Classification of Galaxy Mergers through CNNs

Andrea Borghesi
916202
a.borghesi1@campus.unimib.it

July 9, 2024

Abstract

This study investigates automated methods for classifying galaxy mergers using convolutional neural networks (CNNs). The research was motivated by the need for efficient ways to analyze large astronomical datasets and identify merging galaxies, which play an important role in astronomy and galaxy evolution.

The study utilized simulated galaxy image data from the Illustris-1 cosmological simulations, including both "pristine" and artificially noise-added datasets. Three CNN architectures were evaluated: a custom "FastHeavyCNN", the previously published "DeepMerge" network, and a pre-trained ResNet18 model. Various denoising techniques were also explored, including Fourier transform, morphological operations, and autoencoders.

ResNet18 and the custom FastHeavyCNN achieved the best classification accuracies of $\sim 93\%$ on both pristine and noisy datasets, with DeepMerge underperforming with respect of the latter architectures. Denoising methods provided only marginal improvements in performance, suggesting that the adopted CNNs were already complex and capable enough to extract features from a noisy domain. It is notable the comparable performance of FastHeavyCNN with the pre-trained ResNet18 despite the absence of pretraining, and the usage of about half of the number of parameters.

1 Introduction

Galaxy mergers play a crucial role in the evolution of cosmic structures. They serve as a critical probe for understanding the hierarchical formation of galaxies within the framework of the ΛCDM cosmology paradigm. Key studies have shown that galaxy mergers influence various astrophysical processes, including star formation, black hole activity, and the overall morphology of galaxies [20] [9] [2]. During the epoch known as "cosmic high noon", occurred at redshifts $z \sim$ 2-3, the universe experienced peak star formation rates and significant mass assembly within galaxies [14]. Despite these advancements, the dynamics and rates of galaxy mergers during this period remain not fully understood, as empirical observations often conflict with theoretical models [7] [17] [15].

Traditional methods for detecting galaxy mergers, such as individual visual inspection or selection based on proximity in the sky and redshift, are labor-intensive and suffer from limitations in distinguishing true mergers from flybys [16]. These approaches require high-resolution, multi-wavelength data, which is often only available from space-based observatories. As the volume of astronomical data grows, there is a pressing need for more efficient and automated methods to analyze these data sets.

In recent years, convolutional neural networks (CNNs) have emerged as powerful tools for image classification tasks, demonstrating significant

successes in various domains, including astronomy. CNNs have the ability to automatically extract features and patterns in complex and large-scale datasets without the need for explicitly defined features, making them ideal for analyzing astronomical images.

This project aims to leverage CNNs to predict whether galaxies will merge within the next 500 Myr based on image data. By automating the classification process, we aim to overcome the limitations of traditional methods and provide more accurate and efficient predictions.

The motivation for this study is driven by the potential to achieve higher predictive accuracy through CNNs, denoising, and feature enhancing. Previous research in this area reported an accuracy of 75% utilizing unusual CNN architectures [24]. By optimizing architectures and preprocessing, we aim to improve these results.

2 Data and methodology

2.1 Dataset

Given the difficulty in acquiring sufficient observational data with labeled mergers for supervised learning algorithms, we rely on simulated data. Our approach utilizes images from the Illustris-1 cosmological simulations [22], processed and categorized into two primary datasets within the DeepMerge project [24]: "Pristine" (infinite SNR) and "Noisy". Both datasets originate from the same simulation snapshot. The "Noisy" dataset is created by convolving the pristine dataset with a model point-spread function specific to each filter. Random Gaussian noise is then added to each pixel, simulating shot noise characteristic of instruments like the Hubble Space Telescope (HST) and the James Webb Space Telescope (JWST). Galaxies are labeled as "mergers" if the merging event occurs within a 500Myr window. Each sample presents three channels, obtained using three HST filters: F814W (near-infrared band), F336W (ultra-violet band), and F160W (near-infrared band). The dataset has been split into train, validation, and test sets with a ratio of 70:10:20 respectively.

2.1.1 Disclaimer: likelihood of data leakage in the dataset

The dataset we utilized, sourced from the DeepMerge project, presents potential issues related to data leakage. As described in their article, the original dataset presented a significant class imbalance, with 1624 merger samples compared to 7306 non-merger samples. To address this imbalance, the authors augmented the merger class using simple flips and rotations before proceeding with the dataset splitting (see section 2.2 of [24]). This approach, however, introduces two significant concerns regarding data integrity. Firstly, it creates a high risk of data leakage, as augmented versions of an original image may be distributed across different sets, potentially placing some in the training set and others in the validation set. This overlap compromises the independence of these sets. Secondly, the inclusion of augmented images in the validation and test sets undermines their ability to accurately represent the original data distribution. These sets are intended to evaluate model performance on authentic, unseen data. The presence of augmented images in these sets may skew the statistics derived from them, potentially leading to overly optimistic performance estimates.

