
# Mar Athanasius College of Engineering
# Kothamangalam

## Initial Project Report

# STAR-GALAXY CLASSIFICATION USING DEEP LEARNING

Done by

**AJAY DAS M**

Reg No: MAC23MCA-2008

Under the guidance of

Prof. Nisha Markose

# Abstract

The challenge of accurately classifying astronomical objects as stars or galaxies has been a fundamental task in astrophysics for centuries. Traditional methods relied heavily on visual inspection and morphological analysis, which were labour-intensive and limited by human subjectivity and the capacity to process large data volumes. With the advent of modern sky surveys like the Sloan Digital Sky Survey (SDSS), the volume of astronomical data has grown exponentially, rendering manual classification impractical.

The literature survey across the reviewed papers highlights three algorithms Convolution Neural Network (CNN), deep convolutional neural networks (ConvNets), ContextNet where taken into consideration.

The performance of deep learning architecture Convolution Neural Network (CNN) is used to classify stars and galaxies. Steps include rejecting data with errors, correcting for extinction, aligning images, and centring objects using nMontage and SExtractor.

The Dataset is taken from the Kaggle repository, the dataset contains 3986 data which 942 galaxy 3044 Star data.

Among the three Architecture, the Convolution Neural Network (CNN) is found to be best in terms of model building and computation. Thus, Star-Galaxy Classification Using Deep learning offers significant benefits for star-galaxy classification, including reduced human error, increased scalability, and efficient handling of vast data quantities.

## References:

- Ganesh Ranganath Chandrasekar Iyer Krishna Chaithanya Vastare (2017). Deep Learning for Star-Galaxy Classification
- Kim EJ, Brunner RJ. Star-galaxy classification using deep convolutional neural networks. Monthly Notices of the Royal Astronomical Society. 2016 Oct 17:stw2672.

- Kennamer N, Kirkby D, Ihler A, Sanchez-Lopez FJ. ContextNet: Deep learning for star galaxy classification. In International conference on machine learning 2018 Jul 3 (pp. 2582-2590). PMLR.



**Submitted By:**

Ajay Das M

MAC23MCA-2008

**Faculty Guide:**

Prof. Nisha Markose

Associate Professor

MCA Dept, MACE

# INTRODUCTION

The challenge of accurately classifying astronomical objects as stars or galaxies has been a fundamental task in astrophysics for centuries. Traditional methods relied heavily on visual inspection and morphological analysis, which were labour-intensive and limited by human subjectivity and the capacity to process large data volumes. With the advent of modern sky surveys like the Sloan Digital Sky Survey (SDSS), the volume of astronomical data has grown exponentially, rendering manual classification impractical.

The literature survey across the reviewed papers highlights three algorithms Convolution Neural Network (CNN), deep convolutional neural networks (ConvNets), ContextNet where taken into consideration.

The performance of deep learning architecture Convolution Neural Network (CNN) is used to classify stars and galaxies. Steps include rejecting data with errors, correcting for extinction, aligning images, and centring objects using nMontage and SExtractor.

The Dataset is taken from the Kaggle repository, the dataset contains 3986 data which 942 galaxy 3044 Star data.

Among the three Architecture, the Convolution Neural Network (CNN) is found to be best in terms of model building and computation. Thus, Star-Galaxy Classification Using Deep learning offers significant benefits for star-galaxy classification, including reduced human error, increased scalability, and efficient handling of vast data quantities.

# LITERATURE SUMMARY

**Paper 1**

This project explores a CNN-based classifier to address these limitations. The paper "Deep Learning for Star-Galaxy Classification" (2017) demonstrates that Convolutional Neural Networks (CNNs) can effectively distinguish between stars and galaxies in astronomical images, achieving higher accuracy than traditional methods.

