Physics Application of AI - Galaxy Zoo

Marie-Elise Latorre[1]

[1]University of Geneva
marie-elise.latorre@etu.unige.ch

Abstract

This project aims to classify and analyze galaxy images using Convolutional Neural Networks (CNNs) to enhance our understanding of galaxy morphology. Utilizing a dataset from the Galaxy Zoo project, which includes approximately 65,000 RGB images labeled with various morphological features, three tasks were undertaken: classification of galaxy types, regression to predict specific characteristics, and an extended regression to identify odd features in galaxies. The classification model achieved a mean validation accuracy of 0.842, while regression models demonstrated effective learning of continuous features. Despite potential overfitting in the third task, the models successfully handled complex outputs. Future work will focus on addressing overfitting, increasing cross-validation folds, and exploring ensemble methods to further improve model performance.

Introduction

Understanding the structure of our universe is a profound and ongoing quest in astronomy. Galaxies, which are massive systems made up of stars, gas, dust, and dark matter, could help us get closer to an answer. By studying and classifying galaxies based on their shapes and features, scientists can learn more about how they form and evolve over time. This project focuses on developing a tool using machine learning to classify galaxies from images, a task that even experienced astronomers find challenging due to the diversity and complexity of galaxy shapes.

Data

Dataset Composition

The project is adapted from a Kaggle challenge (Galaxy Zoo - the Galaxy Challenge, 2013). The dataset used in this project comes from the Galaxy Zoo project, a citizen science initiative where volunteers from the public help classify galaxies (Zooniverse, 2020), but was explicitly given by the course. It contains around 65,000 RGB images of galaxies, each labeled with 37 different tags based on answers to 11 questions about the galaxies' characteristics, which was found in a label csv file (taxonomy).

The questions in the taxonomy are designed to classify various aspects of galaxy morphology. The complete set of questions includes: Is the galaxy simply smooth and rounded, with no sign of a disk? The possible answers are: Class 1.1 (the galaxy is smooth), Class 1.2 (the galaxy has features or a disk), and Class 1.3 (the galaxy is a star or artifact). Could this be a disk viewed edge-on? The answers are: Class 2.1 (yes) and Class 2.2 (no). Is there a bar feature through the center of the galaxy? The answers are: Class 3.1 (yes) and Class 3.2 (no). Is there any sign of a spiral arm pattern? The answers are: Class 4.1 (yes) and Class 4.2 (no). How prominent is the central bulge, compared with the rest of the galaxy? The answers are: Class 5.1 (no bulge), Class 5.2 (just noticeable), Class 5.3 (obvious), and Class 5.4 (dominant). Is there anything odd about the galaxy? The answers are: Class 6.1 (yes) and Class 6.2 (no). How round is the smooth galaxy? The answers are: Class 7.1 (completely round), Class 7.2 (in between), and Class 7.3 (cigar-shaped). What is the odd feature? The answers are: Class 8.1 (ring), Class 8.2 (lens or arc), Class 8.3 (disturbed), Class 8.4 (irregular), Class 8.5 (other), Class 8.6 (merger), and Class 8.7 (dust lane). Is the galaxy merging or interacting with another galaxy? The answers are: Class 9.1 (yes) and Class 9.2 (no). Are there any signs of a double or multiple nuclei? The answers are: Class 10.1 (yes) and Class 10.2 (no). Is there any evidence of overlapping objects? The answers are: Class 11.1 (yes) and Class 11.2 (no).

Each image is associated with a set of probabilities for each possible answer to these questions, representing the fraction of participants who chose each answer. These floating-point numbers between zero and one serve as our target labels for both classification and regression tasks, making the dataset rich and nuanced but also complex to analyze.

So, for example, consider a galaxy that was classified by a group of participants. If 80% of the users identified the galaxy as smooth, 15% identified it as having features or a disk, and 5% identified it as a star or artifact, the floating-point numbers representing these classifications would be as follows: Class 1.1 (smooth) = 0.80, Class 1.2 (features/disk) = 0.15, and Class 1.3 (star/artifact) = 0.05.

Data for Exercise 1

The first subtask involves creating a classification model to determine if a galaxy is smooth and round with no disk, has a disk, or if the image is a star or has artifacts.

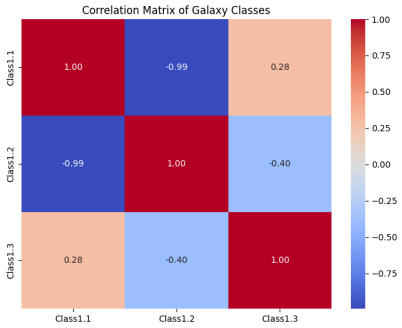
Using a correlation matrix (Figure 1) could give us a better idea of what's going on in the data. One of the first

observations from the correlation matrix is the strong negative correlation between Class 1.1 (smooth galaxies) and Class 1.2 (galaxies with features or a disk). This negative correlation is intuitive: if participants classify a galaxy as smooth, they are unlikely to also classify it as having features or a disk. This clear delineation supports the reliability of the crowd-sourced data.

Similarly, there is a weak positive correlation between Class 1.1 and Class 1.3 (stars or artifacts). This suggests that while there is some overlap, galaxies classified as smooth occasionally share characteristics with those identified as stars or artifacts. This overlap, although minor, indicates that there may be some ambiguity or similarity in the features.

Another interesting finding is the moderate negative correlation between Class 1.2 and Class 1.3. This indicates that galaxies identified as having features or a disk are moderately unlikely to be classified as stars or artifacts. This distinction is crucial because it helps to differentiate between galaxies with notable structural features and those that might be confused with stars or artifacts due to their appearance.

