Classifying Galaxies Using Convolution Neural Network

Rongyan Zhu
Rensselaer Polytechnic Institute
110 8th St, Troy, NY, the U.S. 12180
zhur5@rpi.edu

Qixuan Huang
University of Malaya
50603 Kuala Lumpur, Malaysia
22065325@siswa.um.edu.my

Haoxuan Xie
Macao Polytechnic University
R. de Luís Gonzaga Gomes, Macao, China
p2211355@mpu.edu.mo

Chao Gao
Shanghai University
99 Shangda Road, BaoShan District, Shanghai, China
3194608363@shu.edu.cn

Jiayao Wang
Virginia Polytechnic and State University
Blacksburg, VA 24061, United States
jiayao@vt.edu

Abstract

This study aims to explore the application of Convolutional Neural Networks (CNN) in galaxy classification. The research utilized the Galaxy10 DECals dataset, which contains diverse galaxy images, to gain a basic understanding of CNN principles and applications. Ultimately, by building a CNN model, an analysis of galaxy images from the dataset was conducted, and the performance of CNN in galaxy classification was investigated.

In the experiments, it was demonstrated how CNN extracts features from the data, successfully classifying different types of galaxies. Furthermore, a detailed discussion on how to construct the CNN model was presented, including the application of techniques such as L2 regularization, batch normalization, and data augmentation, to enhance classification performance and solve potential challenges within the dataset.

In the end, the study used standard performance metrics like F1 score, recall, precision, and accuracy to thoroughly evaluate model's performance.

This research provides valuable experience and insights for beginners interested in understanding CNN and conducting galaxy classification research in the field of astronomy, laying a solid foundation for future studies.

Keywords: Convolutional Neural Networks, galaxy classification, image processing, deep learning.

1. Introduction

> Mankind was born on Earth. It was never meant to die here. [1]

This is the famous line from the movie *Interstellar* by the main character Cooper, who sat at the front door of his house, manifesting his determination towards his upcoming voyage across different galaxies in the space. Throughout the history, human beings' exploration in galaxies has always been filled with desires. Understanding them is the first step in our exploration. However, there may be over 2 trillion galaxies in the observable universe, and each of them is unique in shape, color, brightness, distribution of planets, formation, and etc. Without an appropriate, automated classification mechanism, it would be difficult for people to understand galaxies systematically and efficiently. Fortunately, classifying galaxies through machine learning is a proper and available way to do so.

The following research project aims to develop a Convolutional neural network (CNN) model, which calculates out the probabilities of each galaxy's belonging to a particular class of shape, and correctly predicts the shape classes of galaxies by utilizing technique of computer vision and deep machine learning.

To achieve such objective, the research project team utilized the dataset of Galaxy10 DECals from Python package `astroNN`. The dataset of Galaxy10 DECals contained 6 columns: `images`, `ans`, `ra`, `dec`, `redshift`, and `pxscale`. The `images` column was a four-dimensional array in the shape of (17736, 256, 256, 3), referring to

17736 256x256 pixels colored galaxy images (g, r, and z band); `ans` column referred to the 10 defined class labels for each image: Disturbed, Merging, Round Smooth, In-between Round Smooth, Cigar Round Smooth, Barred Spiral, Unbarred Tight Spiral, Unbarred Loose Spiral, Edge-on without Bulge, and Edge-on with Bulge; `ra` column referred to the right ascension coordinates for each galaxy; `dec` column referred to the declination coordinates for each galaxy; `redshift` column referred to the redshift values for each galaxy; `pxscale` column referred to the angular scale of one pixel in each galaxy, with units in arc-seconds per pixel.

The project involves loading galaxy images and their corresponding types into a Jupyter Notebook file, splitting them into training, validation, and testing datasets. A CNN model is then trained and validated using these datasets. The model's performance is evaluated by comparing its predictions on the testing dataset with the actual labels.

In the following parts of the research project essay, the project team are to discuss related works (Part 2), explain project approach (Part 3), report experimental results (Part 4), and summarize the project in conclusion (Part 5) and references (Part 6). Through this project, the research project team hope to contribute to the ongoing efforts in galaxy classification, aiding humanity's quest to understand and explore the galaxies.

