# STELLAR MAPPING

Sivakumar Ramakrishnan
sivakumar.ramakrishnan@colorado.edu
University of Colorado
Boulder, USA

Rahul Prasanna
rahul.prasanna@colorado.edu
University of Colorado
Boulder, USA

Sai Nandini Tata
sai.tata@colorado.edu
University of Colorado
Boulder, USA

## Abstract

Constellations, patterns of stars forming recognizable shapes, have held significance throughout history. They aided navigation and were deeply woven into myths and legends. However, light pollution and reduced stargazing have diminished public knowledge of constellations. This project aims to develop a model using data science techniques to automatically identify constellations in astronomical images. We hope to reignite interest in astronomy and make learning about stars and navigation easier. This model also has educational potential for planetariums, science centers, and even professional astronomers. Additionally, the model tackles common challenges like partial visibility, varying image quality, and changing observation conditions. By solving these problems, the project aspires to improve existing constellation recognition systems, making astronomy more accessible and engaging.

## Keywords

Astronomical Image Processing, CNN, Computer Vision, Constellation Detection, Deep Learning, Digital Planetarium Systems, EfficientNet, Ensemble Learning, Multi-label Classification, Vision Transformers

## 1 Introduction

Throughout human history, constellations have served as celestial landmarks, guiding navigation, marking seasonal changes, and enriching cultural narratives across civilizations. These star patterns, recognized and named by ancient astronomers, have been crucial tools for understanding our position in both space and time. However, in our modern era, increasing urbanization and light pollution have significantly diminished public engagement with the night sky, leading to a declining awareness and recognition of these celestial patterns. This disconnect from celestial observation poses several challenges. First, it impacts astronomy education, making it harder for students and enthusiasts to learn about and identify constellations. Second, it affects cultural preservation, as traditional knowledge about constellations and their significance risks being lost. Third, it creates barriers for amateur astronomers and researchers who rely on constellation recognition for celestial navigation and object location. To address these challenges, we propose an automated constellation detection system that leverages recent advances in deep learning and computer vision. Our system aims to bridge the gap between traditional astronomical knowledge and modern technology, making constellation identification more accessible and engaging. The system processes astronomical images through an ensemble of deep learning models, combining Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs) to achieve robust constellation recognition under varying observational conditions.

## 2 Related Work

### 2.1 Deep Learning in Astronomy

Deep learning applications in astronomy have seen significant growth in recent years, particularly in image analysis and pattern recognition tasks. Dai and Kocevski [**?** ] pioneered the use of convolutional neural networks for galaxy morphology classification, demonstrating the potential of deep learning in astronomical image processing. Their work established foundational techniques for handling the unique challenges of astronomical imagery, including varying light conditions and complex pattern recognition. Recent advances in deep learning architectures have further expanded the possibilities for astronomical applications. The emergence of transformer-based models, as demonstrated by Dosovitskiy et al. [**?** ], has opened new avenues for processing astronomical data with improved attention to spatial relationships and global feature detection. These advances have proven particularly relevant for constellation detection, where understanding spatial relationships between stars is crucial.

### 2.2 Constellation Detection Methods

Traditional constellation detection methods have primarily relied on geometric pattern matching and star brightness analysis. However, modern approaches have increasingly incorporated machine learning techniques. Simonyan and Zisserman [**?** ] introduced deep convolutional networks that proved effective in handling the complex patterns characteristic of constellation recognition. Their work with VGG networks established important principles for feature extraction in astronomical imagery. The field has evolved to address several key challenges:

- **Variable Star Brightness:** Methods for normalizing and accounting for varying stellar magnitudes
- **Pattern Orientation:** Techniques for rotation-invariant pattern recognition
- **Atmospheric Distortion:** Approaches to handle atmospheric effects and light pollution
- **Multi-Scale Detection:** Solutions for identifying constellations at different scales and distances

### 2.3 Vision Transformers and CNNs

The integration of Vision Transformers (ViT) with traditional CNN architectures has marked a significant advancement in image recognition tasks. Dosovitskiy et al. [**?** ] demonstrated that transformer architectures can achieve state-of-the-art performance in image recognition tasks by treating images as sequences of patches. This approach has particular relevance for constellation detection, where understanding both local features (individual stars) and global patterns (constellation shapes) is crucial. Key developments in this area include:

- **Attention Mechanisms:** Enhanced ability to focus on relevant stellar patterns while suppressing noise
- **Hybrid Architectures:** Combination of CNN's local feature detection with ViT's global pattern recognition
- **Transfer Learning:** Adaptation of pre-trained models for astronomical applications

Recent work by Liu et al. [**?** ] on hierarchical vision transformers has further improved the ability to handle multi-scale features, which is particularly relevant for constellation detection across varying field depths and scales.

## 2.4 Multi-label Classification Approaches

The challenge of multi-label classification in astronomical imaging presents unique complexities, as multiple constellations often appear in a single image. Recent advances in this area, particularly the work of He et al. [**?** ] on deep residual networks, have provided frameworks for handling such complex classification tasks. Current approaches focus on several key aspects:

- **Label Dependencies:** Understanding and leveraging relationships between different constellation labels
- **Class Imbalance:** Handling varying frequencies of different constellation appearances
- **Threshold Optimization:** Determining optimal confidence thresholds for multi-label predictions
- **Performance Metrics:** Developing appropriate evaluation metrics for multi-label scenarios

The work of Tan and Le [**?** ] on EfficientNet has been particularly influential in developing efficient architectures for multi-label classification, providing methods to balance model size and performance. Their compound scaling approach has proven effective in handling the complexities of astronomical image classification. These developments in deep learning, combined with domain-specific knowledge of astronomical imaging, form the foundation of our approach to constellation detection. Our work builds upon these advances while introducing novel elements to address the specific challenges of constellation recognition in varying observational conditions.

# 3 Data Collection and Preprocessing

## 3.1 Dataset Description

Our project utilizes multiple datasets to ensure comprehensive coverage of constellation patterns under various conditions. The datasets were carefully selected and combined to provide a robust foundation for training our models.

*3.1.1 Roboflow Universe Dataset.* The primary dataset for this project comes from Roboflow Universe, consisting of approximately 2,000 labeled images of various constellations. Key characteristics include:

- Detailed annotations for individual stars
- Varying image quality and resolution to reflect real-world conditions
- Multiple constellations per image, supporting multi-label classification
- Diverse observational conditions and equipment types

*3.1.2 NOAA Milky Way Panorama.* The NOAA Milky Way Panorama dataset provides high-resolution images that include:

- Complete outlines of 88 major constellations
- Detailed views of constellations' spatial distribution
- Reference material for understanding stellar groupings
- High-quality baseline images for pattern recognition

*3.1.3 Additional Data Sources.* To enhance our dataset's diversity and robustness, we incorporated:

- Astronomy Online resources providing detailed constellation descriptions
- GitHub repository containing over 1,000 additional training images
- Practical insights from computer vision applications in astronomy

## 3.2 Data Analysis

*3.2.1 Class Distribution.* Our analysis of the dataset revealed several important characteristics:

- Total Images: 1,641 night sky images
- Constellation Classes: 16 distinct categories
- Labels per Image: Average of 2.99
- Total Annotations: 4,909 constellation annotations

The class distribution analysis showed varying frequencies:

- **Frequently Observed** (IR < 1.3):
  - Cassiopeia (1.00)
  - Pleiades (1.16)
  - Ursa Major (1.21)
  - Cygnus (1.27)
- **Moderately Represented** ($1.3 \leq$ IR < 2.0):
  - Lyra (1.28)
  - Moon (1.36)
  - Orion (1.46)
  - Bootes (1.96)
- **Under-represented** (IR $\geq$ 2.0):
  - Gemini (2.05)
  - Leo (2.31)
  - Canis Major (2.56)
  - Sagittarius (2.81)

*3.2.2 Image Characteristics.* All images in the dataset maintain consistent properties:

- Dimensions: 640×640 pixels
- Aspect Ratio: 1:1
- Color Space: RGB
- Format: Digital astronomical photographs

## 3.3 Preprocessing Pipeline

*3.3.1 Spatial Augmentations.* We implemented several spatial transformations to enhance model robustness:

- Random rotation (±15 degrees, p=0.5)
- Random 90-degree rotation (p=0.3)
- Random horizontal flip

*3.3.2 Intensity Augmentations.* Two parallel augmentation streams were employed with probabilistic application:

- **Stream 1** (p=0.3):
  - Gaussian noise (variance range: 10.0-50.0, p=0.5)
  - Gaussian blur (kernel size: 3-5 pixels, p=0.5)
- **Stream 2** (p=0.3):
  - Brightness and contrast adjustment (±20
  - Gamma correction (range: 80-120, p=0.5)

*3.3.3 Normalization Techniques.* Image normalization was performed using ImageNet statistics to facilitate transfer learning:

$$\text{mean} = [0.485, 0.456, 0.406] \quad \text{std} = [0.229, 0.224, 0.225] \tag{1}$$

## 3.4 Dataset splitting

The dataset was stratified into three sets:

- Training: 70%
- Validation: 15%
- Test: 15%
  To address the multi-label nature of the data, we employ an iterative stratification strategy [X] that considers co-occurrence patterns between constellations.

## 3.5 Augmentation Visualization

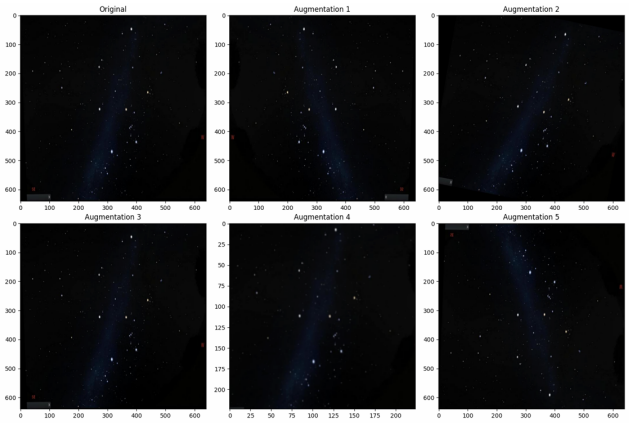



**Figure 1: Augmentation visualization demonstrating the effectiveness of our pre processing pipeline. The original image (top-left) showcases a clear constellation pattern against the night sky. Subsequent augmentations introduce variations in brightness and contrast (augmentations 1-2), geometric transformations that preserve celestial patterns (augmentation 3), random cropping with resolution adjustments (augmentation 4), and combined effects of multiple transformations (augmentation 5). These augmentations effectively preserve crucial astronomical features while introducing beneficial variations that enhance model training.**

This comprehensive preprocessing pipeline ensures robust model training and evaluation while addressing common challenges in astronomical image processing.