| | |
|---|---|
| **Title of the paper** | Ganesh Ranganath Chandrasekar Iyer Krishna Chaithanya Vastare (2017). Deep Learning for Star-Galaxy Classification |
| **Area of work** | Using deep learning, specifically Convolutional Neural Networks (CNNs), for classifying stars or galaxies. |
| **Dataset** | Dataset was taken from the Sloan Digital Sky Survey (SDSS). The dataset contains 30 million images. |
| **Methodology / Strategy** | CNN-based binary star-galaxy classifier involves collecting labelled image data from sources like the SDSS, pre-processing the data by normalizing and resizing images, and splitting it into training, validation, and test sets. A CNN is designed with convolutional and pooling layers for feature extraction, followed by fully connected layers for classification, with a sigmoid output layer for binary classification. The model is trained using binary cross-entropy loss and the Adam optimizer, then evaluated using accuracy, precision, recall, and F1-score metrics. Finally, the trained model is deployed to classify new astronomical data. |
| **Architecture** | Convolutional Neural Networks(CNN) |
| **Result/Accuracy** | CNN(Convolutional Neural Networks) – 99.19 |

**Page 2**

Kim and Brunner (2016) developed a deep CNN approach for classifying stars and galaxies in astronomical images. Their method improves accuracy by effectively learning from the features in the images, outperforming traditional classification techniques.

| | |
|---|---|
| **Title of the paper** | Kim EJ, Brunner RJ. Star-galaxy classification using deep convolutional neural networks. Monthly Notices of the Royal Astronomical Society. 2016 Oct 17:stw2672. |
| **Area of work** | Star-galaxy classification using deep convolutional neural networks. |
| **Dataset** | photometric and spectroscopic data sets with different characteristics and compositions.<br>data sets and the image pre-processing steps for retrieving cutout images |
| **Methodology / Strategy** | The research uses deep convolutional neural networks (ConvNets) to classify astronomical objects from SDSS and CFHTLenS survey data. The ConvNet, with several convolutional and fully connected layers, employs data augmentation and dropout to reduce over fitting. The study compares ConvNet performance to the Trees for Probabilistic Classifications (TPC) algorithm, focusing on accuracy and probabilistic calibration. |
| **Architecture** | Convolutional Neural Networks (ConvNets) |
| **Result/Accuracy** | ConvNet - 99.48 |

**Page 3**

The paper titled "ContextNet: Deep Learning for Star Galaxy Classification" presents a framework for classifying stars and galaxies in astronomical images, specifically for data from the Large Synoptic Survey Telescope (LSST)

| | |
|---|---|
| **Title of the paper** | Kennamer N, Kirkby D, Ihler A, Sanchez-Lopez FJ. ContextNet: Deep learning for star galaxy classification. In International conference on machine learning 2018 Jul 3 (pp. 2582-2590). PMLR. |
| **Area of work** | The work applies ContextNet Architecture to classify stars and galaxies in astronomical images from ground-based surveys like the LSST |
| **Dataset** | The dataset used in the work consists of simulated images from the Large Synoptic Survey Telescope (LSST) observations, generated using the GalSim image simulation package. |
| **Methodology / Strategy** | The methodology uses ContextNet, a three-step neural network framework. It includes a local network for individual object features, a global network for comparing features across objects to capture context, and a prediction network that combines these features for classification. This approach handles non-IID data and improves accuracy by leveraging neural network weight replication for variable object numbers in each exposure. |
| **Architecture** | **Local Network**: Convolutional Neural Networks (CNNs)<br>**Global Network**: Recurrent Neural Networks (RNNs)<br>**Prediction Network**: Fully Connected Neural Networks (FCNs) |
| **Result/Accuracy** | ContextNet - 95% |

# PAPER SUMMARY

From the above three papers, we get to know that different models were used for the classification of Stars and Galaxies. The initial project report on star-galaxy classification using deep learning explores three key research papers that leverage different neural network architectures for this task. The first paper, "Deep Learning for Star-Galaxy Classification" (2017), focuses on using Convolutional Neural Networks (CNNs) to classify astronomical objects. This study utilized a large dataset from the Sloan Digital Sky Survey (SDSS) and demonstrated that CNNs could achieve high accuracy in distinguishing between stars and galaxies, with the model reaching an accuracy of 99.19%. The CNN-based approach was found to be highly effective, emphasizing the strength of deep learning in handling complex classification tasks.

