PHYS378 Final Project:
Predicting Galaxy Flux with Machine Learning

Jinwoo Kim and Jin Wuk Lee

November 2024

1 Abstract

The flux of a galaxy can be described as a measure of the total amount of light it emits as seen from Earth and is a key property linked to brightness. Traditional methods to predict a galaxy's flux, such as light-profile fitting, are computationally intensive and ill-suited for the massive data volumes expected in future astronomical surveys. To overcome these limitations, we leverage single-band 2D images of galaxies, which are simulated to match those imaged using the Hyper Suprime-Cam Survey, alongside their flux values to train and evaluate a convolutional neural network (CNN), offering a scalable alternative to this regression problem.

The methodology includes data preprocessing to enhance input feature model selection through comparative performance analysis and rigorous training, validation, and testing procedures. Model performance is evaluated based on prediction accuracy and sensitivity to key galaxy parameters, such as bulge-to-disk light ratios and galaxy radii. The results of the CNN demonstrate the superiority of machine learning-based approaches, achieving high prediction accuracy while drastically reducing computational costs. This approach offers a scalable, adaptable framework, paving the way for rapid and reliable flux estimation in large-scale astronomical surveys.

2 Introduction

The flux of a galaxy is an important intrinsic physical property, fundamentally linked to its brightness. Traditionally, astronomers have utilized light-profile fitting techniques, such as fitting two-dimensional models to galaxy images (e.g., Geda et al. [1]), to estimate galaxy flux. However, the exponential growth of skysurvey imaging data presents a significant challenge for scaling such techniques to the volumes expected in astronomy over the next decade. This necessitates the development of machine learning (ML)-based frameworks capable of determining galaxy flux efficiently.

Supervised machine learning (ML) enables models to learn from a set of input data paired with corresponding truth values. Instead of explicitly instructing the model on task execution, the algorithm identifies the relationship between inputs and outputs, allowing it to predict unseen data. Among supervised ML methods, deep neural networks—algorithms inspired by the synaptic connections of the brain—are currently among the most widely used techniques.

A specialized class of deep neural networks, convolutional neural networks (CNNs), has demonstrated remarkable efficacy in solving both classification and regression problems. Successful applications of CNNs in classification tasks include the work of Bialopetravicius et al. [2], who constructed a CNN based on the ResNet architecture to derive ages, masses, and sizes of star clusters directly from multi-band images. Similarly, Monsalves et al. [3] employed CNNs to classify variable astronomical objects into eight distinct categories. CNNs have also proven effective in regression tasks. Surana et al. [4] used CNNs to predict stellar mass, star formation rate, and dust luminosity—key properties for studying star formation. Pearson et al. [5] applied CNNs to estimate mass model parameters of strong gravitational lenses.

These diverse applications underscore the versatility and efficiency of CNNs in addressing astronomical problems. In this work, we train a CNN to predict galaxy flux from single-band two-dimensional images, an essential property linked to a galaxy's brightness. Our dataset consists of 40,000 simulated galaxy images, designed to match those obtained by the Hyper Suprime-Cam Survey. The training and testing sets include each galaxy's total flux (in ADUs), radii (in arcseconds), bulge-to-total light ratio, and other relevant properties.

3 Data and Preliminary Testing

3.1 Data

Figure 1 shows the breakdown of the training dataset, which contains the identical columns as the testing dataset. The dataset is structured with the following 17 variables: 'img_number', 'num_components', 'sersic_idx_d', 'R_e_d', 'axis_ratio_d', 'PA_d', 'flux_frac_d', 'sersic_idx_b', 'R_e_b', 'axis_ratio_b', 'PA_b', 'flux_frac_b', 'total_flux', 'log_total_flux', 'file_name', 'R_e', 'Bt'.

