Super resolution and Object detection on Remote sensing imagery

Shizra Tariq
Robotics
tariq044@umn.edu

Manpreet Singh
ECE
sing1174@umn.edu

Seungho
Robotics
shin0369@umn.edu

Anwesha Samaddar
Robotics
samad037@umn.edu

***Abstract*—This project aims to enhance the quality of remote sensing images and improve object detection accuracy in aerial imagery. To address common challenges like low resolution and image blur, the DOTA v1 dataset is used to generate degraded versions of images, simulating real-world conditions. These degraded images are enhanced using the Super-Resolution Generative Adversarial Network (SRGAN), which excels in restoring fine details and improving sharpness through adversarial training and perceptual loss.**

Following the image enhancement, object detection is performed using YOLO-v11 and YOLO-v11 OBB (Oriented Bounding Box), which are designed to detect objects accurately at varying orientations. The proposed method identifies eight distinct object classes, demonstrating that the integration of super-resolution techniques with oriented bounding boxes significantly improves both image quality and object detection performance. Comparative analysis between detections on low-resolution and super-resolved images highlights the effectiveness of this approach, underscoring its potential for applications in remote sensing and aerial surveillance.

***Index Terms*—SRResNet, SRGAN, YOLOv11, OBB, DOTA**

I. Introduction

Remote sensing imagery is vital for applications like military surveillance, urban planning, and environmental monitoring. However, challenges such as low resolution, compression artifacts, and motion blur hinder object detection accuracy, especially for small or densely packed objects. Addressing this, our project combines advanced super-resolution techniques with object detection models. Using SRGAN (Super-Resolution Generative Adversarial Network), we enhance image clarity by upscaling degraded images, while YOLOv11 and YOLOv11-OBB (Oriented Bounding Box) improve detection accuracy by identifying objects in varied orientations. This integration demonstrates significant improvements in both image quality and detection performance, making it highly applicable for real-world remote sensing challenges.

II. Related Work

Super-Resolution in Remote Sensing

Early SR methods focused on spatial and spectral resolution enhancement, leveraging multi-scale image information during upsampling. These approaches were eventually surpassed by deep learning-based methods that significantly improved performance. SRCNN [1] introduced convolutional neural networks (CNNs) for SR, providing foundational insights but limited detail restoration. VDSR [2] enhanced these capabilities by utilizing a very deep convolutional network with residual learning, enabling accurate reconstruction and efficient handling of multiple scaling factors. SRResNet [3] advanced this by incorporating residual blocks and skip connections, enhancing both visual quality and training efficiency. SRGAN [3] further improved results with adversarial training, utilizing a discriminator network to optimize perceptual loss and produce photorealistic textures.

Object Detection in Remote Sensing

Object detection in remote sensing involves identifying objects such as vehicles, buildings, and landscape features. Key challenges include large image sizes, complex backgrounds, small object sizes, and diverse object categories. Methods like Faster R-CNN [4], with its Region Proposal Network (RPN), offer robust localization, while SSD uses multi-scale feature maps for balanced speed and accuracy. Transformer-based models provide global context awareness, handling diverse object orientations effectively.

The YOLO (You Only Look Once) [5] series (Figure 1) stands out for its superior speed and accuracy. Unlike R-CNN and SSD, which separate the processes of region proposal and classification, YOLO adopts a unified framework that predicts bounding boxes and class probabilities in a single pass, enabling real-time detection.

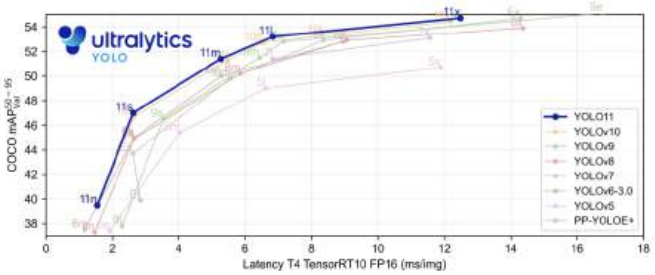


Fig. 1: Benchmarking YOLOv11 Against Previous Versions on COCO dataset [9]

This efficiency has made YOLO a popular choice for remote sensing tasks, with YOLOv8 [6] demonstrating high accuracy and speed for specific applications like vegetation and swimming pool detection.

Building on these advancements, YOLOv11 [7] incorporates multi-scale detection and spatial attention mechanisms, sig-

nificantly improving its effectiveness for datasets like DOTA, which include objects of varying sizes and orientations. Additionally, YOLOv11-OBB extends the YOLO framework by introducing oriented bounding boxes (OBB) [8]. This modification aligns bounding boxes with the actual shapes of objects, particularly in dense scenes with arbitrarily oriented objects, thereby reducing overlaps and improving detection accuracy.

The integration of YOLOv11 and YOLOv11-OBB with super-resolution techniques has further enhanced object detection performance in remote sensing, especially for small and densely packed objects, underscoring their utility in complex aerial imagery analysis.

