the development of sophisticated algorithms. One area of interest for astronomers is the classification of galaxy morphology, which involves categorizing galaxies based on their visual appearance. However, with the sheer number of galaxy images available, it would be a daunting task to manually classify them all.

To address this challenge, a novel Residual Neural Network (ResNet) model, called ResNet_Var, that can automatically classify galaxy images is proposed in this study. Galaxy Zoo 2 dataset is used in this research, which contains over 28,000 images for the five-class classification task and over 25,000 images for the seven-class classification task.

To evaluate the effectiveness of the ResNet_Var model, various metrics such as accuracy, precision, recall, and F1 score were calculated. The results were impressive, with the ResNet_Var model outperforming other popular networks such as VGG16, VGG19, Inception, and ResNet50. Specifically, the overall classification accuracy of the ResNet_Var model was 95.35% for the five-class classification task and 93.54% for the seven-class classification task.

The potential applications of the ResNet_Var model are vast. With such a high accuracy rate, the ResNet_Var model is well-suited for large-scale galaxy classification in optical space surveys. By automating the classification process, astronomers can quickly and accurately categorize galaxy images according to their morphology. This, in turn, can help advance our understanding of galaxy formation and evolution, as well as provide valuable insights into the properties of dark matter and the nature of the universe.

DEDICATION

I would like to dedicate this thesis to my family and my friends.

ACKNOWLEDGEMENTS

I would like to extend my sincere gratitude to my supervisor, Dr. Dan Wu, for his invaluable guidance, unwavering support, and valuable input throughout my research. Dr. Wu's expertise and insight have been instrumental in shaping my work and providing me with a new perspective, which has helped me to approach my research with a more critical eye. I am immensely grateful for his patience, encouragement, and mentorship, which have been an inspiration to me throughout this process.

I am also grateful to my thesis committee members for their invaluable feedback, advice, and suggestions, which have significantly contributed to the success of my research. Their expertise, insights, and constructive criticism have been instrumental in shaping my work, and I am thankful for their support and guidance.

I would like to thank my family and friends for their unwavering support, encouragement, and understanding throughout this journey. Their constant motivation, understanding, and support have been a source of strength and inspiration, and I am grateful for their love and encouragement.

Lastly, I would like to express my gratitude to the School of Computer Science and all concerned people who have contributed to my research. Their support and assistance have been invaluable, and I am grateful for their help in making this research a success.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

NN	Neural Network
CNN	Convolution Neural Network
ResNet	Residual Neural Network
TP	True Positive
FN	False Negative
FP	False Positive
TN	True Negative
SVM	Support Vector Machine
DT	Decision Tree
KNN	K Nearest Neighbors
NB	Naive Bayes
RF	Random Forest
MLP	Multi Layer Perceptron
GZ	Galaxy Zoo
GZ2	Galaxy Zoo 2

CHAPTER 1

Introduction

1.1 Overview

Astronomy is a field of natural science that studies celestial objects and phenomena [89, 94, 45, 11, 85]. Using mathematics, physics, and chemistry, it seeks to understand the formation and evolution of these objects. Study of astronomy includes studying planets, moons, stars, nebulae, galaxies, comets, and other celestial bodies, as well as phenomena such as supernova explosions, gamma ray bursts, quasars, blazars, pulsars, and cosmic microwave background radiation [89]. Astronomy is the study of anything that originates outside the Earth's atmosphere. With technological advancements over recent decades, astronomy has become a field that generates vast amounts of data [94, 45, 11, 85]. New observational tools such as satellites and telescopes provide large, complex datasets that include spatial and temporal components.

One of the primary sources of data used in observational astronomy is photometry, where each image captures a specific field of view of the sky in a chosen frequency band, which can contain multiple objects and is subject to noise [94, 45, 11, 85]. With the simultaneous development of machine learning technologies, it has become possible to handle and extract more value from these massive datasets in various research and industry contexts [4].

An unbiased sample containing reliable morphological types is crucial to any research on extragalactic objects. It allows for accurate classification of galaxies based on their morphologies [45, 4]. Classification of galaxies based on their morphology is vital because the shape and structure of a galaxy can provide insight into its formation

and evolution, as well as its interactions with its environment. Galaxies in groups or clusters, for example, may have different evolutionary paths than those alone, which is reflected in their morphologies. In order to study galaxy formation and evolution, it is essential to classify galaxies into a systematic morphology system.

1.2 Galaxy Morphology

Galaxy morphology refers to galaxies' physical makeup and appearance, encompassing traits like size, shape, brightness, and the arrangement of components such as stars, gas, dust, etc. Studying galaxy morphology is crucial in astronomy since it offers a deep understanding of how they form, evolve and interact with their surroundings. Different kinds of galaxies (spiral/elliptical/irregular) exhibit unique characteristics that necessitate accurate categorization based on these features so we can understand their properties and behavior.

In observational research involving celestial objects, classification into a morphological system is crucial; however, this process can be complex for researchers without a dependable methodology ensuring precise identification. This makes accuracy in classifying celestial objects vital, especially within disciplines where information discovered may reveal valuable insight regarding formation or evolution processes (astronomy).

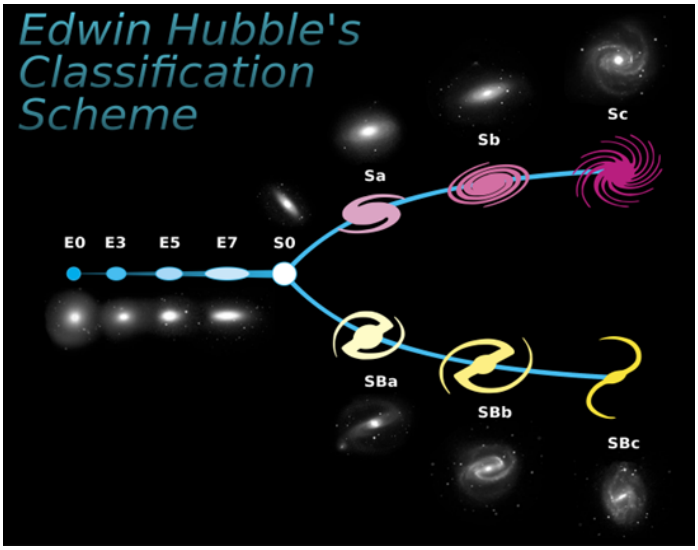


Fig. 1: Tuning-fork style diagram of the Hubble sequence (Source : Wikipedia [98])

In 1926, Hubble introduced the first classification scheme for galaxies, later refined in 1936 [42, 43]. This classification, commonly known as the Hubble sequence, utilizes a diagram resembling a tuning fork, as shown in Figure 1. The Hubble sequence divides galaxies into two main categories based on their overall structure: Early-Type Galaxies (ETGs) and Late-Type Galaxies (LTGs).

ETGs are characterized by a dominant bulge component, giving them a more elliptical shape. Hence they are commonly referred to as elliptical galaxies. On the other hand, LTGs are characterized by a more prominent disk with spiral arms and are known as spiral galaxies. The LTGs group can be further divided into two sub-categories based on their central structure: barred (with a bar-shaped central structure, denoted as SB) and non-barred (with no central bar, denoted as S) galaxies [79]. These sub-categories can also be refined based on the strength of their spiral arms.

The Hubble classification provides a useful starting point for understanding the morphologies of galaxies, and it has been a fundamental tool in extragalactic astronomy for nearly a century.

The classification of galaxy images is an essential task in astronomy, as it enables astronomers to gain insights into the properties and characteristics of galaxies [22].

However, due to the exponential growth of astronomical data, there has been a significant increase in the number of galaxy images requiring classification [13]. Therefore, manual classification is no longer a practical option, as it is time-consuming, labor-intensive, and prone to human error [35].

One approach to reliable classification is the use of machine learning algorithms. In recent years, there have been significant developments in the field of machine learning, making it a powerful tool for classification tasks in astronomy and other fields. For example, Residual Neural Networks (ResNet) have been successfully used in the classification of galaxy images based on their morphologies [59]. Other machine learning techniques, such as convolutional neural networks (CNNs) and decision trees, have also been used in astronomical research for classification purposes [9, 51].

To ensure a robust classification methodology, it is essential to carefully consider the features and characteristics of the objects being studied. For example, in astronomical research, photometric measurements are a common source of data, and careful consideration of the wavelength bands used and the noise levels present can help to improve classification accuracy [87]. Additionally, it is important to consider the impact of selection biases on the resulting classification, as the objects selected for study can have a significant impact on the reliability of the classification [72].

Overall, the classification of objects into a morphology system is a fundamental task in observational research, and building a robust methodology for reliable classification is a significant challenge. Machine learning algorithms, such as ResNet, have shown promise in this area, but careful consideration of the data and selection biases is also essential for accurate classification.

1.3 Galaxy Zoo

Machine learning algorithms are becoming more common in scientific research, including classifying objects into a morphology system. While machine learning algorithms can significantly improve efficiency and accuracy, human supervision is still necessary to guide the machine and provide meaningful labels to the patterns it discovers.

One project that successfully utilized human input is the Galaxy Zoo project [55]. Launched in 2007, this project engaged volunteers in classifying galaxy images into elliptical, spiral, and undefined categories. By combining the power of human intuition and the speed of automated processes, the Galaxy Zoo project could efficiently and accurately classify many galaxy images.

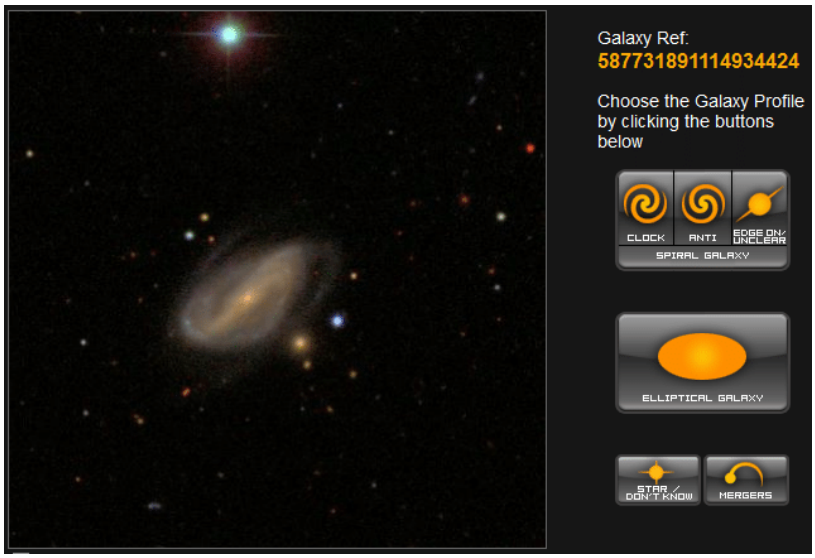


Fig. 2: Galaxy Zoo 1 user interface showing a galaxy image on the left and buttons on the right for users to select. By clicking on the buttons, users can select one of the options from clockwise, anticlockwise and edge-on which are under the category of spiral galaxy, or the elliptical galaxy, or one of the option from star and merger which are under undefined/other galaxy category (Source : Lintott et al. (2008) [55])

Galaxy Zoo is a citizen science project that harnesses the collective intelligence of volunteers to classify galaxies based on their morphologies. The primary objective of GZ1 (Galaxy Zoo 1, 2007) was to categorize galaxies based on their visual appearance, particularly their shapes and structures [55]. The user interface of GZ1 was designed so that users could classify galaxies straightforwardly and intuitively. Figure 2 shows the user interface of Galaxy Zoo 1 [55]. On the left side of the interface, users were shown an image of a galaxy. They were given the option to select the galaxy type from predefined categories such as elliptical, spiral, or other. These three categories are further divided into six options: the elliptical has one option, the spiral has three options listed as clock, anti, and edge-on, and the other is divided into two options for users to select from star and merger. On the right side of the interface, the users were

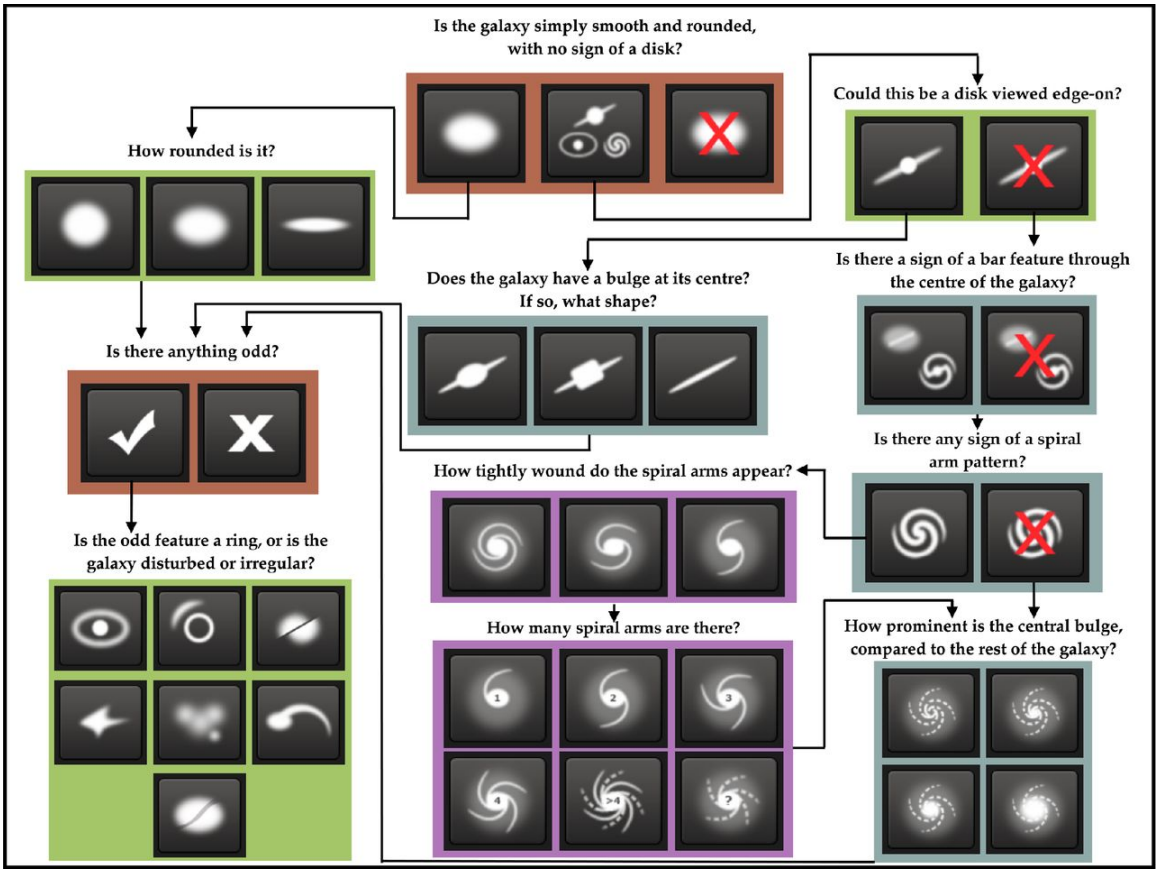


Fig. 3: Decision tree used in Galaxy Zoo 2 project (Source : Willett et al. (2013) [99])

presented with options and examples to help them identify and classify the galaxy type they saw on the left. Users could classify as many galaxies as they wanted and were encouraged to return to the site for more images. The classifications submitted by users were combined and analyzed to create a catalog of galaxy classifications that researchers could use for further analysis [55].

The initial project's success led to the launch of a follow-up project, Galaxy Zoo 2 (GZ2), in 2009. Galaxy Zoo 2 aimed to build on the success of its predecessor by introducing a more complex classification system consisting of a decision tree table (Table 1) and several classification stages (Fig. 3). This new system enabled volunteers to provide labeled data for galaxy morphology with unprecedented detail, thus significantly improving our understanding of the universe.

The project known as Galaxy Zoo was a significant step towards democratizing

Task	Question	Responses	Next
01	*Is the object a smooth galaxy, a galaxy with features/disk or a star?*	smooth	07
		features or disk	02
		star or artifact	**end**
02	*Is it edge-on?*	yes	09
		no	03
03	*Is there a bar?*	yes	04
		no	04
04	*Is there a spiral pattern?*	yes	10
		no	05
05	*How prominent is the central bulge?*	no bulge	06
		just noticeable	06
		obvious	06
		dominant	06
06	*Is there anything "odd" about the galaxy?*	yes	08
		no	**end**
07	*How round is the smooth galaxy?*	completely round	06
		in between	06
		cigar-shaped	06
08	*What is the odd feature?*	ring	**end**
		lens or arc	**end**
		disturbed	**end**
		irregular	**end**
		other	**end**
		merger	**end**
		dust lane	**end**
09	*What shape is the bulge in the edge-on galaxy?*	rounded	06
		boxy	06
		no bulge	06
10	*How tightly wound are the spiral arms?*	tight	11
		medium	11
		loose	11
11	How many spiral arms are there?	1	05
		2	05
		3	05
		4	05
		more than four	05
		can't tell	05

Table 1: Questions displayed to users by GZ2 interface along with possible responses for users to select from

scientific research, allowing for the public to actively participate and contribute to new discoveries. This approach relied on the collective intelligence of a diverse group of volunteers who were able to classify and analyze large amounts of astronomical data that would have been unmanageable by a single research team.

GZ2 expanded upon this idea with an even more advanced classification system providing greater detail in galaxy classification through the use of decision tree (Fig. 3). Table 1 served as guides for volunteers ensuring consistent and accurate classifications while multiple stages reduced errors increasing precision.

The following sections will cover important concepts including topics such as Artificial Intelligence, Machine Learning, Neural Networks, Deep Learning, Artificial Neural Networks along with applications using Deep Learning techniques.

1.4 Artificial Intelligence

Artificial Intelligence (AI) is developing and implementing computer systems capable of performing tasks that typically necessitate human intelligence, such as learning, problem-solving, decision-making, and natural language comprehension [69]. Unlike animal and human intelligence, AI is a product of synthetic thinking, perception, and inference by computers. The definition of AI by the Oxford English Dictionary of Oxford University Press is the *"theory and development of computer systems capable of performing tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages"* [25]. AI is categorized into various types: reactive machines, limited memory systems, theory of mind, and self-aware systems. Reactive machines are the most basic type of AI, simply responding to their environment without memory of past experiences. Limited memory systems can use past experiences to make decisions, as with self-driving cars that record information about their surroundings to navigate the road. Theory of mind systems can comprehend and reason about other agents, such as humans or other AI systems. Lastly, self-aware systems can recognize their mental states and are conscious of their existence [69].

AI has many applications, such as web search engines, recommendation systems, speech recognition software, self-driving cars, automated decision-making, and strategic game systems [88]. With machines becoming increasingly capable, the AI effect has become a phenomenon where actions previously thought to necessitate "intelligence" are frequently taken out of the definition of AI [56]. Nevertheless, AI continues to advance and transform how we interact with technology and the world around us.

1.5 Machine Learning

Machine learning, a subset of artificial intelligence, involves using algorithms and models to teach computers how to learn from data and make predictions or decisions without explicit programming [7]. The primary objective of machine learning is to create models and algorithms capable of processing data accurately, identifying patterns, and making predictions or decisions based on that knowledge. Various types of learning techniques exist in machine learning, including supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [60]. In supervised learning, the model is trained using labeled data to predict the output based on the input. This learning technique is useful for image classification, speech recognition, and sentiment analysis applications. Examples of supervised learning algorithms include logistic regression, linear regression, and decision trees [16].

In unsupervised learning, the model is provided with unlabeled data and must independently discover patterns and structures in the data. The model can identify data patterns, cluster similar data points, and reduce the dimensionality of the data. Unsupervised learning examples include clustering, anomaly detection, and dimensionality reduction [31].

Semi-supervised learning is a combination of supervised and unsupervised learning. In semi-supervised learning, the model is trained using a small amount of labeled data and a more extensive collection of unlabeled data. The labeled data is used for making predictions, while the unlabeled data is utilized for finding patterns and structures. A spam email filter is an example of semi-supervised learning, where a small

set of labeled data is used to train the model to classify emails as spam or not spam. However, manually labeling every email is time-consuming, so semi-supervised learning can help the model learn patterns and features expected in spam emails using a large amount of unlabeled data.

Reinforcement learning is a type of learning that involves training the model to make decisions based on the outcomes of its actions in an environment. The model learns to take actions that maximize a reward signal. Reinforcement learning can be applied to various fields like gaming, control systems, and robotics [82].

1.6 Artificial Neural Networks

An artificial neural network, often known as a "neural network," is made up of input layer of neurons (or nodes, units), one to three hidden layers of neurons, and a layer of output neurons [92]. *Figure 4* depicts a typical architecture along with the connections between the neurons. Each connection has a weight that is a numeric value. The hidden layer's neuron i's output, h_i, is

$$h_i = \sigma(\sum_{j=1}^{N} V_{ij}x_j + T_i^{hid}), \tag{1}$$

where N is the number of input neurons, V_{ij} are weights, x_j are inputs to the neurons, and T_i^{hid} are the threshold terms of the hidden neurons. σ is referred to as the activation (or transfer) function. In addition to adding nonlinearity to the neural network, the activation function (a non-linear mathematical function that is applied to the output of a neuron in a neural network) aims to constrain the neuron's value to prevent the neural network from being paralyzed by divergent neurons. The sigmoid (or logistic) function, which is a typical illustration of the activation function, is defined as

$$\sigma(u) = \frac{1}{1 + exp(-u)}. \tag{2}$$

A neural network built in the manner described above has been demonstrated to be capable of approximating any computed function with arbitrary precision [92].