The second paper, "Star-Galaxy Classification Using Deep Convolutional Neural Networks" (2016), by Kim EJ and Brunner RJ, further advanced the use of deep learning by employing deep convolutional neural networks (ConvNets). This study worked with photometric and spectroscopic datasets from the SDSS and CFHTLenS surveys and incorporated techniques like data augmentation and dropout to enhance model performance. The ConvNet model outperformed traditional classification methods, achieving a remarkable accuracy of 99.48%. This research highlighted the potential of deep learning to improve the precision and reliability of astronomical classifications.

The third paper, "ContextNet: Deep Learning for Star-Galaxy Classification" (2018), introduced a more complex architecture known as ContextNet, designed to handle data from the Large Synoptic Survey Telescope (LSST). ContextNet integrates CNNs, Recurrent Neural Networks (RNNs), and Fully Connected Neural Networks (FCNs) to capture both local and global features of astronomical images. Although this model achieved a slightly lower accuracy of 95%, it offered a sophisticated approach to addressing the challenges posed by non-independent and identically distributed (non-IID) data in astronomical surveys. Together, these studies underscore the effectiveness of deep learning, particularly CNNs, in advancing star-galaxy classification.

# PROJECT PROPOSAL

Astronomical object classification, particularly distinguishing between stars and galaxies, has long been a fundamental challenge in astrophysics. Traditional methods, such as visual inspection and morphological analysis, are labour-intensive and limited in handling the growing volume of astronomical data generated by modern sky surveys. With the advent of deep learning, specifically Convolutional Neural Networks (CNNs), there is now a significant opportunity to automate and improve the accuracy of star-galaxy classification. This project aims to implement a CNN-based classifier, building on the work presented in the paper "Deep Learning for Star-Galaxy Classification" (2017), which demonstrated the effectiveness of CNNs in this domain.

The model will be trained with the help of a dataset taken from kagil repository. Kaggle repository. The dataset contains 3986 sample observations with 942 Galaxies and 3044 Stars photometric data.

The proposed system uses Convolution Neural Network (CNN). The models will classify photometric data under two classes Star and Galaxy. An automated system can be very helpful to offers significant benefits for star-galaxy classification, including reduced human error, increased scalability, and efficient handling of vast data quantities.

## DATASET

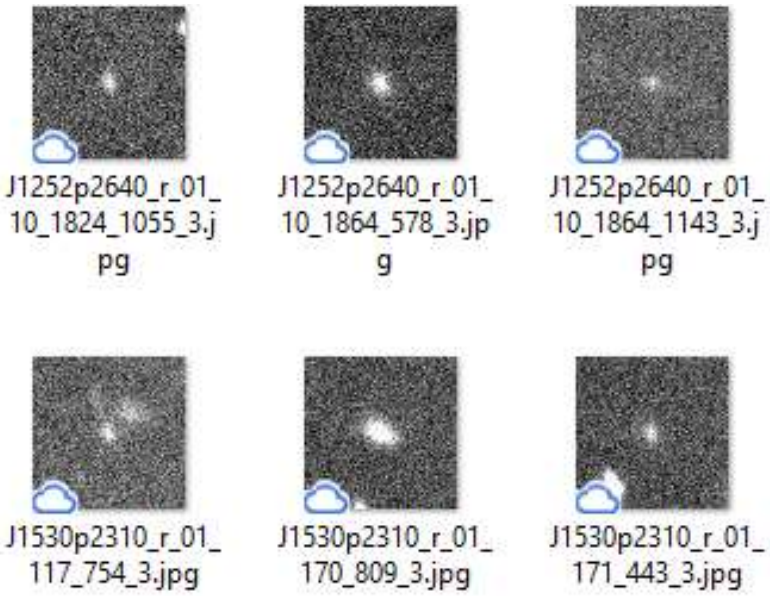
The dataset is taken from the Kaggle repository. The dataset contains 3986 sample observations with 942 Galaxies and 3044 Stars photometric data

The dataset contains a collection of astronomical images captured using a 1.3-meter telescope located in Nainital, India. These images feature stars, galaxies, and other celestial objects. Researchers and data scientists can utilize this dataset for various tasks, including star-galaxy classification, object detection, and image analysis. The dataset provides a valuable resource for exploring the cosmos through machine learning and computer vision techniques.