III. Approach

A. *Proposed Method*

We have three primary sections of our project:

1) *Image Super-Resolution:* In the first stage, we train SRGAN network to upscale low-resolution images by a factor of four
2) *Object Detection:* Next, Object detection models YOLOv11 and YOLOv11-OBB (Oriented Bounding Box) are trained and evaluated on the high-resolution images from DOTA v1 dataset
3) *Comparative analysis*: We compare detection accuracy by evaluating the performance of these models on SRGAN-enhanced high-resolution images (1024×1024) versus the original low-resolution images (256×256).

B. *Dataset Description*

The DOTA (Dataset for Object Detection in Aerial Images) is a large scale dataset designed for object detection in aerial imagery. The DOTA dataset consists of 2806 aerial images collected from a wide range of sensors and platforms. These images have a large resolution of ranging from ranging from 800 × 800 to 10000 x 10000 pixels and include objects across 15 categories. These categories reflect a wide variety of real-world applications such as vehicles, ships, airports, and buildings. The dataset includes 15 object categories with labels for 188,282 objects. Each object is marked with an oriented bounding box (OBB), which can rotate to fit the object more accurately compared to traditional axis-aligned bounding boxes.

The categories are plane, ship, storage tank, baseball diamond, tennis court, basketball court, ground track field, harbor, bridge, large vehicle, small vehicle, helicopter, roundabout, and soccer ball field. A small section of these categories is shown in Fig. 2. The dataset tackles key challenges such as scale variation, crowded scenes, and arbitrary object orientations, providing a diverse range of instances in different sizes and shapes.

On average, DOTA contains 67 instances per image, with a balanced distribution of small, medium, and large objects. The dataset is divided into training, validation, and test sets, with half of the images used for training, one-sixth for validation, and one-third for testing. The training, validation and test sets are publicly available providing a benchmark for object detection models.

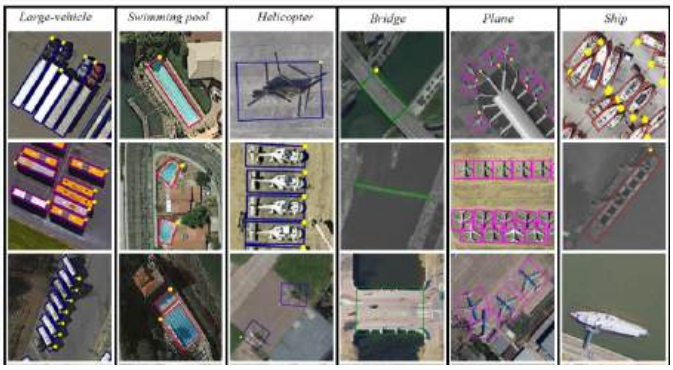


Fig. 2: Example annotations on DOTA dataset [10]

C. *Workflow for Training SRGAN:*

The workflow involves training two key models: SRResNet and SRGAN [3], with the latter leveraging adversarial training for enhanced performance.

1) **Standalone Training of SRResNet:**
   - SRResNet, a fully convolutional network designed for 4x super-resolution, is trained independently as a baseline.
   - The trained checkpoints from this stage are saved for future use.
2) Training of SRGAN:
   - The SRResNet weights are used to initialize the generator of the SRGAN.
   - The SRGAN involves alternate training of the generator and discriminator to iteratively improve the quality of the super-resolved images.

D. *Super-Resolution ResNet (SRResNet) Architecture*

The SRResNet architecture shown in Figure 3 uses residual blocks and skip connections to facilitate optimization in deep networks. This network performs 4x upscaling and serves as a robust baseline for the SRGAN.

Architecture Details:

1) Initial Layer: The input low-resolution image is passed through a 9×9 convolution layer, generating 64 feature maps at the same resolution, followed by a Parametric ReLU (PReLU) activation.
2) Residual Blocks: A series of 16 residual blocks includes 3×3 convolutions, batch normalization, and PReLU activations, maintaining the resolution and channel count. A larger skip connection spans all residual blocks.
3) Upscaling: Two subpixel convolution layers upscale the resolution by a factor of 2 each (net 4x upscaling), with PReLU activations.
4) Upscaling: Two subpixel convolution layers upscale the resolution by a factor of 2 each (net 4x upscaling), with PReLU activations.

5) Final Layer: A 9x9 convolution layer followed by Tanh activation produces the super-resolved RGB image in the [-1,1] range.

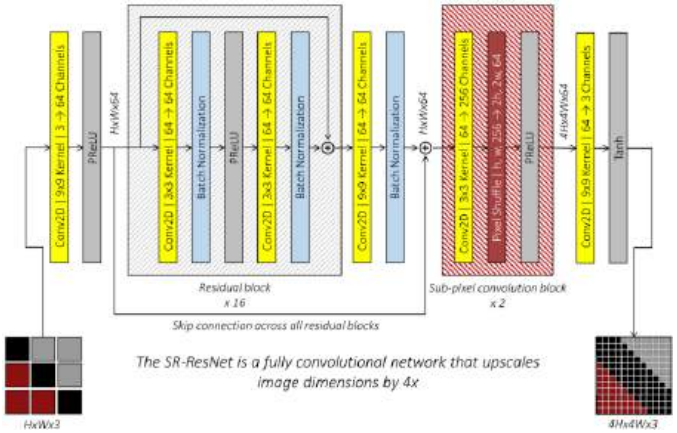


Fig. 3: The SRResNet Architecture (Generator for SRGAN)

E. Super-Resolution Generative Adversarial Network (SRGAN) Architecture

SRGAN comprises two core components: a Generator and a Discriminator. These networks are trained together in a competitive manner.