On the other hand, the dataset that they published has a different cardinality from the one mentioned, totaling 10,798 samples instead of the 15,426 (after augmentation) announced. This is to say that there might be a chance that this dataset might not suffer of the same problems, but we can't be certain about it.

2.2 Data augmentation

To better capture the diversity present in the image distribution, we performed slight data augmentation

on the training dataset. However, to avoid altering the inherent features of each image, and to not add more noise to the noisy dataset, we perform random flips in both axes and random 90-degree rotations. Of course, in order not to leak data, the augmentation is performed on the training set only.

2.3 Denoising

To address the often low Signal-to-Noise Ratio (SNR) in the noisy dataset, we explore several denoising techniques: Fourier transform, white top-hat transform, minimum filter for background estimation, and autoencoders. We evaluate each method's effectiveness by calculating the entropy difference between the original and denoised images, assuming noise reduction corresponds to decreased randomness and entropy. Importantly, we apply these denoising procedures to the entire dataset. The Fourier transform, white top-hat transform, and minimum filter operate independently on each input sample without requiring dataset-wide statistics. This allows us to denoise the dataset without risking data leakage between training and validation sets.

2.3.1 Fourier Transform based denoising

We apply the Fourier transform to each image channel, decomposing them into spatial frequencies. In frequency space, low frequencies at the borders represent major image features, while higher frequencies toward the center correspond to fine details, textures, edges, and noise. We set the inner 35% of rows and columns to zero, aiming to remove most noise without significant information loss. An inverse Fourier transform then produces the denoised image (Figure 1). Observation reveals that this process not only denoises the image but also tends to slightly enhance features due to the blurring effect.

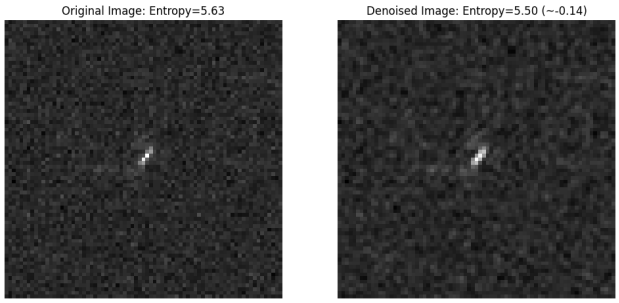

Figure 1: Band 0 of sample no.4369 denoised with a Fourier-based approach.

2.3.2 Morphology-based: White Top-hat Transform

The white top-hat transform is a morphological operation designed to extract "white" or bright elements from an image using a predefined structuring element. In our implementation, we utilized a circular structuring element with a 5-pixel radius. This transform is mathematically defined as the difference between the original image and its morphological opening with respect to the structuring element. The resulting image from this transformation highlights elements that are both smaller than the structuring element and brighter than their surrounding areas (Figure 2). While this approach appears well-suited for our task of identifying bright, compact objects, it comes with certain drawbacks. The method can be quite aggressive, potentially altering the structure of galaxies or even failing to detect them entirely if they don't conform to the size and shape constraints of the structuring element.

2.3.3 Morphology-based: Minimum Filter

To estimate and subsequently remove the background from an image, we employed a minimum filter in conjunction with a structuring element. Our chosen structuring element was a circular shape with a 5-pixel radius. This method aims to preserve the in-

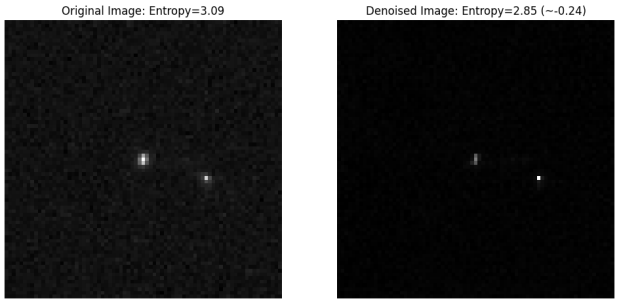

Figure 2: Band 0 of sample no.11604 denoised with a white Top-hat transform.

tegrity of galaxy shapes, resulting in a more conservative approach to background removal (Figure 3). However, it's worth noting that this technique faces challenges when confronted with highly noisy backgrounds, as illustrated in Figure 4. In such cases, the filter's effectiveness in background elimination is somewhat diminished.