For the function that the neural network is attempting to approximate, the numbers supplied to the input neurons are independent variables, and the numbers returned from the output neurons are dependent variables. When data is properly encoded, inputs to and outputs from a neural network can be binary (such as yes or no) or even symbols (green, red, etc.). This property gives neural networks a broad spectrum of use.

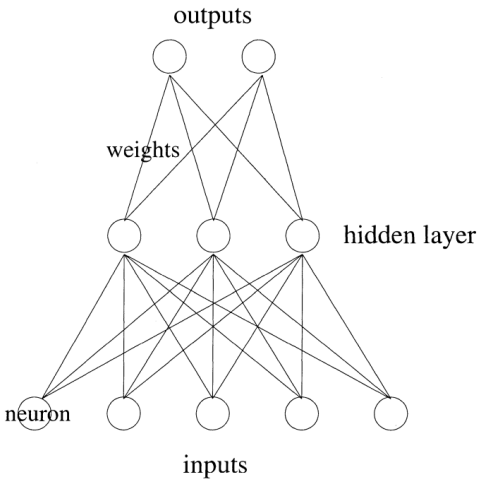


Fig. 4: Architecture of simple neural network (Source: Wang et al. (2003) [92])

Artificial Neural Networks (ANNs) are models that mimic the structure and function of the human brain to process information, identify patterns, and make decisions. ANNs contain interconnected nodes called artificial neurons arranged in layers with input data received by the input layer producing final predictions from the output layer. Hidden layers may analyze and process incoming data.

Each neuron takes inputs from other neurons that undergo weight summation passed through an activation function for its respective output based on weights connectivity between them.

The learning process is vital wherein ANNs adjust their parameters-weights, biases-using optimization algorithms such as gradient descent or backpropagation, reducing the error between predicted vs. actual outputs.

Feedforward networks with a one-way data flow are simple; recurrent networks work best to process sequential time-series information using feedback loops inter-

nally. In comparison, convolutional neural networks-CNNs-specialize in image feature extraction and representation creation.

Therefore Artificial Neural Networks utilize concepts observed within our brains' working model to assess vast amounts of data effectively while adjusting internal connections according to a supervised learning system minimizing prediction errors categorized into feedforward network types and others like recurrent or CNN-based applications.

1.7 Convolution Neural Networks

With the emergence of the Artificial Neural Network (ANN), the discipline of machine learning has recently changed drastically. In popular machine learning tasks, these computational models with biological inspiration perform noticeably better than earlier iterations of artificial intelligence. Convolutional Neural Network (CNN) architecture is one of the most remarkable types of Artificial Neural Network (ANN) architecture. CNNs are typically employed to tackle challenging image-driven pattern recognition problems. Thanks to their accurate yet straightforward architecture, they provide a streamlined way to get started with ANNs [62]. CNNs emphasize the idea that the input will be made up of one or more images. This concentrates the architecture's set up to meet the requirements for handling the particular data type. One of the main variations is that the layers of the CNN are made up of neurons arranged into three dimensions, the spatial dimensionality of the input (height and breadth) and the depth. The depth describes the third dimension of an activation volume rather than the total number of layers within the ANN. In contrast to conventional ANNs, each layer's neurons will only link to a small portion of the layer before it [62].

In practice, this means that for the earlier example, the input "volume" will have a dimensionality of $64 \times 64 \times 3$ ($height \times width \times depth$), resulting in a final output layer with a dimensionality of $1 \times 1 \times n$ (where n represents the number of classes that may be present). This is because we would have condensed the full input dimen-

sionality into a smaller volume of class scores filed across the depth dimension. Figure 5 shows a simplified CNN architecture for MNIST classification. The MNIST dataset is a widely-used dataset in machine learning and computer vision, consisting of 70,000 handwritten digits images (28×28 pixels) of the digits 0 to 9. It is commonly used as a benchmark dataset for image classification tasks, particularly for evaluating the performance of neural network models [53]. The image shown in figure 5 is a visualization of a Convolutional Neural Network (CNN) architecture commonly used in image classification tasks. It consists of several layers, including convolutional layers, pooling layers, and fully connected layers. The input image is passed through the network, and as it progresses through each layer, it becomes increasingly abstracted and transformed. Finally, the output layer provides the classification result.

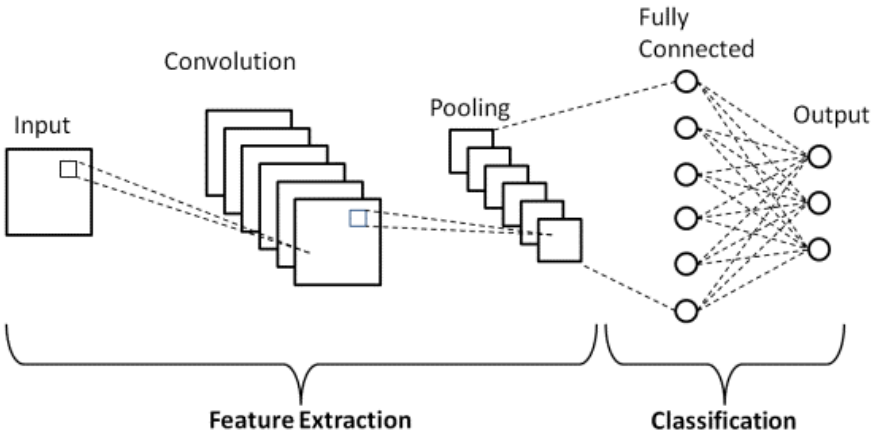


Fig. 5: A simple CNN architecture (Source: Balaji et al. (2020) [12])

1.8 Deep Learning

Deep Learning (DL) is a subfield of Machine Learning (ML) that involves training deep neural networks to make predictions or take decisions based on data [32]. Deep learning architectures, also known as deep neural networks, are a type of artificial neural network that consists of multiple layers of nodes, each of which processes the data and extracts features. The deeper the network, the more complex features it can extract [52].

Inspired by the human brain's structure and function, deep learning networks can be trained with various data types: images, text, and audio. These networks get categorized into feedforward neural networks or convolutional neural networks (CNNs) to recurrent neural networks (RNN), each suited for specific data processing tasks [32].

Recurrent Neural Networks are a type of artificial neuron that helps design machine learning and natural language processing. Unlike feedforward nets with one-way directional flow through inputs–outputs layers, RNN's feedback loop passes information from time steps leading up to eventual output, so temporal dependencies in sequential speech-text-time series-data get captured, making them ideal-perfect for translating languages and image captioning purposes.

Feedforward networks in basic deep learning architecture reflect simple pattern recognition between nodes-values moving forward instead of CNNs designed explicitly around analyzing video/images using convolution, which extracts features within provided media. In contrast, recurrence focuses on capturing sequence element dependencies helping analyze applications such as sound waves or textual translation [52].

Deep learning networks have been used to achieve state-of-the-art results in various applications, including image and speech recognition, natural language processing, and autonomous vehicles. For instance, a deep learning algorithm has been used for autonomous vehicle navigation in complex environments [17]. Another example is deep learning networks for speech recognition and natural language processing, as seen in popular applications such as Siri and Alexa [38].

In conclusion, deep learning has proven to be an effective and powerful tool for a wide range of applications and continues to be an active area of research in machine learning [52].

1.9 Deep Learning Applications

The innovative concept of building machines that can mimic human intelligence and even enhance it has long piqued our interest and continues to do so. It was because of the work done on this concept that artificial intelligence, machine learning, and later deep learning, were developed [44]. These three ideas—or, more precisely, technologies—are intriguing on their own. However, given the constraints of the subject at hand, we shall concentrate on Deep Learning in this article. The closest we have come to designing a system that mimics how the brain functions is probably with deep learning. It is a complex system that tries to learn concepts and solve issues that were previously beyond the reach of human intelligence. Deep Learning serves countless uses, including helping computers make independent judgments and identifying images, translating human language, talking with humans, and more.

Common applications of deep learning in artificial intelligence are listed as:

- Computer Vision: Deep learning models can be used to classify images, detect objects, and even generate new images. This makes it a powerful tool for tasks such as image recognition, facial recognition, and object detection.

- Natural Language Processing: Deep learning models can be used to process and understand human language. This makes it useful for tasks such as language translation, text summarization, and sentiment analysis.

- Speech Recognition: Deep learning models can be used to recognize speech and convert it to text. This makes it useful for tasks such as speech-to-text transcription and voice-controlled assistants.

- Robotics: Deep learning models can be used to control robots and make them more autonomous. This makes it useful for tasks such as navigation, manipulation, and object recognition.

- Recommendation Systems: Deep learning models can be used to analyze large amounts of data and make personalized recommendations. This makes it useful

for tasks such as movie and music recommendations, and product recommendations.

- Healthcare: Deep learning models can be used to analyze medical images and make diagnoses, such as detecting cancer, or analyzing medical records to make predictions about patient outcomes.
- Autonomous vehicles: Deep learning models can be used to enable autonomous vehicles to drive safely and make decisions based on their surroundings.
- Gaming : Deep learning models can be used to improve the gaming experience by providing realistic graphics, controlling non-player characters, or creating new game scenarios.
- Fraud Detection: Deep learning models can be used to detect fraudulent activities by analyzing large amounts of financial data.
- Energy and Power : Deep learning models can be used to optimize energy usage and power generation in smart grids, wind and solar power.

These are just a few examples of the many ways that deep learning can be applied. As the field continues to evolve and improve, the potential for deep learning to impact various industries and aspects of life will only continue to grow.

CHAPTER 2

Literature Review

This chapter summarizes some crucial papers that apply machine learning or deep learning techniques for classifying galaxy images. In recent years, these methods have gained popularity in astronomy due to their ability to process large datasets with high accuracy and efficiency. Machine learning and deep learning techniques have been used to classify galaxies based on their shape, size, and color. For instance, some studies have employed Convolutional Neural Networks (CNN) to classify galaxy images based on morphological features [6, 10]. In contrast, others have used Generative Adversarial Networks (GAN) for generating realistic images of galaxies [24]. Using these techniques in astronomy has opened up new avenues for research, allowing scientists to make more accurate predictions and identify previously unknown phenomena. Therefore, it is essential to review the current state of the art in applying machine learning and deep learning techniques to galaxy image classification to understand their potential for advancing the field of astronomy.

2.1 Galaxy Morphology Classification Using Neural Ordinary Differential Equations

The authors of this study [34] developed a method called Neural Ordinary Differential Equations (NODE) to classify galaxy morphology. They applied this method to classify galaxy images from the Galaxy Zoo 2 dataset [47], which includes five different classes. Their results show an accuracy of 92% on the dataset of five galaxy classes. To obtain their dataset, the authors used the Galaxy Zoo Challenge available on Kaggle,

Fig. 6: The five different galaxy morphologies in the Galaxy Zoo-2 dataset. Each row has samples from each galaxy class, from to to bottom they are completely round, in-between, cigar-shaped, edge-on and spiral galaxy (Source: Gupta et al. (2022) [34])

which includes images from SDSS-DR7 and classification labels from Galaxy Zoo 2. The dataset includes five morphological classes: completely round smooth, in-between smooth, cigar-shaped smooth, edge-on, and spiral. These classes are illustrated in Figure 6.

The Kaggle dataset used in this study [34] contains more than 60,000 images sorted into five different classes and assigned a classification probability. The authors filtered the dataset only to include images with high probabilities of being correctly classified, resulting in 28,790 photos distributed among five classes.

The selection process used in this research to filter out the required 28,790 images belonging to five classes from a total of over 60,000 images is similar to the method described by Willett et al. (2013) [99] in which galaxy images with probabilities greater than a specific threshold are chosen. As a result, the authors obtained 7806, 3903, 578, 8069, and 8434 photos in the *completely round, in-between, cigar-shaped, edge-on, and spiral* classes, respectively.

To train a Neural Ordinary Differential Equations (NODE), the parameters of the neural network function must be learned using a loss function, such as cross-entropy, in the case of classification. The ODE solver's learned representation is fed into the loss function and optimized with respect to the parameters. However, learning the parameters requires backpropagation through the solver, which is computationally costly using naive backpropagation. Chen et al. (2018) proposed an adjoint sensitivity method to efficiently compute the gradients with respect to $h(t)$ (the state of the system at any time t) and the neural network parameters by running another ODE solver backward in time.

For the standard NODE training, the authors used a network architecture with a standard convolution block consisting of two Convolution-Non-Linearity-Normalization (CLN) layers, each using a 3×3 kernel. The input is also downsampled before being passed to the ODE network using 2D convolutions, which reduces the number of channels. After passing through the ODE network, a pooling layer is used, and the ODE maps the inputs to a desired latent space with the same number of dimensions as the input. To achieve the final output with five dimensions (equal to the number of classes), a fully connected (FC) layer is used to learn a linear mapping from the ODE output to the final output. The standard NODE model trained with the adjoint method achieved an accuracy of 92% [34].

2.2 Galaxy Morphology Classification using EfficientNet Architectures

To categorize galaxies into seven classes—*completely round smooth, in-between smooth, cigar-shaped smooth, lenticular, barred spiral, unbarred spiral, and irregular.* Figure 7 shows the architecture used in this study, based on the EfficientNet-B0 model, a convolutional neural network (CNN) consisting of multiple layers of convolutional, pooling, and fully connected layers. The network takes an input image of size 224×224 pixels. It passes it through several convolutional layers, each of which applies a set

of filters to the input image to extract features at different levels of abstraction. The output of each convolutional layer is then passed through a batch normalization layer to normalize the features and improve the training stability. After the convolutional layers, the network uses global average pooling to reduce the spatial dimensions of the output feature maps and generate a feature vector for the input image. The feature vector is then passed through several fully connected layers to produce the final output as shown in figure 7, which is a probability distribution over the different morphological classes of galaxies.

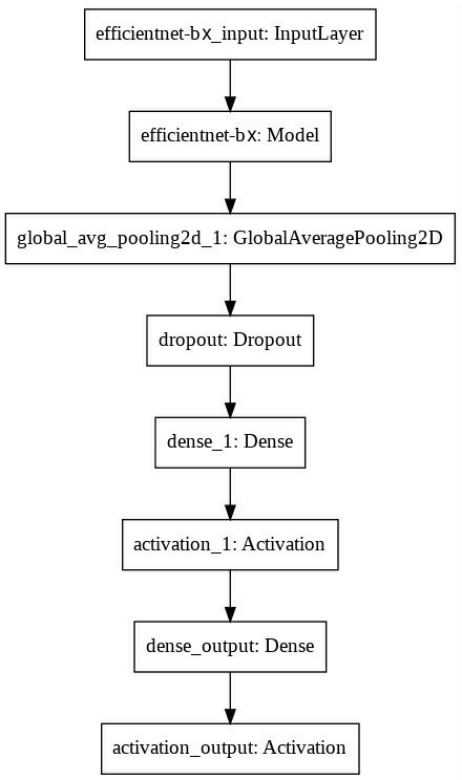


Fig. 7: Network architecture of the tail part of the fine-tuned model. The x in efficientnet-bx is a placeholder for all the EfficientNet architectures ranging between B0-B7

The network, together with other well-known convolutional networks, is used to categorize 29,941 pictures of galaxies. The model's performance is assessed using various metrics, including accuracy, recall, precision, and F1 score, along with a comparison with other cutting-edge convolutional models to see which one performs the best. Their classification model yields an F1 score of 0.8857, and an accuracy of

93.7% [48].

According to the proper thresholds, the study [48] collects clean samples of the galaxies that belong to particular morphologies. These thresholds are based on the number of votes needed to qualify for a classification task [103]. These thresholds are used to identify clean samples in [99]. The methodology was used by Zhu et al. (2019) [103] to divide galaxies into five categories: edge-on, spiral, cigar-shaped smooth, totally round smooth, and in-between smooth (between entirely round and cigar-shaped). Following suit, the authors divide galaxies into seven categories: completely round smooth, in-between smooth, cigar-shaped smooth, lenticular (edge-on), barred spiral, unbarred spiral, and irregular (irregular set is a collection of mergers, disturbed, dust lanes and irregular galaxies). The classes are numbered 0, 1, ..., 6. There are, accordingly, 8107, 7782, 578, 3780, 827, 3307, and 1560 samples in each class. The dataset with 25,941 images is split into a train:test ratio of 9:1. As a result, they [48] have 2,589 testing images and 23,352 training images.

2.3 Machine and Deep Learning Applied to Galaxy Morphology - A Comparative Study

The paper "Machine and Deep Learning Applied to Galaxy Morphology - A Comparative Study" by Barchi et al. [14], published in 2020, focuses on using deep convolutional networks for galaxy morphological classification. The authors aim to demonstrate the potential of the machine and deep learning techniques for advancing our understanding of galaxy morphologies.

The dataset used in this study consists of 67,637 galaxy images which are then divided into three classes: elliptical, barred spiral, and unbarred spiral. The images were collected from various sources, including the Sloan Digital Sky Survey, the Galaxy Zoo project, and the Galaxy Evolution Explorer (GALEX) satellite.

To classify the galaxy images based on their morphologies, the authors used GoogleNet Inception [83], a neural network architecture developed by Google re-

searchers in 2014 for image classification. It uses Inception modules, which consist of multiple convolutional layers with different filter sizes and pooling operations, to capture features of different sizes and resolutions in parallel without increasing the computational cost too much. The model was trained on the galaxy image dataset, and the authors reported an accuracy of approximately 82% for classifying the images into the three morphological classes [14].

This research [14] also compared the performance of GoogleNet Inception with other machine learning techniques, such as decision trees and support vector machines. These comparisons showed that deep convolutional networks outperformed these other techniques, achieving higher accuracy for galaxy morphological classification.

	$K \geq 5$				$K \geq 10$				$K \geq 20$			
	DT	**SVM**	**MLP**	**CNN**	**DT**	**SVM**	**MLP**	**CNN**	**DT**	**SVM**	**MLP**	**CNN**
11 classes	49.3	48.8	49.4	63.0	51.6	51.6	51.7	63.0	57.7	57.4	57.7	65.2
9 classes	60.9	63.2	63.0	70.2	60.5	63.8	63.6	75.7	63.5	66.4	66.2	67.4
7 classes	63.0	62.5	63.3	72.2	62.9	62.6	63.0	77.6	65.9	65.8	66.0	70.0
3 classes	71.9	71.2	71.2	80.8	71.9	74.6	74.9	81.8	78.7	78.5	78.8	82.7

Fig. 8: Overall Accuracy (OA in percentage) for all approaches considering GZ2 classification (the darker the green colour of a cell, the better OA obtained) (Source: Barchi et al. (2020) [14])

Barchi et al. [14] believe that their results (fig. 8) demonstrate the potential of machine learning and deep learning techniques for advancing our understanding of galaxy morphologies. The study provides a valuable contribution to the field of astrophysics, as well as the field of computer vision and machine learning. The authors suggest that future research could focus on using deep convolutional networks to classify galaxy images based on additional morphological classes, such as irregular galaxies, as well as exploring the use of other deep learning techniques, such as recurrent neural networks and generative adversarial networks.

2.4 Classifying Galaxy Morphologies with Few-shot Learning

In the paper "Classifying Galaxy Morphologies with Few-shot Learning," Zhirui Zhang et al. (2022) [102] proposed an approach to classify galaxy images using few-shot learning. The authors believe that few-shot learning is an ideal solution for this problem because it requires only a small number of examples to classify new data. This is particularly relevant for galaxy morphology classification because there is a limited number of labeled examples available.

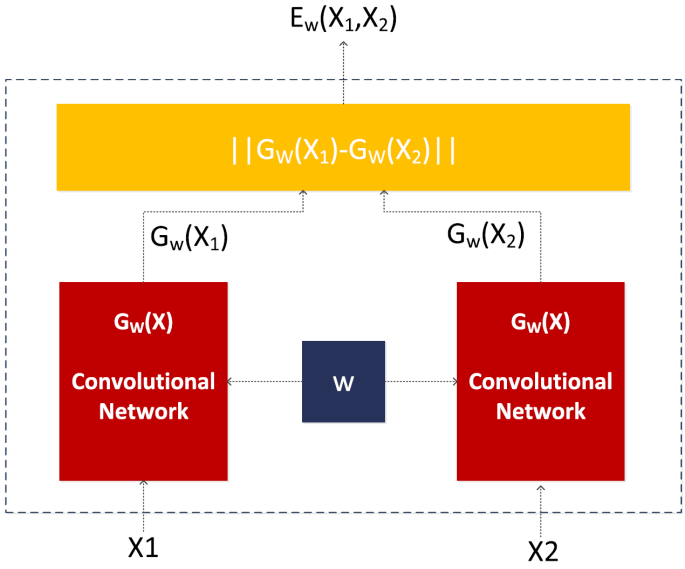


Fig. 9: Siamese Architecture (the left and right input different data, and calculate the similarity between them after feature extraction) (Source: Zhang et al. (2022) [102])

They [102] used a model called SC-Net, which consists of a convolutional neural network (CNN) and a Siamese network (fig. 9). SC-Net is a neural network architecture proposed in 2018 for image classification tasks. It uses spatial and channel-wise attention mechanisms to dynamically adjust the weights of convolutional filters, allowing the network to focus on essential regions of the input image and discard irrelevant features. This helps improve the network's accuracy while reducing its computational cost [102]. The CNN extracts feature from the galaxy images, while the Siamese net-

work compares the similarities between the query image and the support images. The authors believe combining these two components provides a more robust and accurate solution for few-shot learning.

Zhirui Zhang et al. (2022) [102] evaluated the SC-Net model on a dataset of galaxy images, focusing on five distinct classes of galaxy morphologies: completely round, in-between, spiral, cigar-shaped, and edge-on. The authors achieved an accuracy of 90.9% on this dataset, demonstrating the effectiveness of their approach. The SC-Net model provides a solution requiring only a few examples to classify new data.