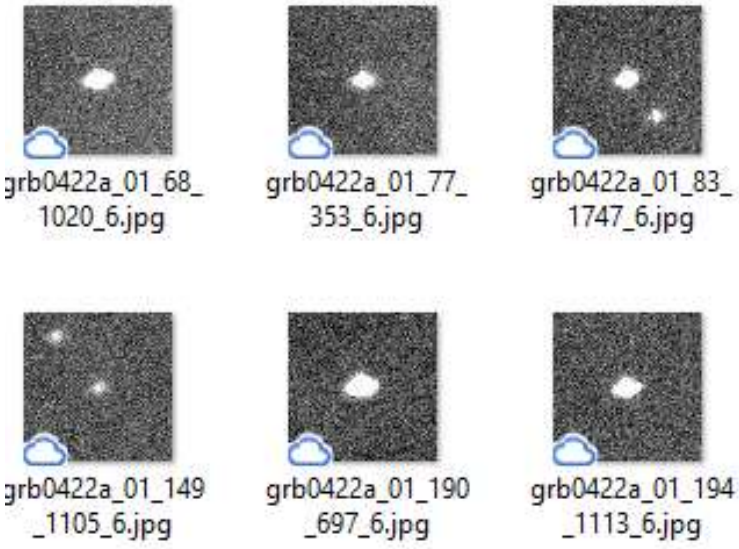
**Dataset**: https://www.kaggle.com/datasets/divyansh22/dummy-astronomy-data

## EXPLORATORY ANALYSIS

The dataset contains 3986 sample observations with 942 Galaxies and 3044 Stars photometric data




**Galaxy data example**




**Star data example**

# DATA PREPROCESSING

## Resizing Image

**Uniform Image Size**: Since CNNs require fixed-size input images, all images in the dataset must be resized to a uniform size. A common choice for this type of classification task is 64x64 or 128x128 pixels, although the size can be adjusted based on the computational resources available and the complexity of the objects in the images.

**Aspect Ratio Consideration**: Ensure that resizing doesn't distort the images, particularly if the original images have different aspect ratios. In some cases, padding the images to maintain aspect ratios might be necessary.

## Normalize

**Pixel Value Scaling**: CNNs perform better when input data is normalized. Typically, image pixel values are scaled from their original range (0-255 for 8-bit images) to a range of 0-1 or -1 to 1. This is done by dividing the pixel values by 255.

Normalization helps in speeding up the convergence during training by ensuring that the input features have a similar scale.

## Data Augmentation

Data augmentation artificially increases the size of the training dataset by creating modified versions of images in the dataset. This helps the model generalize better and become more robust to variations.

Technique:

**Rotation**: Randomly rotate images within a certain range to simulate different orientations of celestial objects.

**Flipping**: Horizontally or vertically flip the images to introduce symmetry variations.

**Zooming**: Randomly zoom in on images to simulate different scales.

**Brightness/Contrast Adjustments**: Modify the brightness and contrast of the images to account for different lighting conditions in the data.

**Translation**: Shift images horizontally or vertically to simulate positional variance.

**Splitting the Database**

**Training Set**: 80% of the dataset is used for training. This is the subset of data the model will learn from.

**Test Set**: The remaining 20% is reserved for testing the model after training to evaluate its performance on unseen data.