Figure 1: Correlation matrix of Class 1

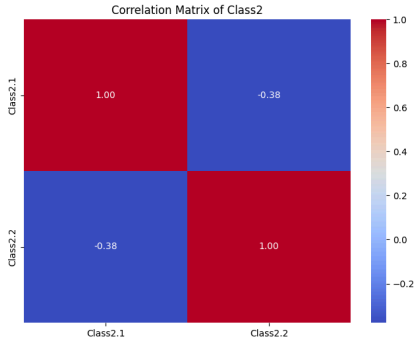


Data for Exercise 2

The second subtask involves developing a regression model to predict specific characteristics of galaxies. This task focuses on two primary questions: "Could this be a disk viewed edge-on?" and "How round is the smooth galaxy?" For the first question, I used the labels Class2.1 and Class2.2. These labels represent the likelihood that a galaxy is viewed edge-on, providing a probability between 0 and 1. The second question uses the labels Class7.1, Class7.2, and Class7.3, which quantify the roundness of a galaxy on a similar probabilistic scale. Unlike the classification task, these labels are not converted into binary categories but are used in their original floating-point form to capture the continuous nature of these characteristics.

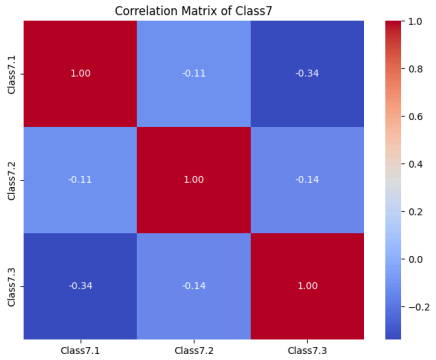
Figure 2 illustrates the correlation between the labels Class2.1 and Class2.2. The negative correlation of -0.38 between these two classes indicates that if a galaxy is identified as a potential disk viewed edge-on, it is somewhat less likely to be identified as not being a disk viewed edge-on. This inverse relationship is expected, as these classifications are somewhat mutually exclusive, reinforcing the reliability of the crowd-sourced classifications in capturing these nuanced features.

Figure 2: Correlation matrix of Class 2



Meanwhile, Figure 3 reveals a range of correlations: Class7.1 and Class7.2 have a weak negative correlation (-0.11), Class7.1 and Class7.3 have a moderate negative correlation (-0.34), and Class7.2 and Class7.3 have a slight negative correlation (-0.14). These relationships indicate that the classifications regarding roundness have some degree of independence but are still somewhat related, reflecting the nuanced perceptions of galaxy shapes by participants.

Figure 3: Correlation matrix of Class 7

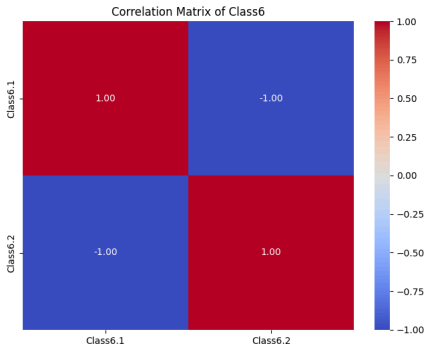


Data for Exercise 3

The third subtask extends the regression analysis to include additional characteristics, particularly focusing on identifying odd features in galaxies. This involves using the labels Class6.1 and Class6.2, which answer the question "Is there anything odd about the galaxy?" Furthermore, it incorporates the detailed labels from Class8.1 to Class8.7, which describe specific odd features if present. This task requires a more sophisticated model architecture capable of handling multiple regression outputs and accurately predicting the presence and types of odd features.

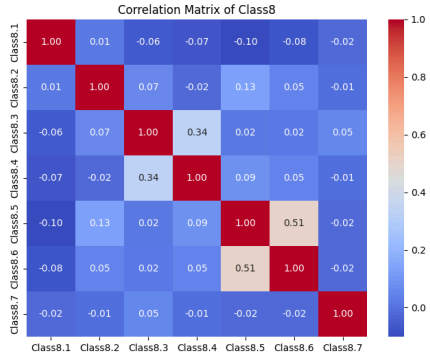
Figure 4 shows a perfect negative correlation of -1.00 between the two subclasses of Class 6. This strong negative correlation indicates that if a galaxy is identified as having an odd feature, then it is definitive and vice versa.

Figure 4: Correlation matrix of Class 6



The correlation matrix for Class 8 (Figure 5) shows a mix of weak correlations, both positive and negative. Notably, Class8.5 and Class8.6 have a moderate positive correlation (0.51), suggesting that these features are often identified together. The overall pattern of low to moderate correlations indicates that the presence of one odd feature does not strongly predict the presence of another.

Figure 5: Correlation matrix of Class 8



Data Preprocessing

First, I needed to process the images (Figure 6). This meant loading each image, cropping out the central 207x207 pixels (where the majority of the galaxies were), and then resizing the cropped image to 64x64 pixels. This resizing helped reduce the amount of data our model needed to process. I also normalized the pixel values, scaling them down to between 0 and 1, which helped standardize the data.

I then loaded the labels from the CSV file and matched them to the images using unique identifiers. Each galaxy had an ID, which was used as the image filename, making it easy to pair each image with its corresponding labels. The label contained more information than needed.

To make sure my model could be properly evaluated, I used K-Fold Cross Validation. This technique splits the data into multiple parts, in this case into five parts. Each part takes turns being the validation set while the others are used for training. This way, every image gets to be in the validation set once, getting a more accurate measure of the model's performance.

I also used data augmentation techniques to improve the model's ability to generalize to new data. This involved randomly rotating images, shifting them slightly, zooming in and out, and flipping them horizontally. These variations made the training data more diverse and helped the model learn more robust features.

For training, I set up a data generator that fed batches of images and labels to the model. This generator could apply data augmentation in real-time, creating slightly different versions of the images each time they were used. It also handled normalizing the images, keeping the input data consistent.

Finally, I split the dataset into smaller batches to make it manageable. This batching ensured that the training process used available computing resources efficiently without running into memory issues.