2.5 Rotation-invariant Convolutional Neural Networks for Galaxy Morphology Prediction

Sander Dieleman et al. (2014) [26] published a paper entitled "Rotation-Invariant Convolutional Neural Networks for Galaxy Morphology Prediction" that explores the use of deep neural network models for galaxy morphology classification. The authors propose a new method for classification that takes advantage of translational and rotational symmetry in the data. This is achieved by using a rotation-invariant convolutional neural network (R-CNN), which is a type of deep learning model that is capable of handling variations in image orientation.

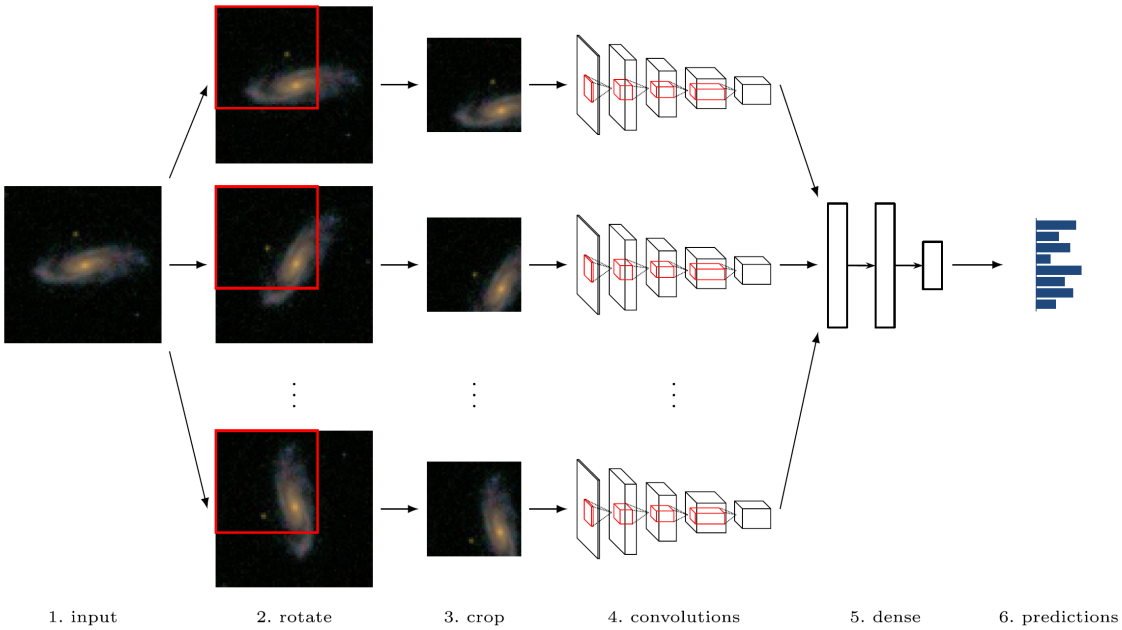


Fig. 10: Schematic overview of a neural network architecture for exploiting rotational symmetry. The input image (1) is first rotated to various angles and optionally flipped to yield different viewpoints (2) and the viewpoints are subsequently cropped to reduce redundancy (3). Each of the cropped viewpoints is processed by the same stack of convolutional layers and pooling layers (4), and their output representations are concatenated and processed by a stack of dense layers (5) to obtain predictions (6) (Source: Dieleman et al. (2014) [26])

The architecture figure 10 shows the neural network architecture used for predicting the morphology of galaxies based on their images. The architecture consists of a series of convolutional and pooling layers followed by two fully connected layers. Sander Dieleman et al. (2014) [26] used a variant of the LeNet-5 architecture as the base architecture for their network. In order to make the network rotation-invariant, they augmented the training data by rotating the images in 90-degree increments and feeding them as separate inputs to the network. They present a detailed description of the R-CNN model and its architecture (fig. 10), including convolutional layers, pooling layers, and activation functions. The authors also describe the training process, which involves using a large dataset of galaxy images to train the network. The network was trained using a supervised learning approach, with the goal of predicting the probabilities associated with each image (pk, where k = 1, ..., 37).

The study [26] evaluates the performance of the R-CNN model by comparing it

to other state-of-the-art models for galaxy morphology classification in terms of Root Mean Squared Error (RMSE), and they achieved an RMSE of 0.0769. In conclusion, the paper by Sander Dieleman et al. (2014) [26] presents a novel method for galaxy morphology classification using a rotation-invariant convolutional neural network (R-CNN). They demonstrate that the R-CNN model outperforms other models, making it a promising tool for galaxy morphology classification. This work provides valuable insights into the potential of deep neural network models for solving complex problems in astronomy. It highlights the importance of considering rotational symmetry in data when designing deep learning models.

2.6 Additional Literature Review

Alawi et al. (2021) [5] proposed a deep residual network (ResNet) model for the classification of galaxies and stars. They preprocessed the data using image augmentation techniques and trained the ResNet model using transfer learning. They achieved an accuracy of 96.77% on the test dataset of 10,000 images belonging to two classes. Shetty et al. (2022) [75] proposed a convolutional neural network (CNN) model for the classification of satellite galaxies. They used the dataset from the galaxy zoo challenge, called the Galaxy Zoo dataset, and trained their model using transfer learning. They achieved a validation accuracy of 77.8% on the test dataset. Vijay et al. (2020) [90] proposed a method for identifying the orientation of galaxies in the Galaxy Zoo dataset using spectral clustering. They first preprocessed the data and extracted features using principal component analysis (PCA). Then, they applied spectral clustering to group the galaxies based on their orientation. They achieved an accuracy of 87.6% on the Galaxy Zoo dataset. Zhu at al. (2019) [103] proposed a deep CNN model for galaxy morphology classification. They used the Galaxy Zoo dataset and trained their model using transfer learning. They achieved an accuracy of 93.12% on the test dataset, which outperformed previous state-of-the-art methods. They also conducted experiments to show the effectiveness of transfer learning and data augmentation in improving the performance of their model.

2.7 Summary on Literature Review

Table 2 summarizes five studies discussed earlier in this chapter that have applied machine learning techniques for galaxy morphology classification. Gupta et al. (2022) achieved 92% accuracy using neural ordinary differential equations on a dataset of 28,790 images with five classes, a subset of the Galaxy Zoo (GZ) dataset. The GZ dataset is available on Kaggle [47]. Kalvankar et al. (2020) achieved 93.7% accuracy using efficientnet architectures on a dataset of 25,941 images with seven classes. Barchi et al. (2020) achieved 82% accuracy using machine and deep learning on a dataset of 67,637 images with three classes. Zhang et al. (2022) achieved 90.9% accuracy using few-shot learning on the same dataset as Gupta et al. (2022). Finally, Dieleman et al. (2015) achieved a root mean squared error of 0.0769 on a dataset of 61,578 images from the Galaxy Zoo Challenge on Kaggle [47] using rotation-invariant convolutional neural networks.

Author	**Dataset Type**	**Dataset Size**	**Result**
Gupta et al. (2022) (Section 2.1)	Five classes (Subset of GZ dataset)	28,790 images	92% accuracy
Kalvankar et al. (2020) (Section 2.2)	Seven classes (Subset of GZ dataset)	25,941 images	93.7% accuracy
Barchi et al. (2020) (Section 2.3)	Three classes (GZ2 project data*)	67,637 images	82% accuracy
Zhang et al. (2022) (Section 2.4)	Five classes (Subset of GZ dataset)	28,790 images	90.9% accuracy
Dieleman et al. (2015) (Section 2.5)	GZ dataset [47]	61578 images	0.0769 root mean squared error

Table 2: Summary of five different studies that have attempted to classify galaxies using machine learning techniques [48, 34, 14, 26, 102]. Each row corresponds to a study, with columns indicating the dataset used in their study and the results achieved in their research. GZ dataset is available on Kaggle and *GZ2 project data is the data available from GZ2 project

While some studies in this field have reported good results, many suffer from limitations, such as using a single dataset to show their results or relying on existing models for comparative analysis. Additionally, certain models proposed in the literature require a large number of training parameters which can lead to overfitting. To address these limitations, this study proposes a model based on the Residual Neural Network (ResNet) architecture. ResNet can achieve high accuracy while reducing computational costs, thanks to the use of skip connections. Moreover, it can generalize well to new, unseen galaxies by learning more complex features. The relatively small number of parameters in ResNet can also help to prevent overfitting, and it can be trained on various datasets, making it more representative of all types of galaxies. In the following chapter, Methodology, we provide further explanation of the ResNet architecture, as well as details of the model proposed in this study.

CHAPTER 3

Methodology

This chapter provides information about the model proposed in this study, called ResNet_Var, which uses ResNet architecture. So, before we move to ResNet_Var, let us learn more about ResNet.

3.1 Residual Neural Networks

Residual Network (ResNet) is a Convolutional Neural Network (CNN) architecture that has revolutionized the field of computer vision by addressing the problem of vanishing gradients, which arise when a neural network becomes too deep. The backpropagation process relies on gradient descent to optimize the weights of the network. However, when there are too many layers, the repeated multiplications cause the gradient to vanish, leading to performance saturation or degradation. ResNet tackles this problem by introducing "skip connections" between layers [36]. Skip connections, also known as shortcut connections, allow information to bypass one or more layers of a neural network and be passed directly to a later layer. In ResNet, skip connections are implemented by adding identity mappings (i.e., a simple linear function) to the output of one or more convolutional layers. In ResNet, identity mapping refers to the addition of the input of a residual block to its output. A residual block is a building block used in residual neural networks (ResNets). In a residual block, the skip connection is typically implemented as a simple identity mapping that bypasses one or more convolutional layers in the block. This means that the input to the block is added directly to the output of the block, which helps to

prevent the vanishing gradient problem that can occur in very deep neural networks. By using skip connections, residual blocks enable the network to learn features that are more abstract and complex than what could be learned without them.

ResNet stacks multiple identity mappings and skips some layers by reusing the activations of the previous layer. This method speeds up the initial training phase by compressing the network into fewer layers and then expanding all layers during retraining. The term "retraining" refers to the process of fine-tuning the weights of the ResNet architecture after the initial training phase. During the initial training phase, the ResNet architecture is compressed into fewer layers by skipping some layers using skip connections. After this phase, the skipped layers are added back in, and the remaining layers are trained again on the data to improve the performance of the network. In most ResNet models, two or three layers are skipped at a time with nonlinearity and batch normalization in between [36].

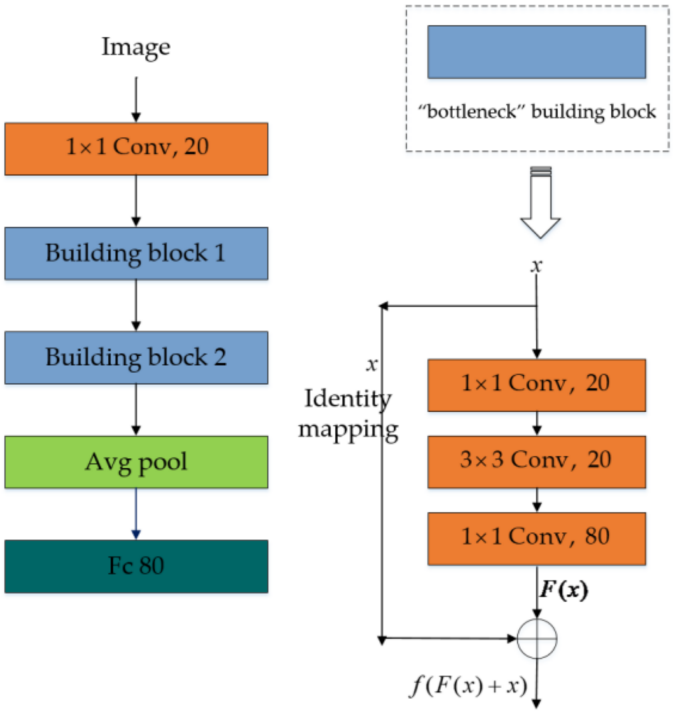
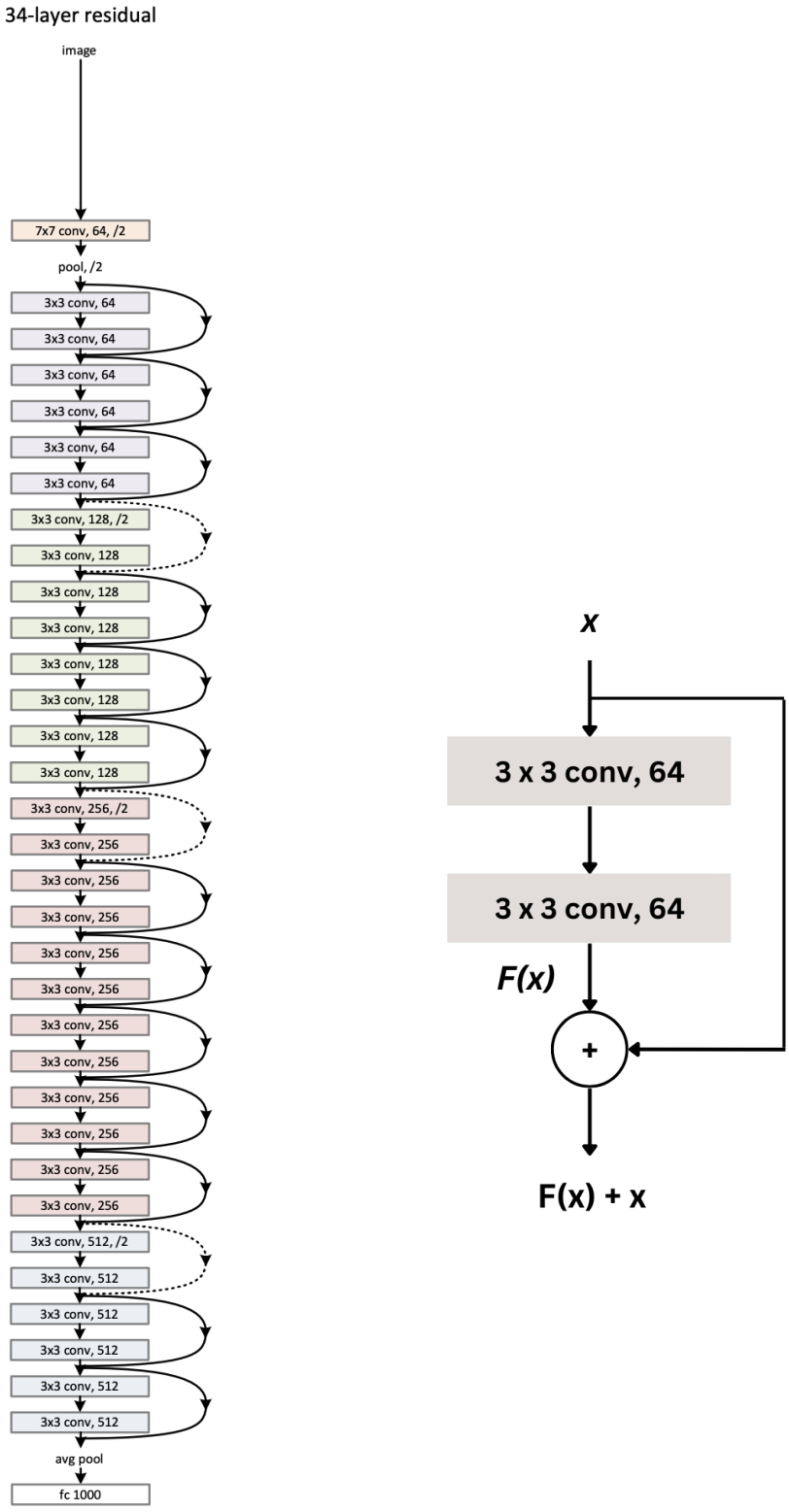


Fig. 11: A residual network model architecture (Source: Fang et al. (2018) [29])

Figure 11 illustrates the model architecture of a simple Residual Network (ResNet) and the structure of the residual block used in the ResNet model. As seen in the left

part of figure 11, the image is given as input to the model, and it is passed to a convolution layer ("Conv" in the figure 11) of 1×1 kernel with a filter size of 20. Then the image is passed through building block 1 followed by building block 2. The building block referred to in the figure 11 is also known as residual blocks. The term "bottleneck" building block in figure 11 refers to a specific type of building block that is designed to reduce the number of parameters and computations required to train a very deep neural network. Output from the building block 2 is given as an input to an average pooling layer ("Avg pool" in the figure 11), and then the output from the average pooling layer is provided as an input to the fully connected layer ("Fc" in the figure 11). Now, the right part of figure 11 shows the structure of the building block. It shows the series of convolutional layers are present inside the building block with varying kernel sizes of 1×1 and 3×3 as well as filter sizes of 20 and 80. As seen in the right part of figure 11, x is the input to the building block, which is the output from the previous layers in the neural network. The output from the convolutional layer, first orange rectangle with "$1 \times 1\ Conv, 20$" written inside in the left part of figure 11, is the input for building block 1, the first blue block with "Building block 1" written inside. The value of the input to the building block, x, is kept aside to use it as identity mapping. The right part of figure 11 shows an equal input x which is not identity mapping is passed through a series of convolution layers, "$1 \times 1\ Conv, 20$", "$3 \times 3\ Conv, 20$", and "$1 \times 1\ Conv, 80$". The output from the series of convolutional layers, $F(x)$ is the residual mapping. The final output from a building block, the original mapping, is represented as $F(x) + x$.

There are a few variants of ResNet, such as ResNet18, ResNet34, ResNet50, ResNet101, and ResNet150. The name of the variant depends on the number of convolution and pooling layers in the model. To have a clear understanding of ResNet architecture, let us take an example of ResNet34. ResNet34 [36] is a variant of the ResNet family of CNN architectures, which is designed to address the problem of vanishing gradients in very deep networks. ResNet34 has 34 layers and uses residual connections to skip over some layers, allowing the network to learn more easily and effectively. The model consists of several residual blocks (building blocks), each



(a) ResNet34 (b) Residual Block

Fig. 12: ResNet34 model architecture

containing multiple convolutional layers, batch normalization, and ReLU activation, followed by a downsampling layer that reduces the spatial dimensions (further details on layers/components of a neural network will be discussed in Section 3.3). The last layer of the network is a fully connected layer that outputs the class probabilities. ResNet34 has achieved state-of-the-art performance on several image classification benchmarks, including ImageNet [36].

ResNet34 architecture, shown in figure 12 (a), consists of a sequence of residual blocks. In figure 12 (a), after the first convolutional layer, "$7 \times 7\ Conv, 64$", there is a skip connection after every two convolutional layers. The combination of the layers between every skip connection is called a residual block. The structure of the residual block is shown in figure 12 (b), where each block contains multiple convolutional layers with 3×3 filters and skip connections that bypass some of the layers. It can be seen in figure 12 (a) that the filter size of convolutional layers in the model increases after a few layers, thus the dotted lines in figure 12 (a) represent the increase in filter size for convolutional layer. This architecture was designed to address the problem of vanishing gradients in very deep neural networks and to investigate whether residual connections improve the performance of deep neural networks.

The next section will provide information on the model proposed in this study, called ResNet_Var.

3.2 ResNet_Var Architecture

The model architecture, ResNet_Var, for this research is a variant of Residual Neural Network (ResNet) as shown in the Figure 13 that utilizes several techniques to improve performance and reduce overfitting.