# 4 Visualizations

## 4.1 Distribution of Image Brightness

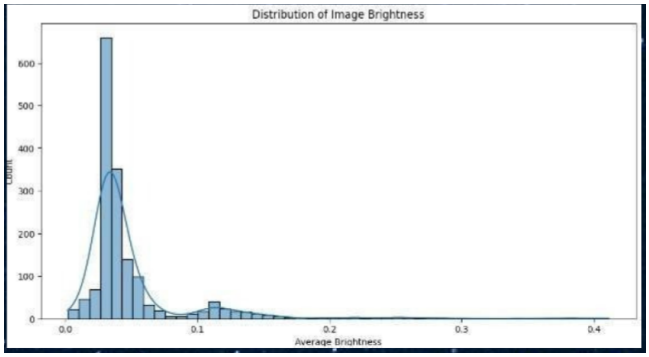



**Figure 2: A histogram of average pixel brightness for each image reveals whether the images are generally dark, bright, or have a good spread of brightness levels. This information helps determine if brightness normalization is necessary during preprocessing and can identify potential issues with overexposed or underexposed images.**

## 4.2 Color Channel Distributions

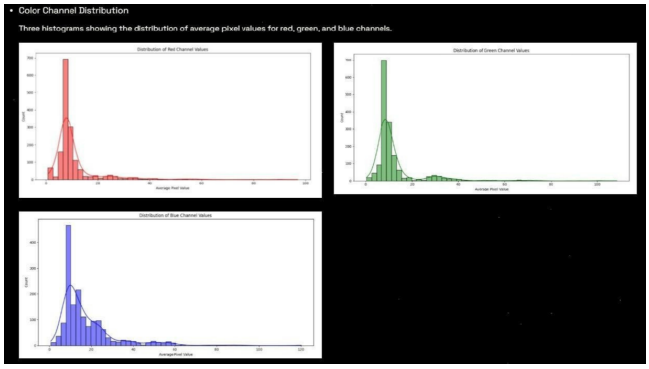



**Figure 3: Three histograms showing the distribution of average pixel values for red, green, and blue channels reveal color biases in the dataset (e.g., if images tend to be more red or blue). This information helps in deciding if color normalization or balancing is needed and can indicate issues with color consistency across the dataset.**

### 4.3 Sample Images

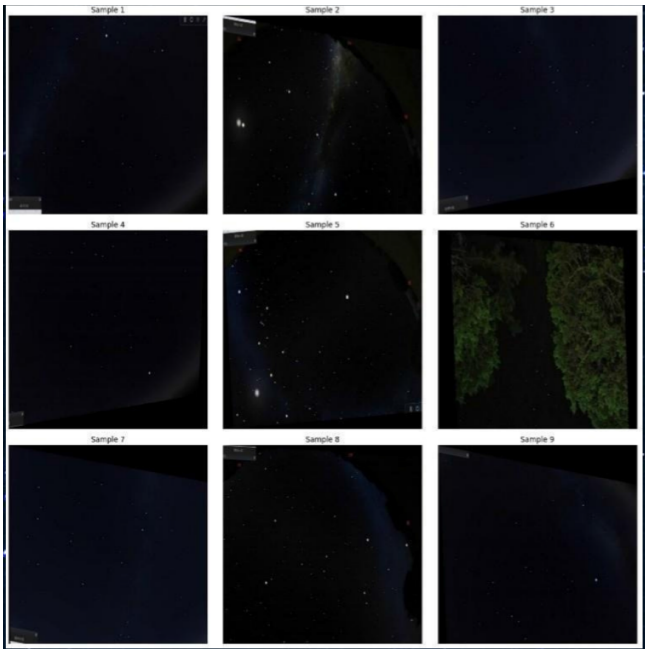



**Figure 4: A grid of 9 randomly selected images from the dataset provides a visual inspection of the types of images in the dataset. This helps in identifying any obvious issues with image quality or content and gives a sense of the variety in the dataset.**

### 4.4 Distribution of Image Complexity

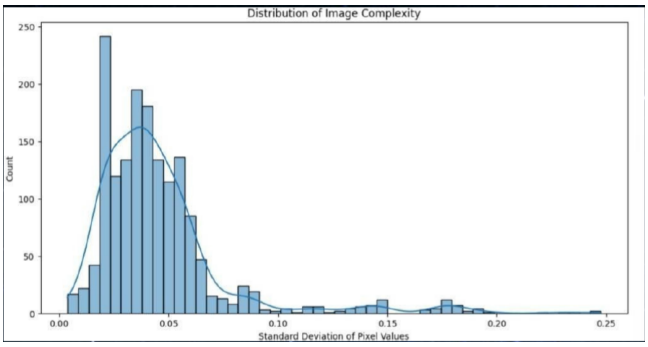



**Figure 5: A histogram of image complexity, measured by the standard deviation of pixel values, indicates the range of image complexities in the dataset. This helps identify if there's a good mix of simple and complex images and can guide decisions on data augmentation or model complexity.**

### 4.5 PCA Visualization of Images

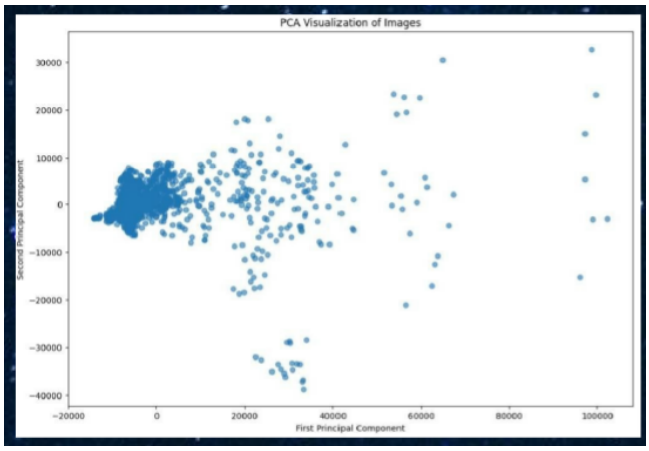



**Figure 6: A scatter plot of images projected onto their first two principal components reveals clusters or patterns in the dataset that might not be apparent from other plots. This can indicate if there are distinct groups of similar images and helps in understanding the overall structure and variability of the dataset.**

### 4.6 Image Statistics Distributions

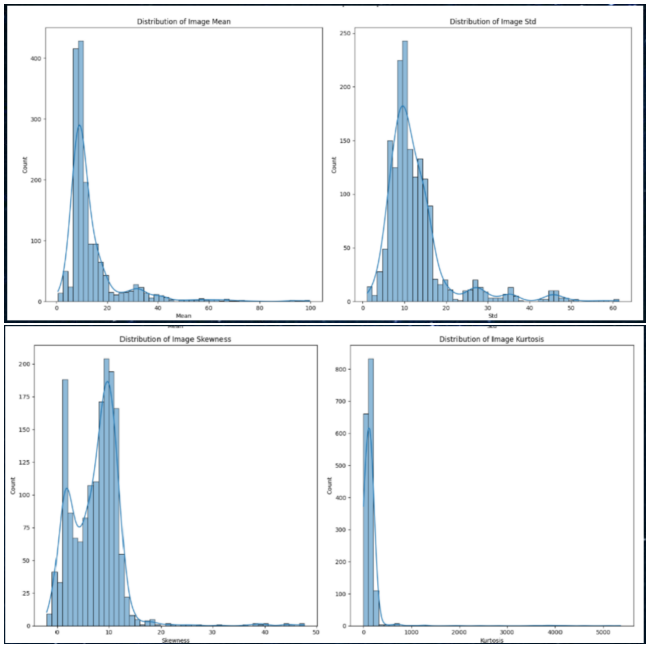



**Figure 7: Histograms of mean, standard deviation, skewness, and kurtosis of pixel values across all images reveal key dataset characteristics. These statistics help identify outliers, assess image brightness, contrast, and distribution, and guide preprocessing decisions.**

## 4.7 Image Count Distribution by Class

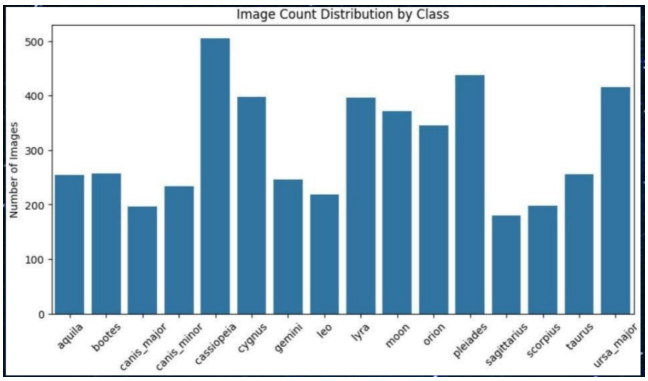



**Figure 8: Understanding the distribution of images across different classes is crucial for model training. Class imbalances, where one class has significantly more images than another, can lead to biased model performance. Cassiopeia dominates with approximately 500 instances, while Sagittarius represents the minority class with around 180 instances. A gradual decline in frequency is observed across other classes. This clear evidence of moderate class imbalance, with a max-min ratio of 2.81:1, necessitates careful consideration during model training and evaluation.**

## 4.8 Class-wise Mean and Variance of Pixel Intensity

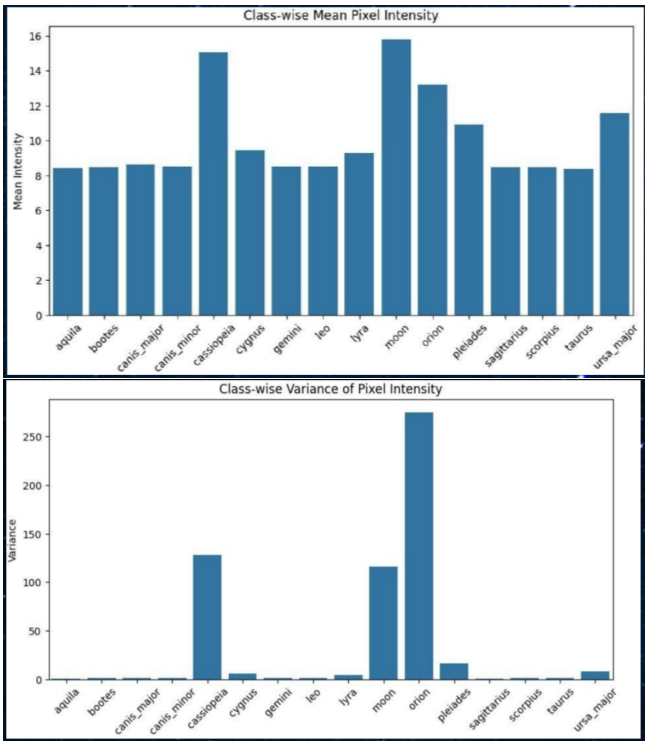



**Figure 9: Evaluating the average and variability of pixel intensities for each class can reveal differences in brightness or contrast between classes. This information can inform preprocessing strategies and help optimize model training.**