Figure 3: Band 0 of sample no.11604 denoised by subtracting the estimated background through a minimum filter.

2.3.4 Autoencoders: U-Net

Autoencoders have demonstrated their effectiveness in various Self-Supervised Learning tasks, including denoising [1]. For this application, we implemented a straightforward U-Net architecture, as shown in Table 1. The choice of a simple design was deliberate, as

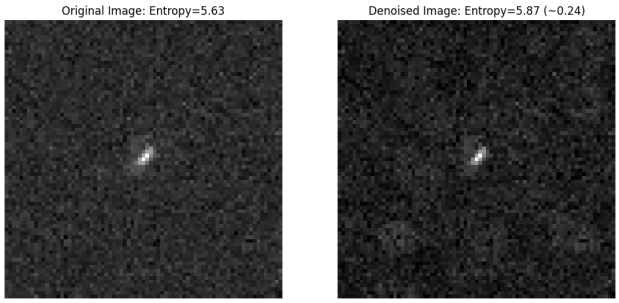

Figure 4: Band 0 of sample no.4369 denoised by subtracting the estimated background through a minimum filter. The circular structuring elements left footprints on the background that most likely increased the entropy.

increased complexity could potentially lead to noise reproduction beyond the bottleneck. We opted for a U-Net over a classic Autoencoder due to its layer concatenation feature, which tends to preserve finer details [18]. Figure 5 illustrates a sample of the denoising capability achieved with this technique. It's crucial to note that careful training is necessary to prevent the network from learning to reproduce noise. In our case, we limited the training to 4 epochs using a training set obtained through a seeded split to avoid data leakage. This approach balances effective denoising and preventing overfitting to noise patterns.

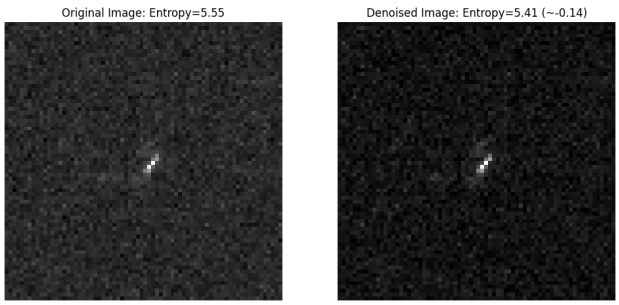

Figure 5: Band 0 of sample no.4369 denoised by a shallow U-Net.

Layer	Details	Output Shape
Input Image: $72 \times 72 \times 3$		
Conv1	Conv2d(3, 8, 3x3, stride=1, padding=same)	$72 \times 72 \times 8$
	ReLU()	$72 \times 72 \times 8$
	Conv2d(8, 8, 3x3, stride=1, padding=same)	$72 \times 72 \times 8$
	ReLU()	$72 \times 72 \times 8$
	MaxPool2d(2x2, stride=2)	$36 \times 36 \times 8$
Bottleneck	Conv2d(8, 16, 3x3, stride=1, padding=same)	$36 \times 36 \times 16$
	ReLU()	$36 \times 36 \times 16$
	Conv2d(16, 16, 3x3, stride=1, padding=same)	$36 \times 36 \times 16$
	ReLU()	$36 \times 36 \times 16$
UpConv1	UpSampling2d(2x2)	$72 \times 72 \times 16$
	Concatenate()	$72 \times 72 \times 24$
	Conv2d(24, 8, 3x3, stride=1, padding=same)	$72 \times 72 \times 8$
	ReLU()	$72 \times 72 \times 8$
	Conv2d(8, 8, 3x3, stride=1, padding=same)	$72 \times 72 \times 8$
	ReLU()	$72 \times 72 \times 8$
Output	Conv2d(8, 3, 1x1, stride=1, padding=same)	$72 \times 72 \times 3$

Table 1: Simple UNet architecture intended for denoising. No. of parameters: 6643

2.4 CNN Architectures

Three architectures have been tested: DeepMerge (Table 3), the architecture presented in [24], the renowned ResNet18 [5] pre-trained on ImageNet and a custom CNN "HeavyNet" (Table 2). Optimizers, schedulers, and metrics for training and validating, that will be discussed in the following sections, are kept the same for each of the presented networks.

2.4.1 Optimizers and Schedulers

As for the optimizer we settled for AdamW, an optimization of Adam [10], it differs on the weight decay, achieving a better regularization and potentially a faster convergence [12]. As for the scheduler we went for the Cosine Annealing, we also tried Cosine Annealing with Warm Restarts [11] but achieved very similar results. The learning rate used is set to 10^{-4} and as for the schedulers we set the `T_max` parameter to the same amount as the epochs trained for a smooth annealing cycle.