	img_number	num_components	sersic_idx_d	R_e_d	axis_ratio_d	PA_d	flux_frac_d	sersic_idx_b	R_e_b	axis_ratio_b	PA_b	flux_frac_b	total_flux	log_total_flux	file_name	R_e	bt
0	27248	2	0.827	0.106	0.628	-42.580	0.024	4.113	1.427	0.857	-38.070	0.976	23092.184	4.363465	27248.fits	1.372930	0.976
1	67161	2	0.840	2.082	0.265	-0.769	0.564	4.631	1.533	0.918	10.391	0.436	7516.552	3.876019	67161.fits	2.169138	0.436
2	27247	2	1.148	0.552	0.368	36.512	0.755	4.531	1.399	0.973	43.542	0.245	9944.074	3.997564	27247.fits	0.690142	0.245
3	7257	2	0.952	1.315	0.374	40.700	0.107	4.951	0.151	0.983	35.853	0.893	28716.708	4.458135	7257.fits	0.193666	0.893
4	85233	2	0.984	0.283	0.851	80.519	0.158	4.586	0.232	0.667	88.126	0.842	18224.146	4.260647	85233.fits	0.245004	0.842
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29995	36465	2	1.147	1.657	0.959	-89.785	0.882	4.159	0.112	0.560	-81.184	0.118	17776.562	4.249848	36465.fits	1.452797	0.118
29996	43881	2	1.131	1.542	0.503	45.036	0.078	4.705	1.034	0.603	39.433	0.922	13802.180	4.139948	43881.fits	1.129090	0.922
29997	72287	2	1.088	2.830	0.817	76.970	0.326	4.356	1.908	0.321	82.387	0.674	26411.818	4.421798	72287.fits	2.493842	0.674
29998	27010	2	1.116	0.345	0.448	-37.761	0.002	4.370	0.116	0.504	-25.391	0.998	19798.688	4.296636	27010.fits	0.128702	0.998
29999	2378	2	1.135	2.402	0.571	-57.108	0.994	4.034	1.721	0.979	-46.737	0.006	17584.672	4.245134	2378.fits	2.484759	0.006

30000 rows x 17 columns

Figure 1: Training dataset

Figure 4 presents an example of a 2D image, one of 40,000 grayscale images in the images folder. Each image has varying levels of light that, as seen later in the CNN, act as strong predictors of the total flux value.

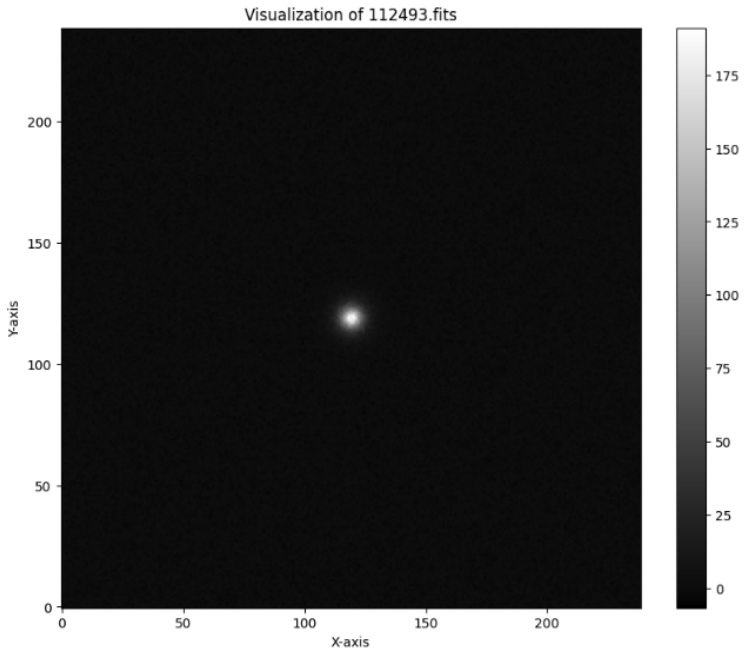


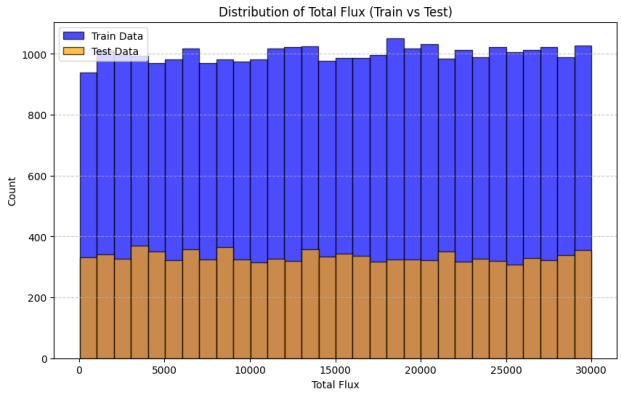
Figure 2: Visualization of an astronomical image