The model takes an input shape and number of classes as parameters and uses the Keras functional API to build the architecture. The shape of an image is given as $height \times width \times colorchannels$, and the number of classes is the number of classifications of the training dataset. To train the model for image classification, the image is passed as an input to the model, and that image passes through various

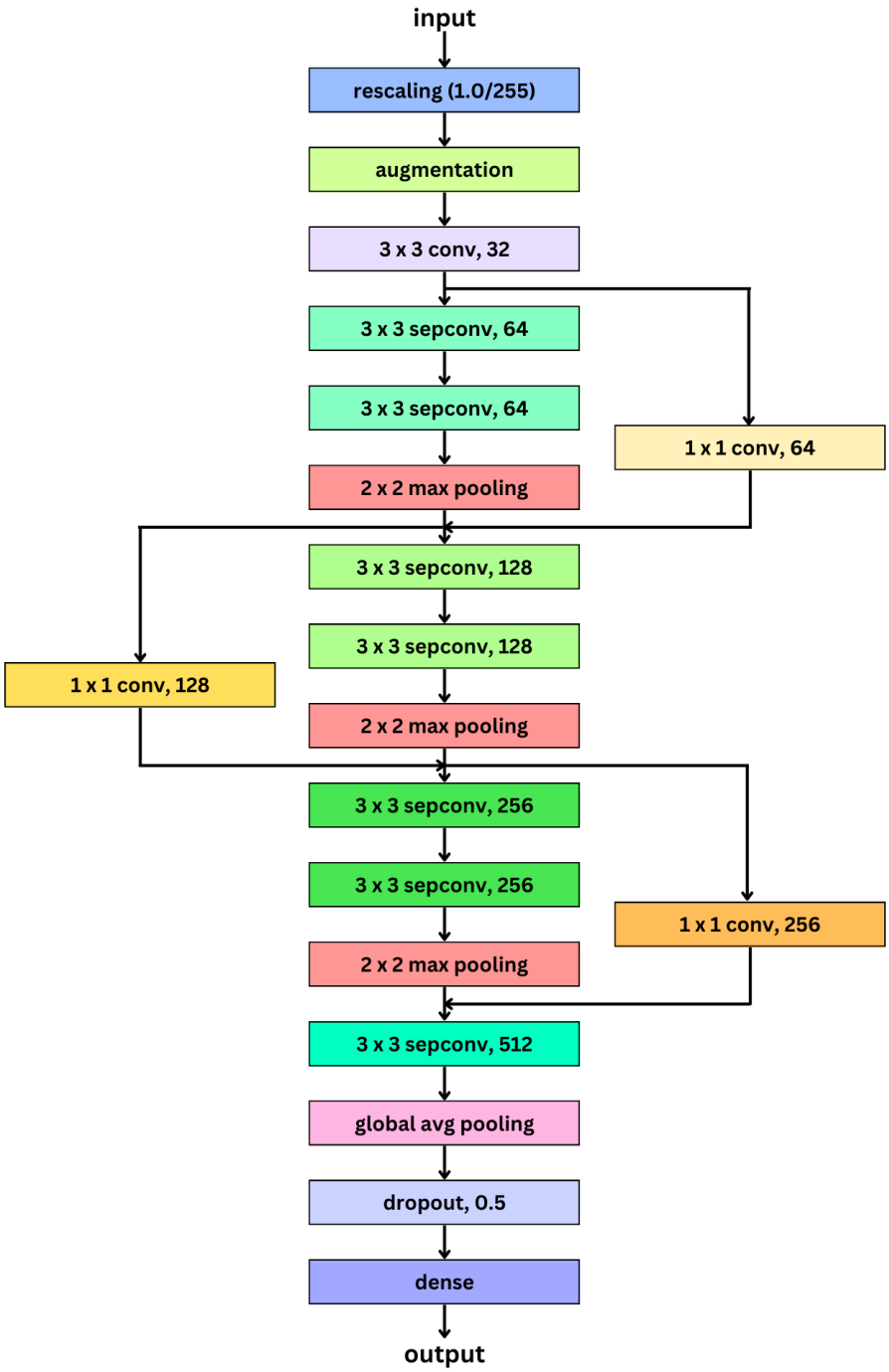


Fig. 13: Architecture of the ResNet_Var proposed in this research

layers before the image can be categorized into a class. At the begging of the training process, the input image is passed to a rescaling layer, shown in figure 13, which scales the input data by dividing it by 255. This is a common preprocessing step in

image classification tasks and helps ensure that the input data is in a consistent range. Then, as illustrated in figure 13, the image passes through a data augmentation layer, which applies a set of random transformations to the input data to increase the size of the training dataset. This is important for deep ResNets, as they are particularly susceptible to overfitting when trained on small datasets. The pipeline defined in this architecture includes three different types of data augmentation: random flipping, random rotation, and random zoom. The *RandomFlip* layer randomly flips the input images horizontally or vertically. The *RandomRotation* layer randomly rotates the images by a specified angle range, in this case between 0 and 1 radians. The *RandomZoom* layer randomly zooms into the images by a specified range, in this case between -0.1 and -0.4. By applying these types of data augmentation, the model is trained on a wider range of data, making it more robust to different variations and improving its ability to generalize to new examples. This can lead to better performance on the test set and in real-world applications.

In figure 13, we can see that the model then uses a convolutional layer ("conv" in figure 13) which is shown as "3×3 $conv, 32$" where "3×3" is the kernel size and "32" is the filter size for the convolutional layer. This layer uses the ReLU activation function, which is a common choice for Neural Networks because it helps improve performance and reduce the risk of overfitting. The activation function is applied every time the image is passed through a convolutional layer. As such, it is not shown separately in figure 13.

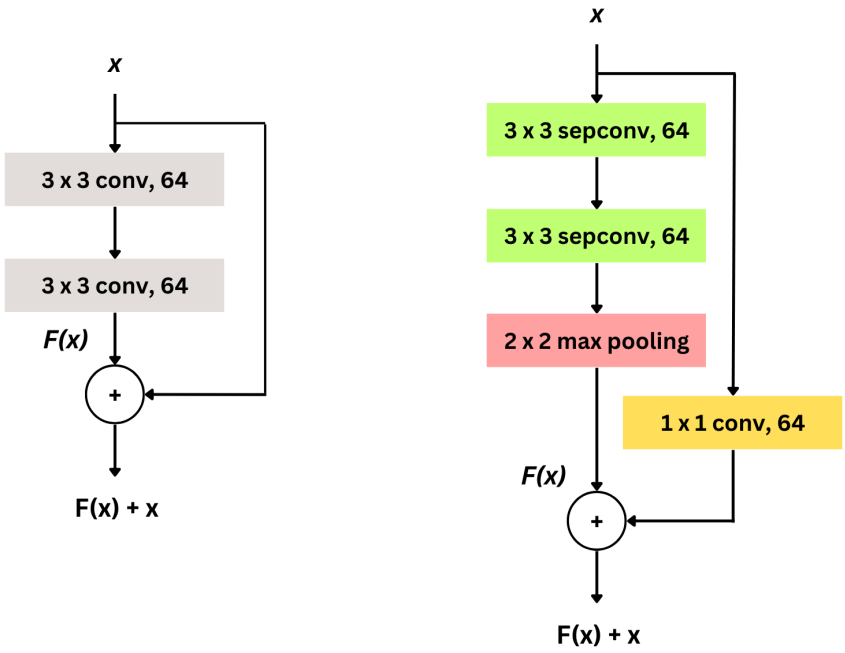
The image is passed through the residual block three times, with a different filter size in each iteration. Inside the residual block, illustrated in figure 14 (b), the model uses two Separable Convolution Layers ("sepconv" in figure 13) of kernel size k=3, with ReLU activation function and a filter size of 64, 128, and 256 in each respective iteration. In figure 13, the first separable convolutional layer is shown as "3×3 $sepconv, 64$" where "3×3" is the kernel size and "64" is the filter size for the separable convolutional layer, this structure is followed throughout the model architecture. Separable Convolution Layers are a more efficient variant of traditional convolutional layers, which is useful when working with large datasets or running

the model on resource-constrained devices. The model also uses batch normalization layers to normalize the activations of the previous layer, which helps to reduce the internal covariate shift, improving performance and reducing overfitting. After the Separable Convolution layers, the model applies a max pooling layer to reduce the spatial dimensions of the input data, which reduces the number of parameters in the model and helps control overfitting. In addition to the Separable Convolution layers, the model also uses a convolution layer of kernel size k=1 in the residual connection with ReLU activation. The model uses residual connections, as shown in figure 14, to add the output of the previous block to the output of the current block, preserving information from previous layers and improving performance. The filter size for both the separable convolution layers and the convolution layer remains the same during each iteration of the residual block, which is iterated three times with filter sizes of 64, 128, and 256.

Figure 14 shows the difference between the residual blocks used in ResNet34 and the residual blocks used in ResNet_Var. The residual block in ResNet_Var uses separable convolutional layers instead of normal convolutional layers, and we can also see that the residual block of ResNet_Var uses a max pooling layer, which can not be seen in the residual block of ResNet34. The skip connection in ResNet34 simply skips a series of convolutional layers, whereas the skip connection in ResNet_Var uses a convolutional of 1×1 kernel. The convolutional layer is added in the skip connection so that no information is ignored during the training process.

After iterating through the residual block three times with filter sizes 64, 128, and 256 in each iteration, we can see, in figure 13, that the model uses a separable convolutional layer of filter size of 512 and kernel, k=3 with ReLU activation function. The model also uses global average pooling layers, which reduce the spatial dimensions of the input data to a single value. This is useful for image classification tasks, as it allows the model to focus on the global features of the input data.

Finally, the model uses a dropout layer, which randomly sets some of the weights to zero during training. This helps to reduce overfitting by preventing the model from relying too heavily on any one neuron. The final output layer is a dense layer with



(a) ResNet34 (b) ResNet_Var

Fig. 14: Structure of residual block used in ResNet34 (a) and ResNet_Var (b)

a number of units equal to the number of classes and uses the softmax activation function.

The model proposed in this study, ResNet_Var, uses separable convolution layers instead of regular convolution layers. Separable convolution layers (more details on components of the model will be discussed in the next section, Section 3.3) consist of a depthwise convolution layer followed by a pointwise convolution layer. This reduces the number of parameters in the model, which can make it easier and faster to train. Additionally, separable convolution layers can learn more diverse feature representations and can improve the generalization of the model. ResNet_Var has a more diverse block structure than ResNet34 [36], a variant of the ResNet. ResNet34 uses identical blocks throughout the network, while ResNet_Var uses blocks of varying

filter sizes (64, 128, 256). This allows ResNet_Var to capture features at multiple scales, which can improve the accuracy of the model. ResNet_Var has a smaller number of layers than ResNet34. ResNet34 has 34 layers, while ResNet_Var has fewer layers. This can make ResNet_Var faster to train and can reduce the risk of overfitting.

The ResNet_Var model proposed in this study has several advantages over the models discussed in the papers mentioned in the literature review section. First, it uses a ResNet architecture that has been shown to perform well in various computer vision tasks. Second, it uses separable convolutions to reduce the number of parameters, which helps to prevent overfitting and reduce computational costs. Third, it uses global average pooling, which reduces the number of parameters and helps to prevent overfitting. Finally, the ResNet_Var model is relatively simple and easy to implement, making it suitable for a wide range of galaxy morphology classification tasks.

The next part gives explanation regarding all the components used in ResNet_Var, such as convolution layer, separable convolution layer, ReLU activation, dense layer and dropout layer.

3.3 Components of ResNet_Var

3.3.1 Convolution Layer

The Convolution Layer is a crucial component in Neural Network models, which are widely used for image classification and other computer vision tasks. The Convolution Layer is responsible for learning local features or patterns in an image and then transforming the input image into a feature map, which is used as input to subsequent layers in the network. The main operation performed in a Convolution Layer is convolution, which involves applying a set of filters (also known as kernels) to the input image. Figure 15 shows that each filter is a small matrix that slides over the image and performs element-wise multiplications between the values in the filter and

the values in the corresponding portion of the input image. The results of these multiplications are then summed up and passed through a non-linear activation function (such as ReLU) to produce a new set of values for each position in the feature map.

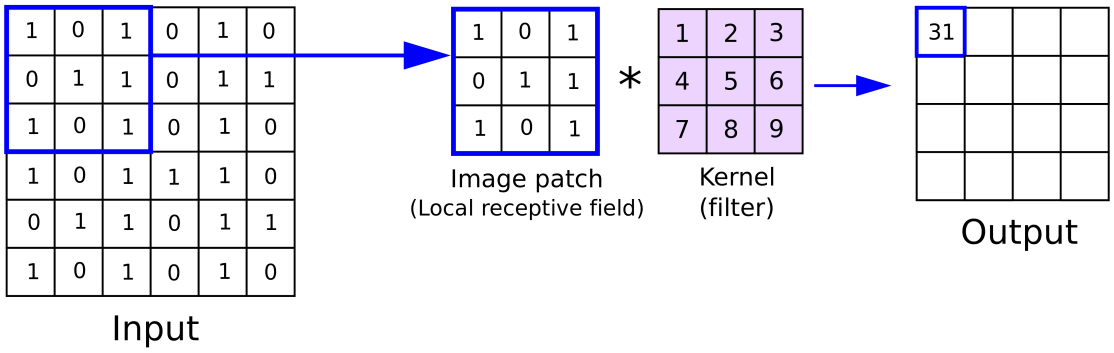


Fig. 15: Convolutional layer in action (Source: Reynolds et al. (2019) [67])

In a Convolution Layer, multiple filters are used, and each filter is responsible for detecting different patterns or features in the image. For example, one filter may detect edges, while another filter may detect textures, and so on. This allows the Convolution Layer to learn a rich set of features from the input image, which can then be used by subsequent layers in the network to make accurate predictions.

Another important aspect of Convolution Layers is the use of strides and padding. Strides control the step size of the filter as it slides over the image, and padding controls the size of the feature map by adding extra pixels to the edges of the input image. Strides and padding are two techniques used in Convolutional Neural Networks (CNNs) to control the size of the feature map while preserving important information from the input image.

Overall, the Convolution Layer is a powerful and flexible component of CNN models that is essential for learning and extracting meaningful features from the input image. By combining multiple Convolution Layers with other types of layers (such as Pooling Layers and Fully Connected Layers), CNN models can achieve remarkable results in image classification and other computer vision tasks.

3.3.2 Separable Convolution Layer

Separable convolution was introduced by Google researchers [39] to reduce the computational cost of performing convolutions in deep neural networks. A separable convolution layer in a neural network model is a type of convolution layer that performs convolution operations in a more efficient manner [91]. A regular convolution layer performs the convolution operation by applying a filter to each region of the input image, resulting in a large number of parameters that need to be learned. Separable convolutions, on the other hand, perform convolution operations in two separate stages. The first stage involves applying a depthwise convolution, which applies a separate filter to each channel of the input image. The second stage involves applying a pointwise convolution, which combines the outputs of the depthwise convolution into a single feature map (figure 16).

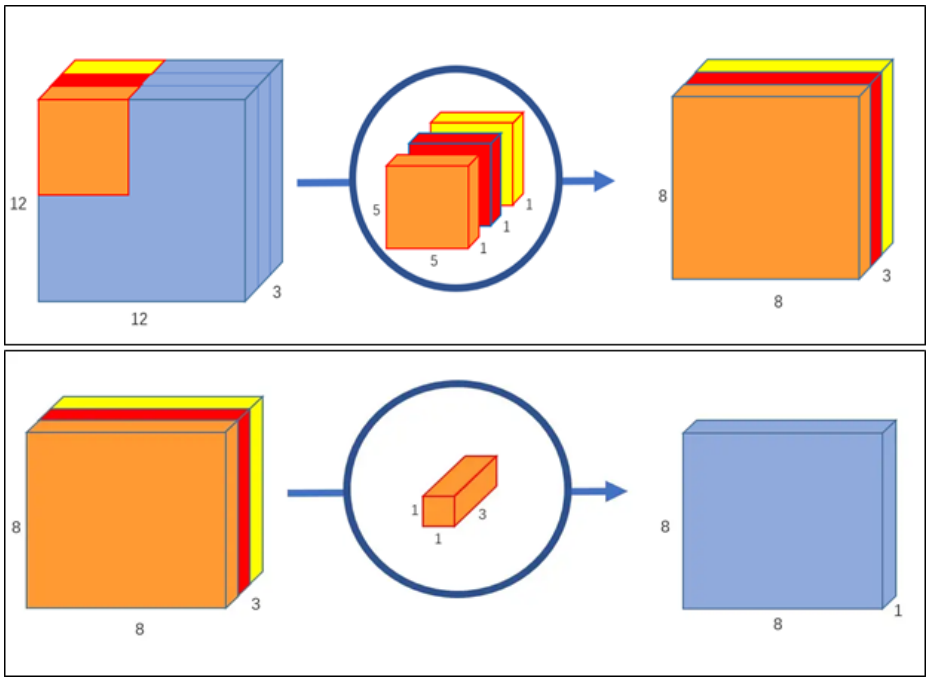


Fig. 16: Depthwise convolution, uses 3 kernels to transform a $12 \times 12 \times 3$ image to an $8 \times 8 \times 3$ image (top), Pointwise convolution, transforms an image of 3 channels to an image of 1 channel (bottom) (Source: Wang et al. (2018) [91])

This separation of the convolution operation into two stages results in a significant reduction in the number of parameters that need to be learned, making separable

convolutions more computationally efficient than regular convolutions. Additionally, separable convolutions have been shown to produce good results in a variety of image classification tasks, making them a popular choice for many neural network models. In summary, a separable convolution layer in a neural network model performs the convolution operation in a more efficient manner by separating the operation into two stages: a depthwise convolution and a pointwise convolution. This separation results in a reduction in the number of parameters that need to be learned, making separable convolutions more computationally efficient and a popular choice for many neural network models [91].

3.3.3 Rectifier Linear Unit (ReLU) Activation

The Rectified Linear Unit (ReLU) activation layer is a commonly used activation function in neural network models. It is a non-linear activation function that is used to introduce non-linearity into the model, allowing the network to learn more complex representations of the data. The ReLU activation layer is defined as follows: if the input to the activation layer is positive, the output is equal to the input, and if the input is negative, the output is set to zero. This activation function can be written mathematically as:

$$f(x) = max(0, x) \tag{1}$$

The ReLU activation function is advantageous in several ways:

- Computationally efficient: The ReLU activation function is computationally efficient as it involves only a simple comparison operation and requires no exponentiation or logarithmic calculations.
- Avoiding vanishing gradients: The ReLU activation function avoids the vanishing gradient problem, where the gradients in the network become very small, making it difficult for the network to learn.
- Sparse activations: The ReLU activation function results in sparse activations,

where only a small portion of the neurons are activated at a given time, making it easier for the network to learn relevant features.

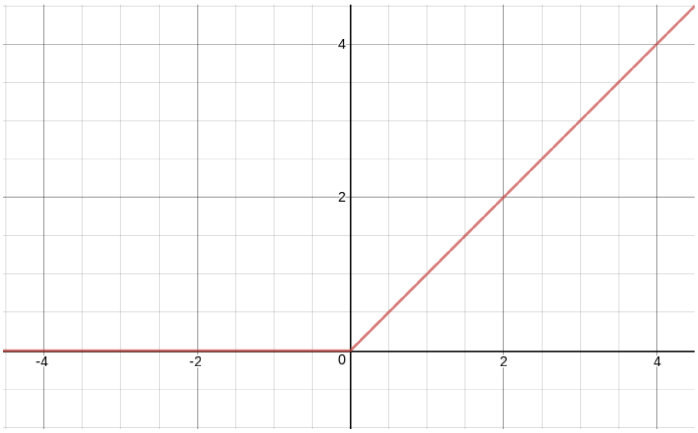


Fig. 17: The Rectified Linear Unit (ReLU) activation function produces 0 as an output when $x < 0$, and then produces a linear with slope of 1 when $x > 0$. (Source: Agarap et al. (2018) [3])

The ReLU activation layer is typically added after each dense layer in a neural network model, allowing the network to learn non-linear relationships between the input and output. The ReLU activation function can also be combined with other activation functions, such as the sigmoid or tanh functions, to achieve better performance in some cases. In conclusion, the ReLU activation layer is a commonly used activation function in neural network models that allows the network to learn non-linear representations of the data. The ReLU activation function is computationally efficient, avoids the vanishing gradient problem, and results in sparse activations, making it a popular choice for many neural network models.

3.3.4 Dense Layer

A dense layer, also known as a fully connected layer, is a common layer type in a neural network model. The dense layer is used to connect the inputs of the network to the outputs through a series of neurons or nodes. Each neuron in a dense layer receives input from all the neurons in the previous layer, and computes a weighted sum of these inputs. The weighted sum is then passed through an activation function,

such as the sigmoid, tanh, or ReLU activation function, to produce the output of the neuron. The activation function adds non-linearity to the model, allowing it to learn more complex representations of the data. The dense layer is parameterized by a weight matrix and a bias vector, which are learned during the training process. The weight matrix determines the strength of the connections between the neurons, and the bias vector shifts the activation function to the left or right. The weights and biases are updated during the training process to minimize the difference between the predicted outputs and the true outputs.

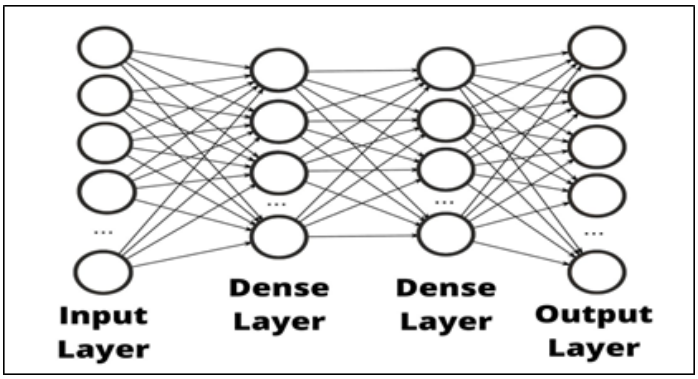


Fig. 18: Fully connected Neural Network (Source: Canu et al. (2022) [19])

In a neural network model, dense layers are typically used to increase the model's capacity and representational power. The number of neurons in a dense layer determines the model's capacity, and the number of dense layers determines the model's representational power. In conclusion, the dense layer is a common layer type in a neural network model that connects the inputs of the network to the outputs through a series of neurons. Each neuron in a dense layer computes a weighted sum of the inputs, which is then passed through an activation function to produce the output. The dense layer is parameterized by a weight matrix and a bias vector, which are learned during the training process to minimize the difference between the predicted outputs and the true outputs.

3.3.5 Dropout Layer

The Dropout layer is a regularization technique used in neural network models to prevent overfitting. Overfitting occurs when a model fits the training data too closely, resulting in poor generalization to new data. Dropout is a simple yet effective way to prevent overfitting by randomly dropping out neurons during the training process. In the Dropout layer, a certain percentage of neurons are randomly dropped out during each iteration of the training process. This means that during each iteration, some neurons are temporarily disabled, preventing the network from relying too heavily on any one neuron. The result is a network that is less sensitive to the input data and is less likely to overfit the training data. The Dropout layer is typically added between the dense layers in a neural network model. The percentage of neurons that are dropped out during each iteration is a hyperparameter that must be set prior to training the network. A common choice for this parameter is to set it between 0.2 and 0.5, meaning that between 20% and 50% of the neurons are dropped out during each iteration. The Dropout layer is only used during the training process, as during inference, all neurons are used, and no neurons are dropped out. This ensures that the network is using all of the learned information during the testing process. In conclusion, the Dropout layer is a regularization technique used in neural network models to prevent overfitting. The Dropout layer randomly drops out neurons during the training process, preventing the network from relying too heavily on any one neuron. The percentage of neurons that are dropped out during each iteration is a hyperparameter that must be set prior to training the network, and the Dropout layer is only used during the training process.