## 4.9 Constellation Mapping

```
In [ ]: fig = plt.figure(figsize=figsize)
        ax = plt.axes(projection=ccrs.PlateCarree(180))

        for index, row in const_names.iterrows():
            ax.text(row['ra']*360/24, row['dec'], row['name'],
                    transform=ccrs.Geodetic(), ha='left', va='top', fontsize=8, color=nonzodiac_color)

        ax.set_xlim(ax.get_xlim()[::-1])
        set_save_image(fig, './figures/constellation_names.pdf')
```

**Figure 10: The constellation map provides a visual representation of the relationships between different constellations. By visualizing the proximity and overlap of constellations, we can gain insights into their spatial distribution and potential correlations. This visualization can be useful for understanding the overall structure of the celestial sphere and identifying potential challenges in constellation recognition tasks, such as overlapping star patterns or ambiguous boundaries.**

## 4.10 Constellation Boundaries

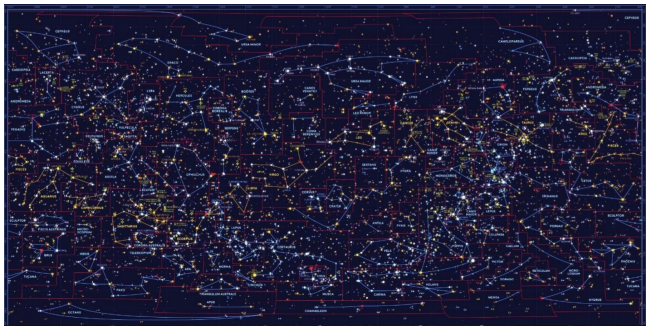


**Figure 11: This visualization provides a detailed view of the 88 officially recognized constellations, outlining their boundaries and highlighting their relative positions in the night sky. By clearly delineating the boundaries, this map helps to distinguish between different constellations and avoid confusion, especially in areas where star patterns overlap. This visualization is a valuable tool for stargazers and astronomers alike, aiding in the identification and study of celestial objects.**

## 4.11 Distribution of Labels

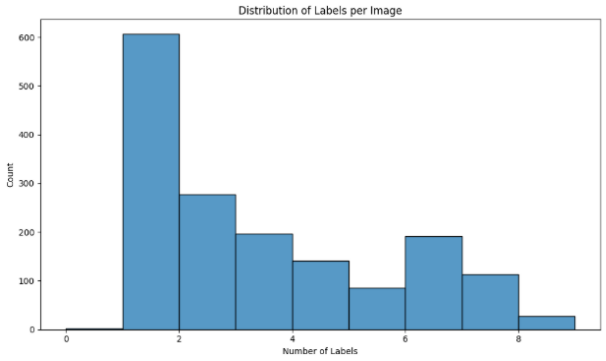



**Figure 12: Understanding the number of labels assigned to each image is crucial for designing the model architecture and training strategy. The multi-label nature of the dataset, with a range of 1 to 8 labels per image, necessitates careful consideration of label sparsity and ensuring effective representation of all label combinations. The modal value of 2 labels per image, with a right-skewed distribution and a median of approximately 3 labels, provides insights into the dataset's complexity and the potential challenges in accurately predicting multiple labels.**

## 4.12 Co-occurrence Analysis

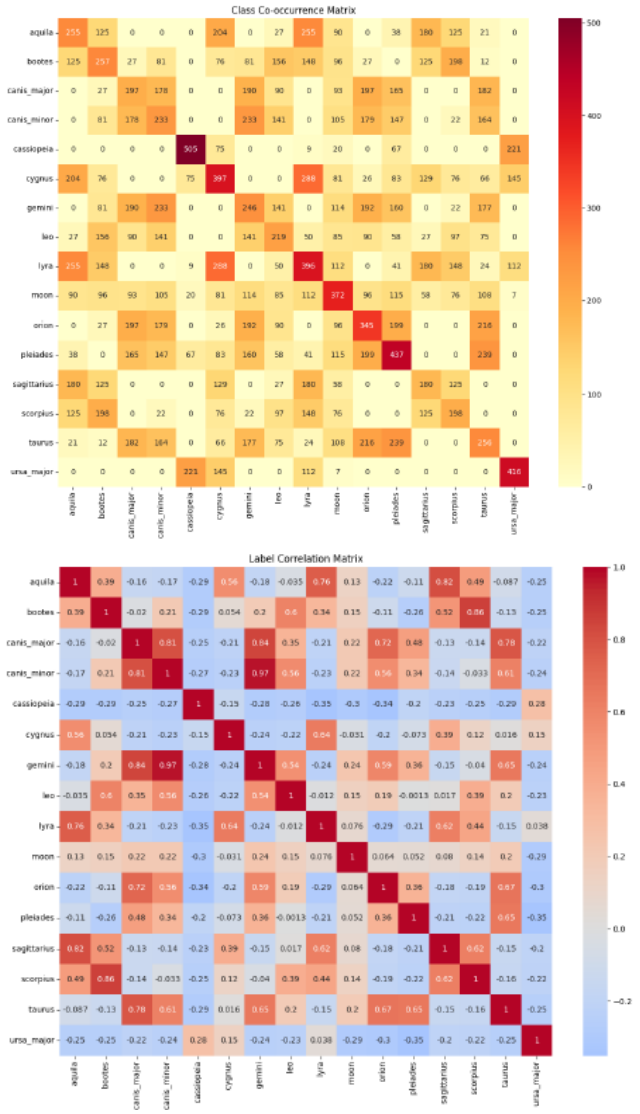



**Figure 13: The co-occurrence matrix reveals significant patterns in the relationships between constellations. Strong correlations are observed between Canis Major and Canis Minor (178 co-occurrences), Gemini and Canis Minor (233 co-occurrences), and Lyra and Cygnus (288 co-occurrences). Clusters also emerge, with the winter constellations (Orion, Taurus, Canis Major/Minor) grouping together and the summer constellations (Lyra, Cygnus, Aquila) forming another cluster. Additionally, isolated patterns are seen, such as Cassiopeia's limited co-occurrence, primarily with Ursa Major. Some constellations never appear together due to spatial and seasonal constraints.**

# 5 Model Architecture

In this section, we describe the four deep learning architectures developed for multi-label constellation classification: a modified CNN with spatial attention, Vision Transformer (ViT), an ensemble approach combining CNN-ViT, and EfficientNet.

## 5.1 CNN with Spatial Attention

We propose a CNN architecture that leverages transfer learning and incorporates spatial attention for improved feature extraction from astronomical imagery. The architecture, illustrated in Figure X, consists of four main components

*5.1.1 Feature Extraction.* Our model utilizes ResNet50 [31] as the backbone, pretrained on ImageNet [32]. We remove the final fully connected layer to extract feature maps that capture both low-level characteristics (edges, gradients) and high-level astronomical patterns. This modification enables the network to better adapt to the unique characteristics of constellation imagery.

*5.1.2 Spatial Attention Module.* To enhance the model's ability to focus on relevant regions of the night sky, we integrate a spatial attention mechanism. The module is defined as:

$$A = \sigma\left(\mathrm{BN}\left(\mathrm{Conv}\left(\mathrm{GAP}(F)\right)\right)\right) \tag{1}$$

$$F' = F \otimes A \tag{2}$$

where $F$ represents input features, GAP denotes global average pooling, Conv is a $7 \times 7$ convolutional layer, BN is batch normalization, $\sigma$ is the sigmoid activation, and $\otimes$ represents element-wise multiplication. This mechanism generates attention weights $A$ that highlight salient features while suppressing less informative regions.

*5.1.3 Classification Head.* The classification component employs a hierarchical structure:

(1) Initial dropout ($p = 0.5$) for regularization.
(2) Linear transformation: $\mathbb{R}^d \to \mathbb{R}^{512}$ with ReLU activation.
(3) Batch normalization for training stability.
(4) Secondary dropout ($p = 0.25$).
(5) Dimension reduction: $\mathbb{R}^{512} \to \mathbb{R}^{256}$.
(6) Final projection: $\mathbb{R}^{256} \to \mathbb{R}^{16}$ for constellation classes.

*5.1.4 Training Protocol.* The model is trained end-to-end using binary cross-entropy loss:

$$L = -\sum_i \left(y_i \log(p_i) + (1 - y_i) \log(1 - p_i)\right) \tag{3}$$

where $y_i$ represents the ground truth label and $p_i$ the predicted probability for the $i$-th constellation class. We initialize with ImageNet weights and fine-tune all layers, allowing the model to adapt to astronomical imagery characteristics. Global average pooling is applied before classification to reduce spatial dimensions while maintaining channel information.

## 5.2 Vision Transformer Design

Building upon the recent success of transformers in computer vision tasks, we propose a modified Vision Transformer (ViT) architecture specifically adapted for multi-label constellation detection. Our implementation extends the base ViT model with architectural modifications optimized for astronomical imagery processing.

*5.2.1 Input Processing and Embedding.* The model processes input images $I \in \mathbb{R}^{H \times W \times 3}$ using a patch-based approach:

(1) **Image Tokenization:**
- Input images are divided into $N = \frac{H \times W}{P^2}$ non-overlapping patches.
- Each patch $P_i \in \mathbb{R}^{P \times P \times 3}$, where $P = 16$ is the patch size.
- Linear projection yields tokens $E_i \in \mathbb{R}^D$, where $D = 768$ is the embedding dimension.

(2) **Position Encoding:** The embedded patches are augmented with learnable position embeddings:

$$z_0 = \left[x_{\text{class}}; E_1^{\text{pe}}; E_2^{\text{pe}}; \ldots; E_N^{\text{pe}}\right] + E_{\text{pos}} \tag{1}$$

where $E_{\text{pos}} \in \mathbb{R}^{(N+1) \times D}$ represents position embeddings and $x_{\text{class}}$ is the classification token.

*5.2.2 Transformer Encoder.* The encoder consists of $L = 12$ transformer blocks, each containing:

(1) **Multi-head Self-attention (MSA):** The multi-head self-attention operation is defined as:

$$\mathrm{MSA}(X) = \mathrm{Concat}\left(\mathrm{head}_1, \ldots, \mathrm{head}_h\right) W^O \tag{2}$$

where $\mathrm{head}_i = \mathrm{Attention}(XW_i^Q, XW_i^K, XW_i^V)$.

