Image Classification of Stellar Objects

Valerie Chavez

Final Project
STAA 578
Spring 2024

1 Introduction

1.1 Background

With the expansion of data collection and storage options, the fields of observational astronomy and astrophysics are increasingly inundated with large amounts of data. Projects like the Hubble, James Web, and Sloan Digital Sky Survey can collect millions of images within a relatively short frame of time. Conventional techniques for image classification could greatly limit scientific potential. Machine Learning allows scientists to work with large volumes of imaging data more efficiently. A classic problem in astrophysics is the categorization of stars vs galaxies. Stars can have the appearance of "point spread" when observed through the earth's atmosphere, while images of galaxies taken from a great distance can resemble stars. With the vast number of observations "the task would take simply too long for large data sets, and therefore there is a very good case for the use of ML (Kembhavi et al 2020)." This analysis will compare the use of a random forest (as a "shallow" baseline) and a convolution neural network in their performance of stellar classification.

1.2 Dataset and preparation

To explore this problem, data from the Aryabhatta Research Institute of Observational Sciences (ARIES), Nainital, India (Agrawal 2021) is explored. The images are given in 64x64 cutouts and represent real-word observations of stars and galaxies. There are a total of 3044 stars and 942 galaxies within the dataset (N = 3986), with an approximately 76%/24% split between stars and galaxies. The images were labeled, reformatted, and normalized then spit into a training (80%) and validation (20%) dataset, with both having the same proportion of stars and galaxies as the full dataset.