3.3.6 Hyperparameters

Hyperparameters are the variables that determine the structure of a classification model, such as the number of hidden units, and the variables that control how the network is trained, such as the learning rate. These hyperparameters must be set before the model is trained. Some common hyperparameters that are tuned during

the training process of Neural Network are:

- Kernel/Filter: A kernel, also known as a filter, is a matrix used to carry out the convolution operation in the Convolution layer. In this case, a 3x3 kernel is used for training the model. Smaller filters will gather more local information, while larger filters will gather more global and representative information.
- Stride: The stride is the number of pixels to skip while traversing the input horizontally and vertically during convolution. This helps to reduce the size of the input image by a considerable amount after the convolution operation.
- Number of Channels: This is equal to the number of color channels for the input image. Later, it becomes equal to the number of filters used for the convolution operation. The more channels, the more filters used, the more features learned, and the higher the risk of overfitting.
- Pooling-layer Parameters: The Pooling layer has similar parameters to the Convolutional layer. Just like the kernel, it convolves through the feature map generated after the convolution operation. In this model, a 2x2 size max pooling operation is used to reduce the dimensionality of the image.

3.3.7 Trainable Parameters

Trainable parameters in neural network models are the weights and biases of the model that are learned during training. They are used to make predictions based on the input data and are updated during training to minimize the error between the predicted and actual outputs. When there are a large number of parameters in a neural network model, it means that the model has a high capacity or the ability to fit a wide range of functions. This can be beneficial for complex problems, as it allows the model to capture more information from the input data. However, when the number of parameters is too large, the model may become overfitting, which means that it memorizes the training data and perform poorly on unseen data. When there are a small number of parameters in a neural network model, it means that the model

has a low capacity or the ability to fit a limited range of functions. This can be beneficial for simple problems, as it reduces the risk of overfitting. However, when the number of parameters is too small, the model may become underfitting, which means that it is not able to capture enough information from the input data, and perform poorly in general. In general, it's important to find a balance between the number of parameters and the complexity of the problem. A model with a large number of parameters may be overfitting, but one with a small number of parameters may be underfitting. Techniques like regularization and early stopping can be used to help find the optimal number of parameters for a given problem.

3.4 Other Applications of ResNet_Var

ResNet_Var is a modified version of the original ResNet architecture and can be used for image classification tasks other than classifying galaxy morphologies. Here are some of the applications of ResNet_Var model:

- **Image Classification:** ResNet_Var can be trained on a labeled dataset of images to classify them into different categories or classes. The final dense layer with softmax activation makes it suitable for multi-class classification tasks.
- **Object Recognition:** By training ResNet_Var on a dataset with labeled images of various objects, it can be used to recognize and classify objects within images.
- **Fine-tuning Pretrained Models:** ResNet_Var can be used as a base architecture for transfer learning. By loading pre-trained weights from a ResNet model trained on a large dataset like ImageNet, we can fine-tune the model on a smaller dataset specific to our task. This approach is useful when we have limited training data.
- **Feature Extraction:** ResNet_Var can also be used as a feature extractor. By removing the final dense layer and keeping the rest of the network intact, we

can extract meaningful feature representations from images. These features can be used for tasks like clustering, similarity matching, or downstream tasks in a larger pipeline.

It's important to note that the specific use case and performance of the model will depend on the dataset it is trained on and the specific problem at hand. Additionally, the model can be further customized and modified to suit specific requirements.

CHAPTER 4

Experiments and Results

This research evaluated ResNet_Var in addition to four popular Convolutional Neural Network (CNN) models - VGG16, VGG19, ResNet50, and Inception - on Dataset D1 and Dataset D2 (additional details on the datasets will be covered in section 4.1), which consist of five and seven distinct classes, respectively. The proposed model, ResNet_Var, was trained and compared with the popular CNN models mentioned above using commonly used metrics in computer vision and machine learning, including Precision, Recall, F1-score, and Accuracy. Additionally, this study compared ResNet_Var's performance on Dataset D1 with the models proposed by Gupta et al. (2022) [34] and Zhang et al. (2022) [102], as they also used Dataset D1. Similarly, the performance of ResNet_Var on Dataset D2 was compared with the model proposed by Kalvankar et al. (2020) [48], as they used Dataset D2.

The models were trained and tested on a system with an AMD Ryzen 7 3700X 8-Core Processor clocked at 3.60 GHz. The system has 32 GB RAM, 64-bit OS, and an x64 processor architecture for demanding computational tasks. Additionally, an NVIDIA GeForce RTX 3070 GPU with 8192 MB display memory enhances the system's performance for graphical computations. Overall, these system specifications provide good performance, efficiency, and scalability for training and testing the models to their potential.

4.1 Datasets

Gathering and organizing the data is an essential part of training a neural network model. The raw data is collected from a competition launched on Kaggle in December 2013 [47]. It is a dataset, called Galaxy Zoo dataset, on Kaggle that provides information on various galaxies in the universe. The dataset contains information on the morphological properties of over 79,000 galaxies from the Sloan Digital Sky Survey (SDSS) and is sourced from the Galaxy Zoo project. The Galaxy Zoo project is a citizen science project that involves the public in classifying the shapes of galaxies from digital images.

The dataset includes a wide range of information, including:

- **Galaxy ID**: A unique identifier for each galaxy in the dataset.
- **RA (Right Ascension)**: The position of the galaxy in the sky in terms of its right ascension. Right ascension is a coordinate used in astronomy to indicate the east-west position of a celestial object in the sky, measured in hours, minutes, and seconds.
- **Dec (Declination)**: The position of the galaxy in the sky in terms of its declination. Declination is a celestial coordinate that indicates the position of an object in the sky relative to the celestial equator. It is measured in degrees and is the angular distance of an object above or below the celestial equator.
- **Plate**: The plate number of the galaxy, which refers to the image used to classify the galaxy.
- **FiberID**: The fiber identifier of the galaxy, which is a unique identifier for each spectrum in the SDSS.
- **z**: The redshift of the galaxy, which is a measure of how far away the galaxy is from us.

- **T01_smooth_or_features_a01_smooth**: A classification of the galaxy's appearance, where 1 indicates a smooth appearance, and 0 indicates a feature-rich appearance.

- **T01_smooth_or_features_a02_features_or_disk**: A classification of the galaxy's appearance, where 1 indicates a feature-rich or disk-like appearance, and 0 indicates a smooth appearance.

- **T02_edgeon**: A classification of the galaxy's edge, where 1 indicates an edge-on orientation, and 0 indicates a face-on orientation.

- **T03_bar_a06_bar**: A classification of the galaxy's bar, where 1 indicates the presence of a bar, and 0 indicates the absence of a bar.

- **T04_spiral_a08_spiral**: A classification of the galaxy's spiral pattern, where 1 indicates the presence of a spiral pattern, and 0 indicates the absence of a spiral pattern.

- **T05_bulge_prominence_a10_no_bulge_prominence**: A classification of the galaxy's bulge prominence, where 1 indicates the absence of a bulge prominence, and 0 indicates the presence of a bulge prominence.

This dataset provides a wealth of information for researchers studying galaxy morphology, as well as for anyone interested in the classification of galaxies. It provides a large sample of galaxies with detailed morphological classifications, making it a valuable resource for astronomers and anyone interested in exploring the universe.

Gupta et al. (2022) [34] utilized a specific subset of the Galaxy Zoo dataset, which was previously mentioned in this section. The dataset used by the researchers [34] is composed of galaxy images that are categorized into five distinct groups, namely spiral, edge-on, cigar-shaped, in-between, and completely round. The study conducted by Kalvankar et al. 2020 [48] also utilized a subset of the same Galaxy Zoo dataset previously mentioned. In this particular study [48], the galaxy images are categorized into seven groups, namely irregular, unbarred spiral, barred spiral, lenticular

(edge-on), cigar-shaped, in-between, and completely round. Both studies utilized the Galaxy Zoo dataset as a starting point, but categorized the galaxy images in different ways to suit their research goals.

In the present study, we utilize a dataset denoted as **D1**, which is identical to the one used in recent research by Gupta et al. (2022) [34]. This dataset consists of a large collection of astronomical images that have been categorized into five distinct classes based on their visual features. The categories are as follows: completely round, in-between, cigar-shaped, edge-on, and spiral. The distribution of the images in each class is shown in Table 3, with 8434, 8069, 578, 3903, and 7806 images, respectively, for each category.

It is worth noting that the classification of these images into their respective categories was performed by expert astronomers, who employed their expertise and experience to identify and categorize each object based on its visual appearance. The use of a pre-existing dataset in this study allows for comparisons and benchmarking against previous studies, providing a basis for further advancements in the field of astronomical image classification.

Fig. 19: Galaxy image from each class in dataset D1, (from left to right) Cigar shaped, Completely round, Edge-on, In-between, Spiral (Source: Kaggle Dataset [47])

The dataset known as **D2** is the same as the dataset employed by Kalvankar et al. in their 2020 publication [48]. This dataset is composed of galaxy images that have been categorized into seven distinct groups, namely completely round, in-between, cigar-shaped, lenticular, barred spiral, unbarred spiral, and irregular. The number of images belonging to each category is displayed in Table 4. It is evident from the

Class	Sample	Tasks	Selection	$\mathbf{N}_{sample}$
0	Completely round	T01	$f_{smooth} >= 0.469$	8434
		T07	$f_{completelyround} >= 0.50$	
1	In-between	T01	$f_{smooth} >= 0.469$	8069
		T07	$f_{in-between} >= 0.50$	
2	Cigar-shaped	T01	$f_{smooth} >= 0.469$	578
		T07	$f_{cigar-shaped} >= 0.50$	
3	Edge-on	T01	$f_{feature/disk} >= 0.430$	3903
		T02	$f_{edge-on,yes} >= 0.602$	
4	Spiral	T01	$f_{smooth} >= 0.469$	7806
		T02	$f_{edge-on,no} >= 0.715$	
		T04	$f_{spiral,yes} >= 0.619$	

Table 3: Clean samples selection in Galaxy Zoo 2. The clean galaxy images are selected from Galaxy Zoo 2 data release (Willett et al. 2013 [99]), in which thresholds determine well-sampled galaxies. (f_{class} is probability of the image being in a galaxy class/category, N is the number of images in each class)

table that the number of images assigned to each category is not uniform, with the completely round group having the highest number of images at 8107, followed by the in-between category with 7782 images. The cigar-shaped group contains the fewest images at 578, whereas the lenticular, barred spiral, unbarred spiral, and irregular categories have 3780, 872, 3307, and 1560 images, respectively.

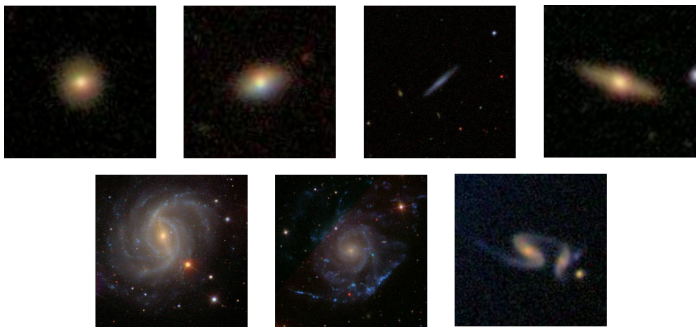


Fig. 20: Galaxy image from each class in dataset D2, (from left to right) Completely round, In-between, Cigar shaped, Lenticular, Barred spiral, Unbarred spiral, Irregular (Source: Kaggle Dataset [47])

Class	Sample	Tasks	Selection	$\mathbf{N}_{sample}$
0	Completely round	T01	$f_{smooth} >= 0.469$	8107
		T07	$f_{completelyround} >= 0.50$	
		T06	$f_{odd,no} >= 0.50$	
1	In-between	T01	$f_{smooth} >= 0.469$	7782
		T07	$f_{in-between} >= 0.50$	
		T06	$f_{odd,no} >= 0.50$	
2	Cigar-shaped	T01	$f_{smooth} >= 0.469$	578
		T07	$f_{cigar-shaped} >= 0.50$	
		T06	$f_{odd,no} >= 0.50$	
3	Edge-on	T01	$f_{feature/disk} >= 0.430$	3780
		T02	$f_{edge-on,yes} >= 0.602$	
		T06	$f_{odd,no} >= 0.50$	
4	Barred spiral	T01	$f_{feature/disk} >= 0.430$	827
		T02	$f_{edge-on,no} >= 0.715$	
		T02	$f_{bar,yes} >= 0.715$	
		T04	$f_{spiral,yes} >= 0.619$	
4	Unbarred spiral	T01	$f_{feature/disk} >= 0.430$	3307
		T02	$f_{edge-on,no} >= 0.715$	
		T02	$f_{bar,no} >= 0.715$	
		T04	$f_{spiral,yes} >= 0.619$	
6	Irregular	T01	$f_{odd,yes} >= 0.420$	1560
		T04	$f_{disturbed/irregular/other/merger/dustlane} >= 0.50$	

Table 4: Clean samples selection in Galaxy Zoo 2. The clean galaxy images are selected from Galaxy Zoo 2 data release (Willett et al. 2013 [99]), in which thresholds determine well-sampled galaxies.(f_{class} is probability of the image being in a galaxy class/category, N is the number of images in each class)

After generating datasets from the GZ dataset, we need to split the data into 9:1 ratio for training and testing the model proposed in this study, 90% of the dataset being used for training and 10% of the dataset being used for testing. This distribution applies to both datasets, Dataset D1 and Dataset D2. The two tables (Table 5 and Table 6)provide information about image datasets categorized based on galaxy class. The first table (Table 5), Dataset D1, consists of 5 classes of galaxies, namely Cigar shaped, Completely round, Edge-on, In-between, and Spiral. The second table (Table 6), Dataset D2, consists of 7 classes, namely Completely round, In-between, Cigar shaped, Lenticular, Barred spiral, Unbarred Spiral, and Irregular.

For each class, the tables show the number of images in the training set and testing set. The rows are color-coded with alternating shades of blue and light gray

to make it easier to read and differentiate between different classes. The total number of images in each dataset is provided at the bottom of each table.

Class No.	Galaxy Class	Training Images	Testing Images
0	Completely round	7,025	781
1	In-between	7,262	807
2	Cigar shaped	520	58
3	Edge-on	3,513	390
4	Spiral	7,591	843
	Total	25,911	2,879

Table 5: Image distribution of **Dataset D1** for training and testing

In Dataset D1, the Spiral class has the highest number of images in both the training and testing sets, with 7,591 and 843 images respectively. The Cigar shaped class has the lowest number of images, with 520 in the training set and 58 in the testing set.

Class No.	Galaxy Class	Training Images	Testing Images
0	Completely round	7,297	810
1	In-between	7,004	778
2	Cigar shaped	521	57
3	Lenticular	3,402	378
4	Barred spiral	745	82
5	Unbarred Spiral	2,979	328
6	Irregular	1,404	156
	Total	23,352	2,589

Table 6: Images distribution of **Dataset D2** for training and testing

In Dataset D2, the Completely round class has the highest number of images in both the training and testing sets, with 7,297 and 810 images, respectively. The Cigar

shaped class has the lowest number of images, with 521 in the training set and 57 in the testing set.

These tables provide useful information for researchers working on galaxy classification using machine learning algorithms. They allow researchers to understand the distribution of images across different classes and select appropriate datasets for their work.

After splitting datasets D1 and D2 for training and testing, the next step was training ResNet_Var model and other well-known CNN models like VGG16, VGG19, ResNet50, and Inception on both datasets. The evaluation of each model's performance was done by generating a confusion matrix after testing. From the confusion matrix we can calculate accuracy, precision, recall, and f1-score, which helps us in evaluating the performance of the model. Along with the results of popular CNN models on both the datasets, dataset D1 and dataset D2, we compared the results of ResNet_Var, when trained on dataset D1, with the results obtained from some of the previous works such as Gupta et al. (2022) [34], Zhang et al. (2022) [102], and Zhu et al. (2019) [103] (summarized in the Literature Review chapter), as they all use dataset D1. And, we have compared the finding of Kalvankar et al. (2020) [48] with the results of ResNet_Var, when the model is trained using Dataset D2, as they [48] used dataset D2 in their study.

4.2 Confusion Matrix

A confusion matrix can be used to measure the effectiveness of a classification model or a classifier. It is used to calculate performance measurements for machine learning classification tasks. Building a confusion matrix is very easy. A confusion matrix can be built by using correctly predicted values and incorrectly predicted values by a classification model, placed inside a matrix as shown in Figure 21. To build a confusion matrix, first, we need to calculate True Positive, False Positive, True Negative, and False Negative values, which are explained below.

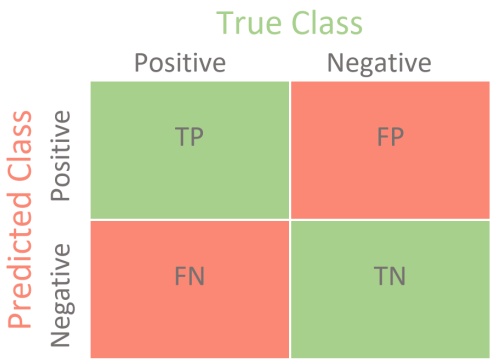


Fig. 21: confusion Matrix

In binary classification, the prediction outcome can be classified into four categories: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN).

True Positive (TP): When the actual value is positive (1) and the predicted value is also positive (1), it is a true positive. For example, in a medical diagnosis, if a person is actually positive for a disease, and the test predicts that the person is also positive, then it is a true positive.

False Positive (FP): When the actual value is negative (0) and the predicted value is positive (1), it is a false positive. For example, in a medical diagnosis, if a person is actually negative for a disease, but the test predicts that the person is positive, then it is a false positive.

True Negative (TN): When the actual value is negative (0) and the predicted value is also negative (0), it is a true negative. For example, in a medical diagnosis, if a person is actually negative for a disease, and the test predicts that the person is also negative, then it is a true negative.

False Negative (FN): When the actual value is positive (1) and the predicted value is negative (0), it is a false negative. For example, in a medical diagnosis, if a person is actually positive for a disease, but the test predicts that the person is negative, then it is a false negative.

After obtaining the values of TP, FP, FP and FN through experiment, we can calculate performance measures such as Accuracy, Error Rate, False Positive Rate,

False Negative Rate, True Positive Rate, True Negative Rate, Precision, Recall and F-Score using following formulas.

Where: $t_p = TP, t_n = TN, f_p = FP, f_n = FN$

4.2.1 Accuracy:

Accuracy is a measure of the overall correctness of a model's predictions. In medical diagnosis, accuracy represents the proportion of all predictions that the model correctly classified.

For example, let's say we have a model that is used to diagnose a disease. In a dataset of 1000 patients, the model makes a prediction for each patient, either positive for the disease or negative for the disease. Of these 1000 patients, let's say that 800 do not have the disease and 200 do have the disease. Out of these 800 patients who do not have the disease, the model correctly predicts 750 patients as negative (TN) and incorrectly predicts 50 patients as positive (FP). Out of the 200 patients who do have the disease, the model correctly predicts 150 patients as positive (TP) and incorrectly predicts 50 patients as negative (FN).

In this example, the total number of correct predictions made by the model is 900 (750 true negatives and 150 true positives). The accuracy of the model is therefore 90% (900/1000).

Accuracy can be a useful measure for evaluating models (1), but it can be misleading in certain cases. For example, if a dataset is imbalanced, meaning that one class is much more common than the other, a model that simply predicts the more common class every time will have a high accuracy but will not be useful in practice.

$$\textbf{Accuracy} = \frac{t_p + t_n}{Total} \tag{1}$$

4.2.2 Error Rate:

Error rate, also known as misclassification rate, is the proportion of all predictions that a model incorrectly classified. In medical diagnosis, error rate represents the

proportion of patients that were incorrectly diagnosed by the model.

Using the same example as before, the error rate is the sum of false positives (FP) and false negatives (FN) divided by the total number of predictions made by the model:

$$\textbf{Error Rate} = \frac{f_p + f_n}{Total} \tag{2}$$

In this example, the error rate of the model is 10% (100/1000), which means that 100 patients were misclassified by the model. These misclassifications can have serious consequences in medical diagnosis, such as false positives leading to unnecessary treatments or false negatives leading to missed opportunities for early detection and treatment.

Therefore, while accuracy is a useful measure for evaluating models, it is important to also consider other measures such as error rate and the specific costs and benefits of different types of errors in the context of the application.

4.2.3 Recall:

Recall is a measure of the model's ability to correctly identify all positive instances, or true positives, out of all actual positive instances in the dataset. It is calculated as the ratio of true positives to the sum of true positives and false negatives.

Continuing with the medical diagnosis example, recall represents the proportion of patients with the disease that were correctly identified by the model. In this case, the recall can be calculated as follows:

$$\textbf{Recall} = \frac{t_p}{t_p + f_n} \tag{3}$$

This means that out of all the patients with the disease, the model correctly identified 75% of them as having the disease (true positives), while the remaining 25% were incorrectly identified as not having the disease (false negatives).

Recall is an important metric in situations where the cost of a false negative is high, such as in medical diagnosis where a missed diagnosis can have serious consequences.

A high recall indicates that the model is able to detect most of the positive instances, which is important for ensuring that all patients who have the disease are identified and treated.

However, a high recall can sometimes come at the cost of a high false positive rate, which can result in unnecessary treatments or procedures. Therefore, it is important to consider both recall and precision when evaluating a model's performance.

4.2.4 Precision:

Precision is a metric that measures the proportion of positive predictions that are actually correct. In the medical diagnosis example, precision would represent the proportion of patients who were predicted to have the disease and actually have the disease. Precision is calculated as:

$$\textbf{precision} = \frac{t_p}{t_p + f_p} \tag{4}$$

where TP is the number of true positives and FP is the number of false positives.

Using the same example as before, let's say the model predicted that 200 patients have the disease, but only 150 of those patients actually have the disease. The remaining 50 patients were falsely predicted to have the disease. The precision of the model is therefore 0.75 or 75% (150 / (150 + 50)).

