Stellar Signatures: Object Detection and Classification in Space and Deep Space Imagery using Deep Learning

Christos Foukanelis (mtn2324)
University of Pireus
NCSR Demokritos
Athens, Greece
christosfouk@yahoo.com

Konstantinos Vytiniotis (mtn2308)
University of Pireus
NCSR Demokritos
Athens, Greece
konstantinos.vytiniotis@outlook.com

Abstract

This report presents a comprehensive study on the application of deep learning techniques for object detection and classification in space and deep space imagery. Utilizing the DeepSpaceYoloDataset as a baseline, we meticulously annotated images of galaxies, nebulae, and star clusters. We trained and evaluated two state-of-the-art pretrained models: YOLO (You Only Look Once) and Faster R-CNN ResNet-50. Furthermore, we developed a custom object detection framework comprising of three neural networks dedicated to feature extraction, bounding box regression, and classification. Our approach addresses the unique challenges posed by astronomical imagery, such as the diffuse nature of nebulae and the large size of some celestial objects. The results demonstrate significant advancements in the accurate identification and classification of celestial objects, contributing to the field of astronomical data analysis.

I. Introduction

The vast expanse of space is filled with numerous celestial objects, each with distinct characteristics and significance. Accurately detecting and classifying these objects in space and deep space imagery is crucial for astronomical research and understanding the universe. In this project, titled "Stellar Signatures: Object Detection and Classification in Space and Deep Space Imagery using Deep Learning," we leverage advanced deep learning techniques to tackle this challenging task. Our experiments conclusively demonstrated that leveraging deep learning methods alongside a sufficiently large dataset with accurate annotations makes the task of detecting the three stellar objects described in our project highly achievable. The results yielded promising levels of accuracy and reliability.

A. Project Overview

Our primary objective is to develop robust models capable of detecting and classifying three types of celestial objects: galaxies, nebulae, and star clusters. To achieve this, we utilized the DeepSpaceYoloDataset, which provides a rich collection of around 4,600 images annotated with approximately 3,600 bounding boxes. Certain images did not contain stellar objects of interest in the context of this project, so they counted towards the empty background class. These annotations were created using Label Studio, focusing on accurately capturing the objects of interest despite the inherent challenges posed by astronomical imagery.

B. Methods Explored

To address the complexities of object detection in deep space imagery, we explored multiple deep learning approaches:

1) Pretrained Models

- **YOLO (You Only Look Once):** Known for its real-time object detection capabilities, YOLO is a single-stage detector that processes images with remarkable speed and accuracy. We fine-tuned this model using our annotated dataset to enhance its performance on celestial objects.
- **Faster R-CNN ResNet-50:** This two-stage detector combines the region proposal capabilities of Faster R-CNN with the powerful feature extraction of ResNet-50. By training this model with our dataset, we aimed to achieve high precision in detecting and classifying galaxies, nebulae, and star clusters.

2) Custom Object Detection Framework

To address the complexities of object detection in deep space imagery, we explored multiple deep learning approaches:

- **Feature Extraction Network:** This network is responsible for extracting relevant features from the input images, leveraging advanced convolutional neural network (CNN) architectures.
- **Bounding Box Regression Network:** Focused on accurately predicting the coordinates of bounding boxes, this network ensures precise localization of celestial objects.
- **Classification Network:** This network classifies the detected objects into the predefined categories (galaxy, nebula, and star cluster), utilizing the features extracted by the first network.

C. Challenges and Innovations

The unique characteristics of astronomical objects presented several challenges, particularly in the annotation and detection processes. Nebulae, with their diffuse and irregular shapes, required careful handling to define accurate bounding boxes. Additionally, some objects covered entire images, complicating the annotation and detection tasks. Our custom framework and the fine-tuning of pretrained models were specifically designed to address these issues, resulting in improved detection accuracy and classification performance.

Through the combination of state-of-the-art pretrained models and a custom-designed object detection framework, our project makes significant strides in the automated detection and classification of celestial objects. The methodologies and results presented in this report highlight the potential of deep learning to advance astronomical research and enhance our understanding of the universe.

D. Related Work

In recent years, there has been significant advancement in the application of deep learning techniques for astronomical data analysis. Notable among these is AstroCV **[3]**, a specialized computer vision library designed to process large-scale astronomical datasets. AstroCV offers efficient algorithms in Python and C++ for object recognition, segmentation, and classification tasks. By leveraging deep learning techniques and GPU acceleration, AstroCV automates the detection and categorization of galaxies with superior performance compared to traditional methods. The library is open-source and provides pre-trained models and Python tutorials, facilitating easy integration and use in various astronomical projects.

Another significant contribution in this domain is AstroYOLO **[4]**, a hybrid model combining convolutional neural networks and Transformers, specifically designed for detecting Blue Horizontal-Branch stars (BHBs) in large-scale datasets, such as those from the Sloan Digital Sky Survey (SDSS). AstroYOLO capitalizes on the global receptive field of Transformers and the local feature representation of convolutions, offering a robust solution for astronomical object detection that surpasses the capabilities of traditional methods.

Additionally, there have been efforts to integrate the YOLO model into the AstroCV framework for galaxy detection **[5]**. Researchers trained this model using diverse, large-scale public datasets containing stellar images, demonstrating the effectiveness of YOLO in handling astronomical image data. This integration not only enhances the detection capabilities of AstroCV but also showcases the versatility of the YOLO model in different astronomical applications.

II. Dataset

The DeepSpaceYoloDataset is a specialized astronomical dataset designed to facilitate the development and testing of machine learning models for object detection in deep space images. This dataset comprises approximately 4,600 images, capturing various celestial objects, and includes around 3,600 handcrafted annotations, focusing on three primary classes: Galaxy, Nebula, and Star Cluster. The annotations were meticulously created using Label Studio, ensuring accurate bounding boxes for the objects of interest.

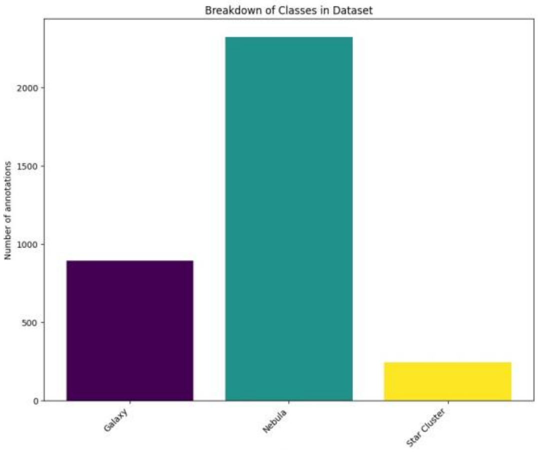


Figure 1. Class breakdown of annotations.

A. Description

The DeepSpaceYoloDataset was curated to support advanced research in the field of astronomy, particularly in the automated detection and classification of celestial objects. The images in this dataset cover a broad range of astronomical phenomena, captured using high-resolution telescopes. Each image is annotated with bounding boxes that indicate the location and extent of the objects, providing essential data for training object detection models. Each RGB image has a resolution of 608 x 608 pixels and corresponds to the capture of different zones of the night sky visible in Northern Hemisphere.