2.5 Metrics and Validation

The task is binary classification, and given that the dataset is slightly unbalanced with $5684 : 5114$, therefore we opted for a weighted Binary Cross-Entropy to balance out such class imbalance.

2.6 FastHeavyCNN

Our proposed network is a moderately deep architecture comprising 13 layers, of which 10 are convolutional. The overall structure draws inspiration from VGG16 and AlexNet. A notable design choice is the use of LeakyReLU as the activation function [13], selected to mitigate the risk of the "dead neuron" problem associated with ReLU [4]. While GELU was considered as an alternative due to its reported superior performance in empirical tests, its significantly higher computational cost [6] led us to test it for this task, revealing no significant performance improvements. Another crucial design element is the incorporation of Batch Normalization layers, which ensure smoother gradient flow and maintain normalized distribution, thereby reducing internal covariate shift [8]. The network's complexity was deliberately chosen to exceed that of the DeepMerge architecture while remaining below that of ResNet18, aiming to compete with the latter in this specific task. In total, our architecture comprises 6,633,249 parameters.

2.7 ResNet18

ResNet18, part of a family of networks that has demonstrated remarkable success across numerous tasks, made significant contributions to the ImageNet competition [5]. For our purposes, we chose to utilize this smallest version of the ResNet architecture. We employed transfer learning by starting with a ResNet18 model pre-trained on ImageNet, then replaced its classifier head with a custom one (Table 4). To leverage the pre-trained features while allowing adaptation to our specific task, we froze the weights of the first 9 convolutional layers and kept the subsequent 8 layers and the classifier head unfrozen. This approach balances the use of general features learned from ImageNet with the need for task-specific fine-tuning. The resulting architecture totals 12,227,137 parameters, of which 11,544,065 are trainable.

Layer	Details	Output Shape
Input Image		
Conv1	Conv2d(3, 32, 3x3, stride=1)	[-1, 32, 75, 75]
	BatchNorm2d(32)	[-1, 32, 75, 75]
	LeakyReLU()	[-1, 32, 75, 75]
	Conv2d(32, 32, 3x3, stride=1)	[-1, 32, 75, 75]
	BatchNorm2d(32)	[-1, 32, 75, 75]
	LeakyReLU()	[-1, 32, 75, 75]
	MaxPool2d(2x2, stride=2)	[-1, 32, 37, 37]
Conv2	Conv2d(32, 64, 3x3, stride=1)	[-1, 64, 37, 37]
	BatchNorm2d(64)	[-1, 64, 37, 37]
	LeakyReLU()	[-1, 64, 37, 37]
	Conv2d(64, 64, 3x3, stride=1)	[-1, 64, 37, 37]
	BatchNorm2d(64)	[-1, 64, 37, 37]
	LeakyReLU()	[-1, 64, 37, 37]
	MaxPool2d(2x2, stride=2)	[-1, 64, 18, 18]
Conv3	Conv2d(64, 128, 3x3, stride=1)	[-1, 128, 18, 18]
	BatchNorm2d(128)	[-1, 128, 18, 18]
	LeakyReLU()	[-1, 128, 18, 18]
	Conv2d(128, 128, 3x3, stride=1)	[-1, 128, 18, 18]
	BatchNorm2d(128)	[-1, 128, 18, 18]
	LeakyReLU()	[-1, 128, 18, 18]
	Conv2d(128, 128, 3x3, stride=1)	[-1, 128, 18, 18]
	BatchNorm2d(128)	[-1, 128, 18, 18]
	LeakyReLU()	[-1, 128, 18, 18]
	MaxPool2d(2x2, stride=2)	[-1, 128, 9, 9]
Conv4	Conv2d(128, 256, 3x3, stride=1)	[-1, 256, 9, 9]
	BatchNorm2d(256)	[-1, 256, 9, 9]
	LeakyReLU()	[-1, 256, 9, 9]
	Conv2d(256, 256, 3x3, stride=1)	[-1, 256, 9, 9]
	BatchNorm2d(256)	[-1, 256, 9, 9]
	LeakyReLU()	[-1, 256, 9, 9]
	Conv2d(256, 256, 3x3, stride=1)	[-1, 256, 9, 9]
	BatchNorm2d(256)	[-1, 256, 9, 9]
	LeakyReLU()	[-1, 256, 9, 9]
	MaxPool2d(2x2, stride=2)	[-1, 256, 4, 4]
FC	Linear(in_features=256*4*4, out_features=1024)	[-1, 1024]
	LeakyReLU()	[-1, 1024]
	Dropout(0.5)	[-1, 1024]
	Linear(in_features=1024, out_features=512)	[-1, 512]
	LeakyReLU()	[-1, 512]
	Dropout(0.5)	[-1, 512]
	Linear(in_features=512, out_features=1)	[-1, 1]
	Sigmoid()	[-1, 1]