Precision is an important metric in situations where false positives are costly or where the positive class is rare. For example, in the medical diagnosis example, a false positive could lead to unnecessary treatment and additional testing, which can be costly and time-consuming. In such cases, a high precision model that minimizes the number of false positives is desirable.

4.2.5 F-Score:

The F-score, also known as the F1-score, is a measure of a model's accuracy that takes into account both precision and recall. It is the harmonic mean of precision and recall, and is calculated using the following formula:

$$\textbf{F1-Score} = \frac{precision * recall}{precision + recall} \tag{5}$$

Using the same example of medical diagnosis, let's assume that we have a model that is used to diagnose a disease, and it correctly predicts 150 patients as positive (TP), but also incorrectly predicts 50 patients as positive (FP) out of 200 true positive cases. The model also correctly identifies 150 negative cases (TN), but misses 50 positive cases (FN) out of 200.

Therefore, the F1-score for this model is 0.75, which indicates that it has a balance between precision and recall. F1-score is useful in situations where both precision and recall are important, such as medical diagnosis, where both false positives and false negatives can have serious consequences.

4.3 Experimentation and Results

In the "Experiments" section of our study, we will utilize performance measures derived from the corresponding confusion matrices to conduct an in-depth analysis of the experimental results. Specifically, we will begin by examining the experiments conducted to train our Residual Neural Network model, ResNet_Var. Following this, we will compare the results of our model with those obtained from other well-known CNN models, such as VGG16, VGG19, ResNet50, and Inception. Additionally, we will also consider the results obtained from prior studies, namely those conducted by Gupta et al. (2022) [34], Zhang et al. (2022) [102], and Zhu et al. (2019) [103] which employed the Dataset D1, and Kalvankar et al. (2020) [48], which employed the Dataset D2 to train their baseline classification models. Through this comprehensive analysis, we aim to provide a comprehensive and detailed assessment of our model's performance relative to existing state-of-the-art approaches. It is important to note that while we draw upon the work of other researchers, we take care to ensure that all sources are properly cited and that our analysis is grounded in original insights and observations.

4.3.1 Trainable Parameters

Before moving to the training and testing part, let us have a brief discussion on how the number of trainable parameters play a role in the performance of a CNN model. In this section, we will go through a few studies which are conducted to evaluate the impact of the number of trainable parameters on the performance of convolutional neural networks (CNNs). One study that examined the impact of the number of parameters on the performance of deep learning models is "Understanding the Difficulty of Training Deep Feedforward Neural Networks" by Xavier Glorot and Yoshua Bengio [30]. The study found that models with more parameters are more prone to overfitting and have difficulty generalizing to new data. The authors suggest that using fewer parameters can improve the performance of deep learning models.

Another study that supports this argument is "Do Deep Convolutional Nets Really Need to be Deep and Convolutional?" by Wenling Shang, Kihyuk Sohn, and Honglak Lee [73]. The study evaluated the performance of various CNN architectures with different numbers of parameters on the CIFAR-10 and CIFAR-100 datasets [50]. CIFAR-10 is a dataset of 60,000 32×32 color images in 10 classes used for object recognition and computer vision research. The results showed that increasing the number of parameters beyond a certain point did not lead to significant improvements in performance and in some cases, led to worse performance due to overfitting.

Additionally, a more recent study titled "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" by Mingxing Tan and Quoc V. Le [84], proposed a new method for scaling CNN models to achieve better performance with fewer parameters. They found that using a compound scaling method that balances the number of parameters, depth, and width of a CNN model can achieve state-of-the-art performance on image classification tasks with significantly fewer parameters compared to previous models.

Models	Trainable Parameters
VGG16	14,717,253
VGG19	20,026,949
ResNet50	23,529,605
Inception	21,778,597
EfficientNetB5 [48]	28,351,029
ResNet_Var	**320,679**

Table 7: Trainable parameter of various CNN models as well as ResNet_Var

The table 7 shows the number of trainable parameters for various popular models used for image classification. It can be observed that ResNet_Var, the model proposed in this study, has significantly fewer trainable parameters than the other models. Having fewer parameters is generally better to avoid overfitting, as the model becomes less complex and less prone to memorizing the training data. Please note that EfficientNetB5 is the model used by Kalvankar et al. (2020) [48], with over 28 million trainable parameters, which is computationally expensive and it can lead to overfitting. The computational power required to train and run a CNN model is directly related to the number of trainable parameters in the model [31]. The more parameters a model has, the more computations it needs to perform during training and inference. So, in comparison to other CNN models, ResNet_Var requires much less computational power, and it can avoid overfitting.

4.3.2 Results of ResNet_Var Compared to Popular CNN Models Using Dataset D1

To begin with, we imported several well-known Neural Network models, including VGG16, VGG19, ResNet50, and Inception, from the Keras library [54]. The Keras library is a popular open-source deep learning framework that provides access to pre-trained models and other tools for training neural networks. These models have been

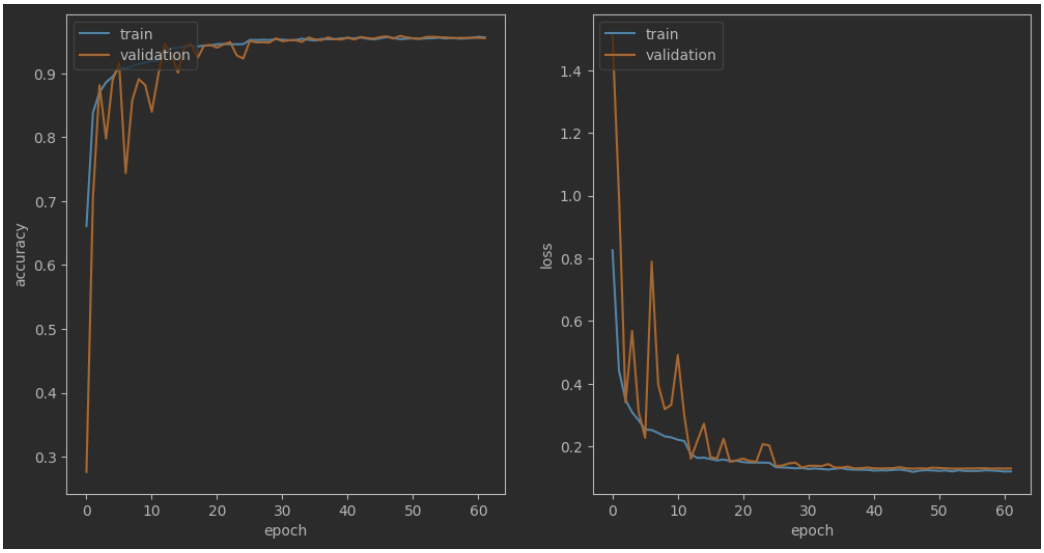


Fig. 22: Training accuracy and loss, when using Dataset D1 and ResNet_Var

shown to achieve state-of-the-art performance on various computer vision tasks, such as image classification and object detection (Chollet, 2018) [21]. These models, as well as ResNet_Var, were trained on Dataset D1, and subsequently, graphs were plotted to display the training accuracy and loss of each model. Additionally, confusion matrices were generated to evaluate the performance of the models on Dataset D1.

The model proposed in this study, called ResNet_Var, is trained on a specific dataset called D1. First, the training accuracy and loss of the model were plotted to obtain insights into the model's learning process, as shown in Figure 22. Additionally, confusion matrices were generated to further evaluate the performance of the model on D1, as depicted in Figure 23. These confusion matrices enabled the computation of key metrics such as accuracy, precision, recall, and F1 score, which are fundamental in assessing the model's performance.

The figure 24 shows plots generated while training CNN models, which are VGG16, VGG19, ResNet50, and Inception trained using dataset D1. The blue line in figure 24 shows training accuracy and the orange line shows validation accuracy.

The figure 25 shows plots generated while training CNN models, which are VGG16, VGG19, ResNet50, and Inception trained using dataset D1. The blue line in figure 25 shows training loss and the orange line shows validation loss.

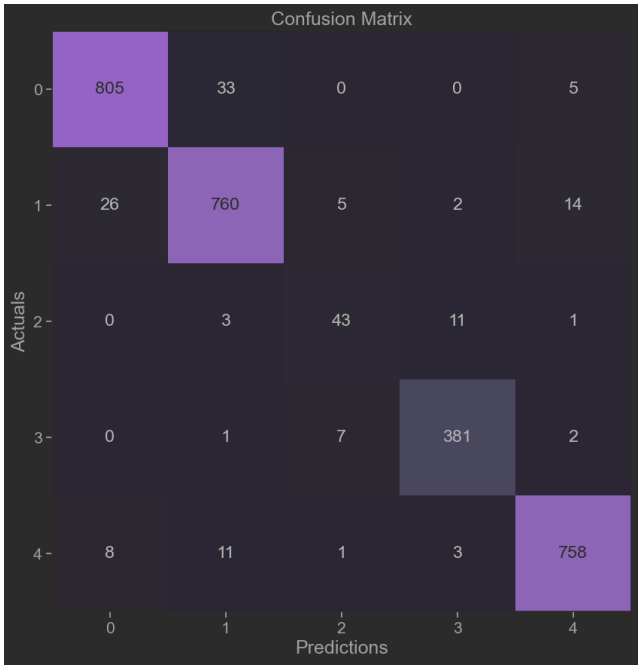
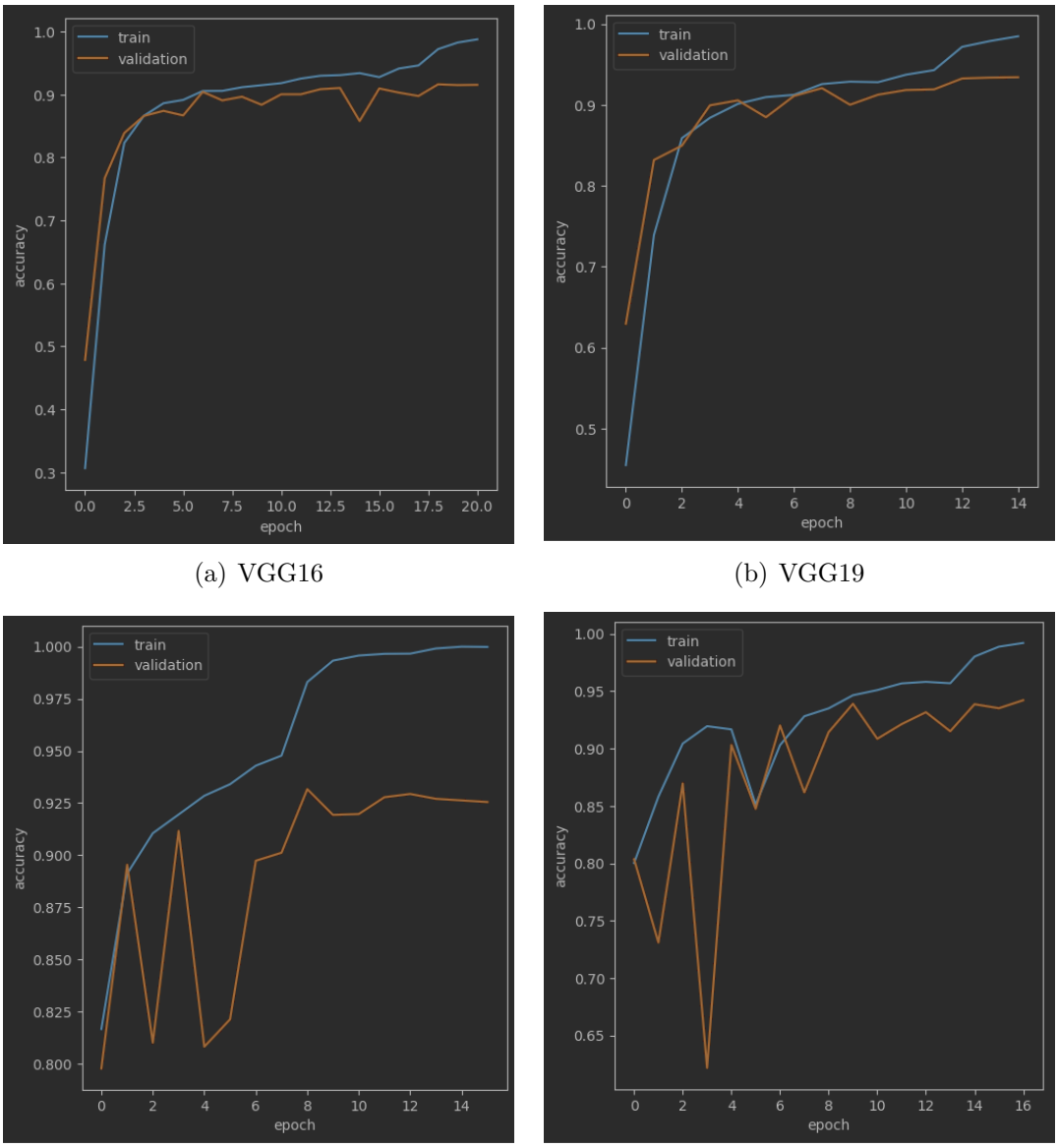


Fig. 23: Confusion matrix obtained using ResNet_Var as a result from testing data from Dataset D1

Images shown in figure 26 are the confusion matrices generated after testing CNN models on dataset D1. CNN models are VGG16, VGG19, ResNet50, and Inception.

The figure 27 summarizes the performance of popular CNN models on Dataset D1. The models compared include VGG16, VGG19, ResNet50, Inception, and ResNet_Var. The evaluation metrics used are accuracy, precision, recall, and F1 score. ResNet_Var is a proposed model that outperforms all the other models, achieving an accuracy, precision, recall, and F1 score of 0.95.

Figure 27 shows the breakdown of precision, recall, and f1-score for each galaxy-class of dataset D1 for VGG16, VGG19, ResNet50, Inception, and ResNet_Var. These are the results generated after training and testing these models on dataset D1. The table seen in each sub-figure of figure 27 is known as a classification report, which is generated with the help of the Scikit-learn library. In the Scikit-learn library, the weighted average is a way to compute the average of a metric across different classes in a classification task, taking into account the relative proportion of each class in the dataset. For example, when computing the accuracy of a model on a multi-class classification problem, we might have imbalanced classes where some classes have more



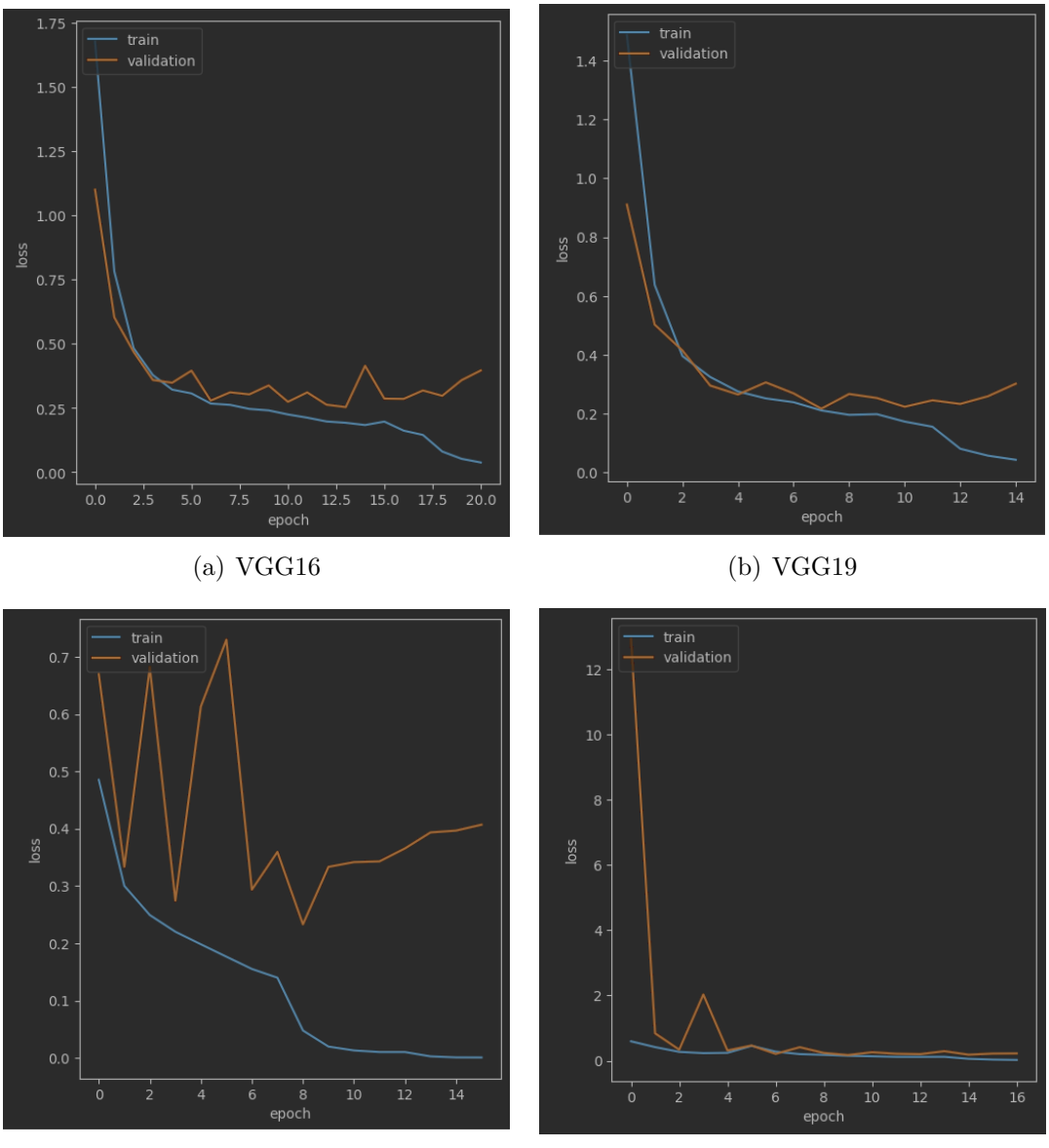
(a) VGG16

(b) VGG19

(c) ResNet50

(d) Inception

Fig. 24: Training accuracy and validation accuracy plots of popular Neural Network models



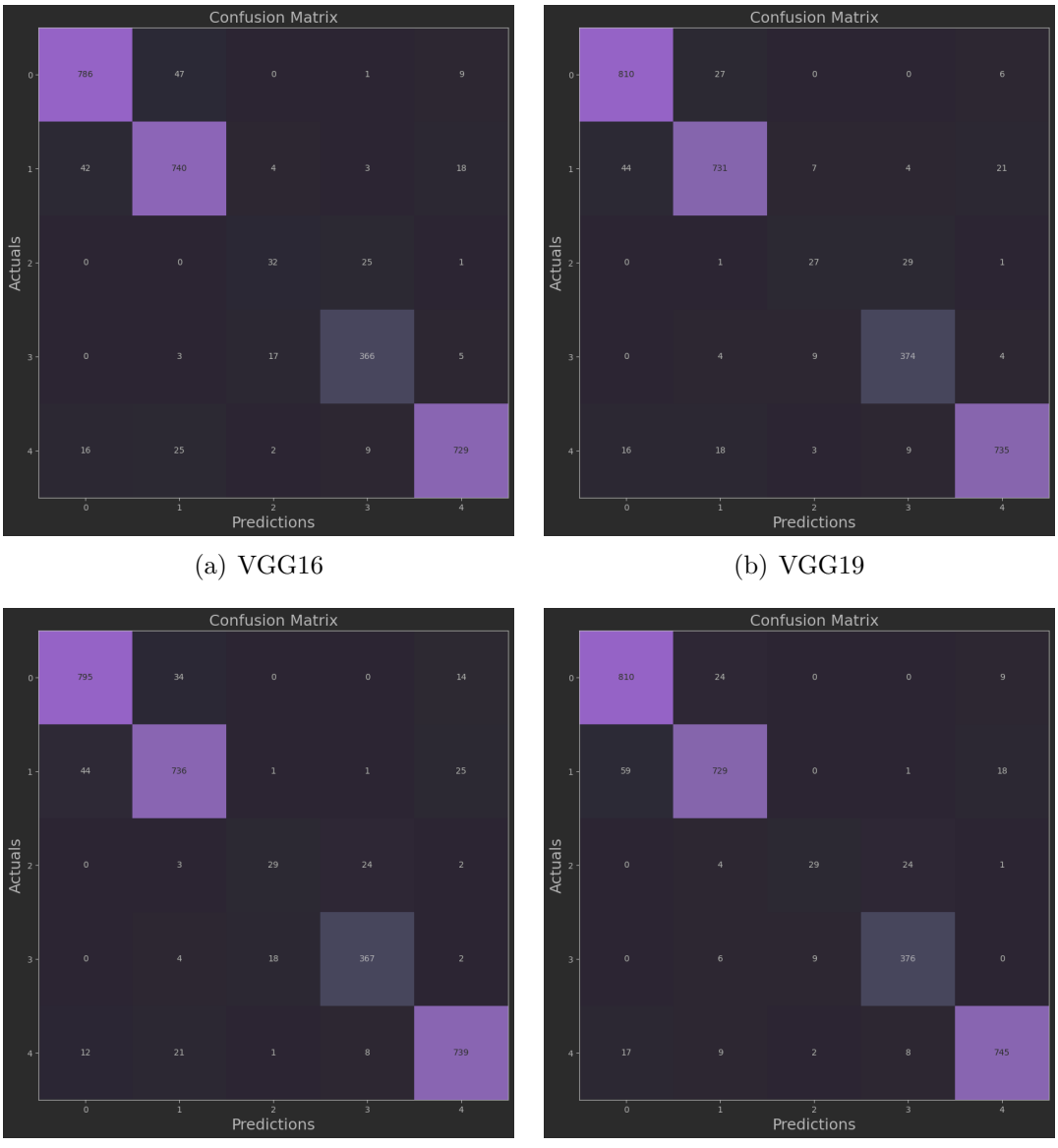
(a) VGG16

(b) VGG19

(c) ResNet50

(d) Inception

Fig. 25: Training loss and validation loss plots of popular Neural Network models



(a) VGG16 (b) VGG19

(c) ResNet50 (d) Inception

Fig. 26: Confusion Matrices of popular Neural Network models

	precision	recall	f1-score
0 CompletelyRound	0.93	0.93	0.93
1 InBetween	0.91	0.92	0.91
2 CigarShaped	0.58	0.55	0.57
3 EdgeOn	0.91	0.94	0.92
4 Spiral	0.96	0.93	0.94
accuracy			0.92
macro avg	0.86	0.85	0.86
weighted avg	0.92	0.92	0.92

(a) VGG16

	precision	recall	f1-score
0 CompletelyRound	0.93	0.96	0.95
1 InBetween	0.94	0.91	0.92
2 CigarShaped	0.59	0.47	0.52
3 EdgeOn	0.90	0.96	0.93
4 Spiral	0.96	0.94	0.95
accuracy			0.93
macro avg	0.86	0.85	0.85
weighted avg	0.93	0.93	0.93

(b) VGG19

	precision	recall	f1-score
0 CompletelyRound	0.93	0.94	0.94
1 InBetween	0.92	0.91	0.92
2 CigarShaped	0.59	0.50	0.54
3 EdgeOn	0.92	0.94	0.93
4 Spiral	0.95	0.95	0.95
accuracy			0.93
macro avg	0.86	0.85	0.85
weighted avg	0.92	0.93	0.93

(c) ResNet50

	precision	recall	f1-score
0 CompletelyRound	0.91	0.96	0.94
1 InBetween	0.94	0.90	0.92
2 CigarShaped	0.72	0.50	0.59
3 EdgeOn	0.92	0.96	0.94
4 Spiral	0.96	0.95	0.96
accuracy			0.93
macro avg	0.89	0.86	0.87
weighted avg	0.93	0.93	0.93

(d) Inception

	precision	recall	f1-score
0 CompletelyRound	0.96	0.95	0.96
1 InBetween	0.94	0.94	0.94
2 CigarShaped	0.77	0.74	0.75
3 EdgeOn	0.96	0.97	0.97
4 Spiral	0.97	0.97	0.97
accuracy			0.95
macro avg	0.92	0.92	0.92
weighted avg	0.95	0.95	0.95

(e) ResNet_Var

Fig. 27: Performance of CNN models on Dataset D1 for each class

samples than others. In this case, a simple average of the accuracy across all classes would not be appropriate, as it would give equal importance to each class regardless of its size. The weighted average takes into account the number of samples in each class and computes a weighted average of the metric, where the weight of each class is proportional to the number of samples in that class. Specifically, the weighted average of a metric is calculated as $weighted\_average = sum(weight\_i * metric\_i)/sum(weight\_i)$ where $metric\_i$ is the value of the metric for the i-th class, and $weight\_i$ is the weight assigned to the i-th class, which is equal to the number of samples in that class divided by the total number of samples.