Figure 2. The Stellina smart instrument during an observation session in September 2023.

B. Annotations and Labeling Process

Annotations were crafted using Label Studio, an open-source data labeling tool. This process involved manually drawing bounding boxes around the objects of interest in each image. The annotations were classified into three distinct categories:

- Galaxy
- Nebula
- Star Cluster

This labor-intensive process ensured high-quality annotations, which are crucial for the performance of machine learning models. However, the annotation process was not without challenges:

- **Nebula Annotations:** Due to the diffuse and irregular shapes of nebulae, accurately delineating their boundaries with bounding boxes proved to be difficult. This often required subjective judgment and fine-tuning **[Fig.3]**.
- **Full Image Coverage:** Some celestial objects, particularly large nebulae and star clusters, spanned the entire image. Creating bounding boxes for such objects was challenging because the boxes often covered the entire image, complicating the annotation process **[Fig.4]**.

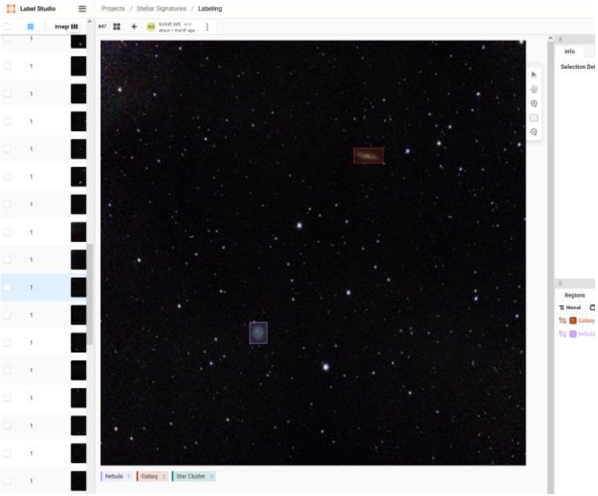


Figure 3. Handcrafted annotations of bounding boxes using Label Studio. Example annotations of a nebulae and a galaxy.

C. *Classes and their Descriptions*

- **Galaxy:** A galaxy is a vast system of stars, stellar remnants, interstellar gas, dust, and dark matter, bound together by gravity. Galaxies are categorized by their shapes and structures into types such as spiral, elliptical, and irregular galaxies. Each galaxy can contain millions to billions of stars and various star systems and clusters. In the DeepSpaceYoloDataset, galaxies are annotated based on their visible structure and brightness, providing critical data for models to learn to identify and classify these complex systems.

- **Nebula:** Nebulae are immense clouds of gas and dust in space, often serving as the birthplaces of stars. They come in various forms, including emission nebulae, which glow due to the ionization of gas, and reflection nebulae, which shine by reflecting the light of nearby stars. The irregular and diffuse nature of nebulae makes them particularly challenging to annotate. Accurate bounding boxes are essential for machine learning models to correctly identify and differentiate nebulae from other celestial objects.

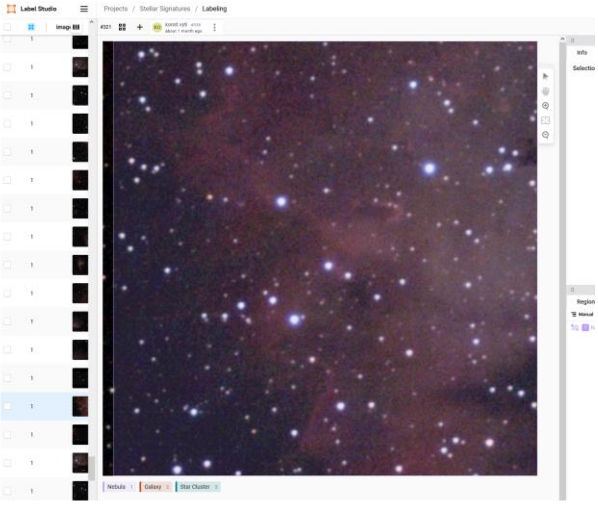


Figure 4. Example of full image coverage issues. Bounding box for a nebulae.

- **Star Cluster:** Star clusters are groups of stars that are gravitationally bound and often formed from the same molecular cloud. There are two main types of star clusters: open clusters, which are loosely bound and contain a few hundred stars, and globular clusters, which are tightly bound and can contain hundreds of thousands of stars. In the dataset, star clusters are annotated by marking the entire group, which can vary significantly in size and density.

III. Methodology

In this section, we present the methodology for our project, focusing on three key models: YOLOv8, Faster R-CNN, and a custom object detection framework. For each model, we provide a detailed overview of its architecture and how it was adapted for our specific task. We then present the performance results obtained from training and evaluating the model on the DeepSpaceYoloDataset. Finally, we analyze these results, highlighting the model's strengths, weaknesses, and overall effectiveness in detecting and classifying celestial objects.

A. *Pretrained Models – YOLOv8*

YOLOv8 (You Only Look Once version 8) is an advanced object detection model known for its high speed and accuracy. The YOLOv8 architecture builds upon previous YOLO versions, incorporating improvements in both network design and training techniques to achieve better performance. The model processes the entire image in a single forward pass, predicting bounding boxes and class probabilities simultaneously.

YOLOv8's architecture **[Fig.5]** includes a series of convolutional layers followed by detection layers. These layers are responsible for extracting features from the input image and making predictions based on these features. The final output layer provides the coordinates of bounding boxes and the associated class probabilities.

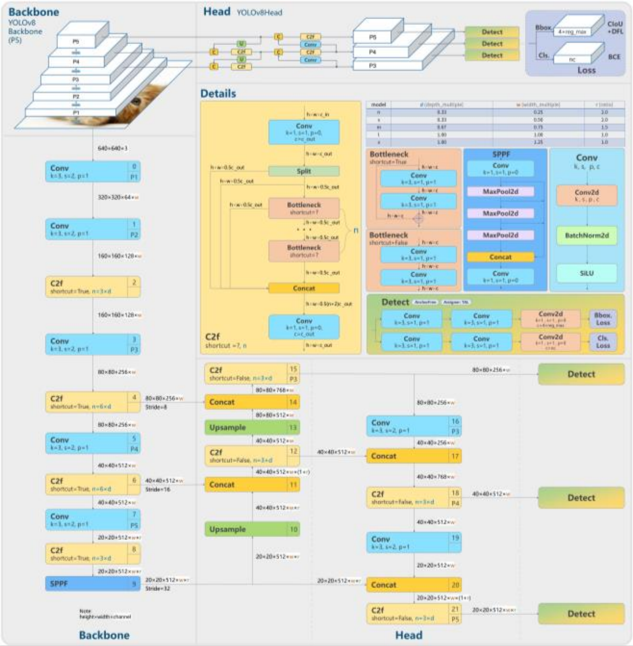


Figure 5. YOLOv8 – High Level Architecture.

YOLOv8 divides the input image into an SxS grid. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect the likelihood that the box contains an object and the accuracy of the bounding box itself. Each bounding box prediction also includes class probabilities for C classes.