Table 2: FastHeavyCNN architecture

Layer	Details	Output Shape
Input Image		
Conv1	Conv2d(3, 8, 3x3, stride=1)	[-1, 8, 75, 75]
	ReLU()	[-1, 8, 75, 75]
	BatchNorm2d(8)	[-1, 8, 75, 75]
	MaxPool2d(2x2, stride=2)	[-1, 8, 37, 37]
	Dropout(0.5)	[-1, 8, 37, 37]
Conv2	Conv2d(8, 16, 3x3, stride=1)	[-1, 16, 37, 37]
	ReLU()	[-1, 16, 37, 37]
	BatchNorm2d(16)	[-1, 16, 37, 37]
	MaxPool2d(2x2, stride=2)	[-1, 16, 18, 18]
	Dropout(0.5)	[-1, 16, 18, 18]
Conv3	Conv2d(16, 32, 3x3, stride=1)	[-1, 32, 18, 18]
	ReLU()	[-1, 32, 18, 18]
	BatchNorm2d(32)	[-1, 32, 18, 18]
	MaxPool2d(2x2, stride=2)	[-1, 32, 9, 9]
	Dropout(0.5)	[-1, 32, 9, 9]
FC	Linear(in_features=2592, out_features=64)	[-1, 64]
	Softmax()	[-1, 64]
	Linear(in_features=64, out_features=32)	[-1, 32]
	Softmax()	[-1, 32]
	Linear(in_features=32, out_features=1)	[-1, 1]
	Sigmoid()	[-1, 1]

Table 3: DeepMerge architecture

Layer	Details	Output Shape
FC1	Linear(in_features, 1024)	[-1, 1024]
	GELU()	[-1, 1024]
	Dropout(0.5)	[-1, 1024]
FC2	Linear(1024, 512)	[-1, 512]
	GELU()	[-1, 512]
	Dropout(0.5)	[-1, 512]
Output	Linear(512, 1)	[-1, 1]
	Sigmoid()	[-1, 1]

Table 4: Classification head attached to ResNet18

Dataset	FastHeavyCNN	DeepMerge	ResNet18
Pristine	0.9355	0.7214	0.9355
Noisy	0.9271	0.6933	0.9313
Fourier	0.9239	0.6972	0.9371
Min. Filter	0.9223	0.6479	0.9258
Top-hat	0.9149	0.6459	0.9358
Autoencoder	0.7752	0.6785	0.7623

Table 5: Test accuracies of each architecture on each variation of the dataset

3 Results

We trained each architecture on every dataset variation for 150 epochs, a duration chosen based on observed convergence around 100 epochs in most experiments. Each training round took approximately 10-15 minutes on a laptop-grade RTX 3060 GPU. Table 5 summarizes the accuracies achieved. Notably, FastHeavyCNN and ResNet18 performed similarly on both pristine and noisy datasets, suggesting their ability to extract crucial features even without denoising assistance.

3.1 Impact of Denoising

While denoising did not significantly improve overall performance, the Fourier denoised dataset consistently yielded marginally better results across all architectures. This subtle enhancement may be attributed to Fourier denoising's dual effect of noise

reduction and feature enhancement, albeit with an introduced blur-like effect. As expected, minimum filters and top-hat transformations produced comparable results, aligning with their similar nature. Surprisingly, autoencoders consistently underperformed. This poor performance might be due to insufficient training of the denoising network or, conversely, excessive network complexity for the task. Visual inspection revealed early noise replication in autoencoder-denoised samples, supporting the latter hypothesis. These findings underscore the challenges in selecting and implementing effective denoising techniques for complex tasks like galaxy merger identification. Figures 6, 7, and 8 illustrate the ROC curves and AUC values for each architecture across datasets, highlighting the similar performance of most denoising procedures, except for autoencoding.