Generator Architecture

1) The Generator shares the same architecture as SRResNet, and we initialize it with trained weights from SRResNet (Figure 3).
2) It produces super resolved images by minimizing content loss and adversarial loss by learning from the Discriminator's feedback.

Discriminator Architecture

Here a convolutional network acts as a binary classifier (Figure 4). Key layers include:

1) Initial convolution with a 9×9 kernel and stride 1, producing 64-channel feature maps.
2) Seven convolutional blocks, each with 3×3 kernels, batch normalization, and leaky ReLU activations.
3) Channels double in even-indexed blocks; dimensions halve in odd-indexed blocks with stride 2.
4) Flattened features are transformed into a 1024-dimensional vector followed by a final output layer, generating a probability score.

F. Model Inputs and Targets - SRResNet and SRGAN

The training of the SRGAN involves careful preparation of both high-resolution (HR) and low-resolution (LR) images. Since DOTA dataset images are of very high dimensions, it is not computationally ideal to train the network on entire images. Instead, we create random patches to form pairs of HR and LR images for efficient training.

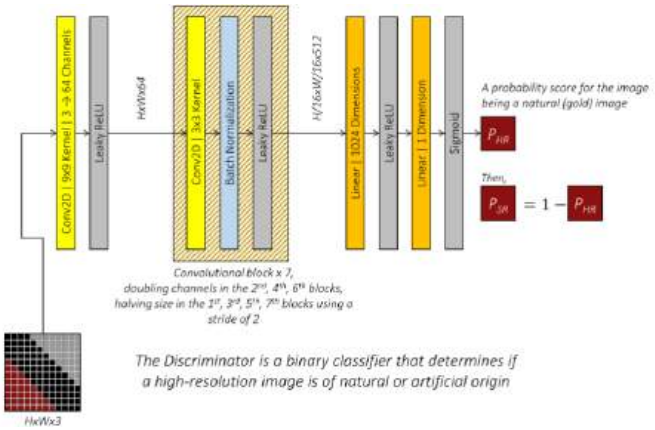


Fig. 4: Discriminator Architecture for SRGAN

High-Resolution (HR) Images:

1) HR images are random patches of dimension 96×96x3 extracted from training set.
2) These are used as targets for the SRResNet training (Generator network of SRGAN), and as inputs to the Discriminator.
3) Normalization: When used for standalone SRResNet training, pixel values are scaled to [-1,1] for MSE calculation in RGB space.
   When used as Generator or Discriminator targets while training SRGAN, pixel values are normalized with **ImageNet** mean [0.485,0.456,0.406] and standard deviation [0.229,0.224,0.225].

Low-Resolution (LR) Images:

1) LR images of dimension 24x24x3 are created by applying 4x bicubic downsampling to HR patches.
2) These serve as **inputs during SRResNet and SRGAN** training.
3) To prevent aliasing we add Gaussian blur, while downsampling. Also LR images are normalized with ImageNet mean and standard deviation.

Few examples of HR and LR image patch pairs are shown in Figure 5.

G. Training of SRResNet:

The training data consists of HR images and their bicubic-downsampled LR counterparts as described in section III-F.

Loss function: The model parameters are updated using the Mean Squared Error (MSE) loss, which ensures that the super-resolved image is closer in appearance to the original high-resolution version. The MSE loss, a content-based metric, compares the predicted super-resolved image with the HR target in the RGB space, guiding the model to produce outputs that closely resemble the original HR images.

Figure 6 shows a visualization of the loss computation process.

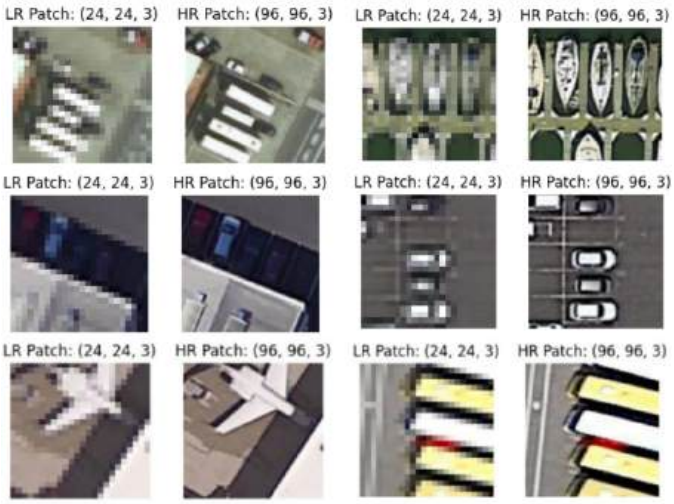


Fig. 5: Examples of HR-LR pairs of image patches created for training SRResNet and SRGAN.