YOLOv8 was initially trained on the COCO dataset, which contains 80 different object classes. The training involves optimizing the model to minimize the loss, which is a combination of localization loss (error in bounding box coordinates), confidence loss, and classification loss.

To adapt YOLOv8 for our specific task of detecting galaxies, nebulae, and star clusters, we fine-tuned a pretrained YOLOv8 model using the DeepSpaceYoloDataset. The following steps outline the methodology:

- **Model Initialization:** We initialized a YOLOv8 model using a pretrained weight file *yolov8n.pt*.
- **Data Preparation:** The dataset was prepared by creating a configuration file (*data.yaml*) that specifies the classes and paths to the training and validation datasets.
- **Training:** The model was trained on our dataset with an image size of 608x608 pixels. The training process involved adjusting the model weights to better detect our specific classes.

The dataset was split into training and validation sets to evaluate the model's performance. A typical split strategy involves using 80% of the data for training and 20% for validation. This split ensures that the model is trained on a diverse set of images while being evaluated on unseen data to measure its generalization ability.

The main hyperparameters which were tuned during the training process are:

- Epochs: The number of training epochs was set to 10 and 20 to compare the results. This parameter determines how many times the entire dataset is passed through the network during training. The number of epochs for pretrained models is usually much lower than training the model from scratch.
- Batch Size: The batch size, which defines the number of images processed before updating the model parameters, was optimized for the available hardware resources.
- Learning Rate: The learning rate controls the step size during the optimization process. A smaller learning rate helps in fine-tuning the model more precisely.

The dataset was split into training and validation sets to evaluate the model's performance. A typical split strategy involves using 80% of the data for training and 20% for validation. This split ensures that the model is trained on a diverse set of images while being evaluated on unseen data to measure its generalization ability.

The performance of the YOLOv8 model was evaluated using several metrics:

- Precision: Measures the accuracy of the positive predictions.
- Recall: Measures the completeness of the positive predictions.
- F1 Score: Harmonic mean of precision and recall, providing a single metric for model performance.
- Confusion Matrix: A detailed breakdown of true positives, false positives, and false negatives for each class.

Following up are the results for the two output models trained with 10 and 20 epochs respectively:

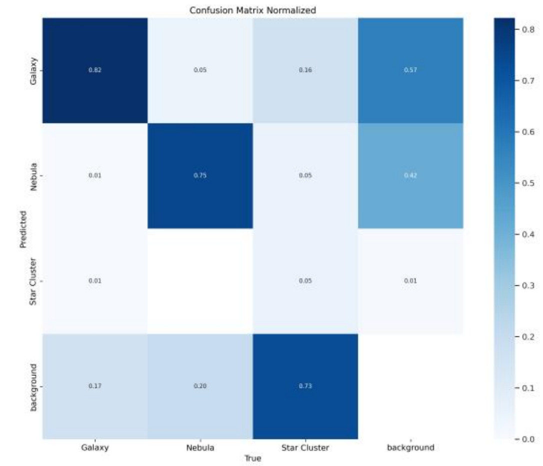


Figure 6. Confusion Matrix for validation – Model YOLOv8 fine-tuned with 10 epochs.

The F1-confidence curve **[Fig.7]** illustrates the relationship between the confidence threshold and the F1 score for different classes of celestial objects (Galaxy, Nebula, and Star Cluster) and for all classes combined.

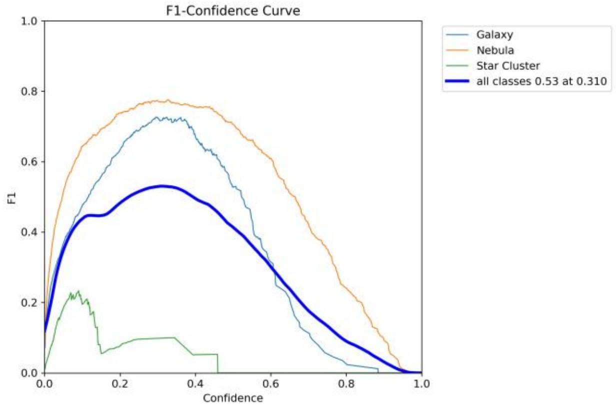


Figure 7. F1-Confidence Curve – Model YOLOv8 fine-tuned with 10 epochs.

- Galaxy (blue curve): The F1 score peaks at a relatively high value before gradually decreasing, indicating good detection performance at an optimal confidence threshold.

- Nebula (orange curve): The F1 score remains high over a broad range of confidence thresholds, showing the model's robustness in detecting nebulae.
- Star Cluster (green curve): The F1 score is lower and more variable, suggesting difficulty in accurately detecting star clusters.
- Overall Performance (bold blue curve): The overall F1 score peaks at 0.53 at a confidence threshold of 0.31, indicating the best balance between precision and recall for the combined classes.

The model performs well in detecting galaxies and nebulae but struggles with star clusters. The optimal confidence threshold for the best overall F1 score is around 0.31. This can guide fine-tuning and confidence threshold adjustments for improved detection accuracy.

The labels correlogram **[Fig.8]** visualizes relationships between bounding box attributes (x, y, width, height) for detected objects.

- x and y Coordinates: Uniform distribution indicates objects are spread throughout images. Histograms show balanced placement of objects across the image.
- Width and Height: Concentrated scatter plots suggest most objects have small bounding boxes. Histograms reveal smaller objects are more common.
- Inter-attribute Relationships: Width and height scatter plots show a triangular distribution, indicating variability in object shapes. Other pairwise plots show no strong correlation between object position and size.

The correlogram shows objects are evenly distributed with a tendency towards smaller sizes, providing insights into dataset characteristics for model tuning.

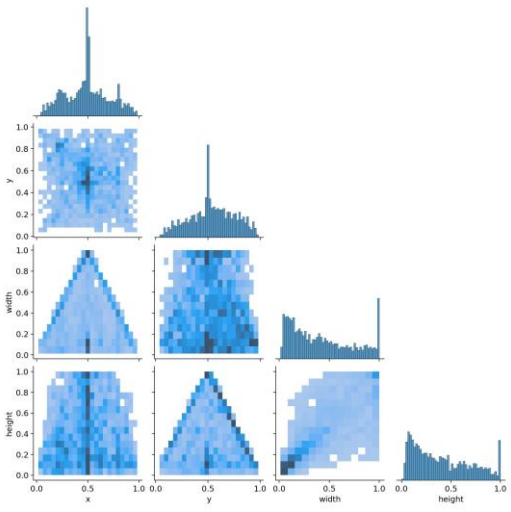


Figure 8. Labels Correlogram – Model YOLOv8 fine-tuned with 10 epochs.

Finally , as observed for all three types of losses **[Fig. 9]**, the training and validation losses decrease over time, indicating no signs of overfitting or underfitting. This trend is accompanied by an improvement in recall, further validating the model's performance.

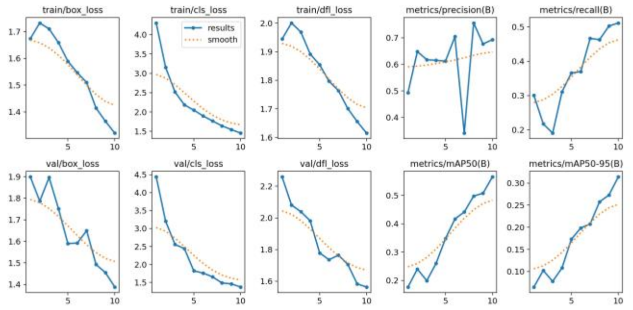


Figure 9. Training Results – Model YOLOv8 fine-tuned with 10 epochs.