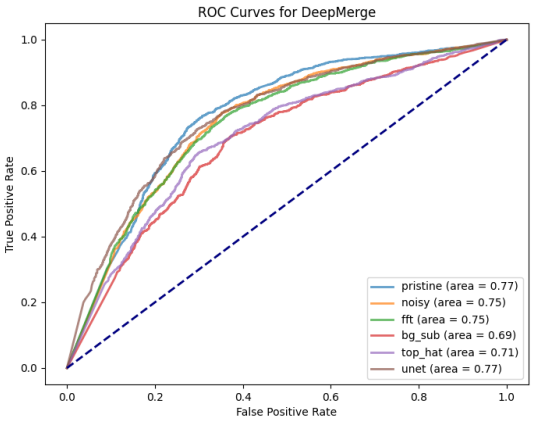

Figure 6: ROC curve of the DeepMerge architecture trained on each dataset.

3.2 Comparison of the architectures

For this analysis, we primarily focus on the performance of each architecture when tested on the noisy dataset. This choice is justified by the observation that architectural behavior across other datasets remains extremely similar, if not identical. The only notable exception in performance occurred when training the architectures on the autoencoder-denoised dataset. However, as previously discussed, these performances were extremely poor due to the aforementioned reasons, warranting no further exploration.

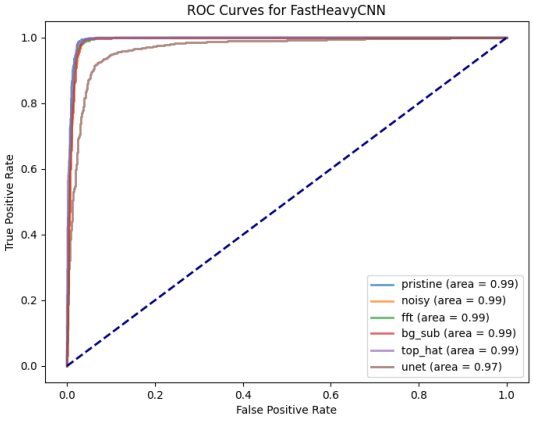

Figure 7: ROC curve of the FastHeavyCNN architecture trained on each dataset.

Among the three architectures tested, the DeepMerge architecture clearly underperformed compared to the other proposals. This behavior was anticipated due to its relative lack of complexity and unconventional structure, particularly the inclusion of double softmax layers. The confusion matrix shown in Figure 9 generalizes the behavior of this architecture across each dataset, revealing a consistent accuracy of around 84% for True mergers and a varying accuracy between 49% - 59% for True non-mergers.

On the other end of the performance spectrum, we find FastHeavyCNN and ResNet18 performing very closely, with ResNet18 consistently producing slightly better results. However, it's important to highlight that FastHeavyCNN achieves these comparable results with approximately half the number

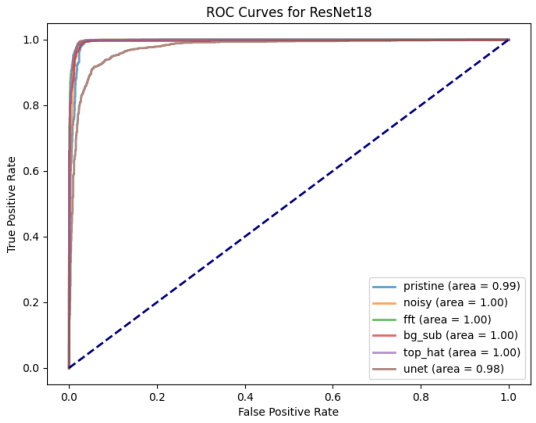

Figure 8: ROC curve of the ResNet18 architecture trained on each dataset.

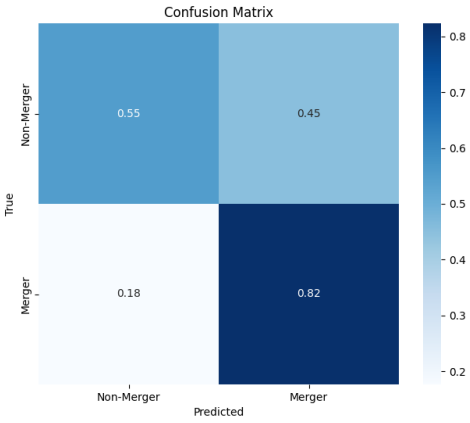

Figure 9: Confusion matrix of Deepmerge architecture tested on 20% of the noisy dataset

of parameters of ResNet18. This observation suggests that ResNet18 might be overreaching in terms of complexity for this specific task. Conversely, the higher complexity of ResNet18 does lead to faster convergence, as evidenced by comparing Figures 12 and 13. Interestingly, both FastHeavyCNN and ResNet18 demonstrate a tendency to classify mergers more easily than non-mergers, as shown in Figures 10 and 11. This behavior might be attributed to the more prominent and distinctive features typically present in merger galaxies compared to non-mergers.