## WORKING OF THE ALGORITHM

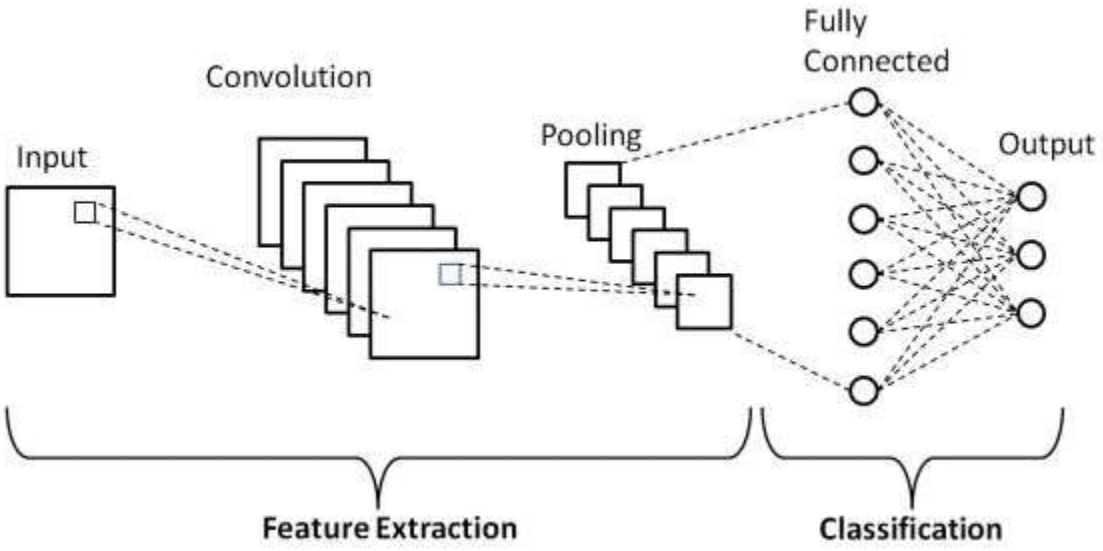
### Network Architecture of CNN

CNNs are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analyzing visual images. Their applications range from image and video recognition, image classification, medical image analysis, computer vision and natural language processing.

Convolutional Neural Networks (CNNs) are deep learning models that extract features from images using convolutional layers, followed by pooling and fully connected layers for tasks like image classification.

The main layers of CNN are:

- Input Layer
- Convolution Layer
- Pooling Layer
- Fully-Connected (dense)Layers
- Output Layer




***Architecture Diagram***

**Dimension Table**

| Type | Filters | Filter Size | Padding | Non-linearity | Initial Weights | Initial Biases |
|---|---|---|---|---|---|---|
| Convolutional | 32 | 5 × 5 | - | Leaky ReLU | Orthogonal | 0.1 |
| Convolutional | 32 | 3 × 3 | 1 | Leaky ReLU | Orthogonal | 0.1 |
| Pooling | - | 2 x 2 | - | - | - | - |
| Convolutional | 64 | 3 x 3 | 1 | Leaky ReLU | Orthogonal | 0.1 |
| Convolutional | 64 | 3 x 3 | 1 | Leaky ReLU | Orthogonal | 0.1 |
| Convolutional | 64 | 3 x 3 | 1 | Leaky ReLU | Orthogonal | 0.1 |
| Pooling | - | 2 x 2 | - | - | - | - |
| Convolutional | 128 | 3 x 3 | 1 | Leaky ReLU | Orthogonal | 0.1 |
| Convolutional | 128 | 3 x 3 | 1 | Leaky ReLU | Orthogonal | 0.1 |
| Convolutional | 128 | 3 x 3 | 1 | Leaky ReLU | Orthogonal | 0.1 |
| Pooling | - | 2 x 2 | - | - | - | - |
| Fully-Connected | 2048 | - | - | Leaky ReLU | Orthogonal | 0.01 |
| Fully-Connected | 2048 | - | - | Leaky ReLU | Orthogonal | 0.01 |
| Fully-Connected | 2 | - | - | Softmax | Orthogonal | 0.01 |

**Layers in CNN Architecture**

**Input Layer**

This layer accepts the raw input images, which is in this case 64 x 64 pixel images in five photometric bands (u, g, r, i, z). This layer provides the initial data (images) to the network, which will be processed and analyzed to distinguish between stars and galaxies.

