Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

DLRV Project Proposal

Shadow Casting Object Segmentation

Sai Mukkundan Ramamoorthy
Shrikar Nakhye

Supervised by

Prof. Dr. Sebastian Houben

September 2025

1 Introduction

Semantic segmentation in aerial imagery is a critical task for smart city planning, urban monitoring, and emergency response. This project explores the segmentation of shadow-casting objects (e.g., buildings, trees) from aerial RGB images, focusing on the trade-off between segmentation accuracy and computational efficiency.

The work was carried out in the context of the Hackathon about Segmentation of Aerial Images / Satellite Images by City of Bonn, with the aim of producing a lightweight segmentation pipeline that can operate under real-world time and hardware constraints.

2 Objectives

- Evaluate deep learning segmentation models (U-Net, Mask-RCNN, YOLO).
- Optimize models for speed and efficiency while maintaining accuracy.
- Perform inference on real aerial images of Bonn City.
- Benchmark models with metrics: IoU, F1, inference time, GPU memory usage.

3 Dataset

The dataset consists of high-resolution aerial imagery provided by the City of Bonn (2024)[3]. Originally captured with an *IGI UrbanMapper-2* camera at 2.5 cm/pixel resolution, the data was downsampled to 10 cm/pixel to comply with privacy regulations. The downsampled dataset (RGB only) is reduced to approximately 70 GB, allowing efficient training and inference on mid-range GPUs.

Preprocessing steps included:

- Cropping into smaller patches for training.
- Removal of NIR channel.
- Integration of hackathon crowd-sourced annotations[3].

4 Hardware and Environment

- OS: Linux (Kernel 6.8.0)
- CPU: Intel Core i5-12400 (6 cores, 12 threads)
- GPU: NVIDIA GeForce RTX 4070 (16 GB, CUDA 12.5) & NVIDIA GeForce RTX 4050 (6 GB, CUDA 12.8)
- Frameworks: PyTorch & Tensorflow v1.5

5 Methodology

6 Model Architectures

- **U-Net:** A classical encoder-decoder segmentation architecture originally designed for biomedical image segmentation [6]. It consists of a contracting path (encoder) that captures context by progressively downsampling the input, and an expansive path (decoder) that enables precise localization through upsampling and concatenation with high-resolution features from the encoder (skip connections). These skip connections allow U-Net to combine semantic information (from deep layers) with fine-grained spatial details (from shallow layers), making it highly effective for pixel-level prediction tasks.

  Why U-Net for this task?

  - Performs well with limited training data due to efficient use of context and localization.
  - Skip connections preserve fine spatial details, crucial for distinguishing small or thin structures.
  - Lightweight compared to more complex models, enabling faster training and inference.
  - Proven strong performance across domains, making it a reliable baseline for remote sensing and 3D vision tasks.

- **Mask R-CNN**: An extension of the Faster R-CNN model, Mask R-CNN is an instance segmentation framework that generates both a bounding box and a pixel-level mask for each distinct object in an image [7, 8]. It is based on a Feature Pyramid Network (FPN) and a ResNet backbone [1, 7].
- **Architecture**: The model is composed of several key stages [8]:
  - **Backbone**: A ResNet-50 or ResNet-101 network acts as a feature extractor. It processes the input image through a series of convolutional layers to generate feature maps at multiple scales.
  - **Region Proposal Network (RPN)**: This network scans the feature maps generated by the backbone and proposes a set of Regions of Interest (ROIs)—areas of the image that are likely to contain an object.
  - **Head Units**: Each ROI is passed to three parallel branches, or ”heads,” for final processing:
    1. **Class Head**: Performs object classification, assigning a label (e.g., ”tree”) to the object within the ROI.
    2. **BBox Head**: Refines the coordinates of the bounding box to tightly enclose the detected object.
    3. **Mask Head**: A fully convolutional network that generates a binary mask for each ROI, segmenting the object at the pixel level.