For a comprehensive overview of each architecture's performance across all datasets, refer to Figure 14.

Given the relatively low resolution of the input images and the moderate complexity of the architectures, we observed impressive inference speeds: approximately 420 FPS for FastHeavyCNN, 270 FPS for ResNet18, and 800 FPS for DeepMerge. These high frame rates suggest that all three models, particularly FastHeavyCNN and DeepMerge, are well-suited for real-time applications, should the need arise.

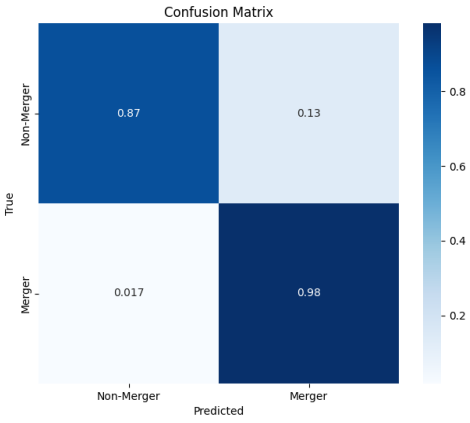

Figure 10: Confusion matrix of FastHeavyCNN architecture tested on 20% of the noisy dataset

3.3 Limitations and Considerations

While our results show promising accuracies, it's important to note that these are achieved on models trained using simulated datasets. Despite the high quality of these simulations, they remain approximations of reality. In real-world applications, we would

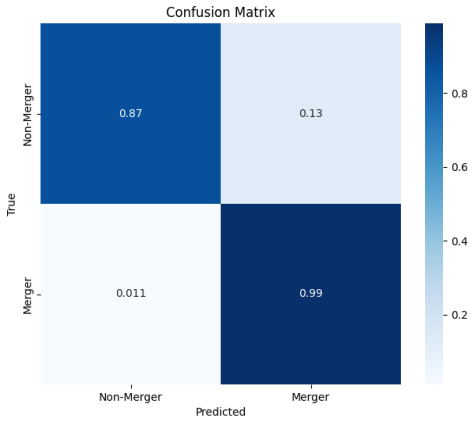

Figure 11: Confusion matrix of ResNet18 architecture tested on 20% of the noisy dataset

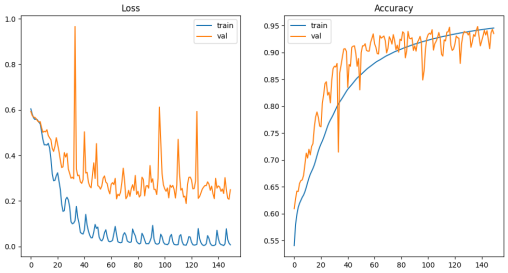

Figure 12: ResNet18's loss and accuracy plots during training on the noisy dataset. Notice how it starts to converge at 90% accuracy around epoch 40.

expect these performance metrics to be lower due to the increased complexity and variability of actual astronomical data.

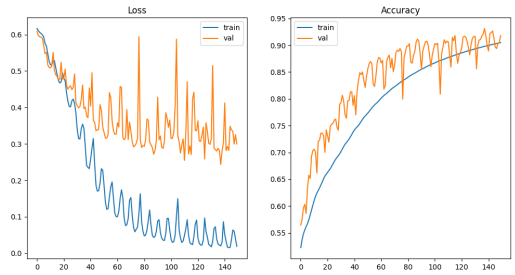

Figure 13: FastHeavyCNN's loss and accuracy plots during training on the noisy dataset. Notice how it starts to converge at 90% accuracy around epoch 80-90.

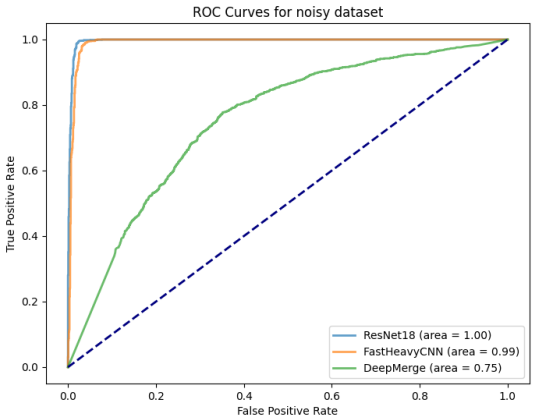

Figure 14: ROC curve of each architecture trained on the noisy dataset.