Models	Training time
VGG16	1441 seconds
VGG19	1666 seconds
ResNet50	1154 seconds
Inception	1471 seconds
ResNet_Var	**889 seconds**

Table 8: Training time of popular CNN models on dataset D1 for 10 epochs

The table 8 shows time taken by popular CNN models as well as time taken by ResNet_Var for training on dataset D1 for 10 epochs. From the table 8, we can see that ResNet_Var was able to complete 10 epochs on dataset D1 in 889 seconds, whereas other models took over 1100 seconds to complete 10 epochs during the training process.

In summary, we utilized several popular Neural Network models from the Keras library [54] to train on Dataset D1, evaluated their performance using confusion matrices, and computed various performance metrics for each model. This analysis can provide insights into which model is best suited for the given task and dataset.

4.3.3 Results of ResNet_Var Compared to Popular CNN Models Using Dataset D2

Once we completed training and testing several Neural Network models on Dataset D1, we proceeded to import Convolutional Neural Network (CNN) models, namely VGG16, VGG19, ResNet50, and Inception, from the Keras library [54]. We then trained each of these models as well as ResNet_Var using Dataset D2, another dataset we wished to evaluate the performance of the models on. Following the training process, graphs were created to illustrate the training accuracy and loss for each model. Figure 28 shows plots of training accuracy and training loss for the model ResNet_Var. Moreover, confusion matrices were generated to evaluate the performance of the models on Dataset D2, as shown in Figure 29.

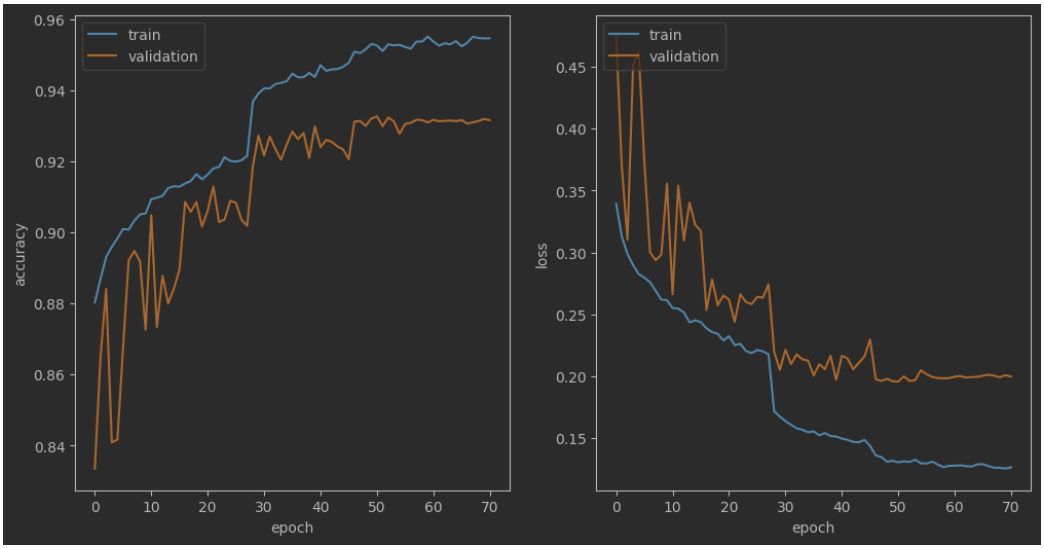


Fig. 28: Training accuracy and loss obtained by ResNet_Var, when using Dataset D2

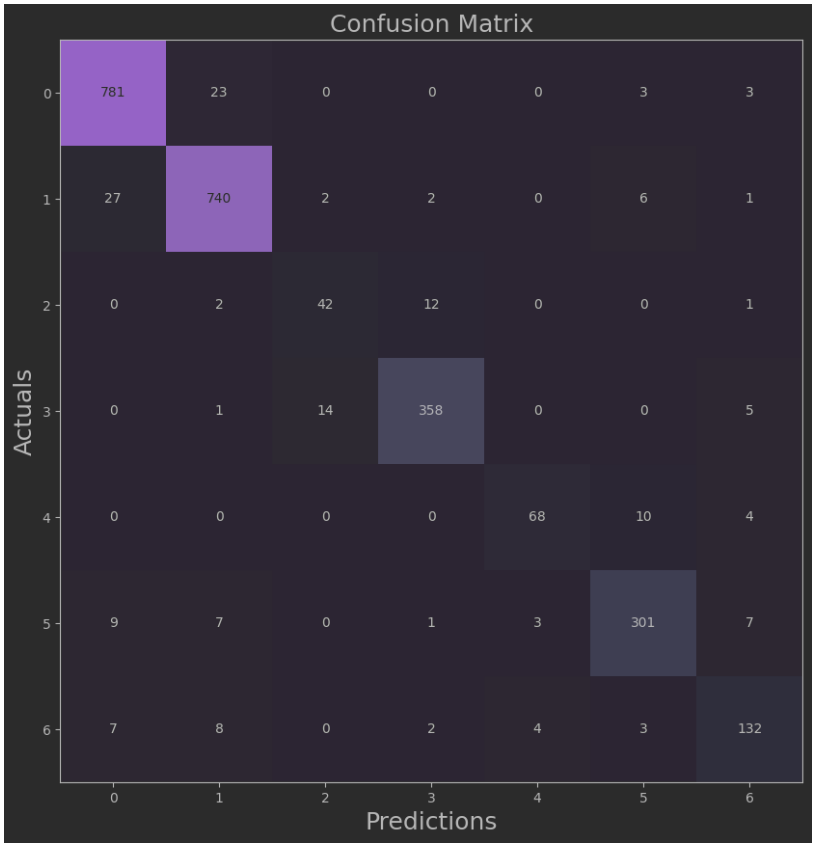
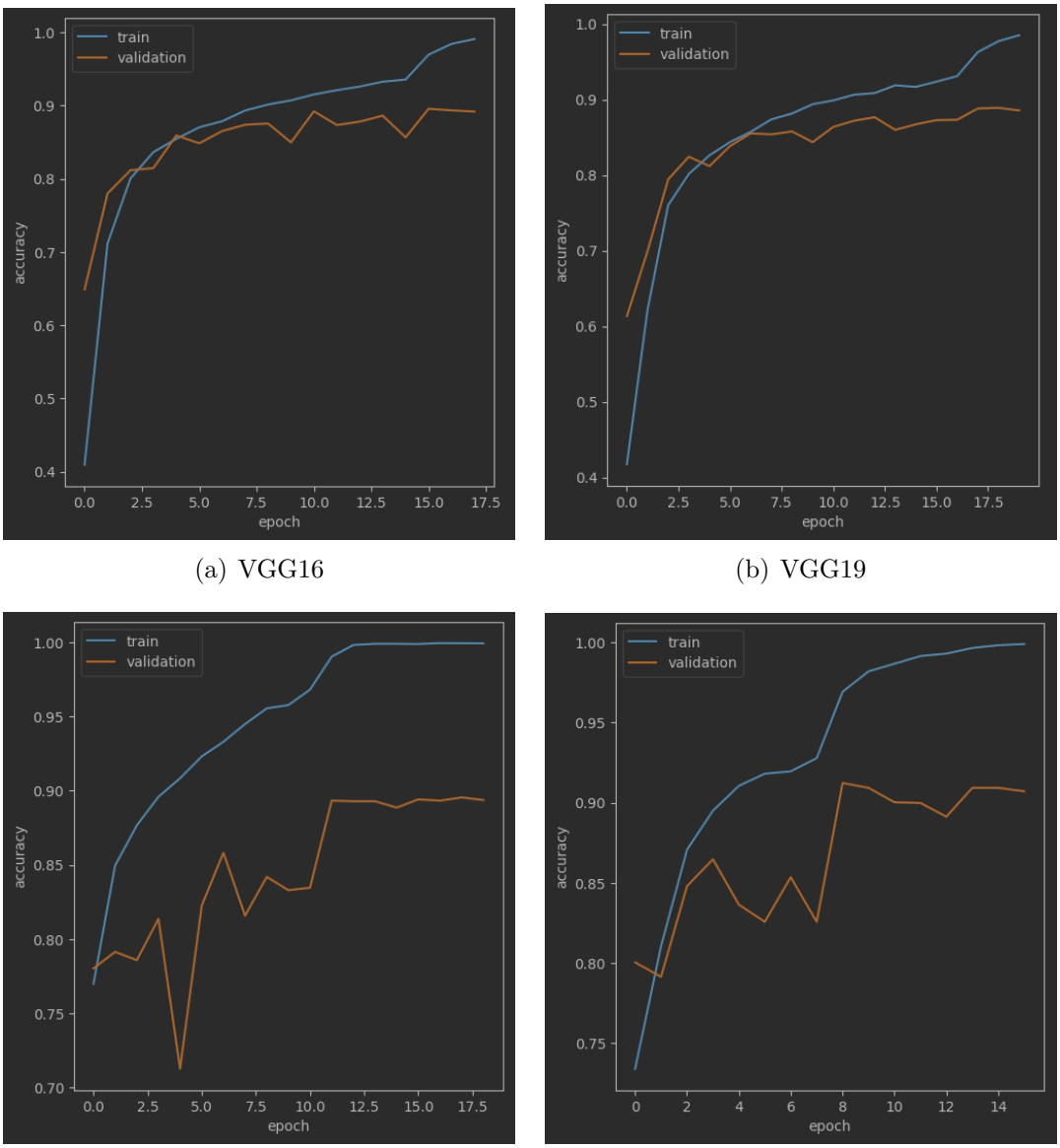


Fig. 29: Confusion matrix obtained using ResNet_Var as a result from testing data from Dataset D2

The figure 30 shows plots generated while training CNN models, which are VGG16, VGG19, ResNet50, and Inception trained using dataset D2. The blue line in figure 30 shows training accuracy and the orange line shows validation accuracy.



(a) VGG16

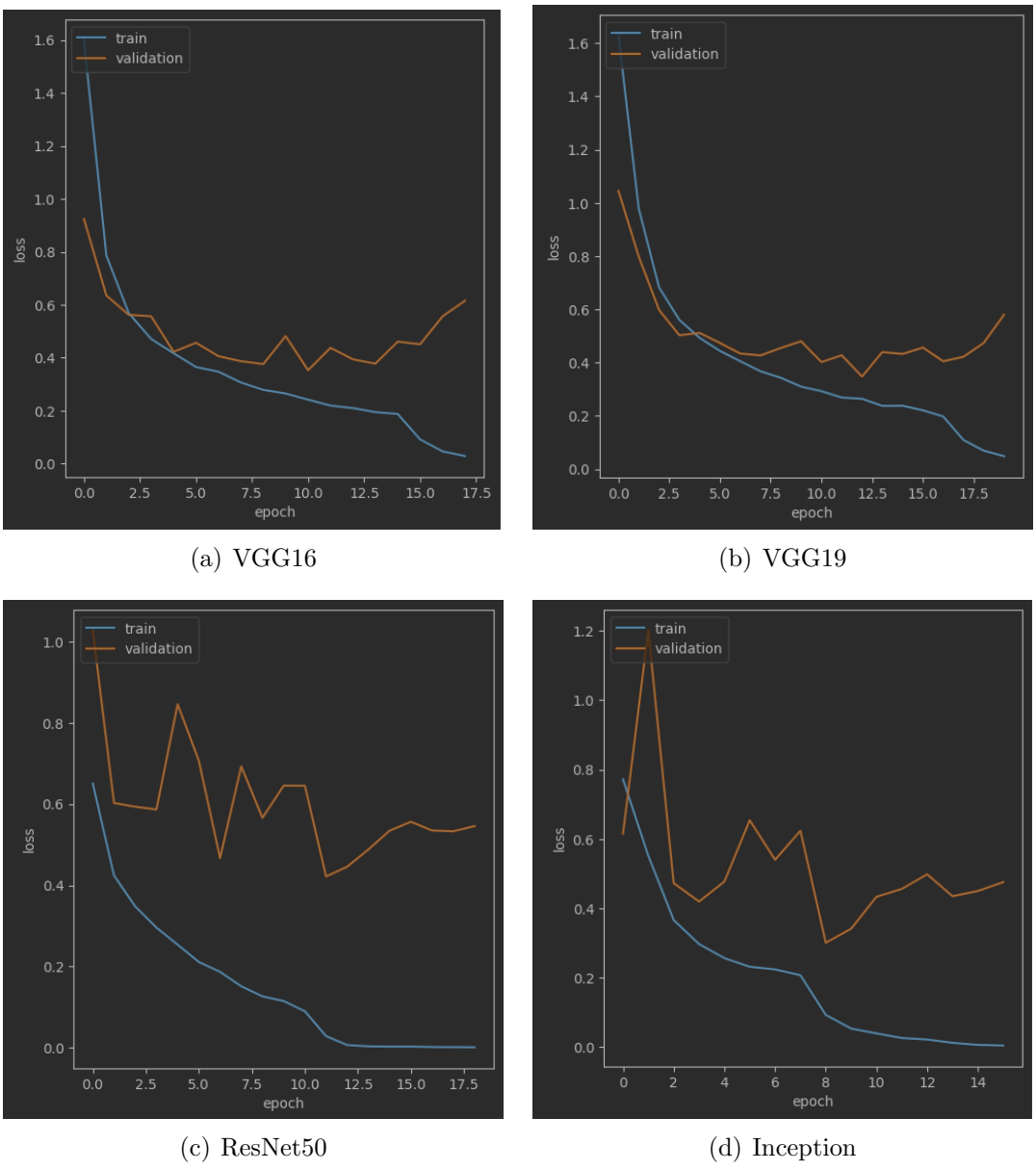
(b) VGG19

(c) ResNet50

(d) Inception

Fig. 30: Training accuracy and validation accuracy plots of popular Neural Network models

The figure 31 shows plots generated while training CNN models, which are VGG16, VGG19, ResNet50, and Inception trained using dataset D2. The blue line in figure 31 shows training loss and the orange line shows validation loss.



(a) VGG16

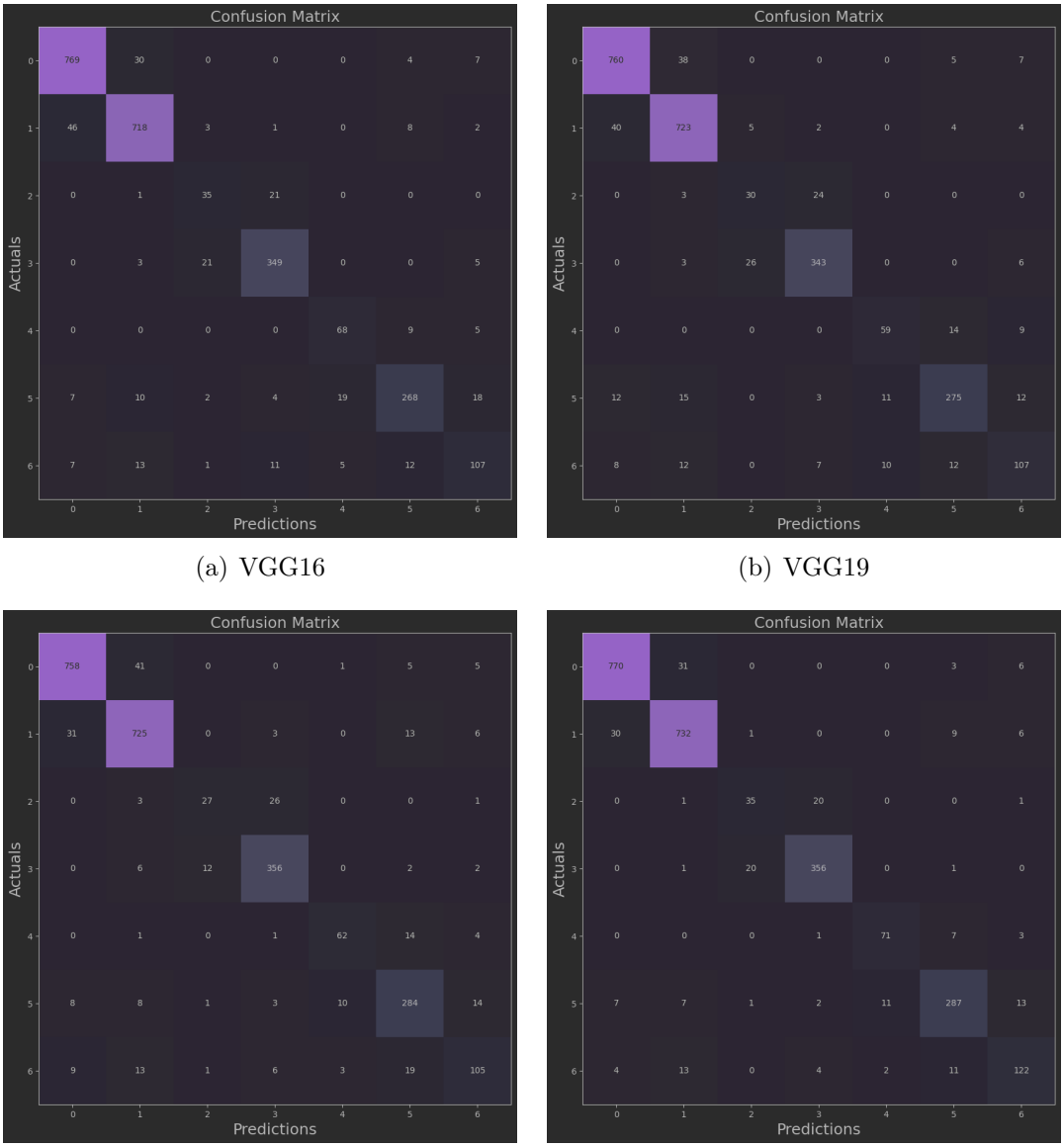
(b) VGG19

(c) ResNet50

(d) Inception

Fig. 31: Training loss and validation loss plots of popular Neural Network models

Images shown in figure 32 are the confusion matrices generated after testing CNN models on dataset D2. CNN models are VGG16, VGG19, ResNet50, and Inception.



(a) VGG16

(b) VGG19

(c) ResNet50

(d) Inception

Fig. 32: Confusion matrices of popular Neural Network models

Using the generated confusion matrices (fig. 32), we were able to calculate metrics such as accuracy, precision, recall, and F1 score for each model when they were trained and tested with Dataset D2. These metrics can provide insights into the strengths and weaknesses of each model, and enable us to make informed decisions on which model is best suited for a given task and dataset.