3.2 Data Preprocessing

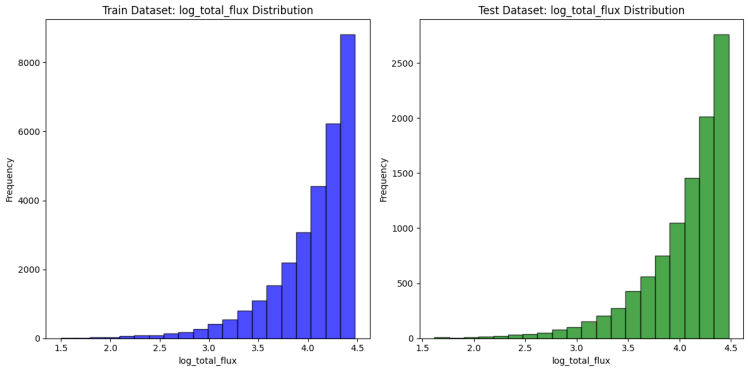
Our dataset consists of 40,000 distinct simulated galaxy images and training/testing datasets containing the ground truth parameters for each galaxy. The images, which are grayscale and two-dimensional, were designed to replicate those obtained by the Hyper Suprime-Cam Survey. The training and testing datasets ("gal_morph_train_val.csv" and "gal_morph_test.csv") contain information such as the total flux, radius, and bulge-to-total light ratio for the galaxy. In our case, we are interested in calculating the total flux.

For the sake of training time, all images were cropped to 143x143 pixels. Since the flux values are rather large and diverse in magnitude, with a uniform distribution of 40,000 samples ranging from a magnitude of 0 to 30,000, we applied logarithmic normalization on the flux values to reduce the range from [0, 30,000] to [0, 5]. Finally, we created a custom data loader that joined the training and testing datasets to the galaxy images, easing future access to the images and corresponding flux values.

Examining the total flux before log normalization in the training and testing datasets reveals that the flux is evenly distributed between 0 and 30,000 ADUs. The training dataset consists of 30,000 data points while the test dataset contains 10,000, resulting in a combined total of 40,000 data points. The log distribution of the datasets reveals a left-skewed distribution in the flux values after normalization.



(a) Before log normalization



(b) After log normalization

Figure 3: Comparison of flux distributions before and after log normalization

3.3 Data Visualizations

We initially started by exploring relationships among the three key variables in our dataset—'total_flux', 'Bt', and 'R_e'. We generated scatter plots comparing Bt vs. Total Flux, R_e vs. Total Flux, and Bt vs. R_e for both the training and test sets. As seen in Figure 3, the visualizations suggest that there is no clear correlation between these variables.