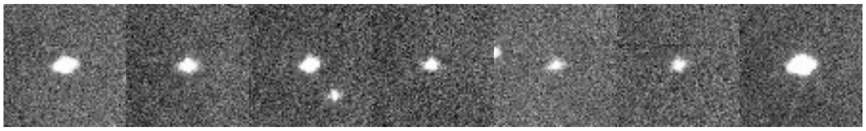


Fig 1.1 Example images of stars

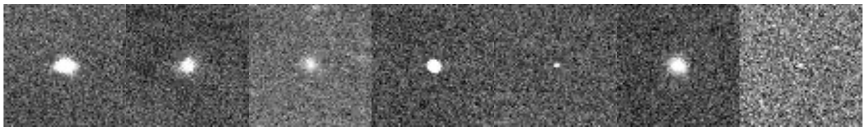


Fig 1.2 Example images of galaxies

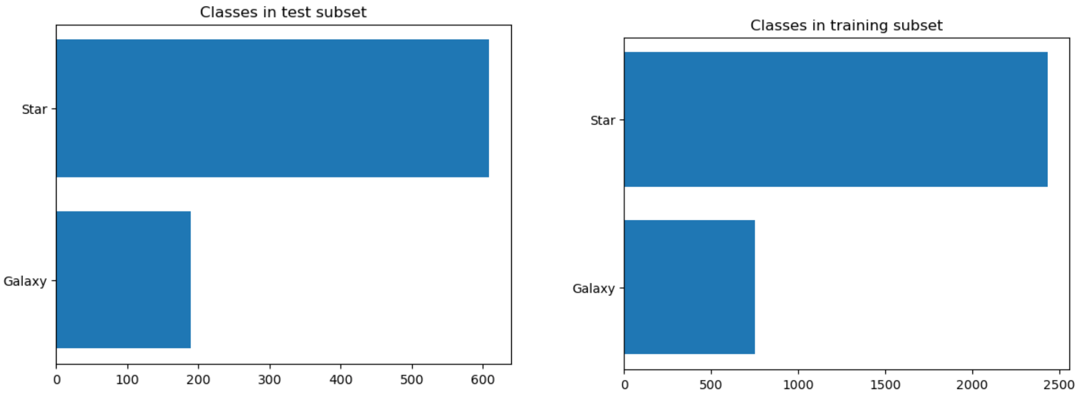


Fig 1.3 Distribution of training and test dataset

2 Method

2.1 Model selection

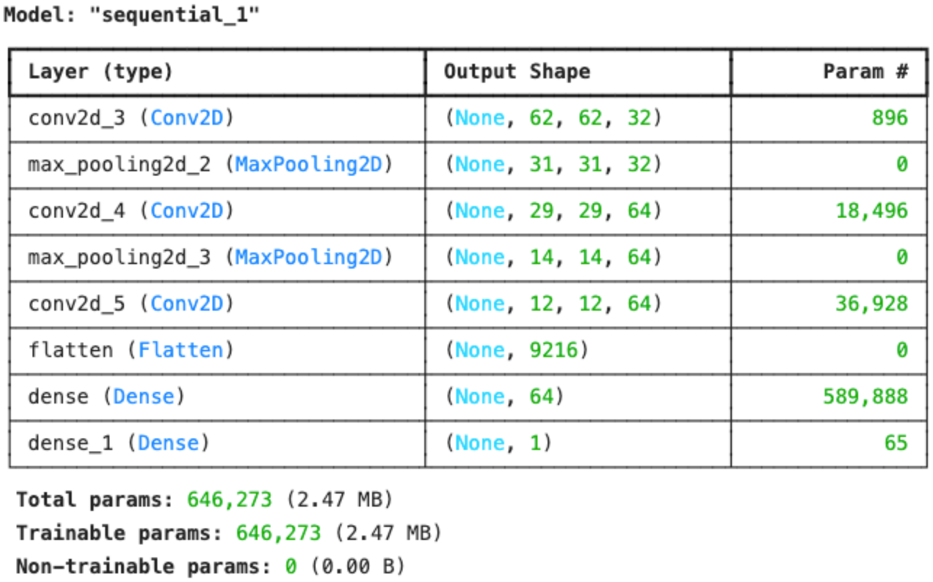
A recent overview of applications of Machine Learning in Astronomy notes the use of methods such as logistic regression, k-nearest neighbors, and Random Forests as satisfactory alternatives to the potentially more time-consuming and computationally expensive deep learning methods (Kembavi et al 2022). There is specific discussion of Random Forests and Convolution Neural Network (CNN) for classification of images, with other methods being more relevant to quantitative observations. As such, both were fit to explore their relative merits, with the Random Forest serving as a "shallow" baseline. Since CNNs are “a type of deep-learning model almost universally used in computer vision applications (Chollet 2017)” with the ability to learn spatial features, this is the natural choice for classification of images.

2.2 Fitting Random Forest

Using scikit-learn in python a random forest was fit with the image data as the dependent variables, and the label (star vs galaxy) as the independent variable. Since this served as a simple baseline and the random search method of cross validation is time-consuming, fine-tuning was not the focus and could be explored further. Using cross validation to compare a few options for number of trees (50, 100, 500) and depth (3,5,10,15), the best performance was with 50 trees and a depth of 15. The performance was comparable to that of a model with the default parameters in the Random Forest Classifier function, so either model could serve as a baseline comparison to the CNN.

2.3 Fitting CNN

The model was fit very similarly to what is described in chapter 5 of *Deep learning with Python* (Chollet 2017) with an input shape of (64,64,3) as this reflects the image size. In both the base model 10 epochs were chosen as the validation accuracy seemed to flatten at around 5 epochs. Since the images fall into two categories, binary cross entropy was chosen as the loss function. Additionally, for the same reason sigmoid was used as the activation function on the last layer. These were compared with a categorical cross entropy and softmax as the loss/activation functions and performed significantly better.



```
Model: "sequential_1"
┌────────────────────────────────┬──────────────────────────┬───────────────┐
│ Layer (type)                   │ Output Shape             │       Param # │
├────────────────────────────────┼──────────────────────────┼───────────────┤
│ conv2d_3 (Conv2D)              │ (None, 62, 62, 32)       │           896 │
├────────────────────────────────┼──────────────────────────┼───────────────┤
│ max_pooling2d_2 (MaxPooling2D) │ (None, 31, 31, 32)       │             0 │
├────────────────────────────────┼──────────────────────────┼───────────────┤
│ conv2d_4 (Conv2D)              │ (None, 29, 29, 64)       │        18,496 │
├────────────────────────────────┼──────────────────────────┼───────────────┤
│ max_pooling2d_3 (MaxPooling2D) │ (None, 14, 14, 64)       │             0 │
├────────────────────────────────┼──────────────────────────┼───────────────┤
│ conv2d_5 (Conv2D)              │ (None, 12, 12, 64)       │        36,928 │
├────────────────────────────────┼──────────────────────────┼───────────────┤
│ flatten (Flatten)              │ (None, 9216)             │             0 │
├────────────────────────────────┼──────────────────────────┼───────────────┤
│ dense (Dense)                  │ (None, 64)               │       589,888 │
├────────────────────────────────┼──────────────────────────┼───────────────┤
│ dense_1 (Dense)                │ (None, 1)                │            65 │
└────────────────────────────────┴──────────────────────────┴───────────────┘
 Total params: 646,273 (2.47 MB)
 Trainable params: 646,273 (2.47 MB)
 Non-trainable params: 0 (0.00 B)
```

Data Augmentation Layer

After the initial model was fit, a data augmentation layer was added to improve performance since we have a relatively small number of observations. This allows us to generate more training data. The data were augmented with a random flip and a shift of 0.1 in height and width. In this case more epochs (30) proved to have better results with the increase in training information

3 Results

3.1 Performance of models

Model	Random Forest	CNN	CCN with data augmentation
Overall Accuracy of model	78%	86%	89%
Accuracy of Galaxy Classification	13%	67%	77%
Accuracy of Star Classification	99%	93%	93%

Table 3.1: Note values are rounded to the nearest percentage

Random forest

With this baseline, the overall model accuracy was about 78%. Recall that with stars representing ~76% of the data, this performance is rather poor, and in fact the algorithm over-classified stars with a relatively high false-positive rate. If we look at the accuracy by subgroup only 13% of galaxies are classified correctly, while 99% of stars are classified correctly. The model is achieving relatively high performance by classifying almost all images (96%) as stars! This would likely not be a useful model for rigorous scientific purposes.