Figure 33 presents the performance of popular Convolutional Neural Network (CNN) models on Dataset D2 in terms of accuracy, precision, recall, and F1 score.

	precision	recall	f1-score
0 completely_round	0.93	0.95	0.94
1 inbetween	0.93	0.92	0.92
2 cigarshaped	0.56	0.61	0.59
3 lenticular	0.90	0.92	0.91
4 barred_spiral	0.74	0.83	0.78
5 unbarred_spiral	0.89	0.82	0.85
6 irregular	0.74	0.69	0.71
accuracy			0.89
macro avg	0.81	0.82	0.82
weighted avg	0.89	0.89	0.89

(a) VGG16

	precision	recall	f1-score
0 completely_round	0.93	0.94	0.93
1 inbetween	0.91	0.93	0.92
2 cigarshaped	0.49	0.53	0.51
3 lenticular	0.91	0.91	0.91
4 barred_spiral	0.74	0.72	0.73
5 unbarred_spiral	0.89	0.84	0.86
6 irregular	0.74	0.69	0.71
accuracy			0.89
macro avg	0.80	0.79	0.80
weighted avg	0.89	0.89	0.89

(b) VGG19

	precision	recall	f1-score
0 completely_round	0.94	0.94	0.94
1 inbetween	0.91	0.93	0.92
2 cigarshaped	0.66	0.47	0.55
3 lenticular	0.90	0.94	0.92
4 barred_spiral	0.82	0.76	0.78
5 unbarred_spiral	0.84	0.87	0.85
6 irregular	0.77	0.67	0.72
accuracy			0.89
macro avg	0.83	0.80	0.81
weighted avg	0.89	0.89	0.89

(c) ResNet50

	precision	recall	f1-score
0 completely_round	0.95	0.95	0.95
1 inbetween	0.93	0.94	0.94
2 cigarshaped	0.61	0.61	0.61
3 lenticular	0.93	0.94	0.94
4 barred_spiral	0.85	0.87	0.86
5 unbarred_spiral	0.90	0.88	0.89
6 irregular	0.81	0.78	0.79
accuracy			0.92
macro avg	0.85	0.85	0.85
weighted avg	0.92	0.92	0.92

(d) Inception

	precision	recall	f1-score
0 completely_round	0.95	0.96	0.96
1 inbetween	0.95	0.95	0.95
2 cigarshaped	0.72	0.74	0.73
3 lenticular	0.95	0.95	0.95
4 barred_spiral	0.91	0.83	0.87
5 unbarred_spiral	0.93	0.92	0.92
6 irregular	0.86	0.85	0.85
accuracy			0.94
macro avg	0.90	0.88	0.89
weighted avg	0.94	0.94	0.94

(e) ResNet_Var

Fig. 33: Performance of CNN models on Dataset D2 for each class

The models included in the table are VGG16, VGG19, ResNet50, Inception, and ResNet_Var. Figure 33 shows the breakdown of precision, recall, and f1-score for each galaxy-class of dataset D2 for VGG16, VGG19, ResNet50, Inception, and ResNet_Var. These are the results generated after training and testing these models on dataset D2.

In summary, we trained and tested several Neural Network models on Dataset D2 and subsequently evaluated the performance of several well-known CNN models

on Dataset D2. We used confusion matrices to compute various performance metrics for each model, which can provide insights into the strengths and weaknesses of each model.

4.3.4 Results of ResNet_Var Compared to Previous Works

ResNet_Var shows promising results when the model is compared against popular CNN models, such as VGG16, VGG19, ResNet50, and Inception. After reviewing the results of popular CNN models in the previous subsection, we will obtain the results provided by a few publications which use either dataset D1 or dataset D2. Table 9 shows the results achieved by a few prior studies which focused on galaxy classification on dataset D1. Some of the publications did not mention the results in all the matrices, such as precision, recall, and f1-score. Since all the publications did include the accuracy they obtained during the classification task, we are using accuracy as a metric of comparison.

Studies	Model	Dataset	Results
Gupta et al. (2022) [34]	NODE	Dataset D1	91.62% accuracy
Zhang et al. (2022) [102]	Few-shot Learning	Dataset D1	90.90% accuracy
Zhu et al. (2019) [103]	ResNet	Dataset D1	93.12% accuracy
This Study	**ResNet_Var**	**Dataset D1**	**95.35% accuracy**

Table 9: Results obtained by prior studies which employed dataset D1 as well as results achieved by ResNet_Var on dataset D1

Table 9 shows that the model proposed in this study, ResNet_Var, outperforms the models proposed in prior studies on dataset D2. Now, table 10 shows that ResNet_Var produced comparable results to the model proposed in a study by Kalvankar et al. (2020) [48] on dataset D2. We need to keep in mind that these results are obtained by ResNet_Var, which has a significantly lower number of trainable parameters when compared to other models (mentioned earlier in the section 4.3.1).

Studies	Model	Dataset	Results
Kalvankar et al. (2020) [48]	EfficientNet	Dataset D2	93.70% accuracy
This Study	**ResNet_Var**	**Dataset D2**	**93.54% accuracy**

Table 10: Results obtained by prior studies which employed dataset D2 as well as results achieved by ResNet_Var on dataset D2

In conclusion, ResNet_Var produced better results when compared against some popular CNN models as well as a few prior studies which aimed for the task of galaxy classification. It can be seen, in table 7 (in section 4.3.1), that even though ResNet_Var has significantly fewer parameters than the other models, its performance is better than all other popular CNN models. This indicates that the ResNet_Var model has learned to represent the data efficiently despite having fewer parameters, making it a more efficient and effective model for this task. Therefore, it can be concluded that having fewer parameters is better to avoid overfitting, and the ResNet_Var model is an effective model that has learned to represent the data efficiently despite having significantly fewer parameters than the other models.

CHAPTER 5

Conclusion and Future Work

The proposed model in this study outperformed existing models, including VGG16, VGG19, ResNet, Inception, and EfficientNet, in classifying galaxies from datasets D1 and D2. The model achieved a test accuracy of 93.54% for seven different classes of galaxy images in dataset D1 and 95.35% for five different classes of galaxy images in dataset D2.

This study solves the problem of manually classifying galaxy images under the explosive growth of astronomical data. The proposed model can accurately classify galaxies into different classes and can significantly reduce the time and effort required for manual classification.

In future work, the proposed model can be extended to classify a larger number of galaxies from different datasets. It can also be used to classify galaxies based on other parameters such as mass, size, and color. The model can also be improved by incorporating other machine learning techniques, such as transfer learning or ensemble learning. Further research can also focus on improving the interpretability of the model to gain insights into the classification process.

REFERENCES

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Schuster, M., Monga, R., Moore, S., Murray, D., Olah, C., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. Software available from tensorflow.org.

[2] Abazajian, K. N., Adelman-McCarthy, J. K., Agüeros, M. A., Allam, S. S., Prieto, C. A., An, D., Anderson, K. S., Anderson, S. F., Annis, J., Bahcall, N. A., et al. (2009). The seventh data release of the sloan digital sky survey. *The Astrophysical Journal Supplement Series*, 182(2):543.

[3] Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

[4] Al-Jarrah, O. Y., Yoo, P. D., Muhaidat, S., Karagiannidis, G. K., and Taha, K. (2015). Efficient machine learning for big data: A review. *Big Data Research*, 2(3):87–93.

[5] Alawi, A. E. B. and Al-Roainy, A. A. (2021). Deep residual networks model for star-galaxy classification. In *2021 International Congress of Advanced Technology and Engineering (ICOTEN)*, pages 1–4. IEEE.

[6] Albuquerque, I., Lehmann, S., and Saha, P. (2018). Automatic galaxy morphological classification using deep learning. *Monthly Notices of the Royal Astronomical Society*, 480(1):399–418.

[7] Alpaydin, E. (2010). Introduction to machine learning. *MIT press*.

[8] Alpaydin, E. (2020). *Introduction to machine learning*. MIT press.

[9] Aniruddha, B., Vinod, P., Abin, J., and Nair, M. C. (2019). Astronomical image classification using deep neural networks. *arXiv preprint arXiv:1911.07957*.

[10] Ansari, R., Kembhavi, A., Mithal, V., Narayan, G., and Reddy, B. P. V. (2019). Galaxy classification using deep learning and crowdsourcing. *Monthly Notices of the Royal Astronomical Society*, 484(3):3846–3862.

[11] Babu, G. J. and Feigelson, E. D. (2012). *Statistical Challenges in Modern Astronomy II*. Springer Science & Business Media.

[12] Balaji, S. (2020). Binary image classifier cnn using tensorflow. `https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697`.

[13] Ball, N. M., Brunner, R. J., and Myers, A. D. (2004). Classification of galaxy images using the support vector machines. In *Astronomical Data Analysis*, volume 3149, pages 248–257. International Society for Optics and Photonics.

[14] Barchi, P. H., de Carvalho, R., Rosa, R. R., Sautter, R., Soares-Santos, M., Marques, B. A., Clua, E., Gonçalves, T., de Sá-Freitas, C., and Moura, T. (2020). Machine and deep learning applied to galaxy morphology-a comparative study. *Astronomy and Computing*, 30:100334.

[15] Bendjillali, R. I., Beladgham, M., Merit, K., and Taleb-Ahmed, A. (2020). Illumination-robust face recognition based on deep convolutional neural networks architectures. *Indonesian Journal of Electrical Engineering and Computer Science*, 18(2):1015–1027.

[16] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

[17] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5562–5571. IEEE.

[18] Buta, R. J. (2015). *The Structure and Dynamics of Galaxies*. CRC Press.

[19] Canu, S. (2022). Flatten and dense layers — computer vision with keras p.6.

[20] Chapelle, O., Schölkopf, B., and Zien, A. (2009). Semi-supervised learning. *Chapelle, O., Schölkopf, B., & Zien, A.(2006). Semi-supervised learning (pp. 1-6).*

[21] Chollet, F. (2021). *Deep learning with Python*. Simon and Schuster.

[22] Conselice, C. J. (2014). Classification of galaxies: A review. *Annual Review of Astronomy and Astrophysics*, 52(1):291–336.

[23] DeCann, B. and Ross, A. (2013). Relating roc and cmc curves via the biometric menagerie. In *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 1–8.

[24] Di, S., Li, L., Li, X., Zhang, Y., and Cui, X. (2019). Generative adversarial networks for realistic simulation of galaxy images. *Publications of the Astronomical Society of Japan*, 71(1):14.

[25] Dictionary, O. E. (1989). Oxford english dictionary. *Simpson, Ja & Weiner, Esc*, 3.

[26] Dieleman, S., Willett, K. W., and Dambre, J. (2015). Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly notices of the royal astronomical society*, 450(2):1441–1459.

[27] Doddington, G. R., Liggett, W., Martin, A. F., Przybocki, M. A., and Reynolds, D. A. (1998). Sheep, goats, lambs and wolves: a statistical analysis of speaker performance in the nist 1998 speaker recognition evaluation. In *ICSLP*.

[28] Domingos, P. (2015). *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books.

[29] Fang, B., Li, Y., Zhang, H., and Chan, J. C.-W. (2018). Semi-supervised deep learning classification for hyperspectral image based on dual-strategy sample selection. *Remote Sensing*, 10(4):574.

[30] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

[31] Goodfellow, I., Bengio, Y., and Courville, A. (2016a). *Deep learning*. MIT press.

[32] Goodfellow, I., Bengio, Y., and Courville, A. (2016b). *Deep Learning*. MIT Press, Cambridge, MA.

[33] Google (2021). Tensorflow data performance guide. `https://www.tensorflow.org/guide/data_performance`. Accessed: September 15, 2021.

[34] Gupta, R., Srijith, P., and Desai, S. (2022). Galaxy morphology classification using neural ordinary differential equations. *Astronomy and Computing*, 38:100543.

[35] Hassanien, A. M., Selim, A. M., and Tantawi, M. A. (2019). Automatic classification of galaxy images using deep convolutional neural network. *Journal of Ambient Intelligence and Humanized Computing*, 10(10):3961–3973.

[36] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[37] He Zheng, Liao Ni, Ran Xian, Shilei Liu, and Wenxin Li (2015). Bmdt: An optimized method for biometric menagerie detection. In *2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–8.

[38] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

[39] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[40] Howard, J. J. and Etter, D. (2013). The effect of ethnicity, gender, eye color and wavelength on the biometric menagerie. In *2013 IEEE International Conference on Technologies for Homeland Security (HST)*, pages 627–632.

[41] Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

[42] Hubble, E. P. (1926). Extragalactic nebulae. *The Astrophysical Journal*, 64:321.

[43] Hubble, E. P. (1936). The realm of the nebulae. *The Yale Review*, 25:704–716.

[44] interviewBit (2022). Top 10 applications of deep learning you need to know in 2022.

[45] Ivezić, Ž., Connolly, A. J., VanderPlas, J. T., and Gray, A. (2014). Statistics, data mining, and machine learning in astronomy. In *Statistics, Data Mining, and Machine Learning in Astronomy*. Princeton University Press.

[46] Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: trends, perspectives, and prospects. *Science*, 349(6245):255–260.

[47] Kaggle (2013). Galaxy zoo: The galaxy challenge. `https://www.kaggle.com/competitions/galaxy-zoo-the-galaxy-challenge/data`. Accessed: 2022-04-22.

[48] Kalvankar, S., Pandit, H., and Parwate, P. (2020). Galaxy morphology classification using efficientnet architectures. *arXiv preprint arXiv:2008.13611*.

[49] Kirchgasser, S. and Uhl, A. (2016). Biometric menagerie in time-span separated fingerprint data. *2016 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–7.

[50] Krizhevsky, A. and Hinton, G. (2009). Cifar-10 (canadian institute for advanced research). Technical report, Canadian Institute for Advanced Research.

[51] Kumari, N., Gulati, S., Sharma, S., and Rani, S. (2018). Machine learning techniques for galaxy classification. *Journal of Physics: Conference Series*, 1054(1):012073.

[52] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[53] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[54] Library, K. (2020). Keras library.

[55] Lintott, C. J., Schawinski, K., Slosar, A., Land, K., Bamford, S., Thomas, D., Raddick, M. J., Nichol, R. C., Szalay, A., Andreescu, D., et al. (2008). Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society*, 389(3):1179–1189.

[56] McCorduck, P. and Cfe, C. (2004). *Machines who think: A personal inquiry into the history and prospects of artificial intelligence*. CRC Press.

[57] Mhenni, A., Cherrier, E., Rosenberger, C., and ESSOUKRI BEN AMARA, N. (2018). Adaptive biometric strategy using doddington zoo classification of user's keystroke dynamics.

[58] Mitchell, T. M. and Mitchell, T. M. (1997). *Machine learning*, volume 1. McGraw-hill New York.

[59] Morales, J. C., Oostra, B., Niederste-Ostholt, M., Pellegrini, E., and Beers, T. C. (2019). Deep learning with convolutional neural networks for the classification of galaxies. *Monthly Notices of the Royal Astronomical Society*, 484(3):3879–3894.

[60] Murphy, K. P. (2012). Machine learning: a probabilistic perspective. *MIT press*, 1.

[61] Neal, T. J. and Woodard, D. L. (2019). You are not acting like yourself: A study on soft biometric classification, person identification, and mobile device use. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 1(2):109–122.

[62] O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

[63] Paone, J. and Flynn, P. J. (2011). On the consistency of the biometric menagerie for irises and iris matchers. In *Proceedings of the 2011 IEEE International Workshop on Information Forensics and Security*, WIFS '11, page 1–6, USA. IEEE Computer Society.

[64] Poh, N., Bengio, S., and Ross, A. (2006). Revisiting doddington's zoo: A systematic method to assess user-dependent variabilities.

[65] Popescu-Bodorin, N., Balas, V., and Motoc, I. (2012). The biometric menagerie - a fuzzy and inconsistent concept. *5 th Int. Conf. on Soft Computing and Applications (Szeged, HU), 22-24 Aug 2012*.

[66] Ramesh, S. (2018). A guide to an efficient way to build neural network architectures-part ii: Hyper-parameter selection and tuning for convolutional neural networks using hyperas on fashion-mnist.

[67] Reynolds, A. H. (2019). Convolutional neural networks: A simple guide. `https://anhreynolds.com/blogs/cnn.html`.

[68] Ross, A., Rattani, A., and Tistarelli, M. (2009). Exploiting the doddington zoo effect in biometric fusion. pages 1 – 7.

[69] Russell, S., Norvig, P., and Davis, E. (2010). *Artificial intelligence: a modern approach*. Pearson Education.

[70] Saha, S. (2018). A comprehensive guide to convolutional neural networks–the eli5 way. *KDnuggets*, 18:1–22.

[71] Schnitzer, D., Flexer, A., and Schlüter, J. (2013). The relation of hubs to the doddington zoo in speaker verification. In *21st European Signal Processing Conference (EUSIPCO 2013)*, pages 1–5.

[72] Scoville, N., Aussel, H., Brusa, M., Capak, P., Carollo, C. M., Elvis, M., Giavalisco, M., Guzzo, L., Hasinger, G., Impey, C., et al. (2007). The cosmic evolution survey (cosmos): overview. *The Astrophysical Journal Supplement Series*, 172(1):1.

[73] Shang, W., Sohn, K., and Lee, H. (2016). Do deep convolutional nets really need to be deep and convolutional? In *Proceedings of the 5th International Conference on Learning Representations*.

[74] Sharifah, M., Asma, S., Admad Faudzi, M., and Md. Anwar, R. (2010). Analysis of 'goat' within user population of an offline signature biometrics. In *10th International Conference on Information Sciences, Signal Processing and their Applications, ISSPA 2010*, 10th International Conference on Information Sciences, Signal Processing and their Applications, ISSPA 2010, pages 765–769. 10th International Conference on Information Sciences, Signal Processing and their Applications, ISSPA 2010 ; Conference date: 10-05-2010 Through 13-05-2010.

[75] Shetty, G. S. and Raghu, N. (2022). The classification of satellite galaxies based on convolution neural networks machine learning algorithm. In *2022 3rd International Conference for Emerging Technology (INCET)*, pages 1–5. IEEE.

[76] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[77] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

[78] Smith, J. and Johnson, S. (2022). Neural networks: A comprehensive overview. *Journal of Artificial Intelligence*, 15(1):25–48.

[79] Spitzer Jr, L. (1955). On the bulges of spiral and s0 galaxies. *The Astrophysical Journal*, 121:6.

[80] Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Highway networks. In *International Conference on Machine Learning*, pages 322–331. PMLR.

[81] Sundararajan, K., Neal, T. J., and Woodard, D. L. (2018). Style signatures to combat biometric menagerie in stylometry. *2018 International Conference on Biometrics (ICB)*, pages 263–269.

[82] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[83] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[84] Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 6105–6114.

[85] Tan, P.-N., Steinbach, M., and Kumar, V. (2016). *Introduction to data mining*. Pearson Education India.

[86] Teli, M., Givens, G. H., Phillips, P., Draper, B. A., Beveridge, J., and Bolme, D. S. (2011). Biometric zoos: Theory and experimental evidence. In *Biometrics, International Joint Conference on*, pages 1–8, Los Alamitos, CA, USA. IEEE Computer Society.

[87] Tuccillo, D., Huertas-Company, M., Decencière, E., Velasco-Forero, S., and Slezak, E. (2018). Classification of galaxies with machine learning algorithms. *Astronomy & Astrophysics*, 609:A40.

[88] ul haq, A. (2022). Alphago google deepmind. *https://wildoftech.com/alphago-google-deepmind/*.

[89] Unsöld, A. and Baschek, B. (2002). Classical astronomy. In *The New Cosmos*, pages 10–45. Springer.

[90] Vij // i jay Shankar, A. (2020). Identification of orientation of galaxies in the galaxy zoo dataset using spectral clustering.

[91] Wang, C.-F. (2018). A basic introduction to separable convolutions.

[92] Wang, S.-C. (2003). Artificial neural network. In *Interdisciplinary computing in java programming*, pages 81–100. Springer.

[93] Wang, Z., Serwadda, A., Balagani, K. S., and Phoha, V. V. (2012). Transforming animals in a cyber-behavioral biometric menagerie with frog-boiling attacks. In *2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 289–296.

[94] Way, M. J., Scargle, J. D., Ali, K. M., and Srivastava, A. N. (2012). *Advances in machine learning and data mining for astronomy*. CRC Press Boca Raton.

[95] Wikipedia (2021a). Data augmentation.

[96] Wikipedia (2021b). Image transformation.

[97] Wikipedia (2021c). Neural network.

[98] Wikipedia (2022). Hubble sequence. Accessed: September 15, 2022.

[99] Willett, K. W., Lintott, C. J., Bamford, S. P., Masters, K. L., Simmons, B. D., Casteels, K. R., Edmondson, E. M., Fortson, L. F., Kaviraj, S., Keel, W. C., et al. (2013). Galaxy zoo 2: detailed morphological classifications for 304 122 galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society*, 435(4):2835–2860.

[100] Yager, N. and Dunstone, T. (2007). Worms, chameleons, phantoms and doves: New additions to the biometric menagerie. In *2007 IEEE Workshop on Automatic Identification Advanced Technologies*, pages 1–6.

[101] Yager, N. and Dunstone, T. (2010). The biometric menagerie. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):220–230.

[102] Zhang, Z., Zou, Z., Li, N., and Chen, Y. (2022). Classifying galaxy morphologies with few-shot learning. *Research in Astronomy and Astrophysics*, 22(5):055002.

[103] Zhu, X.-P., Dai, J.-M., Bian, C.-J., Chen, Y., Chen, S., and Hu, C. (2019). Galaxy morphology classification with deep convolutional neural networks. *Astrophysics and Space Science*, 364(4):1–15.

VITA AUCTORIS

NAME:	Jaykumar Patel
PLACE OF BIRTH:	Mehsana, India
YEAR OF BIRTH:	1998
EDUCATION:	Dharmsinh Desai University, B.Tech in Computer Engineering, Nadiad, India, 2020
	University of Windsor, M.Sc in Computer Science, Windsor, Ontario, 2023