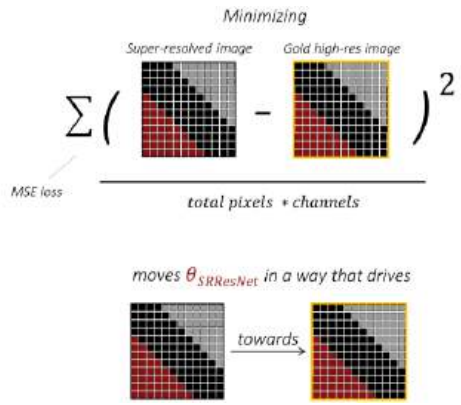


Fig. 6: MSE loss calculation for SRResNet Training

H. Training of SRGAN:

The Generator and Discriminator are alternately updated in each iteration. For each batch we have:

Discriminator Update:

- Minimizes binary cross-entropy loss to improve discrimination ability between real high-resolution image and generated/super-resolved image.

Generator Update:

1) Adversarial Loss: The Generator tries to fool the Discriminator into classifying its output as real.
2) Content Loss: Compares the super-resolved image with the ground truth in VGG19 feature space to preserve fine details without oversmoothing.
3) Perceptual loss: A weighted combination of content and adversarial losses guides the Generator (Figure 7), producing highly detailed, photorealistic images.

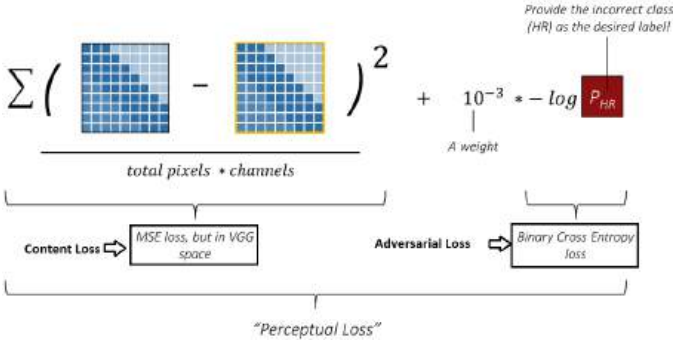


Fig. 7: Computation of Perceptual Loss

I. Dataset preparation - Object detection

To preprocess our dataset, we need to standardize the size of the images, which vary significantly in resolution. For instance, some images are as large as 1200 x 5700 or 6300 x 1800 pixels (see Fig. 9). Simply resizing these images would not be sufficient due to the wide variation in image dimensions and the presence of black borders around some images. These borders could introduce noise and negatively impact our model's performance.

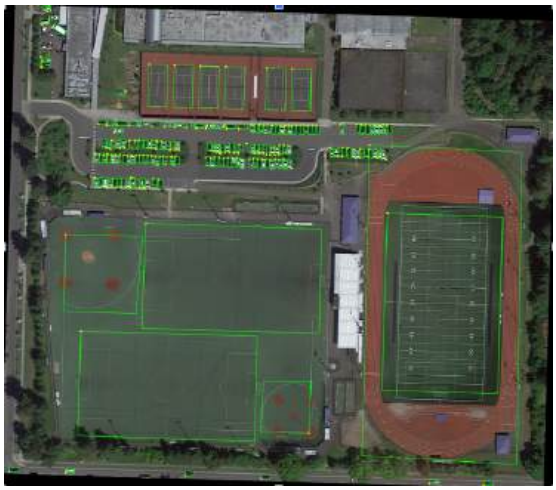
(a) (b)

Fig. 8: Two different size images from original dataset

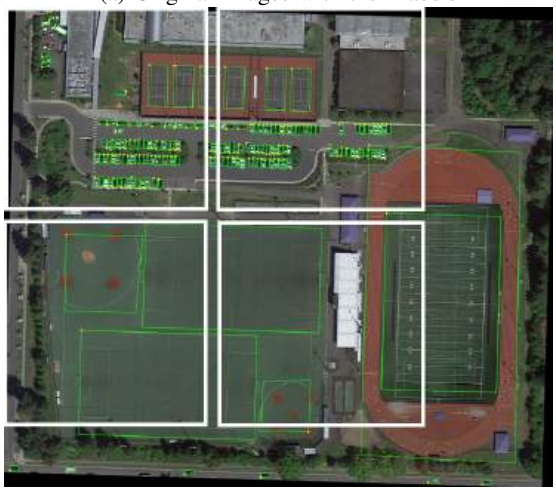
Additionally, our dataset consists of 15 classes, but the classes are imbalanced in terms of the number of images. Some classes have very few examples, and others contain images that are both large and sparse. As a result, we have reduced the number of classes to 8, removing those with low occurrence and large size (such as "Tennis Court" and "Baseball Diamond").

To standardize image sizes, we will crop 1024 x 1024 sections from each image (see Fig. 9). First, images will be padded symmetrically to ensure they are large enough for cropping, preserving the center to avoid losing important content. We will then check for excessive black regions by

setting a threshold. If more than 10% of the crop consists of black pixels, it will be ignored to prevent the model from learning irrelevant background. To optimize training, we will limit large images to a maximum of 10 crops, preventing excessive data generation and overfitting.