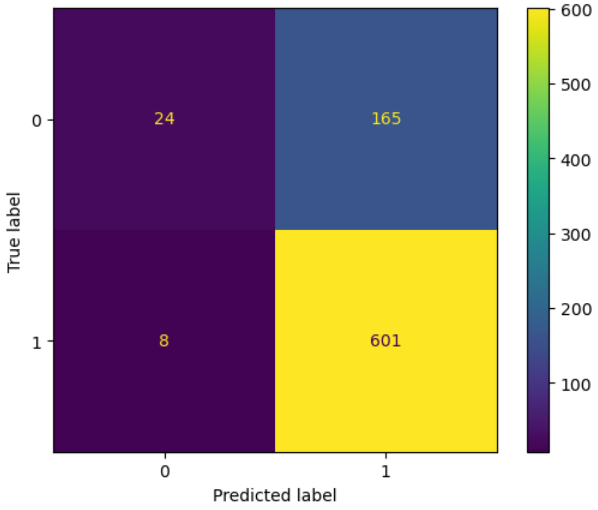


Fig 3.1: Confusion matrix for RF classification (1 = Star, 0 = Galaxy)

CNN

Base model

Fitting the CNN model improved the overall accuracy to about 87% with an improvement in the classification of galaxies to approximately 67%. You can see however, there is a slight trade-off in the accuracy of the stars, but this is overall a much more reliable model.

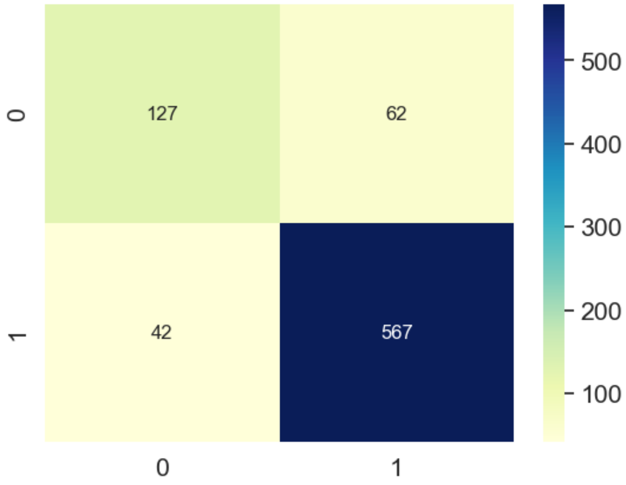


Fig 3.2: Confusion matrix for RF classification (1 = Star, 0 = Galaxy)(row = True label, column = Predicted label)

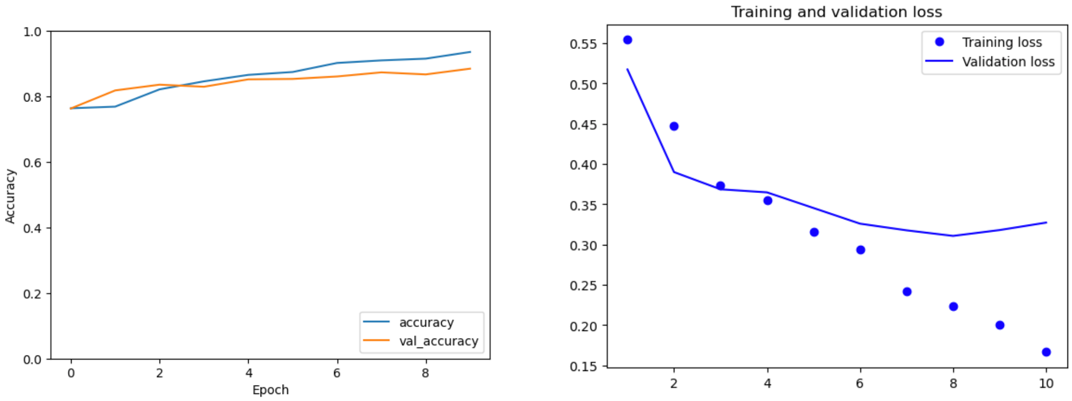


Fig 3.3: CNN accuracy and loss

Model with data augmentation

Adding a data augmentation step showed a modest improvement in the overall accuracy of the model to about 89%. However, there was a meaningful improvement in the classification of galaxies from 67% in the original CNN model to 77% with the performance of star classification remaining relatively consistent. With the data augmentation layer, 30 epochs were used to fit the model.