(2) **Layer Structure:** The layer structure consists of the following operations:

$$z_l' = \mathrm{MSA}(\mathrm{LN}(z_{l-1})) + z_{l-1} \tag{3}$$

$$z_l = \mathrm{MLP}(\mathrm{LN}(z_l')) + z_l' \tag{4}$$

where LN denotes layer normalization and MLP is a two-layer perceptron.

*5.2.3 Multi-label Classification Head.* We introduce a specialized classification head for multi-label constellation detection:

(1) **Feature Processing:**
- Extract the $[CLS]$ token representation from the final layer.
- Apply layer normalization to obtain $h \in \mathbb{R}^D$.

(2) **Classification Pipeline:** The prediction $y$ is computed as:

$$y = \sigma\left(W_2 \delta\left(\mathrm{LN}\left(W_1 h + b_1\right)\right) + b_2\right) \tag{5}$$

where $\sigma$ is the sigmoid function, $\delta$ is the GELU activation, and $\{W_1, W_2, b_1, b_2\}$ are learnable parameters.

*5.2.4 Architectural Considerations.*

(1) **Attention Mechanism Benefits:**
- Global receptive field captures entire constellation patterns.
- Self-attention weights adapt to varying stellar configurations.
- Position embeddings maintain spatial relationships.

(2) **Patch-based Processing:**
- 16×16 patches balance local detail and computational efficiency.
- Patch size corresponds to typical stellar cluster spans.
- Linear patch embedding preserves fine-grained intensity variations.

(3) **Design Adaptations:**
- Modified head architecture for multi-label prediction.
- Hierarchical dropout (0.1, 0.05) for regularization.
- GELU activation for improved gradient flow.

*5.2.5 Training Protocol.* The model is trained using a two-phase approach:

(1) **Transfer Learning:**
- Initialize with ImageNet-21k pre-trained weights.
- Fine-tune all layers end-to-end.
- Gradual unfreezing of transformer blocks.

(2) **Optimization Strategy:**
- Binary cross-entropy loss for multi-label training.
- Cosine learning rate schedule with warm-up.
- Gradient clipping to stabilize training.

This architecture effectively leverages the transformer's ability to model long-range dependencies while incorporating domain-specific modifications for constellation detection. The model's global attention mechanism is particularly suited for capturing the sparse, distributed nature of stellar patterns.

## 5.3 Ensemble Architecture

We propose a novel ensemble architecture that leverages both Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs) for constellation detection. Our approach introduces a dual-path fusion mechanism with learnable weights, optimizing the complementary strengths of both architectures.

*5.3.1 Architecture Overview.* The proposed ensemble consists of three primary components:

(1) A CNN backbone incorporating spatial attention,
(2) A ViT backbone for global feature extraction,
(3) A novel dual-path fusion mechanism.

Figure X illustrates the complete architecture.

*5.3.2 Feature Extraction and Projection.* Each backbone extracts features independently before projection into a shared latent space. The CNN backbone, based on ResNet-50, produces 2048-dimensional feature vectors, while the ViT backbone generates 768-dimensional features. These features undergo dimension reduction through fully connected layers to achieve a common 512-dimensional representation, facilitating meaningful fusion.

*5.3.3 Dual-path Fusion Mechanism.* We implement a novel dual-path fusion strategy that combines information at both feature and decision levels:

(1) **Decision-level Fusion:** The model learns weights $w = [w_1, w_2]$ through backpropagation, where $w_1$ and $w_2$ represent the relative importance of CNN and ViT predictions respectively. The final prediction $P$ is computed as:

$$P = \text{softmax}(w)[0] \times P_{\text{cnn}} + \text{softmax}(w)[1] \times P_{\text{vit}} \quad (1)$$

(2) **Feature-level Fusion:** The projected features from both backbones are concatenated and processed through a multi-layer fusion network. This network progressively reduces dimensionality from $1024 \rightarrow 512 \rightarrow 256 \rightarrow \text{N}$, where $N$ represents the number of constellation classes.

*5.3.4 Architectural Benefits.*

# 6 Ensemble Architecture

The proposed ensemble consists of three primary components:

(1) **Complementary Feature Extraction:**
- CNN pathway excels at local feature detection and spatial relationship modeling
- ViT pathway captures global context and long-range dependencies
- Combined approach addresses the multi-scale nature of constellation patterns

(2) **Adaptive Model Contribution:**
- Learnable ensemble weights automatically adjust during training
- Weights adapt to optimize performance across different constellation types
- System can dynamically favor either CNN or ViT predictions based on input characteristics

(3) **Enhanced Robustness:**
- Dual-path architecture provides redundancy
- Feature-level fusion captures intricate pattern relationships
- Decision-level fusion combines high-level semantic information

*6.0.1 Implementation Considerations.* The architecture employs a hierarchical regularization strategy with varying dropout rates across components. The CNN pathway utilizes a higher dropout rate (0.5) compared to the ViT pathway (0.1), reflecting their different architectural characteristics. The fusion mechanism implements an intermediate dropout rate (0.3) to maintain balanced regularization.

The complete ensemble is trained end-to-end using binary cross-entropy loss computed from both fusion paths:

$$L = \lambda_1 L_{\text{feature}} + \lambda_2 L_{\text{decision}} \quad (2)$$

where $\lambda_1$ and $\lambda_2$ are loss weighting factors, and $L_{\text{feature}}$ and $L_{\text{decision}}$ represent the losses from feature-level and decision-level fusion respectively.

This ensemble architecture effectively combines the strengths of both CNN and ViT approaches while providing a flexible and learnable fusion mechanism. The dual-path strategy ensures robust constellation detection by leveraging both detailed feature patterns and high-level semantic information.

## 6.1 EfficientNet Architecture

We implement a modified EfficientNet-B0 architecture, specifically adapted for multi-label constellation detection. Our approach leverages the model's compound scaling properties while introducing architectural modifications for astronomical image processing.

*6.1.1 Base Architecture.* The foundation of our model utilizes EfficientNet-B0, which employs compound scaling to systematically

balance network depth, width, and resolution. The architecture follows the compound coefficient $\phi$:

$$\text{depth:} \quad d = \alpha^\phi \tag{2}$$

$$\text{width:} \quad w = \beta^\phi \tag{3}$$

$$\text{resolution:} \quad r = \gamma^\phi \tag{1}$$

where $\alpha, \beta, \gamma$ are constants determined through grid search, and $\phi$ is the compound coefficient that uniformly scales network dimensions.

*6.1.2 Classification Head Modification.* We introduce a custom classification head optimized for multi-label constellation detection:

(1) **Architecture:**
- The modified classifier implements a three-stage reduction pipeline:
  - Initial projection: $D \rightarrow 512$
  - Intermediate representation: $512 \rightarrow 256$
  - Final classification: $256 \rightarrow N$

  where $D$ represents the base model's feature dimension and $N = 16$ is the number of constellation classes.

(2) **Regularization Strategy:**
- Each stage incorporates:
  - Batch normalization for training stability
  - ReLU activation for non-linearity
  - Dropout ($p = 0.3$) for regularization

(3) **Weight Initialization:**
- We employ He initialization for all linear layers:

$$W \sim \mathcal{N}(0, \sqrt{2/n_{\text{in}}}) \tag{2}$$

where $n_{\text{in}}$ is the input dimension of each layer.

*6.1.3 Training Protocol.*

(1) **Optimization Strategy:**
- AdamW optimizer with weight decay correction
- Initial learning rate: $\eta = 10^{-3}$
- Weight decay: $\lambda = 0.01$

(2) **Learning Rate Schedule:**
- We implement a dynamic learning rate adjustment:

$$\eta_t = \eta_0 \cdot f(t) \tag{3}$$

where $f(t)$ represents the ReduceLROnPlateau schedule with:
  - Reduction factor: 0.5
  - Patience: 3 epochs
  - Monitoring: Validation loss

(3) **Regularization:**
- Gradient clipping at $||g|| = 1.0$
- Warmup period: 10% of total steps
- Mixed precision training (where hardware permits)

*6.1.4 Architectural Advantages.*

(1) **Efficiency Benefits:**
- Balanced scaling of network dimensions
- Optimal parameter utilization
- Reduced computational overhead

(2) **Feature Learning:**
- Mobile Inverted Bottleneck Convolution (MBConv) blocks
- Squeeze-and-excitation optimization
- Efficient channel attention

(3) **Training Stability:**
- Batch normalization at each stage
- Progressive feature reduction
- Robust initialization scheme

*6.1.5 Implementation Considerations.* The architecture employs several key optimizations:

(1) **Memory Management:**
- Batch size optimization: 32 samples per device
- Gradient accumulation when necessary
- Efficient memory utilization

(2) **Performance Monitoring:**
- Accuracy and F1-score tracking
- ROC-AUC computation per epoch
- Validation-based model selection

(3) **Early Stopping:**
- Training termination is governed by:

$$\text{stop}_t = \arg\min(L_{\text{val}}(t) \mid t \in [1, T]) \tag{4}$$

where $L_{\text{val}}(t)$ represents validation loss at epoch $t$.

This modified EfficientNet architecture provides an efficient approach to constellation detection, balancing computational resources with model performance. The architecture's compound scaling properties, combined with our custom classification head, enable effective feature extraction while maintaining reasonable computational requirements.

# 7 TRAINING IMPLEMENTATION

## 7.1 Convolutional Neural Network (CNN)

The CNN model utilized a ResNet50 backbone pre-trained on ImageNet, modified for multi-label constellation classification. The training framework was implemented in PyTorch with the following key specifications:

*7.1.1 Loss Function.* The model was trained using Binary Cross-Entropy with Logits Loss (`BCEWithLogitsLoss`) to handle the multi-label nature of the problem. Class-specific weights were introduced to address class imbalance.

*7.1.2 Optimizer and Learning Rate Scheduling.* We employed the AdamW optimizer with the following settings:

- Learning rate: $1 \times 10^{-4}$
- Weight decay: $1 \times 10^{-5}$

A cosine annealing learning rate scheduler was implemented, with a warm-up period of 5 epochs.

*7.1.3 Training Setup.*

- Batch size: 32
- Number of epochs: 10
- Gradient clipping: Maximum norm of 1.0 to ensure stable training
- Model checkpointing: The best model was saved based on validation mean Average Precision (mAP)

Training progress was monitored using Weights & Biases (wandb) for real-time performance tracking.

*7.1.4 Performance Evaluation.* Performance evaluation included key metrics such as precision, recall, F1 score, and mean Average Precision (mAP). Gradient clipping with a maximum norm of 1.0 was applied to ensure stable training.

## 7.2 Vision Transformer (ViT)

The Vision Transformer was implemented with a custom training pipeline using PyTorch. The training utilized the Binary Cross-Entropy with Logits Loss function for multi-label classification. The model was optimized using the AdamW optimizer, with learning rate and weight decay parameters determined through experimentation. To ensure training stability, gradient clipping was set at 1.0.