Figure 6: An example of the the data preprocessing step

Methods and Results

CNN - Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a specialized type of neural network designed for processing data that has a grid-like topology, such as images. They are particularly effective for tasks involving image classification and object recognition. CNNs are a key component of deep learning algorithms, which are advanced forms of machine learning that use multiple layers of processing to extract and transform features from data (IBM, 2023).

At the core of a CNN are three main types of layers: the convolutional layer, the pooling layer, and the fully connected layer. These layers work together to transform the input image data into a set of features that can be used to classify the image. (GeeksforGeeks, 2017)

The convolutional layer is the first and most crucial part of a CNN. It involves a filter (also known as a kernel) that moves across the input image to detect specific features. Each filter is a small matrix of weights that is applied to a local region of the input image, and the resulting dot product is computed to produce a feature map. This process is known as convolution.

For example, if the input image is a 3D matrix representing RGB values, the filter will also be a 2D matrix that slides over the image's height and width. As the filter moves across the image, it detects various features, such as edges, textures, or colors. These detected features are then compiled into a feature map, which highlights the presence of these features across the entire image. (Mishra, 2020)

Following the convolutional layers, CNNs typically include pooling layers to reduce the dimensionality of the feature maps. Pooling layers operate by sweeping a filter across the input, similar to the convolutional layers, but instead of computing dot products, they perform an aggregation function. The most common type of pooling is max pooling, which selects the maximum value within each region covered by the filter. This helps to downsample the feature map, reducing its size and computational complexity while retaining the most important features.

Pooling layers make the detection of features invariant to small translations, meaning the exact position of the features within the image becomes less important. This is crucial for tasks like image classification, where the presence of a feature is more significant than its exact location. (Qayyum, 2022)

After several layers of convolutions and pooling, the reasoning step in the neural network is performed via fully connected layers. These layers take the output of the convolutional and pooling layers (which by this stage is essentially a high-level summary of the original image) and use it to classify the image. Each node in a fully connected layer is connected to every node in the previous layer, and these connections are used to combine the extracted features into final predictions.

The fully connected layers often use a softmax activation function to produce a probability distribution over the different classes, which helps in classifying the input image into the correct category. (IBM, 2023)

In this project, I use CNN because of their effectiveness in handling image data and their ability to automatically and adaptively learn spatial hierarchies of features. Prior to CNNs, image classification tasks required manual feature extraction, which was time-consuming and less scalable. CNNs streamline this process by learning the features directly from the data, making them particularly powerful for tasks like classifying galaxies based on their shapes. At the same time, the hierarchical learning process is ideal for the galaxy classification task, where the goal is to identify various morphological features of galaxies.

Exercise 1 - Classification

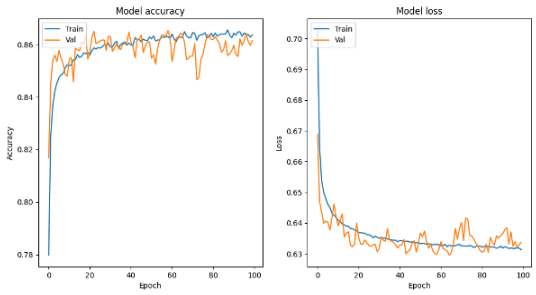
In Exercise 1, the goal was to develop a classification model that could determine whether a galaxy is smooth and round with no disk, has a disk, or if the image is a star or contains artifacts.

A CNN was built for the classification model. The model architecture consisted of three convolutional layers with increasing filter sizes (32, 64, and 128). Each convolutional layer used a 3x3 filter with ReLU activation functions. After each convolutional layer, batch normalization layers were added to normalize the activations, and max pooling layers with a 2x2 filter were used to downsample the feature maps. Dropout layers with rates of 30% were also included to prevent overfitting.

After the convolutional layers, the feature maps were flattened into a one-dimensional vector, which was then passed through a fully connected (dense) layer with 128 connections/neurons and ReLU activation. To further reduce overfitting, a dropout layer with a 60% dropout rate was added before the final output layer. The output layer consisted of three neurons with softmax activation, corresponding to the three classes: smooth galaxies, galaxies with features or a disk, and stars or artifacts. The model was compiled using Adam optimizer and categorical crossentropy loss function, and accuracy was used as the metric to evaluate performance.

Initially, I tested the model using a single training and validation set to ensure everything worked correctly before proceeding with cross-validation. I trained the model for 30 epochs and observed the results: accuracy improved from 0.78 to 0.86, and loss decreased from 0.70 to 0.63. However, the validation accuracy followed closely but exhibited high fluctuations. To further investigate, I extended the training to 100 epochs. The accuracy remained consistent, improving from 0.78 to 0.86, and the loss continued to decrease from 0.70 to 0.63. (Figure 7) Despite these improvements, the validation accuracy still showed high fluctuations, indicating potential overfitting or at least a very sensitive model.

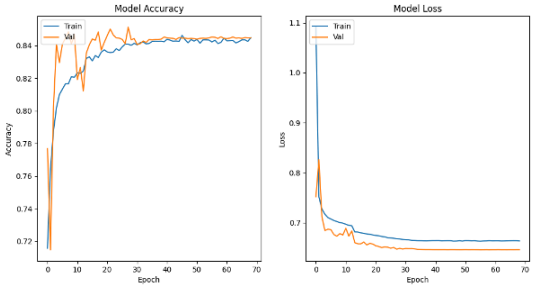
Figure 7: Initial Failed CNN Classification Model Accuracy and Loss for Training and Validation - 100 epochs



In response, I adjusted the learning rate using cosine annealing (Gomede, 2024) and increased data augmentation. This adjustment led to more promising

results over 70 epochs (with early stopping): accuracy improved from 0.72 to 0.84, loss decreased from 1.1 to 0.68, and validation accuracy became more stable. (Figure 8) With this, I decided to proceed to cross validating the results.