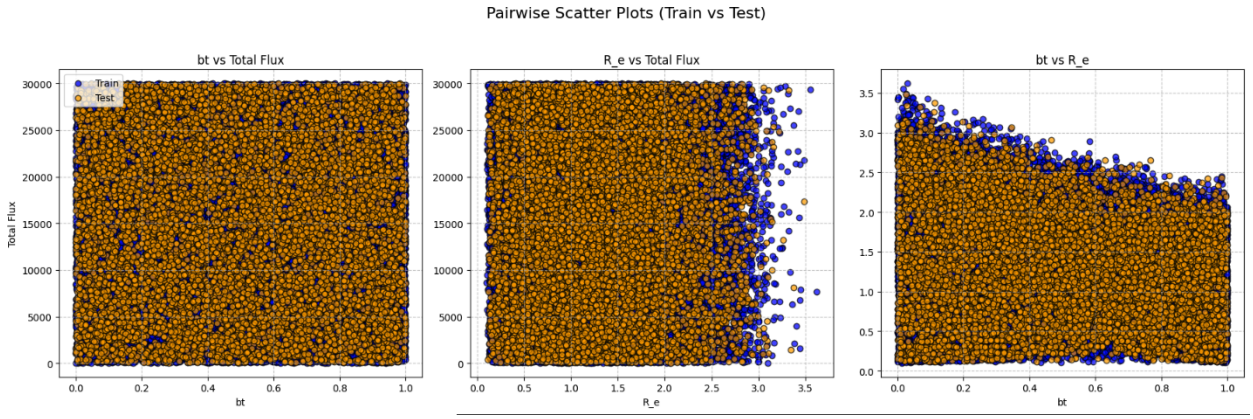
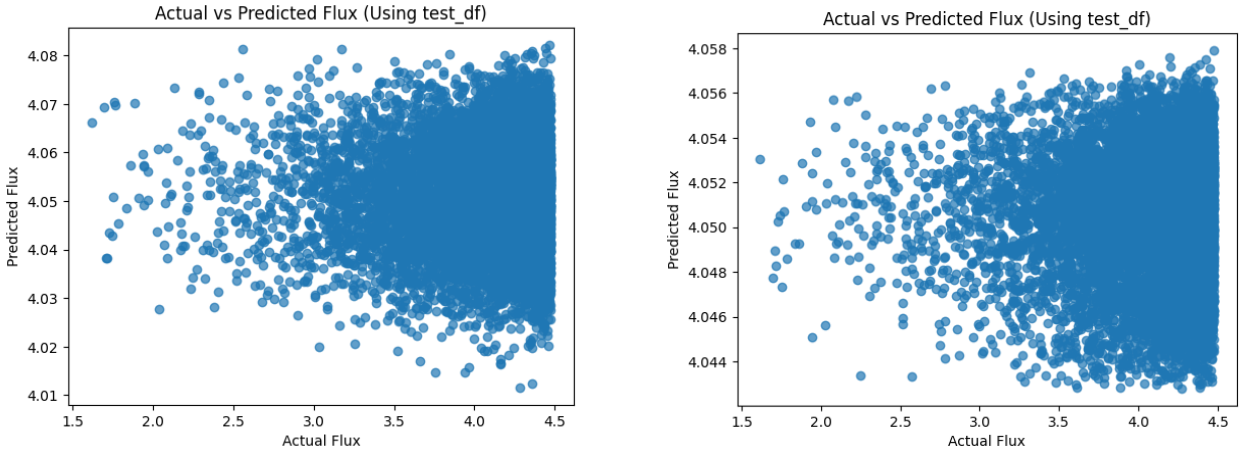


Figure 4: Pairwise scatterplots of Total_flux, Bt, and R_e

3.4 Linear Regression Model

Our first model was a linear regression model, of which we created two versions: The first included all parameters except log_total_flux, total_flux, and file_name, since the file's name does not correspond to the total flux, and the other two variables are directly correlated to the total flux. The second included only the R_e and Bt variables. Both linear regression models performed similarly and poorly, with mean squared error (MSE) values of 0.17889 and 0.17867 and negative R^2 values of -0.001504 and -0.000351 for the first and second models, respectively.



(a) Linear Regression One

(b) Linear Regression Two

Figure 5: Two Linear Regression Models

3.5 Logistic Model

The second model was a logistic regression model. We categorized log_total_flux into three buckets (low, medium, and high) and trained the model with all variables except for log_total_flux, total_flux, file_name, and file_category. The classification report showed a low accuracy of 34%. Additionally, a confusion matrix revealed that the actual and predicted flux values were scattered.

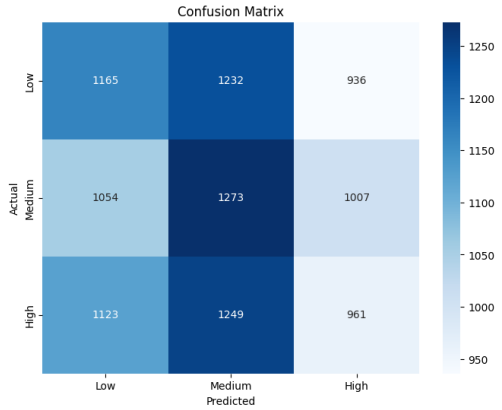


Figure 6: Confusion Matrix from Logistic Regression

3.6 Random Forest Model

Our third model was a Random Forest classification model. We hoped this model would handle non-linear relationships better, but it also performed poorly with a similar accuracy of 34%. The confusion matrix predicted that a majority of flux values were in the high range, where the actual distribution was balanced among "high," "medium," and "low."