Examples of bounding boxes and classification from training are showcased in **[Fig.10]**.

The performance of our model on the test dataset yielded a precision of **0.688**, a recall of **0.716**, and an F1 score of **0.702**. These metrics indicate a good balance between precision and recall, reflecting the model's overall effectiveness in detecting and classifying celestial objects.

Compared to the training phase, the test results meet expectations, demonstrating no signs of overfitting or underfitting, as evidenced by the decreasing training/validation losses and improving recall. However, the model's performance in distinguishing between galaxies, star clusters, and background requires enhancement **[Fig.11]**. Future work should focus on addressing these specific misclassifications to further refine and improve the model's accuracy.

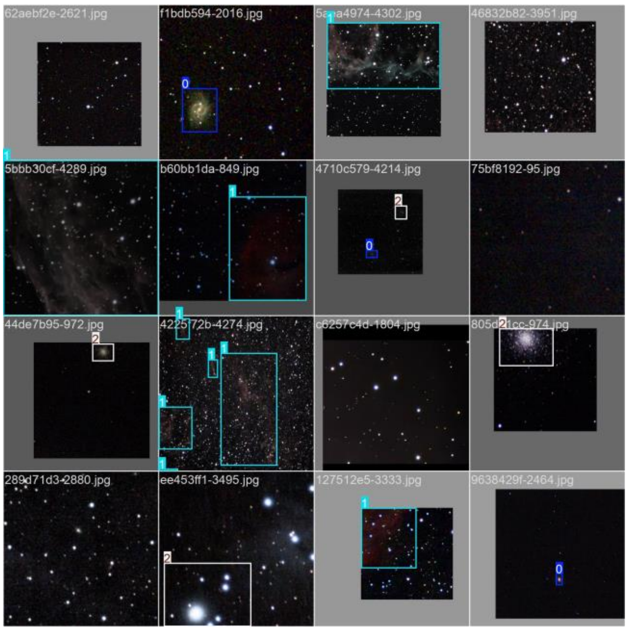


Figure 10. Labels Correlogram – Model YOLOv8 fine-tuned with 10 epochs.

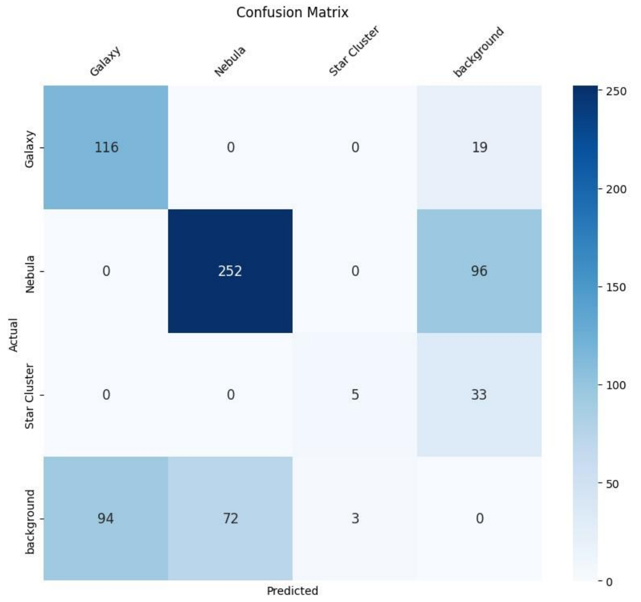


Figure 11. Confusion Matrix | Test Results – Model YOLOv8 fine-tuned with 10 epochs.

After fine-tuning the model for 20 epochs **[Fig.12, Fig.13, Fig.14, Fig.15]** on top of the pretrained weights, the performance on the test dataset improved significantly. The model achieved a precision of **0.727**, a recall of **0.814**, and an F1 score of **0.768**. These metrics indicate a better balance between precision and recall, reflecting enhanced accuracy in detecting and classifying celestial objects.

The results from the 20-epoch training follow a similar pattern to the 10-epoch training, demonstrating continued improvement in performance metrics. This suggests that further training could potentially yield even better results, as the model continues to learn and refine its detection capabilities with more epochs.

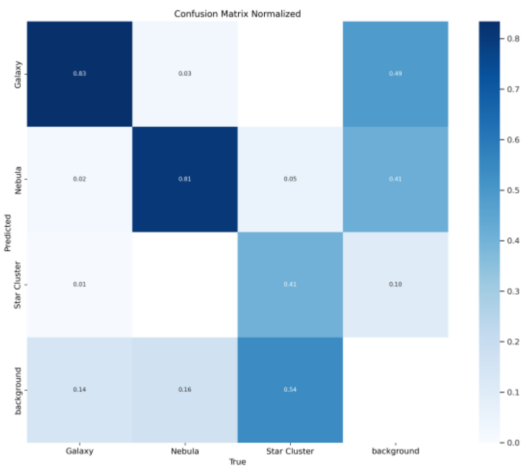


Figure 12. Confusion Matrix for validation – Model YOLOv8 fine-tuned with 20 epochs.

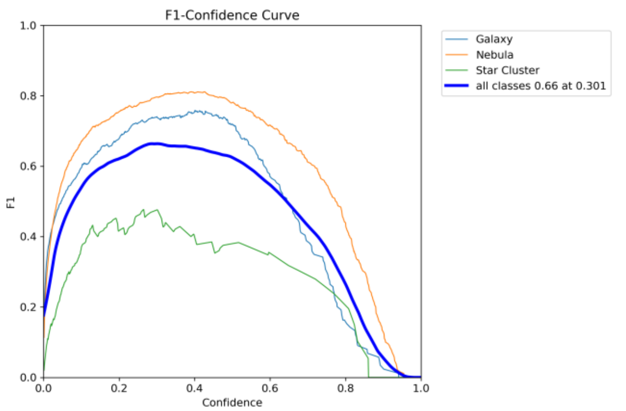


Figure 13. F1-Confidence Curve – Model YOLOv8 fine-tuned with 20 epochs.

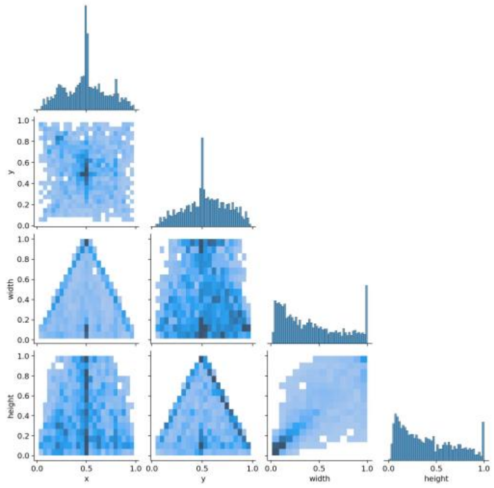


Figure 14. Labels Correlogram – Model YOLOv8 fine-tuned with 20 epochs.