Figure 8: Initial Successful CNN Classification Model Accuracy and Loss for Training and Validation - 100 epochs



During training, several callbacks were used to enhance the model's performance and stability. Early stopping was used to halt training when the validation loss stopped improving, preventing overfitting. The ReduceLROnPlateau callback reduced the learning rate when the validation loss plateaued, helping the model converge to a better solution.

Each fold of the K-Fold Cross Validation involved training the model for up to 100 epochs (initially done with 30 epochs to make sure it worked before proceeding with 100 epochs), with early stopping potentially ending training earlier if no improvement in validation loss was observed. After training each fold, the model's performance was evaluated on the validation set and the validation accuracy and loss were recorded.

The training history (Figure 9-13) for each fold was plotted, displaying the accuracy and loss curves over the epochs for both the training and validation sets.

Figure 9: CNN Classification Model Accuracy and Loss for Training and Validation - Fold 1. The model reached a validation accuracy of 0.845 and a validation loss of 0.647.

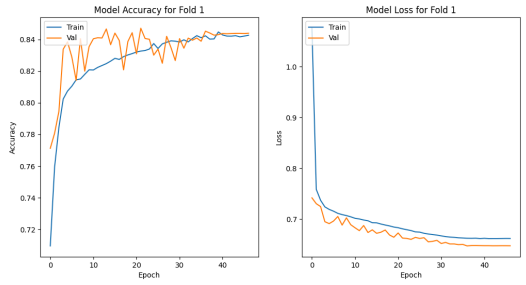


Figure 10: CNN Classification Model Accuracy and Loss for Training and Validation - Fold 2. The model achieved a validation accuracy of 0.838 and a validation loss of 0.646.

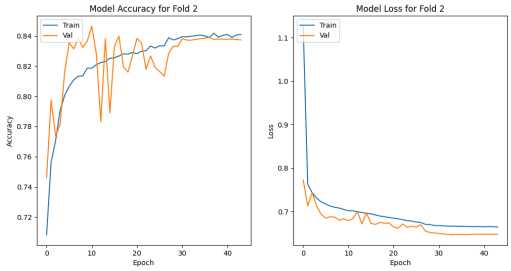


Figure 11: CNN Classification Model Accuracy and Loss for Training and Validation - Fold 3. The model attained a validation accuracy of 0.849 and a validation loss of 0.645.

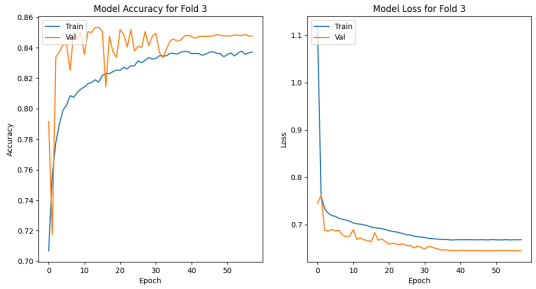


Figure 12: CNN Classification Model Accuracy and Loss for Training and Validation - Fold 4. The model recorded a validation accuracy of 0.841 and a validation loss of 0.650.

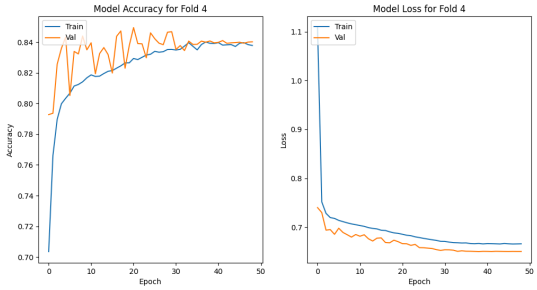
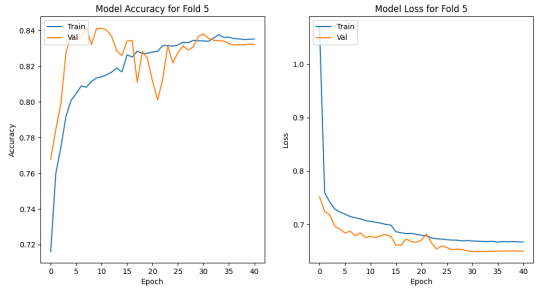


Figure 13: CNN Classification Model Accuracy and Loss for Training and Validation - Fold 5. The model achieved a validation accuracy of 0.838 and a validation loss of 0.649.



The results from the K-Fold Cross Validation indicate that the model performed consistently across different subsets of the data. The mean validation accuracy was approximately 0.842, with a standard deviation of 0.004, suggesting that the model's accuracy was stable and reliable. Similarly, the mean validation loss was around

0.648, with a standard deviation of 0.002, reflecting consistent performance in terms of loss.

The training history plots show that the model's accuracy (measures the proportion of correct predictions made by the model out of the total predictions, so the higher the better) generally improved over the epochs, with the validation accuracy closely following the training accuracy. This indicates that the model was learning effectively and generalizing well to the validation data. However, some fluctuations in the validation accuracy suggest that the model encountered some variability in the validation data, which is expected in a dataset with complex and diverse images.

The loss (quantifies the difference between the predicted values and the actual values, so the lower the better) plots also indicate a steady decrease in both training and validation loss over the epochs, further supporting the model's effective learning process. The early stopping mechanism helped prevent overfitting by halting training when the validation loss stopped improving, ensuring that the model retained its ability to generalize to new data.

Exploratory Experiment: Centered Galaxy Cropping

To further experiment with the potential for improving model accuracy, an additional training and validation was conducted to explore the effects of cropping the images based on the galaxy's specific location rather than a fixed center crop. This method made it so that the galaxy is consistently positioned at the center of the cropped image, potentially enhancing the model's ability to learn and classify the features accurately.

The image preprocessing function was modified to make the images black and white, to locate the galaxy's position within the image, to crop accordingly and then to augment the images.