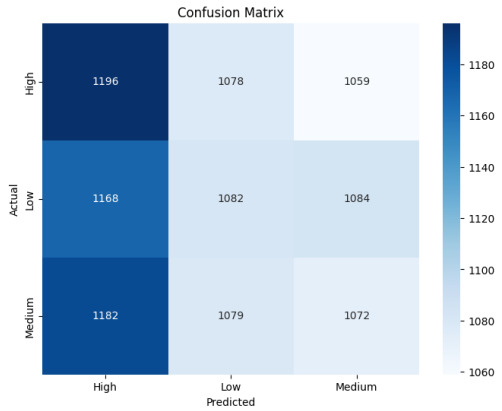


Figure 7: Confusion Matrix from Random Forest

After creating the linear, logistic, and random forest models, we shifted our approach to training a convolutional neural network (CNN), which allowed us to use our image data to predict the total flux of a galaxy. While the three models above were not excellent at prediction, they provided a solid baseline for the MSE, allowing us to compare our CNN's predictions and analyze the results.

4 Methodology

4.1 Model Architecture

Our CNN is trained to predict the log total flux of galaxies by minimizing the mean squared error (MSE) during training. The network consists of the following components:

- **Convolutional layers**. The image tensor of shape (143, 143) is first processed through a series of convolutional layers. Each convolutional layer applies a set of filters to the input image, enabling the network to extract spatial features like edges, textures, and patterns that are relevant to predicting the flux. The first convolutional layer takes an input image with 1 channel (grayscale) and applies 12 filters, each of size 3x3, with a stride of 1 and padding of 1. The second convolutional layer takes the 12-channel output of the first layer and applies 24 filters, each with a size of 3x3, with a stride of 1 and padding of 1. Each convolution layer is followed by a ReLU activation function and 2x2 max pooling. After these layers, the shape of the image becomes (24, 35, 35).
- **Linear layers**. The output of the convolutional layers is flattened to a 1D vector and passed through a fully connected layer with 64 neurons. A ReLU activation function is again applied, and a dropout layer with a rate of 0.2 is added to prevent overfitting. The final dense layer outputs a single continuous value representing the predicted log total flux.
- **Learning rate and epochs**. We are able to adjust the learning rate of the Adam optimizer and the number of epochs. We created three models with the same architecture but with different learning rates and number of epochs. The first model has a learning rate of 0.01 and 7 epochs. The second model has a learning rate of 0.001 and 10 epochs. The third model has a learning rate of 0.001 and 15 epochs.

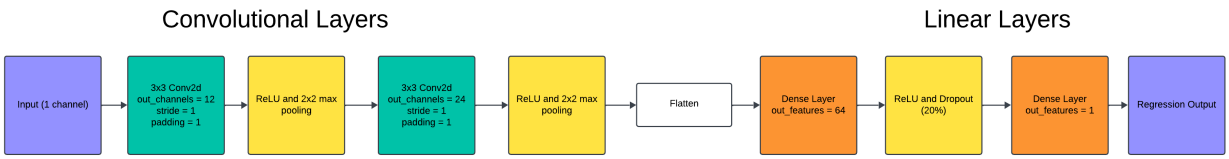


Figure 8: Flowchart of the Convolutional Neural Network (CNN) Architecture.

4.2 Model Training

The training procedure involves the following steps:

- **Forward Pass**: The input images are passed through the convolutional and fully connected layers to generate predictions.
- **Loss Computation**: The MSE loss is computed by comparing the predicted log flux values to the ground truth.
- **Backward Pass**: Gradients are calculated via backpropagation, and the model weights are updated using the Adam optimizer.

4.3 Evaluation

The model's performance is evaluated on the testing set after each epoch. During evaluation, the model is set to inference mode to disable dropout layers, and the test loss is calculated without gradient computation for efficiency. The average training and test losses are logged at the end of each epoch.