**Convolution Layer**

This layer Applies convolution operations to the input data, using a set of filters (kernels) to extract features such as edges, textures, and shapes. These layers detect various features at different levels of abstraction. Early layers might detect basic features like edges, while deeper layers detect more complex structures relevant to differentiating stars from galaxies.

***Activation Function(Leaky ReLU)***

*This is Applies a non-linear transformation to the output of each convolutional layer. Leaky ReLU helps in avoiding the problem of dead neurons by allowing a small, non-zero gradient when the unit is not active. Introduces non-linearity to the model, enabling it to learn from complex data and to improve feature detection and classification.*

**Pooling Layer**

This layer Reduces the spatial dimensions (width and height) of the feature maps, retaining the most critical information while reducing the computational load and controlling overfitting. By reducing the dimensionality, these layers help in abstracting the features detected by convolutional layers and make the network more computationally efficient.

**Fully-Connected (Dense) Layers**

In this layer each neuron in these layers is connected to every neuron in the previous layer, which allows the network to combine the features extracted by the convolutional and pooling layers and make final predictions. These layers are

responsible for the final classification, combining all learned features to distinguish whether an object in the image is a star or a galaxy.

**Output Layer**

This is the final Layer of the architecture this layer Produces the final output, typically using a softmax function in classification tasks to produce probabilities for each class. His layer outputs the probability of the image belonging to either the "star" or "galaxy" class, allowing for final decision-making in the classification task.

## Functions And Packages

**TensorFlow/Keras**: For building and training the CNN model.

- Sequential()
- Conv2D()
- .MaxPooling2D()
- Flatten()
- Dense()
- Dropout()

**NumPy**: For numerical operations and handling arrays.

- array()
- reshape()
- mean()
- std()
- expand_dims()
- argmax()

**Pandas**: For data manipulation and handling datasets

- read_csv()
- DataFrame()
- DataFrame.head()
- DataFrame.describe()

**Matplotlib/Seaborn**: For plotting and visualization

- pyplot.plot()
- pyplot.imshow()
- pyplot.show()
- heatmap()

**Scikit-learn**: For pre-processing, metrics, and splitting data.

- sklearn.model_selection.train_test_split
- sklearn.preprocessing.LabelEncoder
- sklearn.metrics.confusion_matrix
- sklearn.metrics.accuracy_score

- sklearn.metrics.classification_report
- sklearn.metrics.roc_curve
- sklearn.metrics.auc

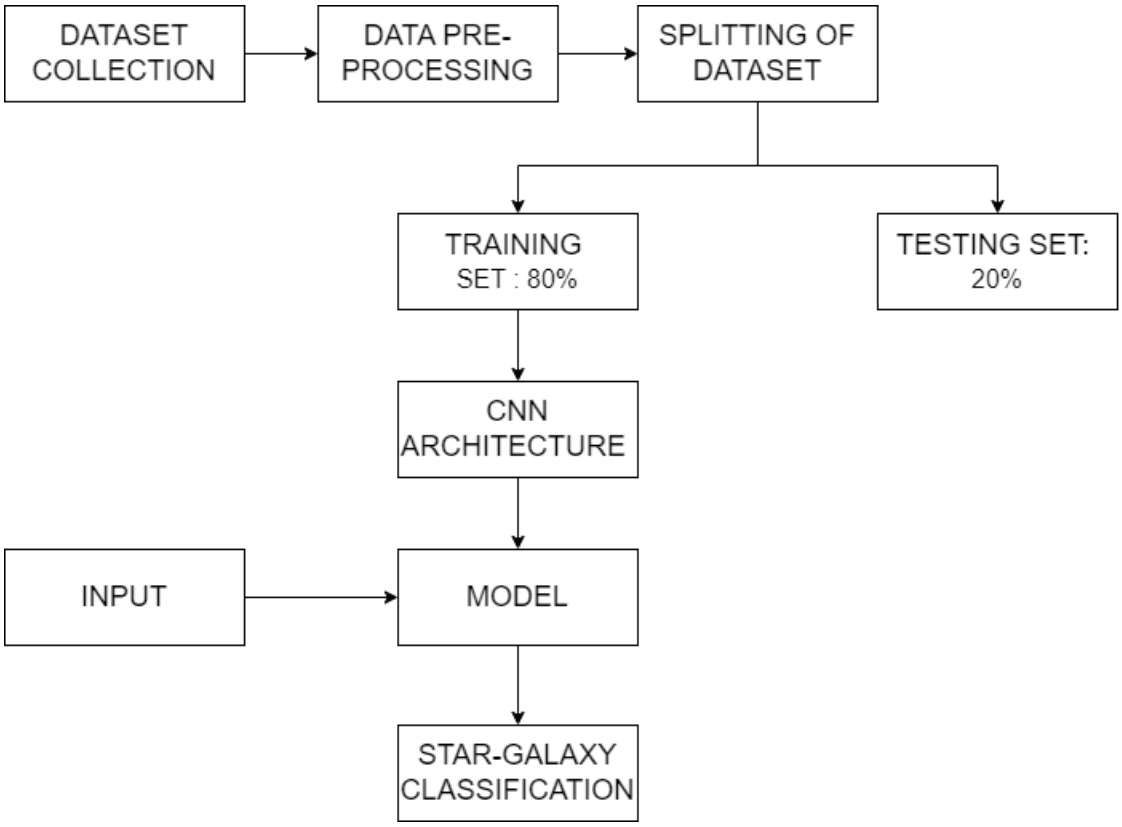
**OpenCV**: For image processing (if needed)