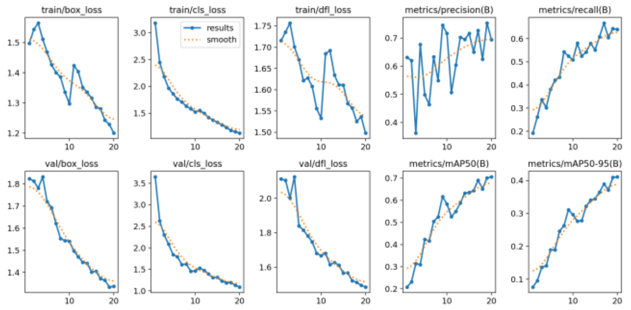


Figure 15. Training results – Model YOLOv8 fine-tuned with 20 epochs.

The confusion matrix of the test results **[Fig.16]** shows that while the model performs well in classifying nebulae, there is a tendency to misclassify galaxies and star clusters as background. This suggests that while the model has learned to identify nebulae robustly, it needs further refinement to improve the detection of galaxies and star clusters.

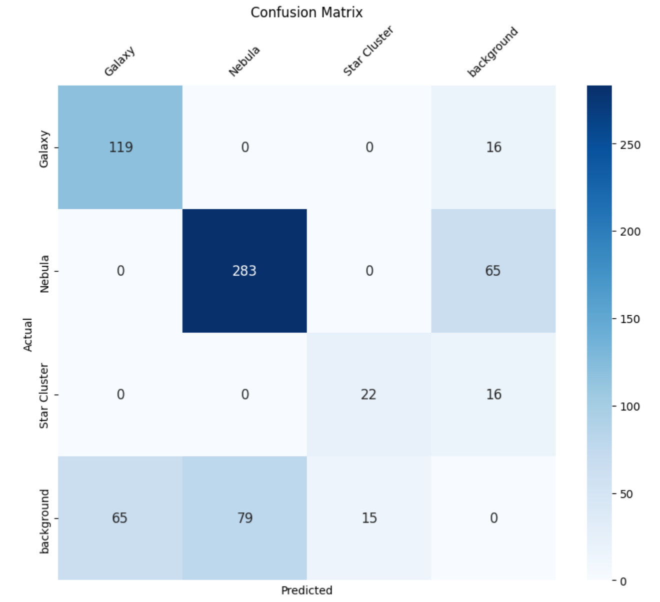


Figure 16. Confusion Matrix | Test Results – Model YOLOv8 fine-tuned with 20 epochs.

B. *Pretrained Models – Faster R-CNN ResNet-50*

Faster R-CNN (Region-based Convolutional Neural Network) is a state-of-the-art object detection model designed to identify and classify objects within images **[Fig.17]**. The model architecture consists of several key components:

- **Backbone Network:** In this case, we use ResNet-50 as the backbone for feature extraction. ResNet-50 is a deep convolutional neural network that captures detailed spatial hierarchies from images.
- **Region Proposal Network (RPN):** This network proposes candidate object bounding boxes (regions) within the image.
- **ROI Pooling:** The Region of Interest (ROI) pooling layer extracts fixed-size feature maps from the proposed regions.
- **Fully Connected Layers:** These layers perform object classification and bounding box regression.

Faster R-CNN operates in two stages:

- The RPN proposes regions where objects might be located.
- The second stage classifies the proposed regions and refines their bounding boxes.

The model was pretrained on the COCO dataset, which includes 80 object classes. This pretraining provides a strong initialization, enabling faster convergence when fine-tuned on custom datasets.

To adapt the pretrained Faster R-CNN model for our task, we initialized the Faster R-CNN model with a ResNet-50 backbone, modifying the final classification layer to match the number of classes in our dataset (Galaxy, Nebula, Star Cluster, and background).

The dataset was loaded using a custom CocoDataset class, which handles image and annotation loading. We defined data transformations, including random horizontal flips and tensor conversions, to augment the training data.

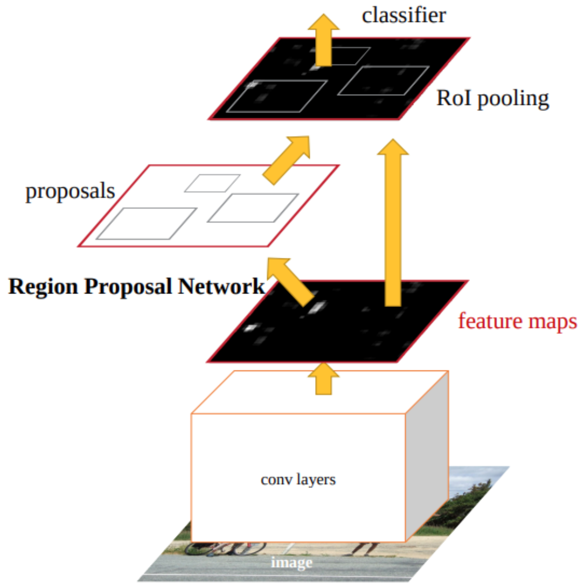


Figure 17. Faster R-CNN is a single, unified network for object detection. The RPN (Region Proposal Network) module serves as the `attention` of this unified network.

The model was trained using stochastic gradient descent (SGD) with a learning rate of 0.009 and weight decay of 0.0005. We utilized a learning rate scheduler to reduce the learning rate by half every two epochs. The training loop included the following key steps:

- Loading the training dataset and initializing the data loader.
- **Forward pass:** Feeding images through the model to obtain predictions.
- **Backward pass:** Calculating the loss and updating the model weights.
- Logging the training loss and other metrics for analysis.

Similar to the YOLOv8 methodology, we further fine-tuned the pretrained model with 10 and 20 epochs. Split strategy was the same.

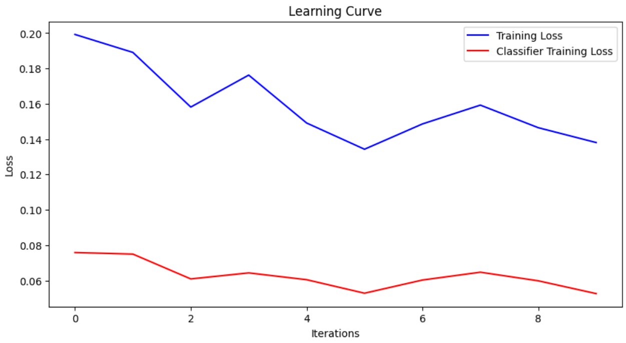


Figure 18. Training Loss – Model Faster R-CNN fine-tuned with 10 epochs.

The results of the test dataset demonstrated a box accuracy of 0.95 which was calculated using the Intersection over Union (IoU) metric. This high accuracy indicates that when the model predicts bounding boxes, they closely align with the ground truth boxes, despite the classification challenges.