5 Results

5.1 Basic models

The linear regression models yielded Mean Squared Error (MSE) values of 0.17889 and 0.17867. The first model demonstrated slightly higher accuracy, likely due to its inclusion of more variables compared to the second model. However, both MSE values were low, indicating that the chosen variables were not linearly correlated and did not effectively represent the total flux.

The logistic regression model achieved a classification accuracy of 34%, suggesting that the relationship between the predictor variables and the outcome was too weak to support reliable predictions. The confusion matrix further highlighted this poor performance, with most predictions falling into the "medium" category, despite the actual distribution being balanced among "high," "medium," and "low."

Similarly, the random forest regression model also underperformed, achieving an accuracy of 34%. The confusion matrix indicated poor predictions, with most values classified in the "high" range, despite the actual distribution being more evenly split among the categories. Among the regression models, we can set the linear regression model MSE of 0.17889 as the baseline performance that we aim to surpass with the CNN results.

5.2 CNNs

The CNN models demonstrated a noticeable improvement in accuracy. All models observed a consistent decrease in training loss with each epoch, indicating no evidence of overfitting. The following table contains the training and testing losses for all three models.

Learning rate	Epochs	Train loss	Test loss
0.01	7	0.1844	0.1781
0.001	10	0.0815	0.0489
0.001	15	0.0561	0.0388

After training over 10 epochs, we observed a consistent decrease in training loss with each epoch, indicating no evidence of overfitting. At the final epoch, the model achieved a training loss of 0.0561 and a test loss of 0.0388. Overall, the training loss for the third model was low and closely aligned with the test loss, which is a strong positive indicator of the model's performance.

An observation to note is that the training loss was greater than the test loss for all three models. This can happen when the model is trained on much more training data, which makes sense in this context as the model is trained on 30,000 data points and tests on 10,000 points.

For each model, we analyzed the absolute differences between the model's prediction and the truth value. We created a table and a histogram for each model, showing numerical values and a visual representation of the model's accuracy.

5.2.1 First CNN

The first CNN had a much higher train and test loss than the other two models, due to its low number of epochs and the high learning rate. This model alerted us that we should increase the number of epochs and reduce the learning rate. Figure 9 displays ten randomly selected images along with their predicted value, actual value, and the differences from the CNN model. The differences range from as low as 0.193549 to as high as 1.560576. The distribution in figure 10 reveals that although the CNN is a significant improvement than the linear regression model, there is still much room for improvement.

File Name	Predicted	Actual	Difference
32643.fits	4.026822	3.712337	0.314485
50987.fits	4.026822	3.833273	0.193549
16675.fits	4.026822	4.444304	0.417482
40761.fits	4.026822	4.070180	0.043358
84531.fits	4.175019	4.436304	0.261284
33715.fits	4.026822	4.440171	0.413349
63863.fits	4.026822	2.466246	1.560576
68747.fits	4.026822	4.356729	0.329906
43169.fits	4.026822	4.331399	0.304577
106470.fits	4.026822	4.299706	0.272884

Figure 9: CNN Results: Predicted, Actual, and Differences (Table)

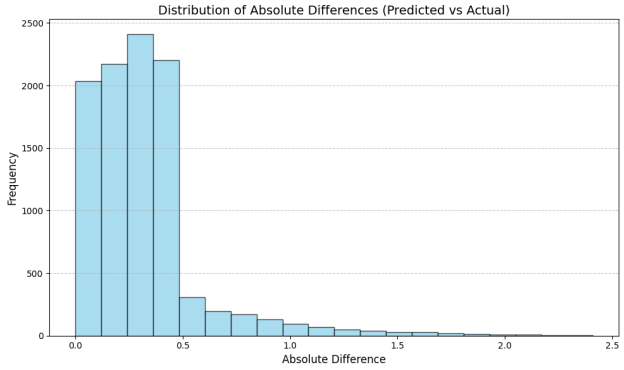


Figure 10: CNN Results: Predicted, Actual, and Differences (Histogram)

5.2.2 Second CNN

The second CNN had a lower train and test loss than the previous model, thanks to increasing the number of epochs and lowering the learning rate. Figure 11 displays ten randomly selected images along with their predicted value, actual value, and the differences from the CNN model. The differences range from around 0.1 to 0.3, without too much deviation. The distribution in figure 12 confirms this result, showing little deviation for most predictions and consistently predicting many within an absolute difference of 0.25.