(a) Original images with their labels



(b) White border represents 1024x1024 cropped area

Fig. 9: Cropped region from original dataset

Next, we need to convert the labels to match the requirements of the two types of models we are using: YOLO and YOLO-OBB. Each model uses a different label format:

- **Original label format:** x1, y1, x2, y2, x3, y3, x4, y4, category, difficult
- **YOLO label format:** class_index, x_center, y_center, width, height
- **YOLO-OBB label format:** class_index, x1, y1, x2, y2, x3, y3, x4, y4

To adapt the dataset, we will convert the original labels into two formats: axis-aligned bounding boxes for YOLO and oriented bounding boxes for YOLO-OBB (see Fig. 15. During this conversion, we may encounter labels near the image edges. If more than 70% of a bounding box lies outside the image boundaries, we will discard that label to avoid including incomplete or irrelevant data.

While cropping may introduce errors for large objects, we have already excluded most of these objects from the dataset, ensuring that the remaining dataset is suitable for training and validation.

(a) YOLO labels (b) YOLO-OBB labels

Fig. 10: Converted labels from original format

IV. Experiments and Results

A. Super resolution

Experiments	SRResNet Training	SRGAN Training
Epochs	140	300
Training+Val Images	1869	1869
Time	5 hrs	10 hrs
Pre-trained Weights	Trained on MSCOCO '14	Trained on MSCOCO '14
Optimizer	Adam	Adam
Learning Rate	10^{-4}	10^{-4}
GPU	RTX 3050 Ti	RTX 3050 Ti

TABLE I: Training parameters for SRResNet and SRGAN.

Table I shows the hyperparameters used for training SRResNet and SRGAN. We ensure consistency and fair comparison by adhering to the recommended settings from the original paper [3].

Evaluation on test set: We evaluate the trained SRGAN model using PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index) on the test set (Figure 11).

Model	Average PSNR	Average SSIM
SRGAN	28.840011	0.785837

Fig. 11: Average PSNR and SSIM of SRGAN Super-resolved images w.r.t. Ground truth

- We have 469 images of dimension 1024x1024 in test set.
- Low-Resolution Conversion: The images are downscaled to 256×256 to create low-resolution (LR) inputs.

- The LR images are given as inputs to SRGAN to get Super-resolved outputs (4x upscaled).
- The super resolved images are compared with the original HR images using PSNR and SSIM.

We compare our results with standard upsampling methods such as Bicubic and Nearest Neighbor. Among all approaches, SRGAN produces results that come closest to the ground truth (GT), as observed qualitatively. This is further validated by the quantitative metrics, where PSNR and SSIM values indicate superior performance of SRGAN over the alternatives.

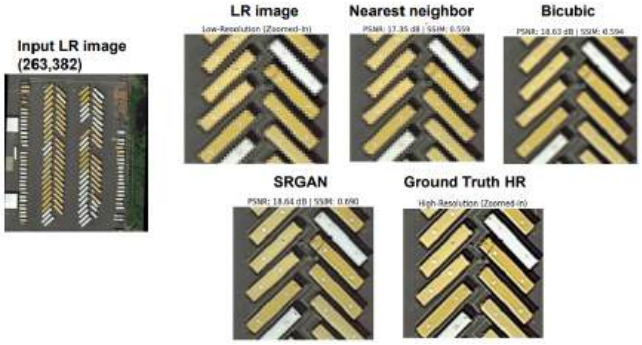


Fig. 12: Example1: Comparing SRGAN performance with traditional upsampling methods

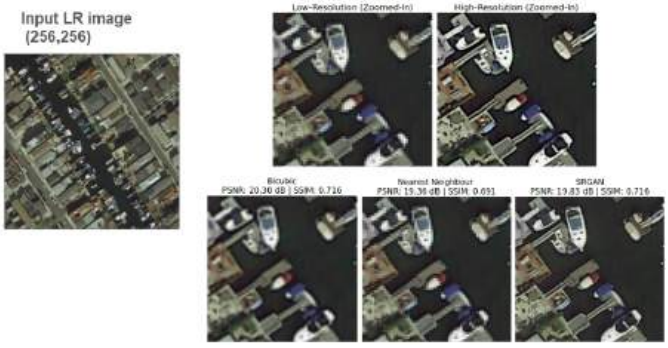


Fig. 13: Example2: Comparing SRGAN performance with traditional upsampling methods

Figures 12, 13 and 14 illustrate three examples where Bicubic and Nearest Neighbor upsampling introduce artifacts, such as blurring, pixelation, and a lack of fine detail. These methods result in noisy and less visually pleasing images, especially in regions with high-frequency textures. In contrast, SRGAN produces results that are closest to the high-resolution (HR) ground truth.

While PSNR and SSIM values for all three techniques might appear close, the visual comparison clearly demonstrates SRGAN's superiority. However, as discussed in the SRGAN [3] paper, PSNR and SSIM are not the most reliable metrics for perceptual quality assessment. To address this limitation, the authors propose a **human opinion-based evaluation**, where subjective ratings from viewers provide a more accurate reflection of the perceived quality of the generated images.