While classifying galaxies is the main focus of the research project, it is worthwhile to consider known works from others upon research of CNN and galaxy classification. Without their efforts in the past, the research project would not make any progresses. Take a look at Part 2, Related Work.

2. Related Work

2.1. Introduction to Convolutional Neural Network

When it comes to image classification problems, the first word that appears in people's minds is CNN (Convolutional Neural Networks). In 1980, Fukushima built a neural network called Neocognitron artificial neural network. This model combines layers containing 2 kinds of cells to practice handwritten character recognition. The first kind of cell would do the convolutional operation on the input, the second one would downsample the input. The convolution part would get weight from input examples. Then, in 1998, the first deep CNN which had seven layers appeared to recognize handwritten characters. [2]

The modern CNN model normally includes the following parts: convolutional layers, activation functions, pooling layers, and fully connected layers. In convolutional layers, a convolution operation is performed on a small part of images (containing multiple pixels), which then outputs a single value for the neuron in the next layer. The key of convolution operation is a small matrix called the kernel. Different kinds of kernels can abstract different features of images such as edge features, texture features, shape features, and so on. Pooling layers are used to reduce the input size while preserving the salient features that we want the network to learn. The kernel is also the main part of the pooling layer, but it performs like getting the max pixel value or average value of a region of pixels. After convolution layers and pooling layers, adding one or more fully connected layers to generate the final classification result. [2] Between the fully connected layers and convolutional layers, we flatten the multidimensional data into one dimension shape which can fit the requirement of fully connected layers.

2.2. Efforts on Galaxy Classification

Galaxy is one of the most attractive topics in image classification. Galaxy classification is crucial in astronomy, as galaxy types reveal information on how the galaxy was formed and evolved. While manually conducting the classification task requires extensive background knowledge and is time-consuming, deep learning algorithms provide a time-efficient and expedient way of accomplishing this task. In the past few years, many people have tried building various CNN models to do Galaxy Classification. But Galaxy Classification is not easy for many reasons: the complexity and diversity of galaxy data, how to abstract features of galaxy, model optimization, noise, and uncertainty. People have different focuses when it comes to galaxy classification, like improving data collection, adjusting the structure of the CNN model itself, tuning and optimizing hyperparameters, applying regularization techniques, and so on. There are three kinds of research on this topic.

2.2.1 Optimization for Galaxy Classification

Fatih Ahmet ¸Senel concentrates on the hyperparameter Optimization for Galaxy Classification. He uses the GWO (Grey Wolf Optimizer) algorithm in solving optimization problems and improving other metaheuristic optimization algorithms. The GWO algorithm is inspired by Grey Wolves' survival and hunting strategies in nature. When performing hunting, different levels of grey wolves act in different roles like main force and assistant. In algorithms, the main force represents the best solution. The GWO algorithm finally got 0.934 train accuracy and 0.852 test accuracy separately which is great compared to other optimization algorithms. [3]

Considering the specialization of the Galaxy dataset, the choice of optimization algorithm is important for a good model.

2.2.2 Comparison on existing models

In this experiment, this research group investigates 11 light-weighted models: the ResNet50, the MobileNetV2, the MobileNetV3L, the EfficientNetV2S, the EfficientNetB0, the EfficientNetB1, the EfficientNetB2, the EfficientNetB3, the EfficientNetB4, the EfficientNetB5, the DenseNet121 and show that the DenseNet121 outperforms other models. It should be noted that the research uses the same dataset as we do – Galaxy10 DECals. They also use Morphological Opening to decrease the noise of images and Data Augmentation to avoid overfitting. In this case, DenseNet121 shows a great advantage over the second-best model EfficientNetV2S. [4]

The most significant feature of DenseNet121 is Dense Connection. It builds direction connection between layers so that every layer can access all the feature maps in previous layers. The comparison is meaningful but this group doesn't explain the reason why DenseNet211 can be the best.