File Name	Predicted	Actual	Difference
32643.fits	3.647793	3.712337	0.064545
50987.fits	3.804711	3.833273	0.028562
16675.fits	4.324530	4.444304	0.119774
40761.fits	3.995665	4.070180	0.074515
84531.fits	4.411434	4.436304	0.024870
33715.fits	4.370896	4.440171	0.069275
63863.fits	3.616690	2.466246	1.150444
68747.fits	4.277138	4.356729	0.079591
43169.fits	4.257374	4.331399	0.074025
106470.fits	4.216940	4.299706	0.082767

Figure 11: CNN Results: Predicted, Actual, and Differences (Table)

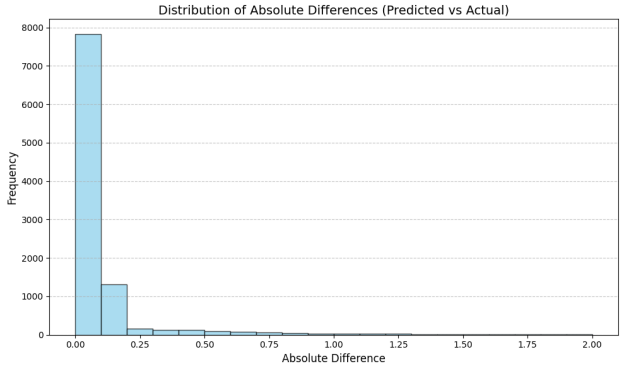


Figure 12: CNN Results: Predicted, Actual, and Differences (Histogram)

5.2.3 Third CNN

The third CNN had a lower train and test loss than the other two models, due to its higher number of epochs, although it maintained the same learning rate. However, Figure 13 and 14 reveal that the differences range from 0.05 to as high as 0.95 among the ten random samples. Although this might be an outlier, the histogram shows that the predictions are not as consistent as our model with 10 epochs. With a higher number of epochs, the model might have overfitted to the training data, which still result in very low training and test losses but poor generalization for unseen samples.

File Name	Predicted	Actual	Difference
32643.fits	3.669793	3.712337	0.042544
50987.fits	3.779774	3.833273	0.053499
16675.fits	4.302043	4.444304	0.142262
40761.fits	3.925186	4.070180	0.144995
84531.fits	4.334299	4.436304	0.102005
33715.fits	4.302124	4.440171	0.138047
63863.fits	3.419301	2.466246	0.953055
68747.fits	4.185405	4.356729	0.171323
43169.fits	4.191720	4.331399	0.139678
106470.fits	4.161193	4.299706	0.138513

Figure 13: CNN Results: Predicted, Actual, and Differences (Table)

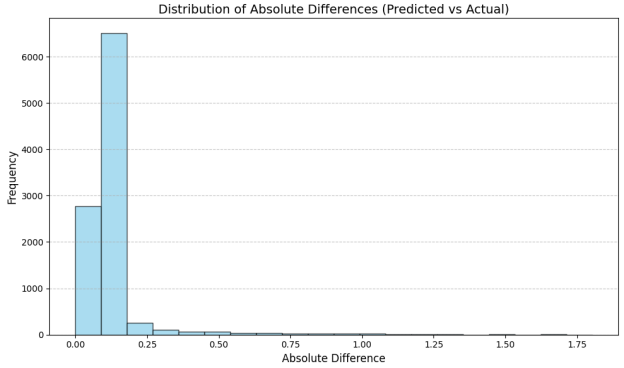


Figure 14: CNN Results: Predicted, Actual, and Differences (Histogram)

6 Discussion

The analysis conducted across multiple models highlights key insights into their predictive capabilities while also exposing several limitations and areas of concern. For the linear regression model,

we found that additional variables contributed to marginal improvements but did not significantly enhance predictive accuracy. The low MSE values reveal that the chosen variables were not linearly correlated and failed to adequately represent the total flux. The logistic regression model's classification of 34% is considerably low, suggesting that the predictor variables do not strongly correlate with the outcome categories. The random forest regression mirrored that of logistic regression, with an accuracy of 34%. The uniformity in prediction for both the logistic and random forest models in their confusion matrices underscores their limitations in adapting to the data.