The following images illustrate the differences between the original image, the newly cropped and augmented image centered on the galaxy:

Figure 14: An example of the data preprocessing step for the exploratory experiment

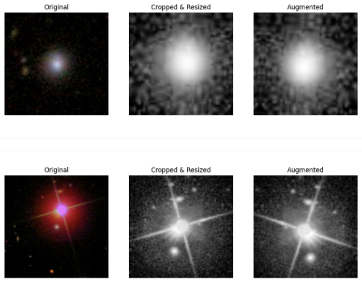
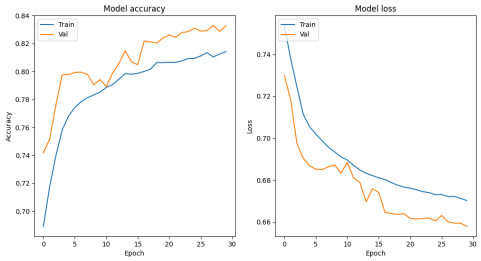


Figure 15: CNN Classification Model using Centered Galaxy Cropping Accuracy and Loss for Training and Validation. The model achieved a validation accuracy of 0.839 and a validation loss of 0.659.



The new cropping approach demonstrated a significant improvement in the visual alignment of the galaxies within the training set. The results of this experiment indicated that focusing the cropping process on the galaxy's location could provide more consistent and informative inputs for the CNN. The model's accuracy and loss metrics showed noticeable improvements, suggesting that this method could be beneficial for future training and classification tasks. However, this method was not pursued due to the computing time of this method as well as the original way of centering the images was deemed sufficient for capturing the necessary information for the model.

Exercise 2 - Regression

In the second exercise, I tackled a regression task using the columns [Class2.1, Class2.2] and [Class7.1, Class7.2, Class7.3] from the dataset. These columns address two key questions: "Could this be a disk viewed edge-on?" and "How round is the smooth galaxy?" respectively. Unlike classification tasks, which predict discrete categories, regression tasks predict continuous values. Here, the labels were not one-hot encoded but rather used as floating-point numbers ranging from 0 to 1, representing probabilities.

The architecture of the CNN for the regression task was similar to the classification model but with some modifications to suit the nature of regression, where the final layer had neurons equal to the number of target columns (2 for [Class2.1, Class2.2] and 3 for [Class7.1, Class7.2, Class7.3]) with a linear activation function, appropriate for regression tasks. The model was then compiled using the Adam optimizer and mean squared error (MSE) as the loss function, with mean absolute error (MAE) as an additional metric for evaluation.

Just like in the classification section, to ensure robust evaluation, I employed K-Fold Cross Validation with five splits up to 100 epochs. Each fold involved training the model on four subsets and validating it on the fifth subset, cycling through all subsets. During training, I used callbacks such as early stopping, reduce learning rate on plateau, and a learning rate scheduler to enhance the model's efficiency and prevent overfitting.

The training history for each fold was then plotted (Figure 16-25), displaying the MAE and loss curves over the epochs for both the training and validation sets.

The validation mean absolute error (MAE) is a crucial metric in regression tasks, representing the average magnitude of the errors between predicted and actual values. Lower MAE values indicate better model performance.

Class 2

Figure 16: CNN Regression Model MAE and Loss for Class 2 Training and Validation - Fold 1. The model achieved a validation MAE of 0.1676 and a validation loss of 0.0473.

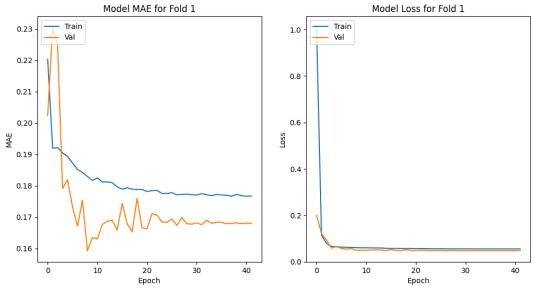


Figure 17: CNN Regression Model MAE and Loss for Class 2 Training and Validation - Fold 2. The model achieved a validation MAE of 0.1328 and a validation loss of 0.0308.

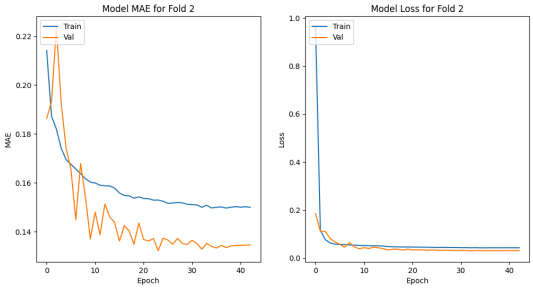


Figure 18: CNN Regression Model MAE and Loss for Class 2 Training and Validation - Fold 3. The model achieved a validation MAE of 0.1544 and a validation loss of 0.0418.

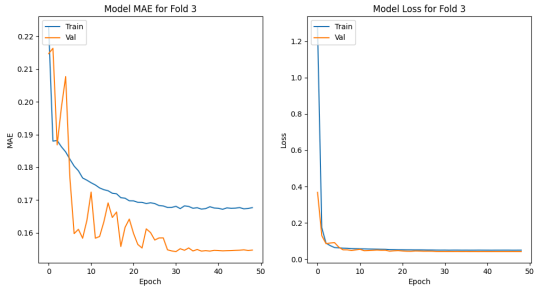


Figure 19: CNN Regression Model MAE and Loss for Class 2 Training and Validation - Fold 4. The model achieved a validation MAE of 0.1597 and a validation loss of 0.0438.