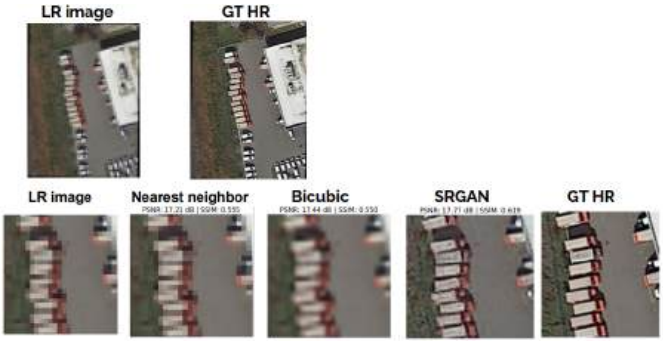
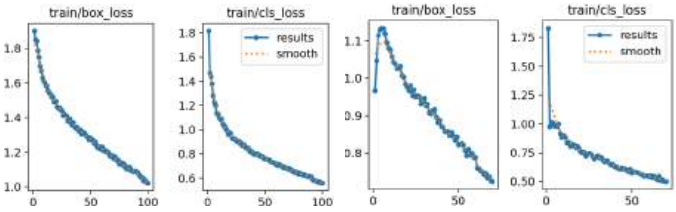


Fig. 14: Example3: Comparing SRGAN performance with traditional upsampling methods

B. Object Detection

YOLO and YOLO-OBB Training: To train YOLO and YOLO-OBB models, we begin by organizing the dataset into separate folders for training and validation, as specified in a YAML configuration file. This YAML file defines the locations of the training and validation folders, the number of classes, class names, and their respective indices, which are required for training the YOLO model. We have around 3200 images for YOLO training and 5000 images for YOLO OBB training. We will use pretrained weights from *yolov11l.pt* and *yolov11l-obb.pt*, both large models with approximately 25 million parameters and a file size of around 50 MB. The YOLO model will be trained for 100 epochs, while the YOLO-OBB model will undergo 70 epochs of training, using an Nvidia 3050 GPU. During training, we monitor the classification loss and bounding box loss, both of which are expected to decrease over time, indicating successful model optimization.



(a) YOLO train loss (b) YOLO-OBB train loss

Fig. 15: Training loss for YOLO network

Result: After training YOLO and YOLO-OBB models, we get following results on test images in Fig. 16

Here, we obtain axis-aligned bounding box results along with the detected classes and their confidence scores. While the results are generally good, in cases where a large number of objects are in close proximity to each other, bounding boxes may overlap. To address this issue and improve visualization, we use Oriented Bounding Boxes (OBB) for better accuracy and clarity.

The results from YOLO-OBB are shown in Fig. 17

In Fig. 17a, we observe that the objects are densely packed. If axis-aligned bounding boxes were used, they would overlap, leading to suboptimal results. This is one of the key advantages

(a) (b)

Fig. 16: YOLO test images result

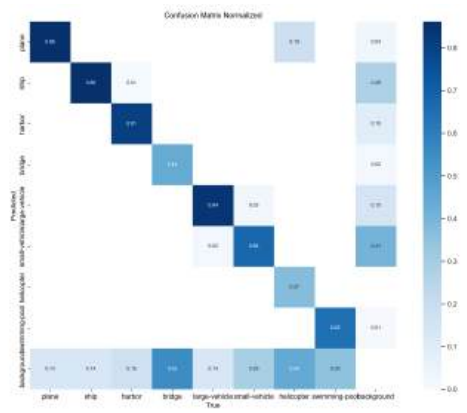
(a) (b)

Fig. 17: YOLO OBB test images result

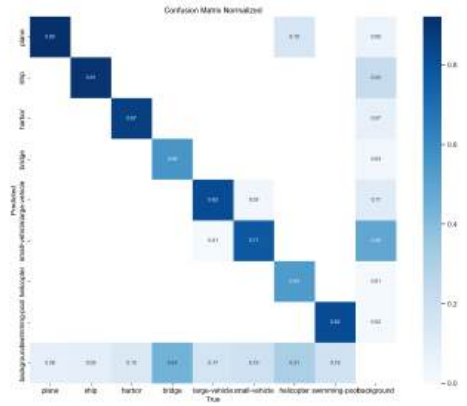
of YOLO-OBB, as it prevents such overlap during training. Additionally, in Fig. 17b, although the objects are spatially separated, they are elongated. If simple axis-aligned bounding boxes were drawn around them, a large portion of the bounding box would cover background areas rather than the actual objects. In contrast, Oriented Bounding Boxes (OBB) are more effective in training, as they focus on regions containing relevant features, improving the accuracy of detection.

Performance: We can analyze the performance of models using a confusion matrix. In Fig. 18, we can observe the confusion matrix for both YOLO and YOLO-OBB. For each class, YOLO-OBB appears to provide better performance. As explained earlier, this is due to YOLO-OBB's superior feature detection compared to YOLO. However, there are some classes, like "bridge," that perform poorly in both confusion matrices. This can be attributed to dataset cropping; since the dataset is cropped, large objects and their corresponding classes are not fully captured, which results in poor performance for this class. For other classes, such as "helicopter," the low number of instances leads to less accurate predictions. Additionally, we notice that many true and predicted classes are labeled as background. This occurs because images might contain small objects, like vehicles, but we don't have labels for them.