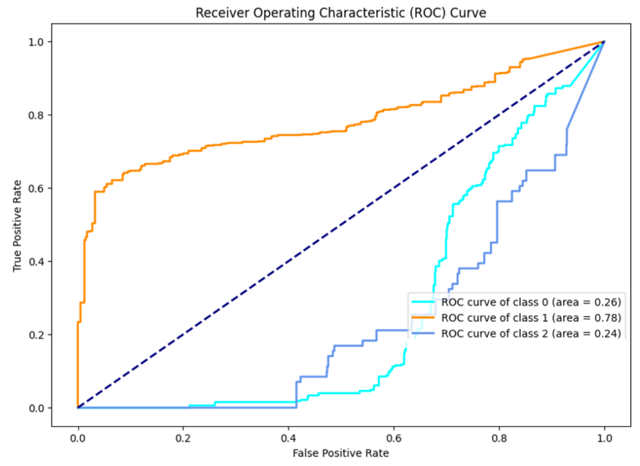


Figure 19. Training Loss – Model Faster R-CNN fine-tuned with 10 epochs.

The ROC curve **[Fig.19]** provides a visual representation of the true positive rate (TPR) versus the false positive rate (FPR) for different classes:

- Galaxy: The area under the curve (AUC) is 0.26, indicating poor performance in distinguishing galaxies.
- Nebula: The AUC is 0.78, showing good performance in detecting nebulae.
- Star Cluster: The AUC is 0.24, indicating significant challenges in identifying star clusters.

The classification report showcases a similar pattern with the ROC curve.

Classification Report	**Metrics**			
	precision	***recall***	***f1-score***	***support***
Galaxy	0.44	0.12	0.19	329
Nebula	0.69	0.87	0.77	697
Star Cluster	0.10	0.18	0.13	71
accuracy			0.60	1097
macro avg	0.31	0.29	0.26	1097
weighted avg	0.58	0.60	0.55	1097

Figure.20 Classiciation Report – Test results – Faster R-CNN (10 epochs)

The Faster R-CNN ResNet50 model shows strong performance in detecting nebulae but faces significant challenges with galaxies and star clusters. The ROC curves and classification metrics highlight the need for further refinement, particularly for improving the detection of galaxies and star clusters. The high box accuracy suggests that the localization of objects is precise, but the classification of these objects requires additional attention and possible adjustments to the model or training process.

The results of our second fine-tuning with 20 epochs showcased similar behavior and results.

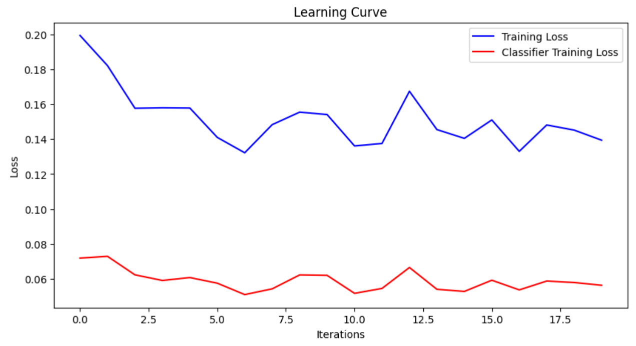


Figure 20. Training Loss – Model Faster R-CNN fine-tuned with 10 epochs.

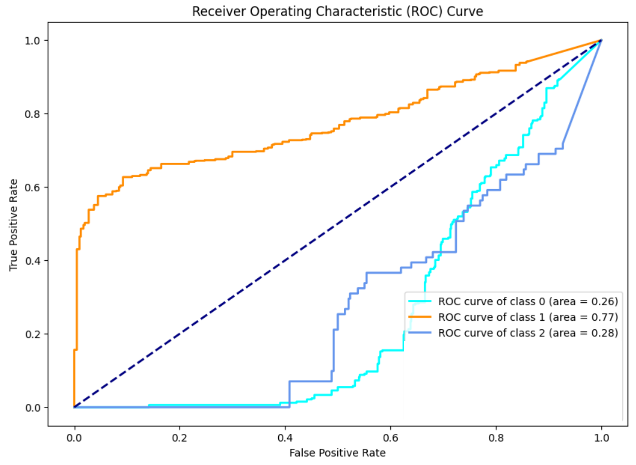


Figure 21. Training Loss – Model Faster R-CNN fine-tuned with 10 epochs.

Classification Report	**Metrics**			
	precision	***recall***	***f1-score***	***support***
Galaxy	0.37	0.11	0.17	329
Nebula	0.68	0.83	0.75	697
Star Cluster	0.12	0.24	0.16	71
accuracy			0.58	1097
macro avg	0.39	0.39	0.36	1097
weighted avg	0.55	0.58	0.53	1097

Figure.22 Classiciation Report – Test results – Faster R-CNN (20 epochs)

C. *Faster R-CNN ResNet-50*

In our final approach, we opted to develop and train a network from the ground up. Our choice was to implement the Yolo v3 architecture, known for its robustness and efficiency in object detection tasks **[6]**.

The YOLO (You Only Look Once) framework is renowned for its speed compared to other state-of-the-art models such as Faster R-CNN. YOLO innovates by integrating feature extraction and object localization within a single block. Moreover, its architecture unites the localization and classification heads, deviating from the three-part structure used in Faster R-CNN **[7]**.

The concept underlying YOLO is that it eliminates the need for classification/detection modules to synchronize with each other and avoids recurring region proposal loops found in earlier two-stage detectors. It primarily relies on convolutions (with occasional max-pooling layers). Instead of extracting areas with high object probability and passing them to a separate network for box detection, YOLO employs a unified monolithic network responsible for feature extraction, box regression, and classification tasks. In contrast to previous models, which typically had two output layers—one for class probability distribution and another for box predictions—YOLO integrates these functions within its single network architecture.

Taking inspiration from ResNet and FPN (Feature Pyramid Network) designs, YOLO-V3 incorporates a feature extractor named Darknet-53, distinguished by its 52 convolutional layers. It integrates skip connections akin to ResNet and features 3 prediction heads akin to FPN, each handling image data at various spatial compressions **[Fig.23]**.

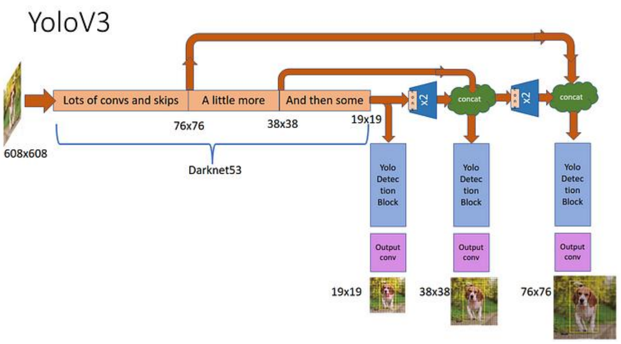


Figure 23. YoloV3 – High Level Architecture.

The Feature Pyramid Network (FPN), introduced by FAIR (Facebook A.I. Research) in 2017, is a topology where the feature map initially decreases in spatial dimension, as typical. However, it later expands and is concatenated with previous feature maps of corresponding sizes. This iterative process continues, with each concatenated feature map being fed into a distinct detection head.