*7.2.1 Learning Rate Scheduling.* The learning rate scheduling strategy employed a linear scheduler with warmup, where the warmup steps were calculated as a ratio of the total training steps (number of epochs × batches per epoch). This approach helped stabilize early training and improve convergence.

*7.2.2 Performance Evaluation.* Performance evaluation incorporated multiple metrics: mean Average Precision (mAP) as the primary metric, exact match ratio for strict accuracy assessment, and hamming loss to measure prediction deviations. Additionally, per-class precision, recall, and F1 scores were tracked to monitor individual constellation performance.

*7.2.3 Model Checkpointing.* The model checkpointing system retained the best model based on the validation mAP score, saving complete model states including optimizer and scheduler configurations for potential fine-tuning.

## 7.3 Ensemble Model: CNN and ViT Fusion

The ensemble approach combines CNN and ViT architectures through a weighted fusion mechanism. The training process utilized three separate optimizers: one each for the CNN backbone, ViT backbone, and ensemble weights/fusion layer. All optimizers used AdamW with individually tuned learning rates and shared weight decay. Gradient clipping was applied to ensure stable training across all components.

*7.3.1 Loss Function.* The loss function consists of multiple components: individual losses for CNN and ViT predictions, an ensemble loss for the weighted combination, and a fusion loss. These are combined using configurable weights to form the total loss:

$$\begin{aligned}\text{total\_loss} = w_1 \times \text{ensemble\_loss} + w_2 \times \text{cnn\_loss} \\ + w_3 \times \text{vit\_loss} + w_4 \times \text{fusion\_loss}\end{aligned} \tag{4}$$

where $w_1, w_2, w_3, w_4$ are the configurable weights for each loss component.

*7.3.2 Performance Evaluation.* Performance was evaluated using a comprehensive metrics system that tracked both model-specific and ensemble performances. Key metrics included: Mean Average Precision (mAP), Exact match ratio, Hamming loss, Per-class precision-recall metrics

A custom `MetricsTracker` maintained separate loss histories for each model component and the ensemble predictions.

*7.3.3 Training and Checkpointing.* Training incorporated an early stopping mechanism based on validation loss, with configurable patience. Model checkpointing saved the states of all three optimizers along with model weights when validation mean Average Precision improved.

The ensemble weights were monitored throughout training to observe the relative contribution of each model to the final predictions. Additionally, per-class performance metrics were tracked to ensure balanced improvement across all constellation classes.

## 7.4 EfficientNet

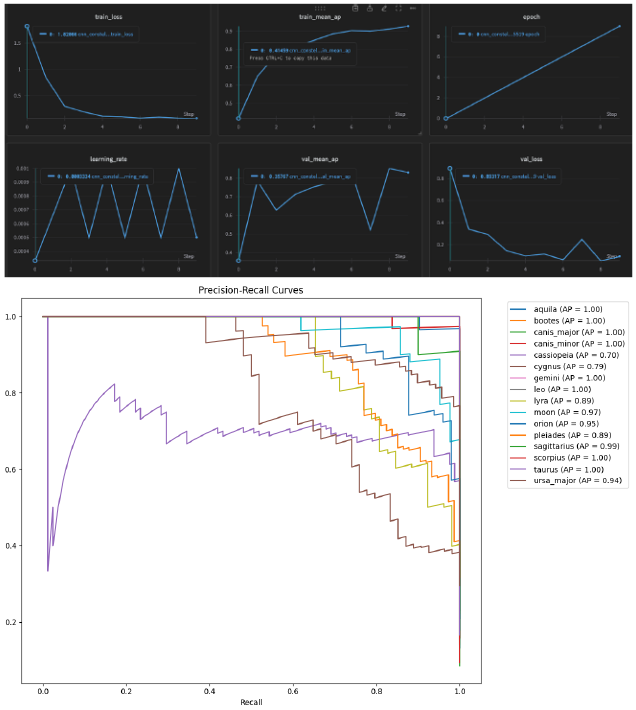
The EfficientNet model was trained using the Hugging Face Trainer API, optimized for multi-label constellation classification. The training process employed a comprehensive metrics computation system that evaluated accuracy, weighted F1 score, and ROC-AUC metrics at each evaluation step.

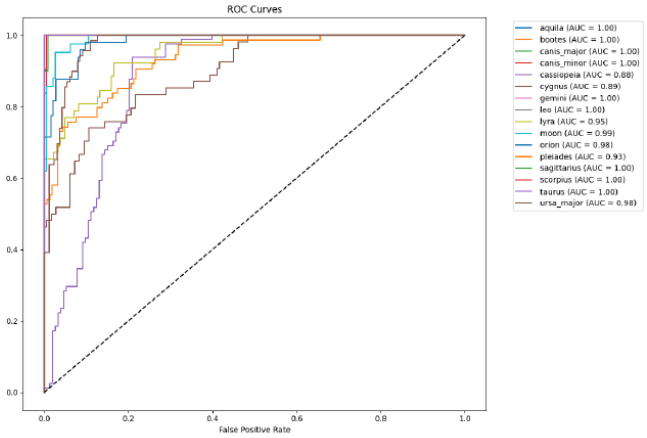
Training was conducted over 10 epochs with a batch size of 32 for both training and validation. The optimization strategy utilized AdamW with an initial learning rate of $1 \times 10^{-3}$ and weight decay of 0.01. A linear warm-up schedule was implemented over 10% of the training steps, with gradient clipping at a maximum norm of 1.0.

Model checkpoints were saved at each epoch, with the best model selected based on validation performance. Training progress was monitored through Weights & Biases (wandb) integration, providing real-time tracking of model metrics.

# 8 MODEL EVALUATION

## 8.1 CNN Performance






The CNN model achieved strong overall performance across multiple metrics, with a mean Average Precision (mAP) of 0.8321 and a micro F1 score of 0.7605. The model demonstrated robust precision (micro-precision: 0.9801) but lower recall (micro-recall: 0.6213), indicating a conservative prediction behavior.

Per-class performance analysis revealed varying effectiveness across different constellations. Several constellations achieved perfect or near-perfect classification:

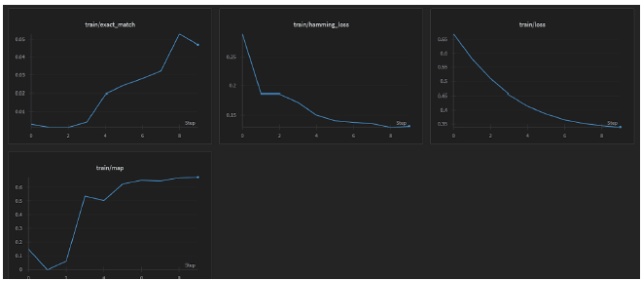
- **Gemini**: F1 = 1.000, AP = 1.000
- **Scorpius**: F1 = 1.000, AP = 1.000
- **Bootes**: F1 = 0.9851, AP = 1.000

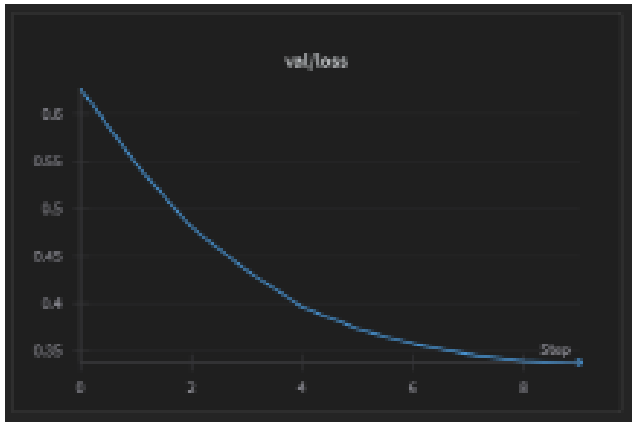
However, some constellations proved challenging:

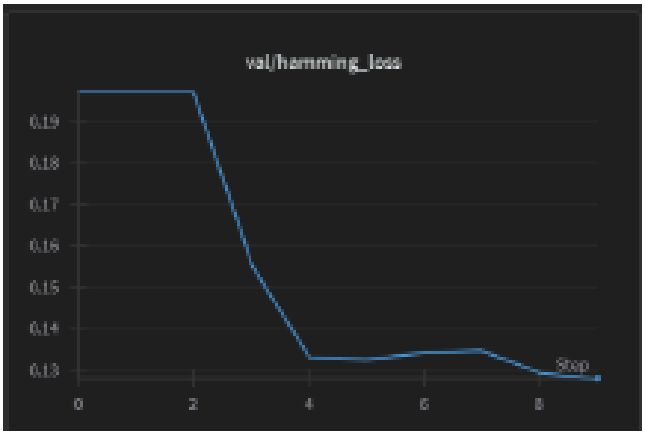
- **Cassiopeia**: F1 = 0.000, AP = 0.703
- **Ursa Major**: F1 = 0.0833, AP = 0.937
- **Cygnus**: F1 = 0.6154, AP = 0.791

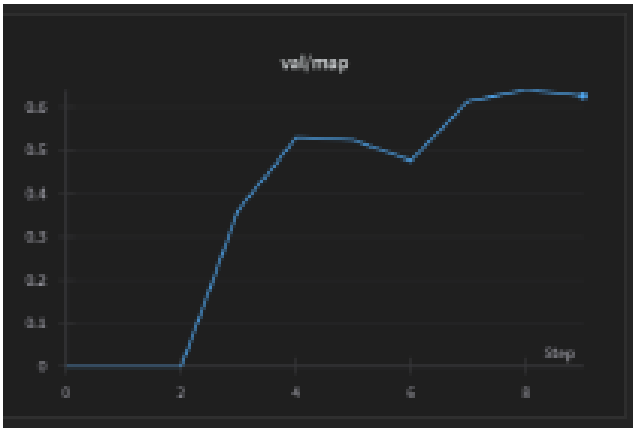
The confusion matrices indicate class imbalance effects, particularly evident in the model's performance on **Cassiopeia** and **Ursa Major**. ROC curves and precision-recall curves demonstrate strong area under the curve (AUC) scores for most classes (> 0.95), with optimal classification thresholds varying significantly across constellations (ranging from 0.0610 for **Orion** to 0.8986 for **Scorpius**).