2.3. Comparison from our work

The main purpose of our work is to compare the efficiencies of different ways of regularization. Not like other data, the images of galaxies have small differences which may lead to overfitting. We build several kinds of models to train the models and consider the results from these points: efficiency, accuracy, is overfitting or not. In the models, Dropout, Data Augmentation, BatchNormalization, and other methods are used. We also use ROC curve and confusion matrix to showcase the performance of models. The overfitting problem badly influences the classification of raw galaxy data, so a more appropriate and efficient solution is necessary.

3. Approach

We apply a classic neural network, CNN(In fact, each of us designs our own model, and the chosen one is the best one), to process images and effectively learn features and finally obtain the probability distributions of galaxies classification. Given a list of RGB image $\mathbb{R}^{15962\times256\times256\times3}$, we first convert the image storage form into a $15962 \times 256 \times 256 \times 3$ tensor. This is because the input should be tensors.

Our network contains three parts: convolutional layers, pooling layers and fully connected layers. The convolutional layers include 6 layers, each of which convolves the input tensors with a kernel or filter and outputs tensors. The process of convolution is to calculate dots between the input pixels and the filter. The final output from series of dots is what we called a feature map after multiple iterations. The pooling layers also involve kernels or filters across the entire input image. But each pooling layer is aimed to reduce the number of parameters in the input and improve the efficiency of computation. The fully connected layers perform the image classification task on the base of the feature maps extracted in the previous layers. All layers use ReLU as the activation function. Because it's concise and efficient enough, and has been shown to produce better results in many image classification tasks. The output layer use softmax as the activation function, because of its ability to provide class probabilities, normalization properties, differentiability, and handling of multiple classes.

Our optimizer is adam. Adam is an optimization algorithm that combines the benefits of RMSprop and momentum algorithms It enhances the convergence rate, generalization ability, and stability of the optimization process. Our loss function is categorical-crossentropy. It measures the dissimilarity between the predicted probability distribution and the true distribution of the target classes and commonly used in multi-classification problems. We also apply regularization techniques, including batch normalization, kernel regularization, and dropout, are commonly used to improve the generalization performance of convolutional neural networks.

3.1. Convolutional Layers

The convolutional layers structure is shown in Figure1.

As mentioned above, our network has 5 convolutional layers. The first convolutional layer in our model takes an input image with a size of 256x256 pixels and three channels (RGB). This layer has a set of 32 filters, each with a size of 3x3x3. During training, the weights of these filters are learned through backpropagation to capture specific patterns or features in the input image.

To compute the output feature maps, each filter slides over the input image, computing the element-wise product between the filter values and the overlapping section of the input, summing up the results, and adding a bias term. This process generates a 2D activation map for each filter, representing the response of the filter at each position in the input image.

After the convolution operation, the ReLU activation function is applied element-wise to the activation maps. This introduces non-linearity to the model and removes negative values.

Next, batch normalization is applied to the output of the convolutional layer. Batch normalization normalizes the activations along the batch dimension, which stabilizes the training process and improves the generalization performance of the model.

After batch normalization, a max pooling layer is applied with a pool size of 2x2.

The above process is repeated for the next three convolutional layers, each with 32 filters and a size of 2x2. The third convolutional layer is followed by another max pooling layer, while the fourth and fifth convolutional layers are

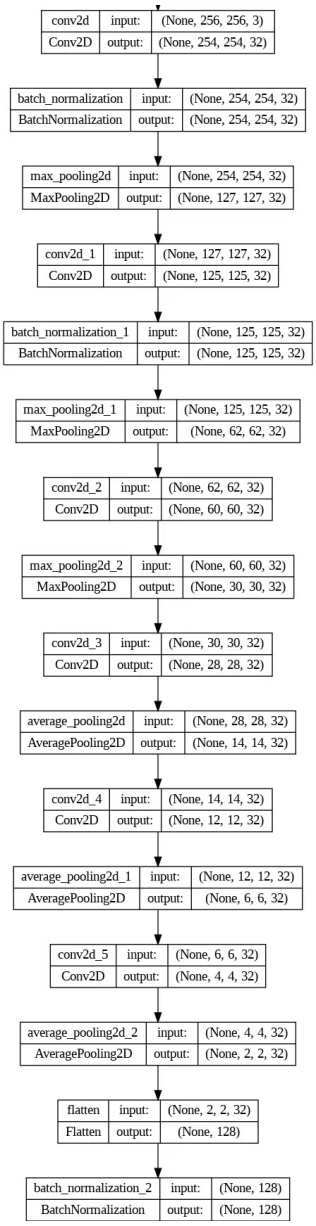

Figure 1. Convolutional Layers

followed by average pooling layers. The outputs of the final pooling layer are then flattened to form a feature vector, which is passed through a fully connected layer for classification.

