**Question 2 :--**

So, according to question when Dataset was given to the YOLO model this was it’s overall performance

according to this context will be answering all the questions given in Question 2.

**Precision (P):** 0.797 overall

**Recall (R):** 0.201 overall (0.0 for Line → ., streaks undetected)

**mAP@0.5:** 0.241 overall (0.102 for Line, 0.381 for Stars)

a . How can the model achieve high accuracy when the available training images are insufficient? What strategies can be used to overcome this limitation?

1. The method I followed, 35 high resolution images were not enough , but I tiled the images for eg - ((4418/1024)^2) = 16 , so 16\*35 = 560 Images , so 560 images were suitable
2. My model can perform very well when I tiled the images according to 512 or 84 , but annotating 2k to 5k images in 4 days was really a tough job so I chose 1024
3. One of the strategies I used Data Augmentation to increase data which reached from 560 to 590
4. In my model Streaks/line are not detected - Only 10 line instances were their . To avoid this I could use Programmatically generate new lines samples by drawing faint linear streaks on background noise tiles (use OpenCV cv2.line).
5. I did not apply more data augmentation technique like crop , noise , blur , flip .. etc because of the same reason of data annotation of 4-5k images in 3-4 days

b .How can model overfitting and underfitting be addressed? What factors contribute to these issues?

**For Overfitting:**

| **Method** |  |
| --- | --- |
| **Extensive data augmentation** | Random rotation, flips, noise injection, elastic deformations simulate more diverse skies and prevent memorization. |
| **MixUp or CutMix** | Especially useful in astronomical imaging to blend overlapping star fields and simulate complexity. |
| **Early stopping on validation loss** | Stops training before model begins to memorize training data. |

**For Underfitting:**

| **Method** |  |
| --- | --- |
| **Class imbalance correction** | Use class weights or oversample rare “Line” (streak) samples |

Contribute to this issue

|  |  |
| --- | --- |

1. inadequate feature learning from grayscale data
2. Too few unique training samples.
3. Simple background in training vs real-world noise.

c .Can the developed model accurately detect streaks and stars in real-sky astronomical images, given varying signal-to-noise conditions? How does this differ from simulated images?

Answer model trained only on clean synthetic tiles: Uniform background Clean, continuous lines Consistent brightness

But But in real-sky astronomical images: Stars may blend into noisy background

So to overcome this I could use following

1 Fine-tune model on a small labeled subset of real-sky images

2 Use test-time **augmentation** to boost robustness (rotate/flip inference tiles).

What is the likelihood of false detections, such as detecting multiple blobs along a streak instead of a continuous line? Will the model correctly report false positives and generate an appropriate error matrix?

According to the process I followed of tiling then feeding to YOLO

Yes, YOLOv11 may misinterpret a single faint streak as multiple disconnected blobs if:

* Contrast is too low
* The streak is broken into faint segments due to tiling.

To correctly report false positives and false negatives:

* Use **confidence thresholds** carefully (not too high to miss detections, not too low to flood false positives).

**Question 1**

**Project Description: Detection & Classification of Stars and Streaks in Synthetic Astronomical Images**

The original dataset consisted of:

* 35 grayscale .tiff images, each of size 4418 × 4418 pixels.
* Each image was a 16-bit unsigned integer single-band format, containing faint stars and streaks**.**

**Preprocessing Steps Performed:**

1. 16-bit Normalization:
   * Since standard visualization and many DL models expect 8-bit images, the 16-bit TIFF images were normalized using OpenCV’s cv2.normalize() and converted to 8-bit.
2. **Contrast Stretching**:
   * Applied **contrast enhancement** (linear stretching) to highlight faint objects (stars, streaks) for better annotation and model input.
3. **Tiling Large Images**:
   * Due to GPU memory limits and the huge size of original images, each image was **divided into smaller tiles (512×512 or 1024×1024)** using sliding windows with overlap.
   * This also served as a form of **data augmentation**, increasing dataset size.
4. **Annotation**:
   * Tiles were annotated manually using tools like **Roboflow**, where:
   * Annotations were saved in YOLO format (bounding boxes).
5. **Saved Tiles as .JPG**:
   * Tiled images were saved in .jpg for compatibility with **YOLOv11 training**, keeping visual quality acceptable.
6. Dataset: 550 + annotated tiles from 35 base images.
7. Classes: 2 (Line, Star)
8. Input Size: 84x84 ( for faster training)
9. Epochs: 50
10. Batch Size: 8

Why Recall , accuracy was low ??

Because I annotating 5k to 6k images were so tough and I completed annoting images manually by making bounding box for Stars and Lines ,but if I really get the time then I could bring a good recall and accuracy , also my precision was nearly good like (0.797 )

**Model Performance**

After training:

| **Metric** | | **Value (All Classes)** | | | |
| --- | --- | --- | --- | --- | --- |
| Precision (P) | | 0.797 | | | |
| Recall (R) | | 0.201 | | | |
| mAP@0.5 | | 0.241 | | | |
| mAP@0.5:0.95 | | 0.0615 | | | |
| **Class** | **Precision** | | **Recall** | **mAP@0.5** | **mAP@0.5:0.95** | |
| Line | 1.0 | | 0.0 | 0.102 | 0.0102 | |
| Stars | 0.594 | | 0.403 | 0.381 | 0.113 | |