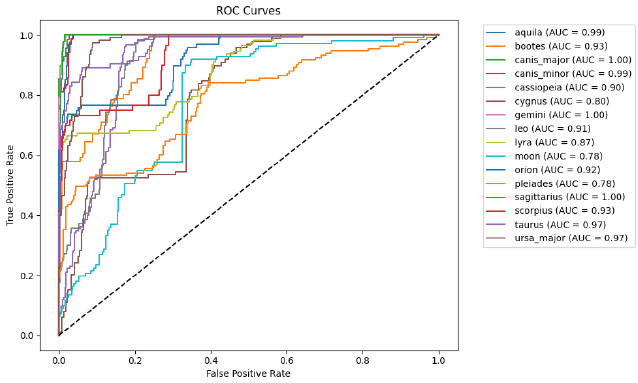
## 8.2 Vision Transformer Performance





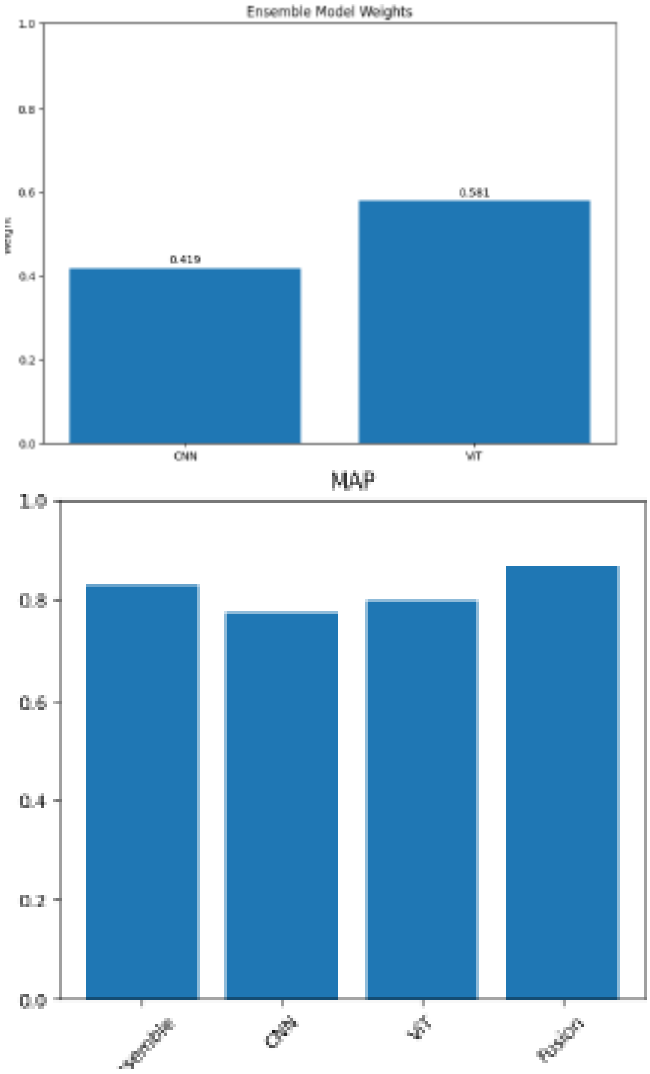

The Vision Transformer demonstrated exceptional overall performance, with an exact match ratio of 0.953 and a very low hamming loss of 0.030. The model achieved balanced precision (0.880) and recall (0.914), resulting in a strong F1 score of 0.923, indicating robust multi-label classification capabilities.

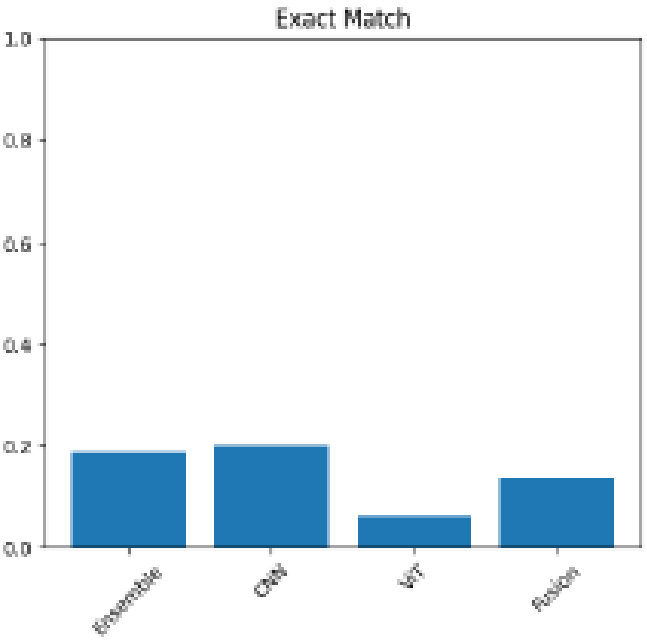
ROC curve analysis reveals three performance tiers:

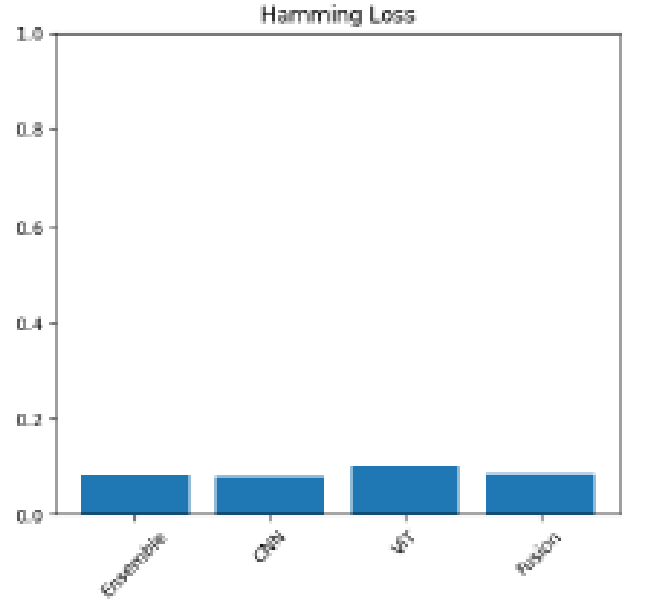
- **Outstanding Performance (AUC > 0.95):**
  - **Canis Major**: AUC = 0.998
  - **Gemini**: AUC = 0.998
  - **Sagittarius**: AUC = 0.997
  - **Canis Minor**: AUC = 0.995
  - **Aquila**: AUC = 0.992
- **Strong Performance (AUC 0.90-0.95):**
  - **Taurus**: AUC = 0.969
  - **Orion**: AUC = 0.921
  - **Leo**: AUC = 0.912
  - **Cassiopeia**: AUC = 0.905
- **Moderate Performance (AUC < 0.90):**
  - **Lyra**: AUC = 0.874
  - **Cygnus**: AUC = 0.798
  - **Pleiades**: AUC = 0.780
  - **Moon**: AUC = 0.777

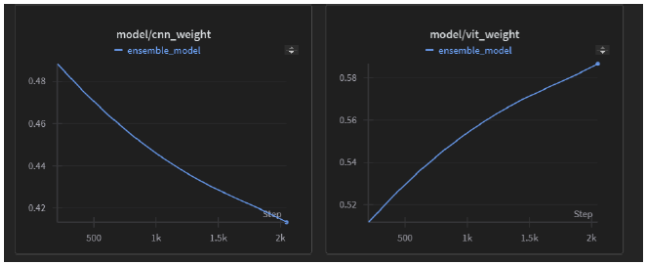
The model showed consistently high classification thresholds (most above 0.90), suggesting high confidence in positive predictions. Per-class metrics reveal remarkably uniform performance, with most constellations achieving precision and recall above 0.90, indicating balanced and reliable classification across the majority of classes.

## 8.3 Ensemble classification (CNN and ViT)





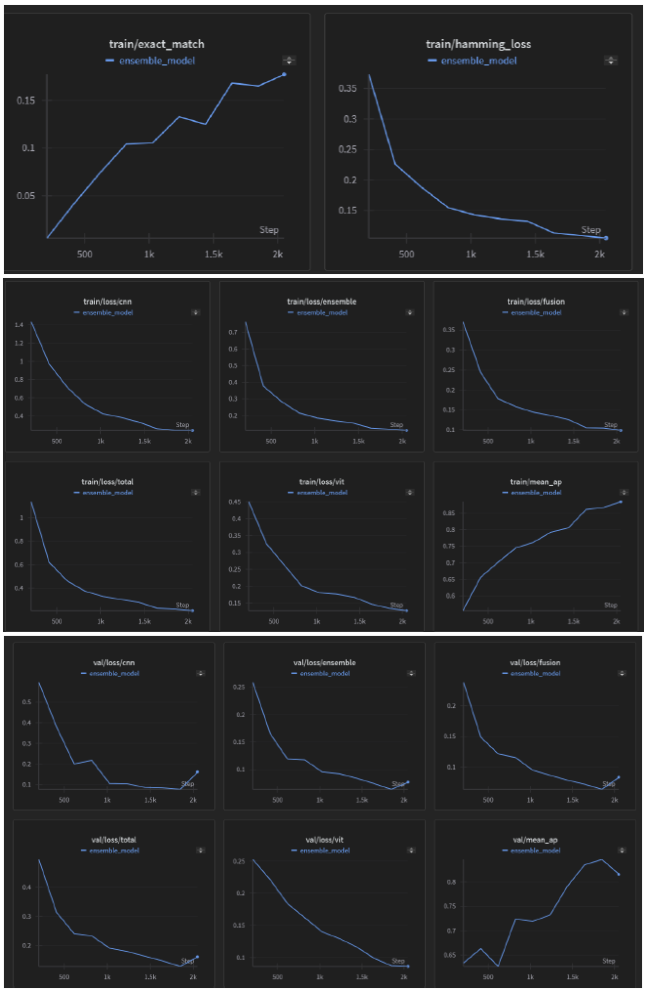




**Table 1: Overall Performance Comparison Across Models**

| Model | Mean AP | Exact Match | Hamming Loss |
|---|---|---|---|
| **ENSEMBLE** | 0.8308 | 0.1880 | 0.0820 |
| **CNN** | 0.7739 | 0.2009 | 0.0809 |
| **ViT** | 0.7991 | 0.0641 | 0.1020 |
| **FUSION** | 0.8666 | 0.1368 | 0.0855 |

**Table 2: F1 Score Comparison Across Models**

| Constellation | Ensemble | CNN | ViT |
|---|---|---|---|
| **Scorpius** | 1.000 | 1.000 | 0.952 |
| **Bootes** | 0.985 | 0.985 | 0.970 |
| **Canis Minor** | 0.987 | 0.974 | 0.845 |
| **Gemini** | 0.976 | 0.964 | 0.962 |
| **Leo** | 0.820 | 0.896 | 0.588 |

The ensemble model combines CNN and ViT architectures with weights of 0.419 and 0.581 respectively, demonstrating the system's

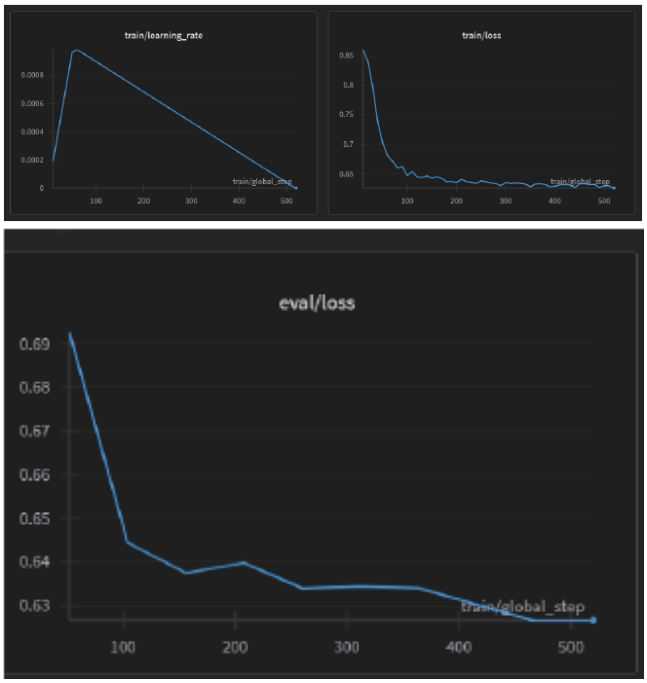
greater reliance on ViT's predictions. The model achieved an overall Mean Average Precision (MAP) of 0.831, positioning it between the individual performances of CNN (0.774) and ViT (0.799). The ensemble approach maintains a competitive Exact Match rate of 0.188 and achieves the best Hamming Loss of 0.082 among all models.