Finally, we can examine the precision-recall curve for both cases in Fig. 19. The area under the curve shows better



(a) YOLO Confusion Matrix



(b) OBB Confusion matrix

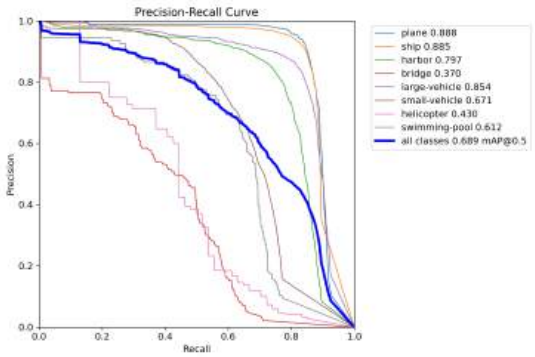
Fig. 18: Confusion matrices

performance for YOLO-OBB across all classes. While some classes, such as "swimming pool" and "helicopter," are not performing well, we can still observe overall good results.

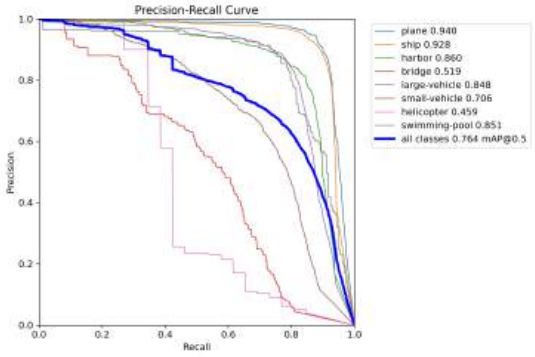
V. Bonus Feature: Streamlit-Based GUI Integration

To enhance the user experience, we have developed a Streamlit-based graphical user interface (GUI) that allows users to interact with the Super-resolution and object detection pipeline.We have also uploaded our files on **Github** for training Super resolution, object detection and GUI files. The GUI supports the following features:

1) **Image Upload and Model Selection:** The user can upload an image from the test set and choose between two object detection models, YOLOv11 and YOLOv11-OBB, for inference. Additionally, users can select between two super-resolution models, SRGAN or SRResNet, to observe differences.
2) **Super-Resolution Results:** After the image upload, the system first shows the super-resolution results using the selected model (SRGAN or SRResNet). The results include the PSNR and SSIM values for the super-resolved image, compared with the ground truth high-resolution image.



(a) YOLO PR curve



(b) OBB PR curve

Fig. 19: Precision Recall curves

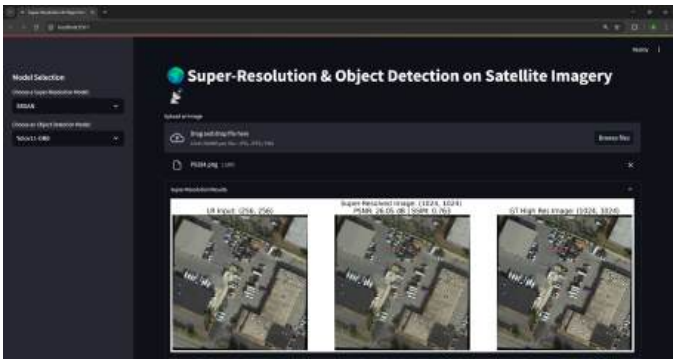


Fig. 20: SRGAN results visualization in GUI

3) **Object Detection on Super-Resolved Image:** Next, we perform object detection using the selected YOLO model on the super-resolved image. The detected objects are displayed along with their bounding boxes and corresponding class labels.
4) **Comparison of Detections on Low-Resolution vs. Super-Resolved Image:** In the third step, we compares the results of object detection on the input low-resolution image (256x256) with the results on the super-resolved image (1024x1024). The comparison includes: !ht
   - Comparing the number of **bounding boxes with a confidence score** $>= 0.5$ for both the low-resolution and super-resolved images.

Fig. 21: YOLOv11-OBB results visualization in GUI

   - Comparing **average confidence score for all detections** in both the low-resolution and super-resolved images.

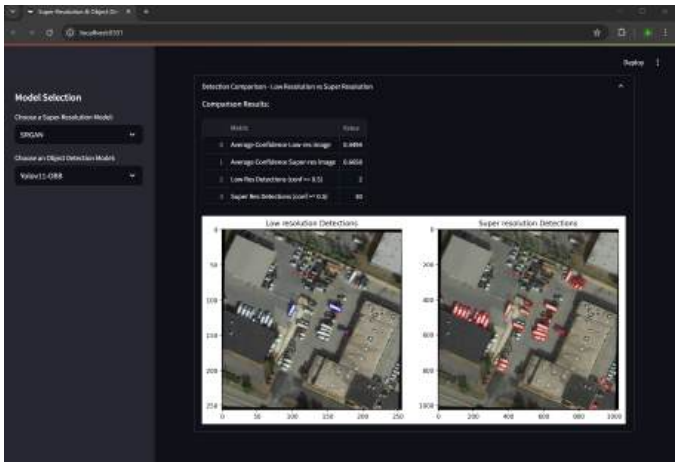


Fig. 22: Comparing OBB results on Low resolution and Super-resolved image

This integration allows users to visually assess the improvements in object detection performance achieved by SRGAN's super-resolution on real test data.
We have uploaded our files on **Github** for training Super resolution, object detection and GUI files.

VI. Conclusions

In conclusion, this project addressed key challenges in remote sensing image analysis, including weak object detection in complex environments, size variation, random orientations, and the difficulty of detecting tiny objects with small dimensions in satellite imagery containing hundreds of objects.