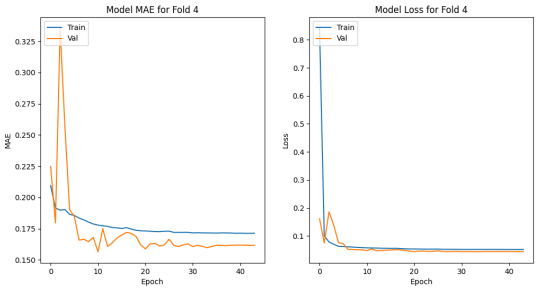
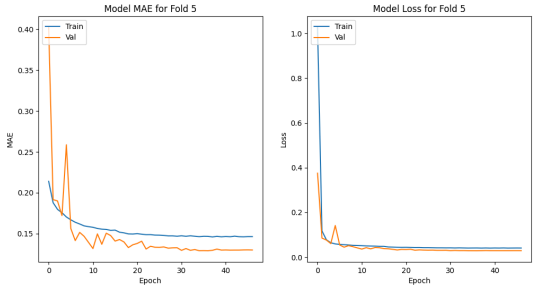


Figure 20: CNN Regression Model MAE and Loss for Class 2 Training and Validation - Fold 5. The model achieved a validation MAE of 0.1291 and a validation loss of 0.0290.



For Class 2, the MAE per fold ranged from 0.1291 to 0.1676, with a mean of 0.1487. This means that, on average, the predicted probabilities deviated from the actual probabilities by approximately 0.1487. The relatively low standard deviation (0.0152) suggests that the model's performance was consistent across the different folds.

The corresponding validation loss (mean squared error) ranged from 0.0290 to 0.0473, with a mean of 0.0385. This metric also indicates a good fit, as lower values denote that the predicted values are close to the actual values.

Class 7:

Figure 21: CNN Regression Model MAE and Loss for Class 7 Training and Validation - Fold 1. The model achieved a validation MAE of 0.1313 and a validation loss of 0.0299.

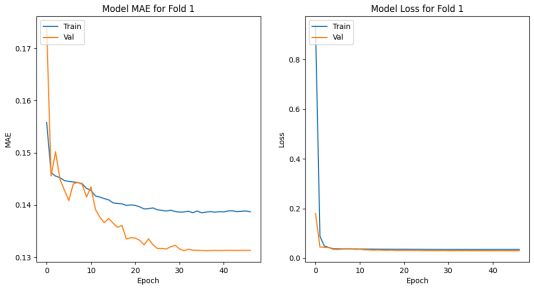


Figure 22: CNN Regression Model MAE and Loss for Class 7 Training and Validation - Fold 2. The model achieved a validation MAE of 0.1490 and a validation loss of 0.0389.

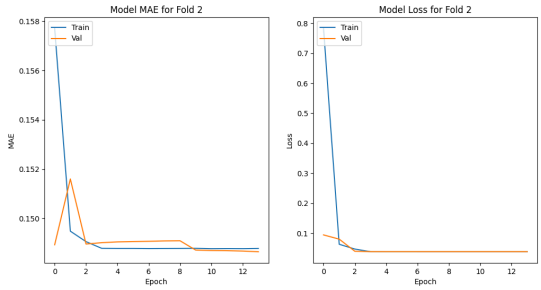


Figure 23: CNN Regression Model MAE and Loss for Class 7 Training and Validation - Fold 3. The model achieved a validation MAE of 0.1230 and a validation loss of 0.0270.

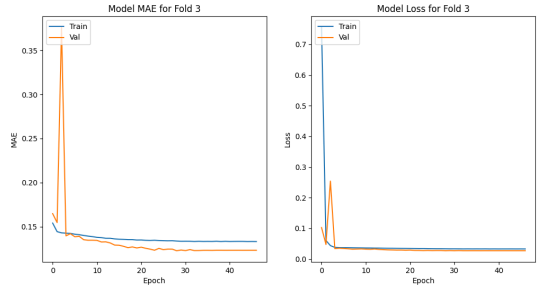


Figure 24: CNN Regression Model MAE and Loss for Class 7 Training and Validation - Fold 4. The model achieved a validation MAE of 0.1485 and a validation loss of 0.0387.

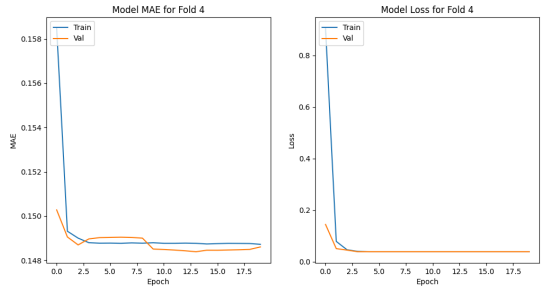
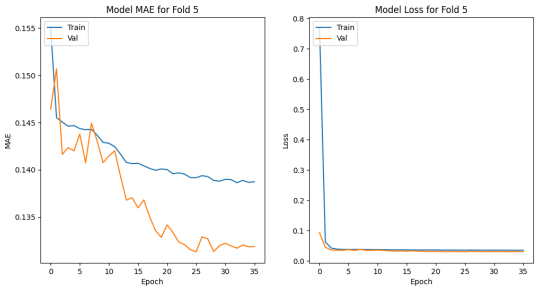


Figure 25: CNN Regression Model MAE and Loss for Class 7 Training and Validation - Fold 5. The model achieved a validation MAE of 0.1313 and a validation loss of 0.0300.



The MAE per fold ranged from 0.1229 to 0.1490, with a mean of 0.1366. This indicates an average deviation of 0.1366 from the actual probabilities, reflecting a slightly better performance compared to the first regression task.

The corresponding validation loss ranged from 0.0269 to 0.0389, with a mean of 0.0329, which further supports the model's accuracy in predicting these probabilities.

The performance metrics (MAE and loss) for both regression tasks are reasonably low, indicating good predictive performance. However, the model seems to perform marginally better on predicting how round the smooth galaxy is (Class 7.1, 7.2, 7.3) compared to predicting if a galaxy could be a disk viewed edge-on (Class 2.1, 2.2).

Bonus:

In the project, the use of regression tasks is to predict the values for [Class2.1, Class2.2] and [Class7.1, Class7.2, Class7.3], which range between 0 and 1. This constraint is critical because these values represent probabilities or normalized scores, making it essential for the model outputs to lie within this range.