3.2. Pooling Layers

We use max pooling as the first pooling operation because it is effective at extracting the most salient features from the feature maps and reducing the spatial dimensions. By selecting the maximum value within each pooling region, max pooling helps to capture the strongest presence of a particular feature in an image. On the other hand, average pooling calculates the average value within each pooling region. It acts as a form of spatial smoothing, which can provide a more global perspective of the features present in the input.

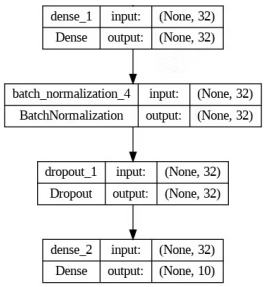

Figure 2. Fully Connected Layers

3.3. Fully connected layers

Our fully connected layers structure is shown in Figure2.

Our network consists of three fully connected (dense) layers with 32 units.

After each dense layer, a batch normalization layer is added. Batch normalization normalizes the output of the previous layer, making the model more stable during training and improving its ability to generalize to new data.

A dropout layer with a dropout rate of 0.2 is added after the batch normalization layers. Dropout randomly sets a fraction of the previous layer's units to zero during training, reducing overfitting by encouraging the network to learn more robust features.

Finally, the model ends with a dense layer of 10 units and a softmax activation function. This layer produces the output probabilities for classification into 10 classes.

4. Experimental Results

4.1. Training and Validation Accuracy

The Training and Validation Accuracy graph provides insights into how well the model is learning and how it performs on unseen data during the training process. The graph, as shown below, plots the accuracy of the model against the number of epochs.

From Figure 3, it can be observed that:

- The training accuracy (green curve) consistently increases as the number of epochs grows, indicating that the model is progressively learning from the training data.

- The validation accuracy (blue curve) follows a similar trend as the training accuracy in the initial epochs but later exhibits fluctuations. This behavior can be attributed to the model's encounters with variations in

Figure 3. Training and Validation Accuracy vs. Epochs

the validation set, which is a subset of the training data due to the validation split set at 0.2 or 20%.

- Around the 30th epoch, the validation accuracy tends to stabilize, suggesting that the model is reaching its optimal performance on the validation data. However, slight oscillations are still present, which is typical in neural network training due to factors like learning rate, batch size, and the stochastic nature of the training process.

- The gap between the training and validation accuracy remains reasonably small throughout the training process. This implies that the model is not overfitting significantly, as a substantial gap would indicate the model's inability to generalize well to new data.

Considering these observations, it's evident that the model's training process is both practical and stable, making it a suitable choice for further evaluations of the test data. The other models either display a divergence between training and validation accuracy, suggesting some over-fitting tendencies, or have relatively low validation accuracy. In contrast, the validation accuracy of our chosen model is relatively higher, indicating a more balanced performance.

4.2. Performance Comparison

The graph in Figure 4 provides a comparison of five different models across four performance metrics: accuracy, precision, recall, and F1 score.

Model	Accuracy	Precision	Recall	F1 Score
Model1	0.72000	0.71000	0.68000	0.68000
Model2	0.63416	0.61251	0.60795	0.59929
Model3	0.60000	0.67000	0.55000	0.57000
Model4	0.56257	0.60759	0.52363	0.51149
Model5	0.21000	0.17000	0.21000	0.21000