- imread()
- cvtColor()
- resize()
- imshow()

**os**: For interacting with the operating system (e.g., for directory management)

- listdir()
- path.join()
- makedirs()

**Project Pipeline**



# SYSTEM DESIGN

## Model Building:

- **Training**

The Convolution Neural Network model is constructed using a deep learning approach to extract features from images and classify them accordingly. Below are the detailed steps involved in the model-building process, including model architecture and training procedure using the training dataset.

**Step 1**: Importing Required Libraries

First, we import the necessary libraries such as TensorFlow and Keras for building and training the Capsule Network.

**Step 2**: Convolution Neural Network Model Architecture

The Convolution Neural Network is designed using the following layers:

1. Input Layer
2. Convolution Layer 1 (32 filters, 3x3 kernel, ReLU activation)
3. MaxPooling Layer 1 (2x2 pool size)
4. Convolution Layer 2 (64 filters, 3x3 kernel, ReLU activation)
5. MaxPooling Layer 2 (2x2 pool size)
6. Convolution Layer 3 (128 filters, 3x3 kernel, ReLU activation)
7. MaxPooling Layer 3 (2x2 pool size)
8. Flatten Layer
9. Dense Layer 1 (128 units, ReLU activation)
10. Dropout Layer 1 (50% rate)
11. Dense Layer 2 (64 units, ReLU activation)
12. Dropout Layer 2 (50% rate)
13. Output Layer (2 units, Softmax activation)

**Step 3**: Model Compilation and Class Weights

After building the model, it is compiled using the Adam optimizer with a learning rate

Schedule. The custom margin loss function is utilized to optimize the training process. Additionally, class weights are computed to address any class imbalance present in the dataset.

• Learning Rate Schedule: This helps in gradually decreasing the learning rate during training.

• Class Weights: Computed to balance the influence of different classes during training.

Step 4: Model Training

The model is trained using the training dataset while monitoring validation loss to prevent over fitting through early stopping.

• Early Stopping: A call back is used to stop training when validation loss does not Improve for a specified number of epochs.

- **Testing**

Model Performance:

The Convolution Neural Network model was evaluated using a test dataset to classify images into two

categories: Star and Galaxy.