4 Conclusions

This study presents a more comprehensive approach to the galaxy merger classification problem, building upon previous work [5]. We placed significant emphasis on denoising the dataset; however, our experiments revealed that the tested architectures were already sufficiently complex to extract crucial information from noisy samples without significant per-

formance gains from denoising procedures. The best-performing network in our study was the renowned ResNet18, pre-trained on ImageNet with the first 8 convolutional layers frozen. We also developed a custom CNN that achieved comparable results to ResNet18, but with half the trainable parameters and faster inference time, albeit at the cost of slower convergence.

4.1 Future Work

As with any machine learning problem, a larger dataset would likely yield more robust results. This is particularly crucial in our case, given the concerns about the generation of the provided dataset, as discussed in Section 2.1.1. Although denoising procedures alone did not provide apparent improvements, future research could explore combining these techniques with specialized feature-enhancement methods. Regarding model architecture, access to more powerful computational resources would enable testing of higher-complexity models such as Vision Transformers (ViT), which have shown promising results in image classification tasks [19] [3] [21] [23]. Additionally, further investigation into the trade-offs between model complexity, convergence speed, and performance could yield valuable insights for optimizing galaxy merger classification systems.

Disclaimer

This report does not contain any form of plagiarism, including content generated or suggested by AI tools such as ChatGPT or similar services. All sources used have been properly cited and referenced.

References

[1] Snigdha Agarwal, Ayushi Agarwal, and Maroti Deshmukh. Denoising images with varying noises using autoencoders. In *Computer Vision and Image Processing: 4th International Conference, CVIP 2019, Jaipur, India, September 27–29, 2019, Revised Selected Papers, Part II 4*, pages 3–14. Springer, 2020.

[2] Christopher J. Conselice. The evolution of galaxy structure over cosmic time. *Annual Review of Astronomy and Astrophysics*, 52(1):291–337, August 2014.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[4] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[7] Philip F Hopkins, Kevin Bundy, Darren Croton, Lars Hernquist, Dusan Keres, Sadegh Khochfar, Kyle Stewart, Andrew Wetzel, and Joshua D Younger. Mergers and bulge formation in λcdm: which mergers matter? *The Astrophysical Journal*, 715(1):202, 2010.

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[9] Guinevere Kauffmann, Simon DM White, and Bruno Guiderdoni. The formation and evolution of galaxies within merging dark matter haloes. *Monthly Notices of the Royal Astronomical Society*, 264(1):201–218, 1993.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[11] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[12] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[13] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, GA, 2013.

[14] Piero Madau and Mark Dickinson. Cosmic star-formation history. *Annual Review of Astronomy and Astrophysics*, 52(1):415–486, 2014.

[15] Allison W. S. Man, Andrew W. Zirm, and Sune Toft. Resolving the discrepancy of galaxy merger fraction measurements at z $\sim$ 0–3. *The Astrophysical Journal*, 830(2):89, oct 2016.

[16] D. R. Patton, C. J. Pritchet, R. G. Carlberg, R. O. Marzke, H. K. C. Yee, P. B. Hall, H. Lin, S. L. Morris, M. Sawicki, C. W. Shepherd, and G. D. Wirth. Dynamically close galaxy pairs and merger rate evolution in the cnoc2 redshift survey. *The Astrophysical Journal*, 565(1):208–222, January 2002.

[17] Jr. R. E. Ryan, S. H. Cohen, R. A. Windhorst, and J. Silk. Galaxy mergers at z $\gtrsim$ 1 in the hudf: Evidence for a peak in the major merger rate of massive galaxies*. *The Astrophysical Journal*, 678(2):751, may 2008.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[19] Uzair Shah, Jens Schneider, Giovanni Pintore, Enrico Gobbetti, Mahmood Alzubaidi, Mowafa Househ, and Marco Agus. Elevit: exploiting element-wise products for designing efficient and lightweight vision transformers.

[20] Alar Toomre and Juri Toomre. Galactic bridges and tails. *Astrophysical Journal, Vol. 178, pp. 623-666 (1972)*, 178:623–666, 1972.

[21] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[22] M. Vogelsberger, S. Genel, V. Springel, P. Torrey, D. Sijacki, D. Xu, G. Snyder, S. Bird, D. Nelson, and L. Hernquist. Properties of galaxies reproduced by a hydrodynamic simulation. *Nature*, 509(7499):177–182, May 2014.

[23] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions, 2021.

[24] A. Ćiprijanović, G.F. Snyder, B. Nord, and J.E.G. Peek. Deepmerge: Classifying high-redshift merging galaxies with deep neural networks. *Astronomy and Computing*, 32:100390, 2020.