Currently, the model's architecture does not explicitly enforce this constraint, as I use a linear activation function in the final layer. This means that the model can potentially produce output values outside the 0 to 1 range. The most straightforward method to achieve this is by using the sigmoid activation function in the output layer. The sigmoid function maps any input to a value between 0 and 1, thus inherently constraining the outputs within the desired range.

Exercise 3 - Regression Continued

In the third exercise, the regression task was expanded by incorporating additional columns into the output, namely [Class6.1, Class6.2] and [Class8.1, Class8.2, Class8.3, Class8.4, Class8.5, Class8.6, Class8.7]. These columns answer the questions "Is there anything 'odd' about the galaxy?" and "What is the odd feature?". As opposed to exercise 2's regression, Class 6 and Class 8 were combined into a single model. This was done due to the nature of the questions being addressed, which are interconnected. By combining these tasks, the model can leverage shared features and dependencies between the

different classes, potentially leading to better performance and more meaningful insights.

In contrast, Exercise 2 handled the classes separately because the questions pertain to distinct and most likely independent characteristics of the galaxies. Treating these tasks separately allows the model to specialize in identifying features relevant to each specific question without interference from the other task.

The CNN model is similar to exercise 2's model with the main difference being a single model is trained to handle all the target columns together, resulting in a more complex output layer with 9 neurons.The model was then compiled using the Adam optimizer and mean squared error (MSE) as the loss function, with mean absolute error (MAE) as an additional metric for evaluation.

Just like in exercise 2, I employed K-Fold Cross Validation with five splits up to 100 epochs. Each fold involved training the model on four subsets and validating it on the fifth subset, cycling through all subsets. During training, I used callbacks such as early stopping, reduce learning rate on plateau, and a learning rate scheduler to enhance the model's efficiency and prevent overfitting.

The training history for each fold was then plotted (Figure 26-30), displaying the MAE and loss curves over the epochs for both the training and validation sets.

The validation mean absolute error (MAE) is a crucial metric in regression tasks, representing the average magnitude of the errors between predicted and actual values. Lower MAE values indicate better model performance.

Figure 26: CNN Regression Model MAE and Loss for Training and Validation - Class 6 and Class 8, Fold 1. The model achieved a validation MAE of 0.0709 and a validation loss of 0.0161.

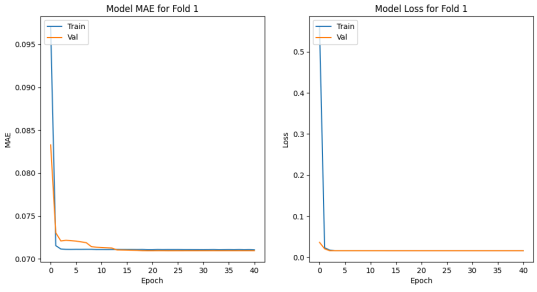


Figure 27: CNN Regression Model MAE and Loss for Training and Validation - Class 6 and Class 8, Fold 2. The model achieved a validation MAE of 0.0708 and a validation loss of 0.0158.

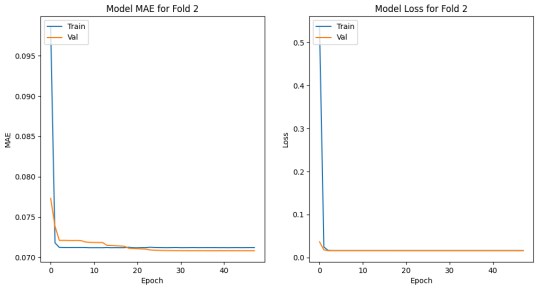


Figure 28: CNN Regression Model MAE and Loss for Training and Validation - Class 6 and Class 8, Fold 3. The model achieved a validation MAE of 0.0725 and a validation loss of 0.0166.

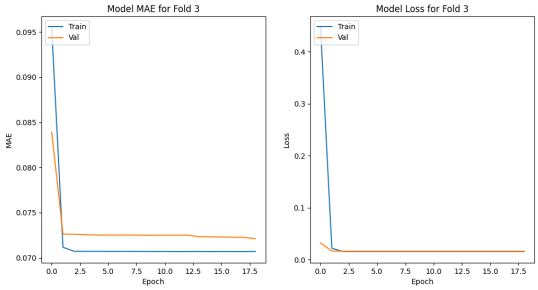


Figure 29: CNN Regression Model MAE and Loss for Training and Validation - Class 6 and Class 8, Fold 4. The model achieved a validation MAE of 0.0706 and a validation loss of 0.0158.

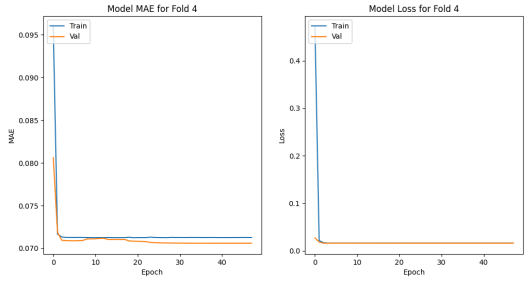
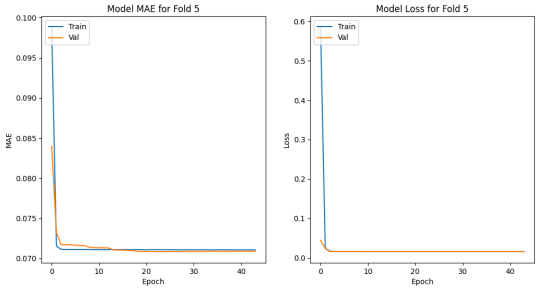


Figure 30: CNN Regression Model MAE and Loss for Training and Validation - Class 6 and Class 8, Fold 5. The model achieved a validation MAE of 0.0709 and a validation loss of 0.0159.