We successfully integrated Super-Resolution techniques with Object Detection models to improve satellite image analysis. By leveraging SRGAN, we upscaled low-resolution images by 4 times, improving image quality and addressing issues like blurriness. Object detection was performed using YOLOv11 and YOLOv11-OBB, with significant improvements in performance when applied to super-resolved images, especially for detecting small, densely packed objects.

We used pretrained weights from MS COCO 14 for SRGAN/SRResNet and YOLOv11 models, as training from scratch was inefficient, requiring extensive iterations, epochs, and computational resources. Pretrained weights reduced training time and improved performance significantly.

Key findings include a marked improvement in object detection accuracy with SRGAN-enhanced images. The comparison between low-resolution and super-resolved images showed SRGAN's ability to restore fine details, leading to clearer detection boundaries. YOLOv11-OBB excelled in detecting objects at various orientations and minimizing bounding box overlap in crowded scenes.

Despite these successes, the study had limitations. The models struggled with extremely small objects in highly cluttered environments, where occlusion and poor image quality led to misdetections. Additionally, while SRGAN improved visual quality, metrics like PSNR and SSIM did not always align with human perception, suggesting that more perceptual evaluation methods could be valuable for future work.

VII. Summary of contributions

- Manpreet Singh: I implemented and trained the SRGAN and SRResNet models for super-resolution, fine-tuning them to achieve optimal performance specifically for DOTA satellite imagery. To assess the models' effectiveness, I evaluated their performance using PSNR and SSIM metrics. Additionally, I integrated the results into a user-friendly Streamlit-based GUI, enabling easy interaction and visualization. I also conducted a detailed performance comparison between low-resolution and super-resolved images, focusing on their impact on object detection tasks.

  I have also researched object detection methods for satellite imagery and experimented with different datasets to identify the best approach. After evaluating options, I selected YOLO and YOLO OBB models. I prepared labels for the cropped dataset and trained both models for 10-15 hours using pretrained weights on a local PC. I also trained the models from scratch to compare performance, though the results were suboptimal. Finally, I conducted testing and performance evaluation for object detection results and contributed to the development of the GUI.

- Anwesha Samaddar: I prepared and preprocessed the dataset for super-resolution by handling tasks such as resizing, data augmentation, and creating high-resolution and low-resolution image pairs. I ensured the consistency and compatibility of the preprocessed dataset with the SRGAN and SRResNet models. Additionally, I verified that the input data met the specific requirements of the training pipeline, which contributed to the overall quality, accuracy, and reliability of the dataset.

  I explored various datasets used in satellite imagery and analyzed the specific challenges associated with preprocessing object detection data. I examined the techniques required for efficient data preparation while considering the limitations of our computational resources. Additionally, I investigated methods to optimize data processing, such as leveraging data augmentation, downsampling, and applying dimensionality reduction, in order to balance performance and computational efficiency.

Through this project, we learned how adversarial training enhances image quality, particularly for super-resolution tasks. Using pre-trained checkpoints like MS COCO 14 for SRGAN/SRResNet gives solid results, but training from scratch may yield better performance. We observed SRGAN's process, where the generator creates high-resolution images and the discriminator evaluates them using content, adversarial, and perceptual losses. Additionally, we found that both SRGAN and YOLOv11 require high GPU resources, with powerful hardware being essential for optimal performance.

For YOLOv11, using oriented bounding boxes (OBB) improved accuracy for rotated images compared to standard YOLOv11, but it still relies on pre-trained checkpoints for state-of-the-art results.

References

[1] C. Dong, C. C. Loy, K. He, and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," 2015. [Online]. Available: https://arxiv.org/abs/1501.00092.

[2] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," *arXiv preprint arXiv:1511.04587*, 2016. [Online]. Available: https://arxiv.org/abs/1511.04587.

[3] C. Ledig et al., "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 105-114, doi: 10.1109/CVPR.2017.19.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 2016. [Online]. Available: https://arxiv.org/abs/1506.01497.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," arXiv, 2016. [Online]. Available: https://arxiv.org/abs/1506.02640.

[6] R. Varghese and S. M., "YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness," 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS), Chennai, India, 2024, pp. 1-6, doi: 10.1109/ADICS58448.2024.10533619.

[7] R. Khanam and M. Hussain, "YOLOv11: An Overview of the Key Architectural Enhancements," 2024. [Online]. Available: https://arxiv.org/abs/2410.17725.

[8] X. Yu, M. Lin, J. Lu, and L. Ou, "Oriented object detection in aerial images based on area ratio of parallelogram," *arXiv preprint arXiv:2109.10187*, 2021. [Online]. Available: https://arxiv.org/abs/2109.10187.

[9] Ultralytics, "Ultralytics YOLOv11," 2024. [Online]. Available: https://docs.ultralytics.com/models/yolo11/s. Accessed: 21-Oct-2024.

[10] J. Ding, N. Xue, G.-S. Xia, X. Bai, W. Yang, M. Yang, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "Object detection in aerial images: A large-scale benchmark and challenges," IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1–1, 2021, doi: 10.1109/TPAMI.2021.3117983.