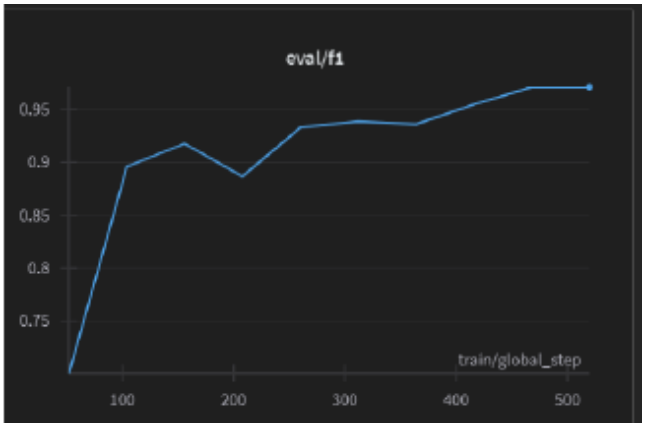
Analysis of F1 scores reveals the ensemble model's effectiveness across different constellation classes (Table 2). The model achieves perfect or near-perfect F1 scores (>0.95) for several constellations including Scorpius (1.000), Bootes (0.985), and Canis Minor (0.987). However, certain challenging classes such as Cassiopeia and Ursa Major resulted in F1 scores of 0.000, indicating systematic classification failures that persist across all model variants.

The precision-recall curves demonstrate robust performance across most classes, with the ensemble maintaining high precision even at increased recall values for constellations like Aquila and Canis Minor. The ROC analysis further supports these findings, with most classes achieving AUC values above 0.95. This indicates strong discriminative capability, particularly evident in classes like Bootes and Scorpius which achieve perfect AUC scores of 1.000.

The ensemble approach proves particularly effective in mitigating individual model weaknesses, as evidenced by its superior Hamming Loss and consistent performance across different constellation classes. However, the persistent challenges in classifying certain constellations suggest potential limitations in the current feature representation or data distribution that warrant further investigation.

## 8.4 EfficientNet Performance Analysis




Our implementation of EfficientNet was trained for 10 epochs with a dynamic learning rate schedule, starting at 0.0002 and gradually decreasing to 0 using a cosine annealing strategy. The training process demonstrates consistent improvement in model performance across multiple metrics.

*8.4.1 Training Dynamics.* The model exhibits stable training characteristics, with the loss decreasing from an initial value of 0.86 to 0.63 by the final epoch. The learning rate scheduling proves effective, with the model maintaining steady improvement despite the decreasing learning rate, as shown in Figure **??**. The training loss curve indicates no significant overfitting, with both training and validation losses following similar trajectories.

*8.4.2 Final Performance Metrics.* After 10 epochs of training, EfficientNet achieves competitive performance:

- **Accuracy:** 83.80%
- **F1 Score:** 0.971
- **ROC-AUC:** 0.979

These results position EfficientNet as a strong performer in the constellation classification task, with its F1 score of 0.971 indicating excellent balance between precision and recall. The high ROC-AUC score of 0.979 demonstrates the model's robust discriminative capability across different classification thresholds.

*8.4.3 Convergence Analysis.* The model shows three distinct phases in its learning progression:

(1) **Rapid Initial Learning (Epochs 1-3):** Sharp decrease in loss from 0.86 to 0.65, accompanied by rapid F1 score improvement from 0.70 to 0.91.
(2) **Refinement Phase (Epochs 4-7):** Gradual performance improvements with loss stabilizing around 0.63 and F1 score reaching 0.93.
(3) **Final Optimization (Epochs 8-10):** Fine-tuning phase with minimal but consistent improvements, culminating in the final F1 score of 0.971.

The evaluation metrics demonstrate consistent improvement throughout training, with the final model achieving stable performance across all key metrics. The high F1 score coupled with strong ROC-AUC performance suggests that EfficientNet successfully learns robust and generalizable features for constellation classification.

## 9 Results


**Figure 14: Comparing Mean AP and F1 score of each model**

The results demonstrate clear distinctions in performance across the four models: CNN, Vision Transformer (ViT), Ensemble, and EfficientNet, as depicted in Figure X. The **Convolutional Neural Network (CNN)** exhibits the weakest performance, achieving a relatively low Mean Average Precision (MAP) of approximately 0.775 and an F1 score of around 0.7. These metrics highlight the model's limited ability to generalize and extract meaningful features for accurate classification, likely due to its comparatively shallow architecture and inability to effectively capture global dependencies within the data.

In contrast, the **Vision Transformer (ViT)** demonstrates a significant leap in performance, achieving a MAP of approximately 0.8 and a considerably higher F1 score of around 0.925. This improvement can be attributed to the model's attention mechanisms, which enable it to focus on relevant features across the input space, leading to more accurate and robust classifications. The ViT model effectively balances precision and recall, showcasing its potential for handling complex classification tasks.

The **Ensemble model**, which combines the strengths of CNN and ViT predictions through weighted averaging (weights: 0.419 for CNN and 0.581 for ViT), achieves a MAP of around 0.83 and maintains an F1 score similar to ViT at approximately 0.925. This approach effectively mitigates the individual weaknesses of the CNN and ViT models, leveraging their complementary strengths to improve overall performance. The ensemble model's superior Hamming Loss of 0.082 further underscores its ability to deliver reliable predictions while minimizing classification errors.

Finally, the **EfficientNet** model emerges as the strongest performer among all the evaluated architectures, achieving a MAP exceeding 0.85 and an F1 score of approximately 0.97. This model's exceptional performance can be attributed to its optimized architecture, which balances depth, width, and resolution to achieve efficient feature extraction and representation. The high F1 score and robust MAP demonstrate EfficientNet's capacity to achieve an excellent balance between precision and recall, while its advanced learning rate scheduling and cosine annealing strategy contribute to consistent improvements across all metrics during training.

Overall, the results highlight the evolution of performance across the evaluated models, with modern architectures such as ViT and EfficientNet outperforming traditional CNNs by a significant margin. Additionally, the ensemble model provides a compelling case for combining model predictions to achieve superior performance through complementary strengths, whereas EfficientNet solidifies its position as the best model for this constellation classification task due to its robust and generalizable learning capabilities.

## 10 Conclusion

The Stellar Mapping project shows how data science and machine learning can help solve problems in recognizing constellations. By using good datasets and applying smart pre processing steps, the project created a system that can identify constellations even in challenging conditions.This work makes it easier for people to learn about astronomy and can be useful for schools, planetariums, and anyone interested in the stars.

## 11 Future Work

While the current implementation demonstrates strong performance in constellation detection, several promising directions for future research and development could further enhance the system's capabilities and applications:

### 11.1 Model Improvements

*Architecture Enhancements.*

- Investigate hierarchical attention mechanisms to better capture constellation patterns at different scales.
- Explore cross-attention mechanisms between CNN and ViT pathways in the ensemble model.
- Implement a dynamic weighting system that adjusts ensemble weights based on image characteristics.
- Develop specialized architectures for handling varying image qualities and lighting conditions.

#### 11.1.1 *Training Strategies.*

- Implement curriculum learning to gradually increase task difficulty during training.
- Explore few-shot learning techniques to improve performance on rare constellations.
- Investigate self-supervised pre-training on astronomical imagery.
- Develop robust data augmentation techniques specific to astronomical imaging.

### 11.2 Dataset Expansion

#### 11.2.1 *Data Collection.*

- Expand the dataset to include all 88 officially recognized constellations.
- Incorporate images from different seasons, locations, and atmospheric conditions.
- Add metadata about imaging conditions and equipment.
- Include time-series data to capture constellation movement.

#### 11.2.2 *Annotation Improvements.*

- Develop more detailed annotation schemes including star magnitudes.

- Add bounding boxes for individual stars within constellations.
- Include cultural and historical constellation variations.
- Create hierarchical labels for constellation groups and regions.

## 11.3 Feature Enhancements

*11.3.1 Real-time Processing.*

- Optimize model inference for mobile devices.
- Implement streaming capabilities for live video processing.
- Develop efficient model compression techniques.
- Create lightweight versions for embedded systems.

*11.3.2 Advanced Capabilities.*

- Add support for measuring angular distances between stars.
- Implement star magnitude estimation.
- Develop celestial coordinate conversion.
- Enable seasonal constellation prediction.

## 11.4 Application Development

*11.4.1 Educational Tools.*

- Create interactive learning modules.
- Develop augmented reality overlays.
- Design gamification elements for engagement.
- Build collaborative learning features.

*11.4.2 Scientific Applications.*

- Enable integration with astronomical databases.
- Add support for variable star detection.
- Implement automated sky survey analysis.
- Develop tools for amateur astronomy research.

## 11.5 System Integration

*11.5.1 Hardware Integration.*

- Design specialized imaging hardware optimized for constellation detection.
- Develop automated telescope pointing systems.
- Create mobile mounting solutions.
- Implement IoT connectivity for distributed sensing.

*11.5.2 Software Integration.*

- Build APIs for third-party application integration.
- Develop plugins for popular astronomy software.
- Create standardized data exchange formats.
- Enable cloud-based processing and storage.