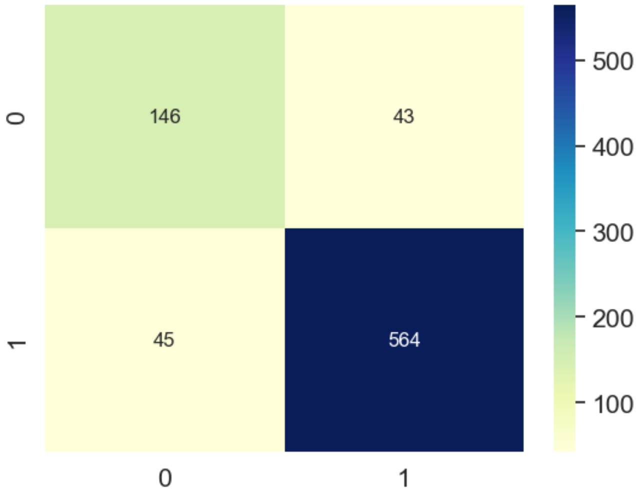


Fig 3.4: Confusion matrix for RF classification with data augmentation(1 = Star, 0 = Galaxy)(row = True label, column = Predicted label

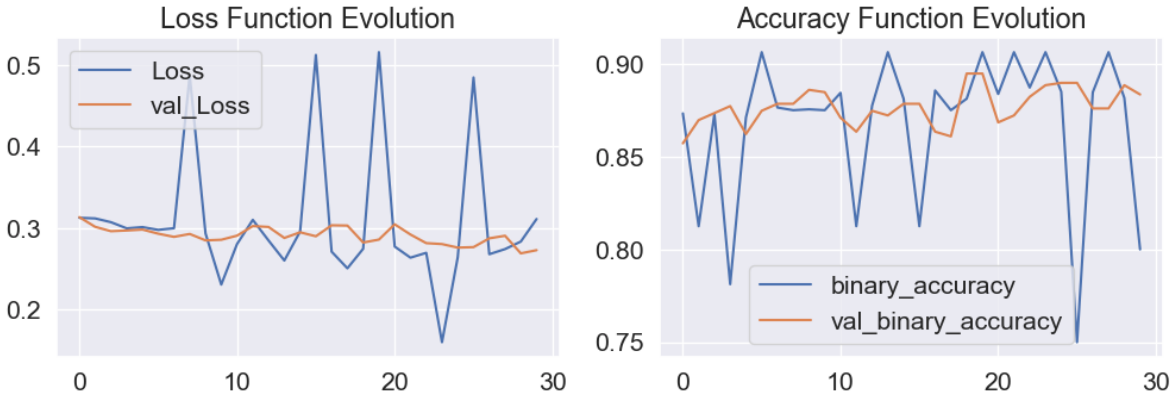


Fig 3.4: CNN accuracy and loss

3.2 Conclusion

In this exploration, the use of CNN has vastly superior performance to the simpler random forest. In this particular dataset a major constraint was the relatively small number of galaxies compared to stars, which resulted in a high rate of misclassification in both models, though it was greatly improved by the CNN model. While it was not explored here, adjustments to the makeup of the training set (with a more even representation of galaxies and stars) could result in better classification of galaxies. The results so that a random forest could be problematic for most scientific purposes where high error of a particular subcategory could have major consequences. With the use of deep learning, the CNN approach could be applied beyond the simple classification of stars vs galaxies to subtypes within stars and galaxies in addition to other astronomical phenomena.

References

Data Source:
Divyansh Agrawal. (2021). Star-Galaxy Classification Data [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DS/1396185

Articles on Machine Learning in Astronomy:
Kembhavi, A., & Pattnaik, R. (2022). Machine learning in astronomy. Journal of Astrophysics and Astronomy, 43(2), 76.

Rajesvari, M., Sinha, A., Saxena, V., & Mukerji, S. A. (2020). Deep learning approach to classify the galaxies for astronomy applications. OSR-JEEE., 15, 35-9.

Information on Random Forest and CNN:
Random Forest Classification with Scikit-Learn
https://www.datacamp.com/tutorial/random-forests-classifier-python

Image classification using Sklearn (RandomForest)
https://www.kaggle.com/code/kkhandekar/image-classification-using-sklearn-randomforest

François Chollet. Deep Learning with Python Video Edition. Manning Publications, 2017.

*For code and details see Jupyter notebook (note that due to the random nature of splitting the test/training data there could be some small discrepancies between the values reported here and those found when running the code)