• Test Accuracy: 88.59%

• Test Loss:  28.75%

Training Metrics:

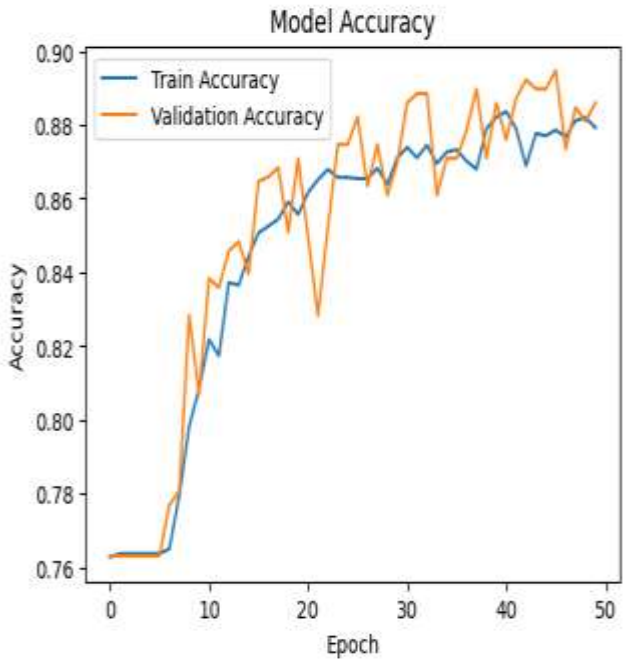
• Training Accuracy: 87.72%

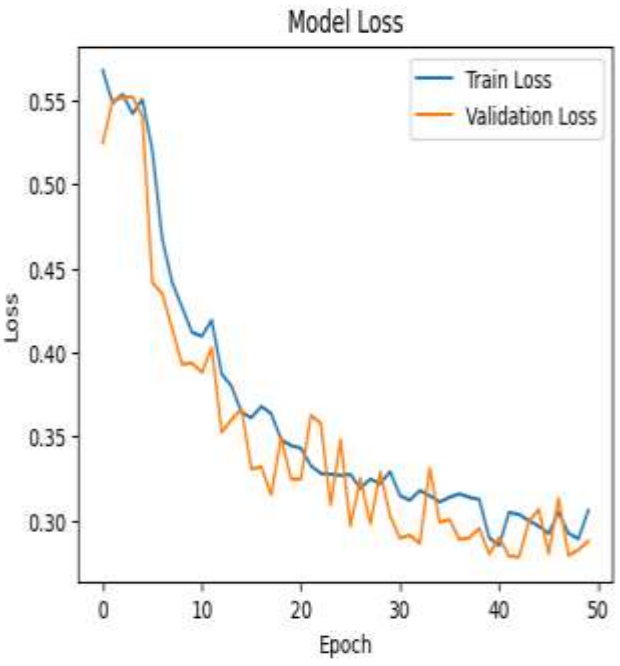
• Training Loss: 29.91%

**Evaluation Matrix:**

| Metric | Galaxy | Star | Accuracy | Macro Average | Weighted Average |
|---|---|---|---|---|---|
| **Precision** | 0.24 | 0.76 | | 0.50 | 0.64 |
| **Recall** | 0.21 | 0.79 | | 0.50 | 0.66 |
| **F1-Score** | 0.23 | 0.78 | 0.66 | 0.50 | 0.65 |
| **Support** | 189 | 609 | 798 | 798 | 798 |

**Learning Curves:**



## Project TimeLine

- ➢ Submission of project synopsis with Journal Papers - 22.07.2024
- ➢ project proposal approval - 26.07.2024
- ➢ presenting project proposal before the Approval Committee - 29.07.2024 & 30.07.2024
- ➢ Initial report submission - 12.08.2024
- ➢ Analysis and design report submission - 16.08.2024
- ➢ First project presentation - 21.08.2024 & 23.08.2024
- ➢ Sprint Release I - 30.08.2024
- ➢ Sprint Release II - 26.09.2024
- ➢ Interim project presentation - 30.09.2024 & 01.10.2024
- ➢ Sprint Release III - 18.10.2024
- ➢ Submission of the project report to the guide - 28.10.2024
- ➢ Final project presentation - 28.10.2024 & 29.10.2024
- ➢ Submission of project report after corrections - 01.11.202