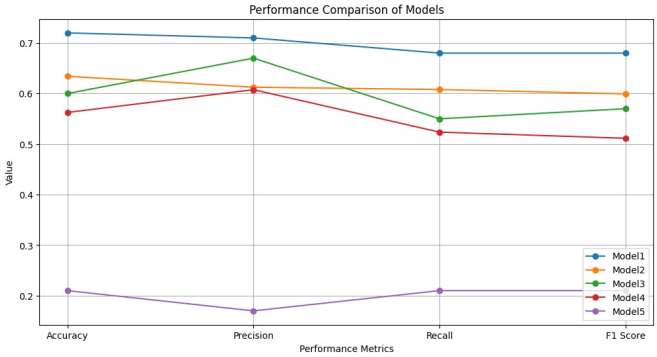

Figure 4. Performance Comparison of Models across Different Metrics

From the graph:

- **Model1**: Exhibits the best performance across all metrics.
- **Model2**: Consistently hovers around the 0.6 mark for all metrics.
- **Model3**: Has an accuracy around 0.65 with its other metrics, notably the F1 score, below this value.
- **Model4**: Metrics are around the 0.6 range with F1 score slightly lower.
- **Model5**: Performs the poorest among all models with values consistently below 0.3 for all metrics.

Given the metrics, **Model1** stands out as the superior choice. **Model5** might require significant reconsideration given its consistently low performance.

4.3. ROC Curve

The ROC (Receiver Operating Characteristic) curve is a graphical representation of a classifier's performance. It plots the true positive rate (sensitivity) against the false positive rate (1-specificity). The area under the curve (AUC) measures the model's ability to distinguish between the classes. From Figure 5, we can observe:

- **Micro-average ROC curve (area = 0.96)**: This computes the ROC AUC by considering the total number of true positives, false positives, and false negatives across all classes. An AUC of 0.96 indicates an excellent classification performance on a micro-level.

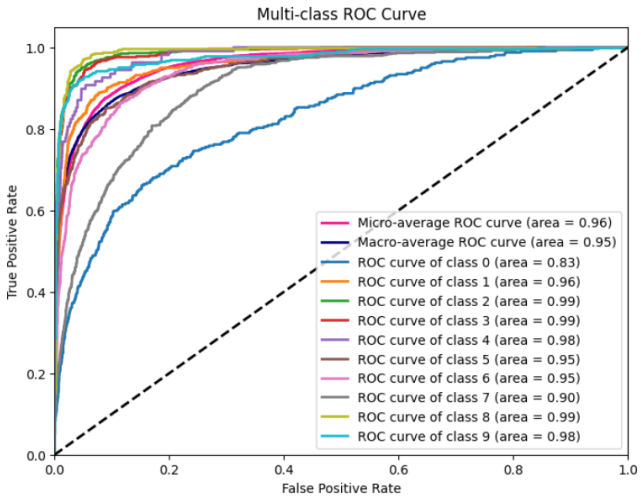

Figure 5. Multi-class ROC Curve

- **Macro-average ROC curve (area = 0.95)**: This computes the average ROC AUC score for each class and then takes the average. An AUC of 0.95 signifies excellent classification performance on a macro level.
- **Individual Class ROC AUC**: In binary classification, an AUC of 0.5 means that the model's predictions are no better than random guessing. An AUC of 1.0 indicates perfect predictions. From the individual class AUCs:
  - Class 2, 3, 4, 8, and 9 have AUCs close to 1 ($\geq$ 0.98), showcasing excellent discrimination.
  - Classes 5, 6, and 7 have AUCs of 0.95, indicating very good performance.
  - Class 1 has an AUC of 0.96.
  - Class 0 has the lowest AUC of 0.83, which is decent but suggests potential overlaps or misclassifications for this class compared to others.

In conclusion, the chosen model exhibits robust classification performance with AUCs significantly greater than 0.5 for all classes. An AUC closer to 1 means the model effectively distinguishes between positive and negative cases. This model performs exceptionally well across all classes.

4.4. Confusion Matrix

The Confusion Matrix provides a detailed visual representation of the model's performance across different classes. Each row represents the instances of an actual class, while each column corresponds to the instances as predicted by the model.