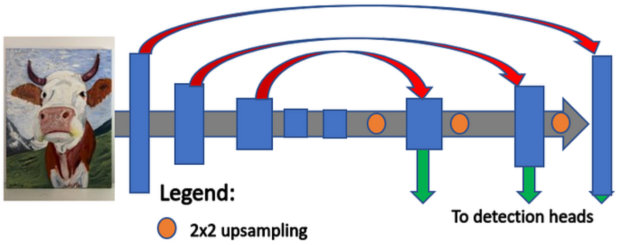
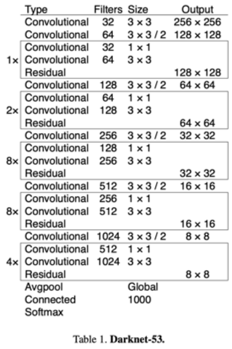


Figure 24. Classification heads illustration.

The YOLO network adopts a new backbone architecture in comparison to YOLOv2, known as Darknet-53. It combines elements of Darknet-19 with residual blocks and shortcut connections, creating a hybrid architecture that enhances performance and efficiency. The complete architecture is illustrated in **[Fig.25]**.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Table 1. **Darknet-53.**

Figure 25. YOLOv3 Backbone.

In YOLO (You Only Look Once), anchors refer to predefined bounding boxes of a fixed size and aspect ratio. These anchors are placed throughout the input image at various positions and scales. During training, YOLO predicts offsets to these anchor boxes rather than predicting absolute bounding box coordinates directly. This approach allows YOLO to efficiently handle object detection by focusing on predicting offsets relative to a set of anchor boxes, which helps in handling objects of different sizes and aspect ratios effectively.

Each cell in the output layer's feature map predicts 3 boxes in the case of Yolo-V3 one box per anchor. Each box prediction consists of:

- 2 values for box center offsets(in x a y, relative to cell center)
- 2 values box size scales (in x and y, relative to anchor dimensions)
- 1 value for objectness score (between 0 and 1), number-of-classes values for class score (between 0 and 1).

For each spatial cell, for each box prediction in centered in that cell, the loss function finds the box with the best IoU with the object centered in that cell. This distinguishing mechanism between best boxes and all the other boxes is in the heart of the YOLO loss.

The bounding boxes are predicted at three different points in this network and on three different scales or grid sizes. The idea behind this approach is that the small objects will get easily detected on smaller grids and large objects will be detected on larger grid. In our implementation we used [19, 32, 78] as grid sizes with images of size 608 x 608 pixels.

Our network implementation comprises three primary building blocks:

- The convolution block, which includes a convolutional layer followed by batch normalization and leaky ReLU activation.
- The residual block, composed of stacked convolution blocks that progressively reduce input size by a factor of 2 each time, featuring residual/skip connections.

Additionally, we incorporated the scale block, which consists of two convolutional layers separated by batch normalization and leaky ReLU activation. This block outputs predictions (image and boxes with labels) at different scales. The output is organized into grids, resulting in a shape of (batch size, 3, grid size, grid size, num classes + 5), where 3 represents the number of anchors.

Finally, we constructed the YOLOv3 network using the three aforementioned building blocks as specified in the YOLOv3 paper.

For the training loss, we implemented a composite loss function described in the paper. This loss function combines four components:

- **No object loss:** Penalizes confidence predictions for background regions.
- **Object loss:** Penalizes confidence predictions for object detection.
- **Box coordinate loss:** Penalizes errors in bounding box coordinates.
- **Class loss:** Penalizes errors in class predictions.

In **[Fig.26]** is a plot showing the average of these combined losses across epochs. It demonstrates that the model generalizes effectively and fits well to the data.

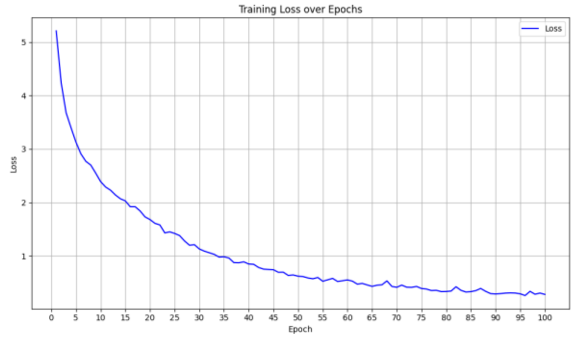


Figure 26. Training loss – YOLOv3.

Intersection over Union (IoU) is a metric used to evaluate the accuracy of bounding box predictions in object detection. It measures the overlap between a predicted bounding box and a ground truth bounding box. IoU is calculated as the ratio of the intersection area (overlap) to the union area (combined area). A higher IoU indicates better alignment between the predicted and ground truth bounding boxes, with IoU values ranging from 0 (no overlap) to 1 (perfect overlap). Typically, IoU thresholds like 0.5 are used to determine if a prediction is considered correct in many object detection applications.

After making bounding box predictions, we applied Non-Maximum Suppression (NMS), a technique used in object detection to select the most confident and accurate bounding boxes while suppressing overlapping ones. NMS works by sorting the bounding boxes based on their confidence scores and then iteratively selecting the boxes with the highest scores. During this process, boxes that have high overlap (measured by IoU) with previously selected boxes are suppressed, leaving behind the most relevant detections.

After applying NMS, there were still numerous overlapping bounding boxes that did not meet the IoU threshold of 80% overlap or more. Therefore, to address highly overlapping boxes with the same label, we retained only one box per group. All remaining bounding boxes had a confidence score of 90% or higher.

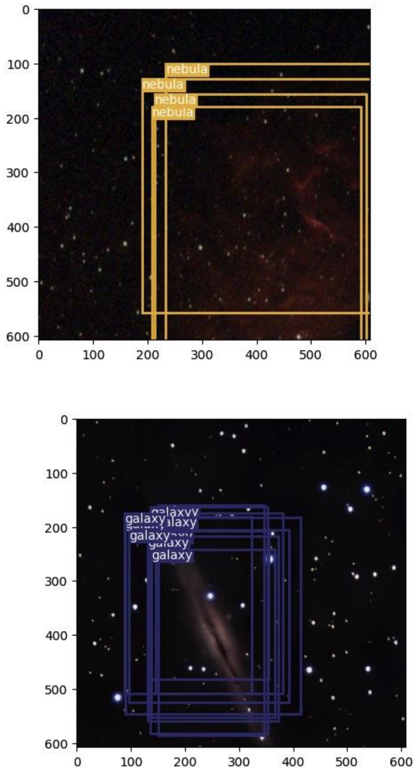


Figure 27. Images with unfiltered overlapping boxes.

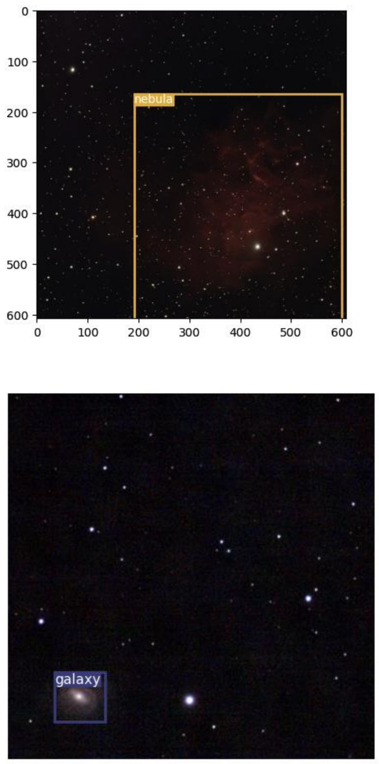


Figure 28. Images with single kept bounding box.