The CNN models demonstrated substantial improvement, with all three models achieving losses under 0.1. The second and third models achieved significantly lower training and testing losses than the first model, highlighting the importance of sufficiently many epochs and low learning rates. The third model achieved the lowest losses (train loss of 0.0561 and test loss of 0.0388), but showed signs of overfitting due to the predictions becoming less consistent. The second model had slightly more loss, but predicted a larger portion of samples within an absolute difference of 0.3. Therefore, we believe that the second model with a learning rate of 0.001 and 10 epochs performed the best out of all three models.

For future work, we could consider implementing a model that incorporates variables such as R_e, Bt, and other potential ones on top of our current image-processing CNN model. In this project, we concentrated on key variables from the original data file, including R_e, Bt, and total flux, but we did not find any meaningful relationships between them using linear, logistic, and random forest models. It may be worthwhile to perform Principal Component Analysis (PCA) and factor analysis to reduce the dimensionality of our variables and explore any subtle correlations that could potentially enhance the machine learning model.

7 Conclusion

The analysis across different models highlighted their strengths and limitations in predicting galaxy flux. The linear regression model indicated some improvements with low MSE values, but the chosen variables did not capture the complexity of the total flux. The logistic and random forest models performed poorly, with low classification accuracy and confusion matrices showing significant limitations in adapting to the data. In contrast, the CNN model performed exceptionally well, achieving low MSE values and better generalization to unseen data. We demonstrated that using a learning rate of 0.001 and training for 10 epochs achieved promising results, highlighting the effectiveness of supervised machine learning techniques in predicting galaxy properties. Despite these advances, there are concerns about sensitivity to outliers and the test set size. Future efforts could refine the CNN model by exploring additional variables and applying dimensionality reduction techniques like PCA to identify and implement correlations in data alongside the image-processing model.

8 Acknowledgement

8.1 Presentation

Jinwoo: Related work, Methodology, CNN, Results, References
Jin Wuk: Background, Project Solution, Data, Graphs, Models, Conclusion

8.2 Final Report

Jinwoo: Introduction, Methodology, References
Jin Wuk: Abstract, Data, Models, Results, Discussion, Conclusion

8.3 Notebook

Jinwoo: Linear, Logistic, CNNs, Plots
Jin Wuk: Random Forest, Image Downloading

References

[1] Geda, R., Crawford, S. M., Hunt, L., Bershady, M., Tollerud, E., & Randriamampandry, S. (2022). PetroFit: A Python Package for Computing Petrosian Radii and Fitting Galaxy Light Profiles. In The Astronomical Journal (Vol. 163, Issue 5, p. 202). American Astronomical Society. https://doi.org/10.3847/1538-3881/ac5908

[2] Bialopetravičius, J., Narbutis, D., & Vansevičius, V. (2019). Deriving star cluster parameters with convolutional neural networks. In Astronomy & Astrophysics (Vol. 621, p. A103). EDP Sciences. https://doi.org/10.1051/0004-6361/201833833

[3] Monsalves, N., Jaque Arancibia, M., Bayo, A., Sánchez-Sáez, P., Angeloni, R., Damke, G., & Segura Van de Perre, J. (2024). Application of Convolutional Neural Networks to time domain astrophysics. 2D image analysis of OGLE light curves. In Astronomy & Astrophysics (Vol. 691, p. A106). EDP Sciences. https://doi.org/10.1051/0004-6361/202449995

[4] Surana, S., Wadadekar, Y., Bait, O., & Bhosale, H. (2020). Predicting star formation properties of galaxies using deep learning. In Monthly Notices of the Royal Astronomical Society (Vol. 493, Issue 4, pp. 4808–4815). Oxford University Press (OUP). https://doi.org/10.1093/mnras/staa537

[5] Pearson, J., Li, N., & Dye, S. (2019). The use of convolutional neural networks for modeling large optically-selected strong galaxy-lens samples. arXiv. https://doi.org/10.48550/arxiv.1904.06199