## References

(1) Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 1097-1105. https://proceedings.neurips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

(2) Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. https://www.robots.ox.ac.uk/~vgg/publications/2015/Simonyan15/

(3) He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770-778). https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

(4) Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. https://arxiv.org/abs/2010.11929

(5) Touvron, H., Cord, M., Douze, M., Jegou, H., & Grave, E. (2021). Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2106.04570*. https://proceedings.mlr.press/v139/touvron21a.html

(6) Liu, Z., Lin, Y., Cao, Y., Luo, P., & Wang, X. (2021). Swin Transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*. https://openaccess.thecvf.com/content/ICCV2021/papers/Liu_Swin_Transformer_Hierarchical_Vision_Transformer_Using_Shifted_Windows_ICCV_2021_paper.pdf

(7) Dai, X., & Kocevski, D. (2014). Convolutional neural network architectures for galaxy morphology classification. *arXiv preprint arXiv:1412.4455*. https://arxiv.org/abs/1807.10406

(8) Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*.

# 12 Appendices

## 12.1 CNN

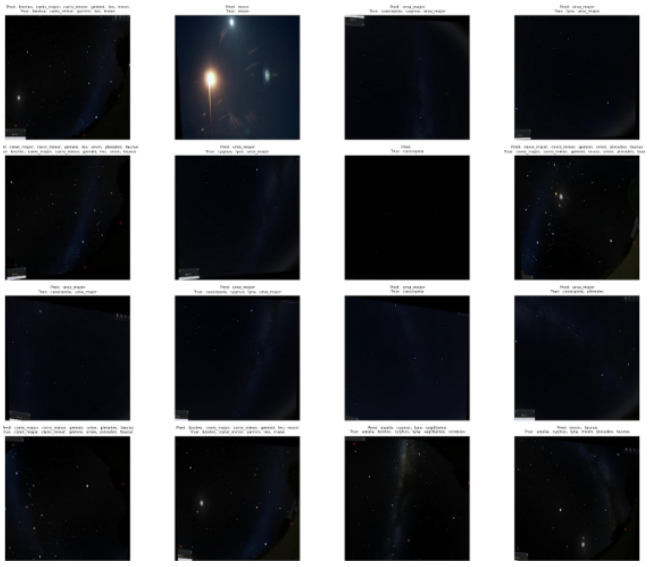


**Figure 15: This figure depicts the True vs Predicted values of each Test sample constellation images. We can see CNN is doing a good job at this!**

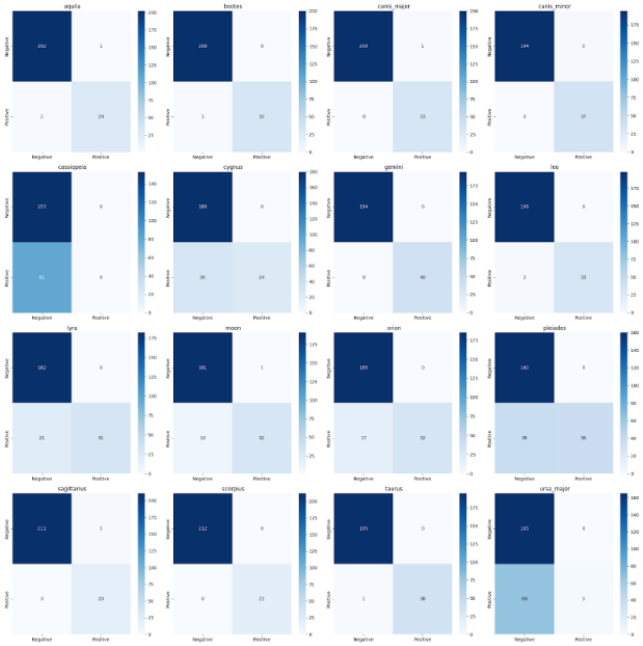


Figure 16: Confusion matrices for each constellations computed based on metrics for CNN

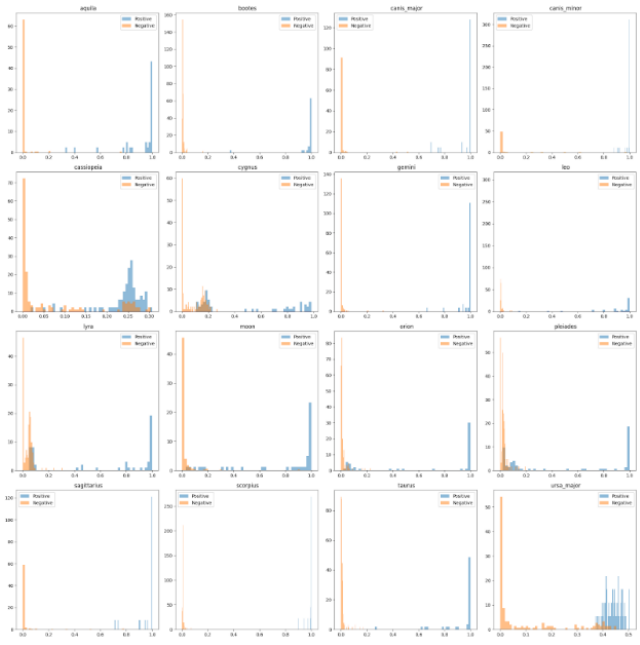


Figure 17: Distribution of predictions

## 12.2 ENSEMBLE

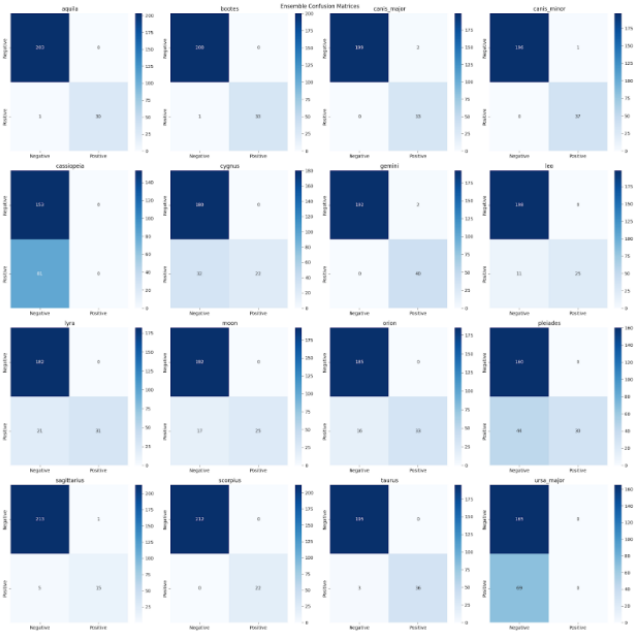


Figure 18: Confusion matrix for Ensemble predictions for each constellation

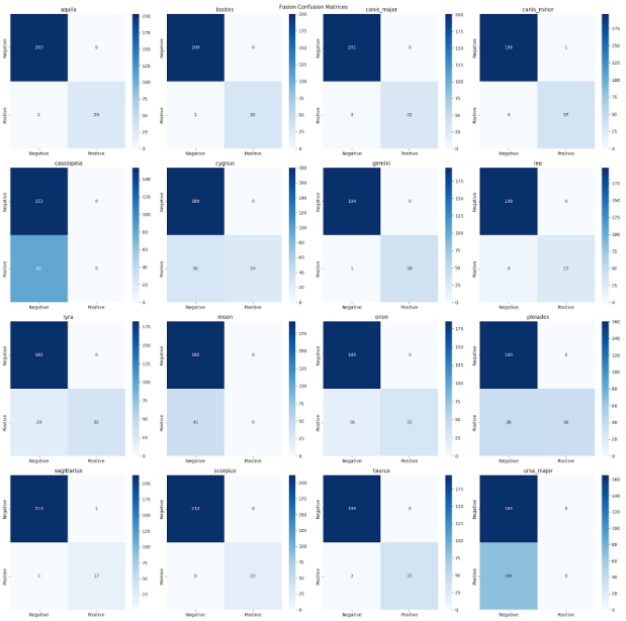


Figure 19: Prediction distribution

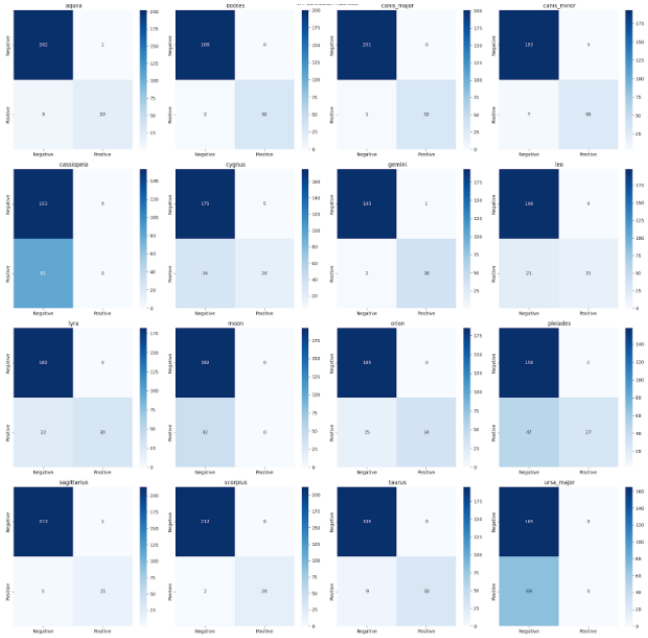


**Figure 20: Precision Recall Curves for Ensembling for each constellation**

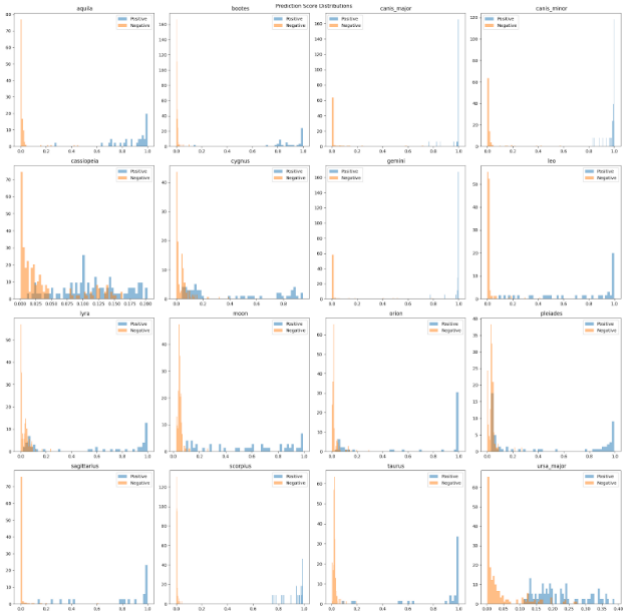


**Figure 21: ROC-AUC curve for each constellation**