Following the approach described in the YOLOv3 paper, our model generates predictions using smaller cells across the image, assigning bounding boxes and predictions to each cell. To reconstruct the final bounding boxes, we concatenate predictions from cells containing detections.

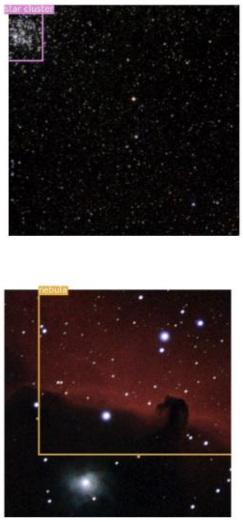


Figure 29. Sample predictions from unseen images.

Anchors used:

- [(0.28, 0.22), (0.38, 0.48), (0.9, 0.78)]
- [(0.07, 0.15), (0.15, 0.11), (0.14, 0.29)]
- [(0.02, 0.03), (0.04, 0.07), (0.08, 0.06)]

Trained for 100 epochs with a batch size of 32 and learning rate of 0.0001. All images had three RGB channels, and ground truths were annotated across three scales of the image.

For train transform, the goal is to enhance model robustness through diverse data augmentation techniques. Images are resized to image size while maintaining aspect ratio, and padding ensures they are exactly image size x image size. Color jittering introduces random variations in brightness, contrast, saturation, and hue, improving the model's adaptability to different lighting conditions. Horizontal flipping further diversifies training data by randomly mirroring images. Normalization sets pixel values with a mean of 0 and a standard deviation of 1, converting images into PyTorch tensors. Bounding boxes, if present, adhere to YOLO format for consistent annotation handling.

In contrast, test transform focuses on standardizing evaluation conditions. Images undergo resizing and padding identical to train transform to maintain uniform input dimensions. Unlike training, it skips augmentation to evaluate the model's performance on unaltered data. Normalization and tensor conversion ensure compatibility with model input requirements. Bounding box parameters remain consistent with training settings to ensure accurate object localization during evaluation. This approach balances training diversity with consistent evaluation conditions, essential for developing and testing robust computer vision models effectively.

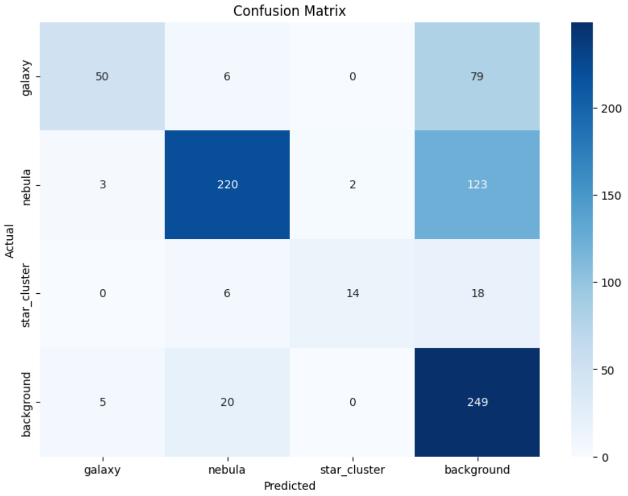


Figure 30. Confusion matrix on test data – YOLOv3.

Classification Report	**Metrics**			
	precision	***recall***	***f1-score***	***support***
Galaxy	0.86	0.37	0.52	135
Nebula	0.87	0.63	0.73	348
Star Cluster	0.88	0.37	0.52	38
Background	0.53	0.91	0.67	274
accuracy			0.67	795
macro avg	0.79	0.57	0.61	795
weighted avg	0.75	0.67	0.66	795

Figure.31 Classiciation Report – Test results – YOLOv3 (100 epochs)

IV. Results

Model	***YOLOv8 (10ep)***	***YOLOv8 (20ep)***	***FRCNN (10ep)***	***FRCNN (20ep)***	***YOLOv3 (100ep)***
F1-Score	0.702	0.814	0.26	0.36	0.61
Training Time (A100)	~28m	~45m	~1h	~2h	~8h

Figure.32 Classiciation Report – Test results – YOLOv3 (100 epochs)

We selected the F1 score to assess performance across three classes and evaluate how well each model manages class imbalance.

As observed, the top performer is the YOLOv8 model, which achieved superior results due to its state-of-the-art and updated architecture. It demonstrated satisfactory performance with just 10 epochs, leveraging its fast inference capabilities and benefiting from pretrained weights focused on object detection tasks. YOLOv3, while a strong contender, didn't match YOLOv8's performance, partly because it stssarts training from scratch and utilizes an older architecture. This highlights YOLO's efficiency in capturing features effectively even with a smaller dataset of around 4000 images and limited epochs. In contrast, Faster RCNN struggled to achieve comparable results, likely due to its complexity, suggesting it may require more extensive training data and fine-tuning epochs to optimize performance effectively.

Additionally, the scores indicate that the primary challenge for the models lies in handling class imbalance, particularly evident in Faster RCNN. This suggests that further fine-tuning and additional data could potentially enhance their performance significantly.

As a future improvement, leveraging transfer learning and utilizing larger datasets could enhance our ability to capture the distinct characteristics of each class more effectively. Additionally, exploring more complex architectures could further refine the model's capabilities and improve overall performance.

V. Conclusions

In summary, our study demonstrates that solving the task of object detection and accurately identifying stellar objects with their distinct and complex characteristics is highly achievable using a medium-sized dataset. By leveraging pretrained models for optimal performance or training new models with satisfactory results, we've identified areas for potential improvement. These advancements can significantly aid in automating stellar observations, providing valuable tools for astrophotography and advancing astronomy research.

References

[1] Olivier Parisot, “DeepSpaceYoloDataset: Annotated Astronomical Images Captured with Smart Telescopes”, Luxembourg Institute of Science and Technology, January 2024.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, arXiv:1506.01497, June 2015.

[3] González, R. E. and Muñoz, R. P., “AstroCV - A computer Vision Library for Astronomy”, in Astronomical Data Analysis Software and Systems XXVII, 2020, vol. 522, p. 425.

[4] Yuchen He, Jingjing Wu, Wenyu Wang, Bin Jiang, Yanxia Zhang, AstroYOLO: A hybrid CNN–Transformer deep-learning object-detection model for blue horizontal-branch stars, Publications of the Astronomical Society of Japan, Volume 75, Issue 6, December 2023, Pages 1311–1323, https://doi.org/10.1093/pasj/psad071

[5] R.E. González, R.P. Muñoz, C.A. Hernández, Galaxy detection and identification using deep learning and data augmentation, Astronomy and Computing, Volume 25, 2018, Pages 103-109,ISSN 2213-1337, https://doi.org/10.1016/j.ascom.2018.09.004.

[6] Joseph Redmon, Ali Farhadi, "YOLOv3: An Incremental Improvement", University of Washington, April 2018

[7] YOLO V3 Explained