From Figure 6, several observations can be made:

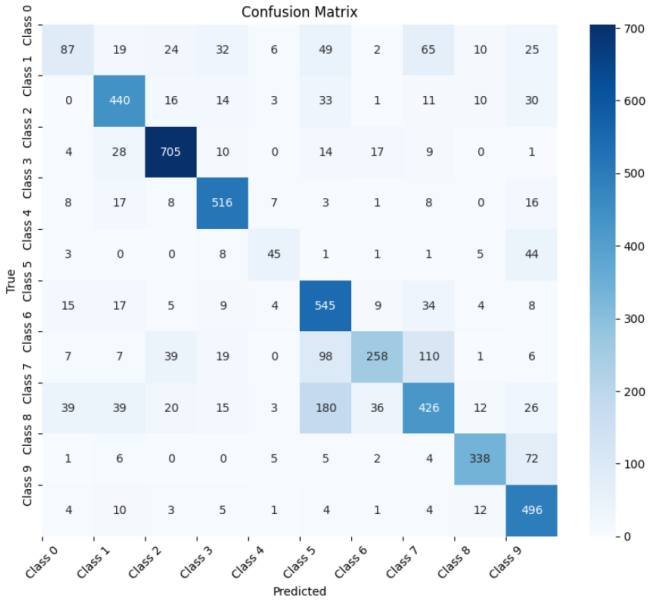

Figure 6. Confusion Matrix of the Model's Predictions

- **Class 0**: The model predicted 87 instances correctly but had notable misclassifications with Classes 5 and 7.
- **Class 1**: Achieved 440 correct predictions, with some confusion primarily with Classes 4 and 9.
- **Class 2**: Showcased an excellent performance, predicting 705 correctly.
- **Class 3**: Correctly predicted 516 instances, which is also excellent.
- **Class 4**: The model seems to be confused here. Only correctly classified 45 instances and have nearly half misclassifications with labeled class 9.
- **Class 5**: Predicted 545 instances accurately, but 34 instances were classified as Class 7.
- **Class 6**: Here, the model seems to be confused, with only 258 instances correctly classified out of 528. Notably, 110 instances were classified as Class 8.
- **Class 7**: The model did well with 426 correct predictions, but there was confusion with Class 5 where it classified 180 instances.
- **Class 8**: Correctly predicted 338 instances, but 72 instances were classified as Class 9.
- **Class 9**: The performance was strong with 496 correct predictions.

The diagonal from the top left to the bottom right of the matrix indicates the number of correct predictions for each class. It's evident that while the model has strong performance for specific classes, there are areas of confusion, especially between Class 6 and Class 7.

5. Conclusion

This study represents a collaborative effort by a team of researchers aimed at exploring the performance of multiple CNN models in galaxy classification tasks. Each team member designed a unique model, and the key findings are as follows:

Model A demonstrated excellent accuracy and exhibited outstanding performance in galaxy classification, particularly excelling in identifying elliptical galaxies.

Model B performed averagely in multi-class galaxy classification tasks, without any standout advantages, and it experienced overfitting issues.

Model C performed moderately in multi-class galaxy classification tasks.

Model D performed moderately in multi-class galaxy classification tasks.

Model E exhibited lower accuracy due to the increased time cost caused by the use of data augmentation techniques.

Through these independently designed models, a variety of solutions were considered for galaxy classification, enriching understanding of CNN technology in the field of astronomy.

Future research can extend these models, exploring additional designs and improvements to enhance the accuracy and efficiency of galaxy classification. Collaboration is encouraged, especially in the research of integrating multiple models to further enhance performance.

This study represents a collective effort and offers valuable insights for future CNN learning endeavors.

References

[1] Christopher Nolan. Interstellar, 2014. 1

[2] Burger Becker, Mattia Vaccari, Matthew Prescott, and Trienko Grobler. Cnn architecture comparison for radio galaxy classification. *Monthly Notices of the Royal Astronomical Society*, 503(2):1828–1846, 2021. 2

[3] Fatih ŞENEL. A hyperparameter optimization for galaxy classification. *Computers, Materials and Continua*, 74(2), 2023. 2

[4] Wuyu Hui, Zheng Robert Jia, Hansheng Li, and Zijian Wang. Galaxy morphology classification with densenet. In *Journal of Physics: Conference Series*, volume 2402, page 012009. IOP Publishing, 2022. 3