The mean validation MAE across all folds was 0.0711, with a standard deviation of 0.0007. The mean validation loss was 0.0160, with a standard deviation of 0.0003.

The results indicate a consistent performance across all five folds, with minor variations in MAE and loss. The low standard deviations for both MAE (0.0007) and loss (0.0003) suggest that the model's performance is stable and reliable across different subsets of the data. This consistency is crucial for a model intended to classify rare object classes, as it demonstrates the model's strength in handling varied data distributions.

However, the observation that the training and validation metrics are close to zero requires further investigation. This could indicate potential issues such as overfitting, data leakage, or improper validation data handling.

Studies

Model Performance Across Tasks:

The same CNN architecture, with modifications primarily in the output layer, was used for classification and regression tasks involving different sets of galaxy characteristics. The results indicated that while the architecture performed consistently across different tasks, the regression tasks in Exercise 3 exhibited near-zero performance metrics, raising concerns about potential overfitting or data leakage. This suggests that while the overall architecture works pretty well, specific tasks may require more tailored adjustments to ensure more reliable performance.

Impact of Augmentations:

The use of data augmentation significantly improved the classification model's performance. By introducing variations in the training images, the model learned more generalized features, reducing overfitting and improving validation stability. This indicates that augmentations are crucial not only during training but could also enhance performance at test time by making the model more resilient to diverse data conditions.

Sequential Task Informing:

The concept of using the output from one model to inform subsequent predictions was not directly implemented in this project. However, the consistent performance across different tasks suggests potential. For instance, classifications about the smoothness or disk presence could serve as features or prior information for regression tasks predicting galaxy roundness or odd features. Integrating these insights could enhance the overall predictive power and provide a more holistic analysis of galaxy characteristics.

Future Steps

As mentioned above, exercise 3's regression seems to be a little questionable, due to how close to 0 the performance metrics were. It would be wise to check if there is overfitting and if there is potentially some sort of data leakage, just to verify that there is no leakage between the training and validation datasets. This can happen if the same images are used in both training and validation or if there is some systematic error in how the data is split. If this fails, it would be worthwhile to revisit the model's architecture and try to improve it.

Additionally, increasing the number of folds in the cross-validation process could provide a better assessment of the model's performance.

Another potential improvement is to consider using ensemble methods, such as bagging or boosting, to combine multiple models (Analytics Vidhya, 2018) . Ensembles can often achieve better generalization by reducing the variance of the predictions. Bagging (Bootstrap Aggregating) involves training multiple versions of a model on different subsets of the training data and then averaging their predictions. Boosting, on the other hand, involves training models sequentially, each one correcting the errors of its predecessor. These methods can help enhance the robustness and accuracy of the predictions, especially when dealing with complex and diverse data like galaxy images.

Conclusion

The project aimed to develop a machine learning tool capable of classifying and analyzing galaxy images, using convolutional neural networks (CNNs) to tackle these tasks. The main findings from this project demonstrate that CNNs, with tailored architectures, can effectively classify and predict galaxy features, albeit with some areas needing further refinement.

Exercise 1 - Classification:

The CNN model successfully classified galaxies into three categories: smooth and round with no disk, having a disk, or being a star or artifact. Through a structured training and cross-validation process, the model achieved a mean validation accuracy of approximately 0.842. The incorporation of data augmentation and learning rate adjustments improved the model, reducing overfitting and stabilizing validation performance, showing that the model is effective in handling the complexity and diversity of galaxy shapes.

Exercise 2 - Regression:

The regression tasks in Exercise 2 focused on predicting the likelihood of a galaxy being viewed edge-on (Class2.1, Class2.2) and the roundness of smooth galaxies (Class7.1,

Class7.2, Class7.3). The CNN model, adapted for regression, performed well with mean absolute errors (MAE) suggesting reasonable predictive accuracy. For Class 2, the MAE per fold ranged from 0.1291 to 0.1676, while for Class 7, the MAE ranged from 0.1229 to 0.1490. These results indicate that the model was able to capture the continuous nature of these galaxy characteristics effectively.

Exercise 3 - Extended Regression:
In Exercise 3, the regression task was extended to include additional characteristics related to odd features in galaxies (Class6.1, Class6.2, and Class8.1 to Class8.7). The model showed consistent performance across folds, with a mean validation MAE of 0.0711 and mean validation loss of 0.0160. However, the near-zero performance metrics suggest possible overfitting or data leakage. Despite these issues, the results indicate the model's capability to handle multiple regression outputs simultaneously, specifically using shared features and dependencies between the different classes.

References

Galaxy Zoo - The Galaxy Challenge. (2013, December 21). Kaggle.com. https://www.kaggle.com/c/galaxy-zoo-the-galaxy-challenge/overview

Zooniverse. (2020). Zooniverse.org. https://www.zooniverse.org/projects/zookeeper/galaxy-zoo/

IBM. (2023). What are Convolutional Neural Networks? | IBM. Www.ibm.com; IBM. https://www.ibm.com/topics/convolutional-neural-networks

GeeksforGeeks. (2017, August 21). Introduction to Convolution Neural Network. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-convolution-neural-network/

Mishra, M. (2020, August 27). Convolutional Neural Networks, Explained. Medium; Towards Data Science. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

Qayyum, R. (2022, August 16). Introduction To Pooling Layers In CNN – Towards AI. TowardsAI. https://towardsai.net/p/l/introduction-to-pooling-layers-in-cnn

Gomede, E. (2024, January 12). Cosine Annealing: Enhancing Deep Learning Through Dynamic Learning Rate Adjustment. **AI monks**.**io**. https://medium.com/aimonks/cosine-annealing-enhancing-deep-learning-through-dynamic-learning-rate-adjustment-d60957fe7115

Analytics Vidhya. (2018, June 18). A Comprehensive Guide to Ensemble Learning (with Python codes). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/