Why Mask R-CNN for this task?

- **Instance-level Segmentation**: Unlike models that perform semantic segmentation (classifying every pixel into a category), Mask R-CNN provides instance segmentation. This is critical for tasks like counting individual trees, as it generates a separate mask for each object instance [7].
- **Rich Feature Extraction**: The use of a deep backbone like ResNet combined with an FPN allows the model to detect objects at various scales, which is essential for aerial imagery where object sizes can vary significantly.

  - **Transfer Learning**: The model can be initialized with weights pre-trained on large datasets like MS COCO. This approach, used in the project, leverages existing knowledge to achieve better performance and faster convergence on a custom dataset [2, 7].
  - **Multi-Task Learning**: By simultaneously training for classification, bounding box regression, and mask generation, the model develops a more robust understanding of the objects, which can improve the accuracy of all three tasks [8].

6.1 Training Strategy: U-Net

- **Loss Function:** Cross-Entropy Loss.
  - Suitable for multi-class segmentation (foreground vs. background).
  - Penalizes pixel-level misclassification and provides stable gradients.
  - Efficient and supported in PyTorch.
- **Training Approach:**
  - Dataset split: 80% training, 20% validation using `train_test_split`.
  - Parameters updated via backpropagation with Adam optimization.
  - Validation monitored with IoU and F1-score.
  - Metrics and system resource usage logged for reproducibility.
  - Training for 20 epochs (balanced cost vs. convergence).
- **Evaluation Metrics During Training:**
  - Intersection over Union (IoU).
  - F1-score (precision-recall balance).
- **Hyperparameters:**
  - Learning Rate: 1×10^{-4}
  - Optimizer: Adam

  - Batch Size: 4
  - Number of Epochs: 20
  - Image Size: 512×512
  - Encoder Backbone: ResNet-34 pre-trained on ImageNet

- **Why This Strategy?**
  - Balances stability (small LR, Adam) with speed (transfer learning).
  - Cross-Entropy provides a strong baseline.
  - IoU and F1 ensure segmentation quality is captured beyond pixel accuracy.
  - Hyperparameters tuned for limited GPU memory.

6.2 Training Strategy: Mask R-CNN

- **Loss Function: Cross-Entropy Loss** The default loss function for the mask head is Binary Cross-Entropy [8].
  - **Suitability**: It is well-suited for the pixel-level classification task within the mask head, where each pixel in a proposed region is classified as either belonging to the object (foreground) or not (background).
  - **Function**: It effectively penalizes predictions that are confidently incorrect, providing stable gradients for backpropagation.
  - **Weighting**: In the project's configuration, the `mrcnn_mask_loss` is assigned a weight of 0.5, balancing its contribution with the other losses in the network [7].
- **Training Approach**
  - **Dataset Split**: The dataset is divided into training and validation sets to monitor for overfitting. The project uses separate `train` and `valid` directories [7]. A dataset of 2077 split into 1,444 for training and 633 for validation.

  - **Data Preprocessing**: YOLO-format polygon annotations are converted on-the-fly into binary masks during runtime. This is a memory-efficient approach that avoids storing large mask files on disk [7].
  - **Optimization**: Model parameters are updated via backpropagation. The training log indicates a momentum of 0.9 and weight decay of 0.0005 with an SGD optimizer [7].
  - **Training Strategy**: Training is initiated using pre-trained COCO weights, a form of transfer learning that fine-tunes the model on the custom aerial tree dataset [1][7].

- **Evaluation Metrics During Training**
  - **Average Precision (AP)**: The primary metric for evaluation, calculated based on Intersection over Union (IoU) thresholds. The `eval_ap.py` script is used for this purpose [7].
  - **Intersection over Union (IoU)**: Also referred to as mean Intersection over Union (mIoU), this metric measures the percentage of overlap between the predicted mask and the ground-truth mask, directly evaluating segmentation accuracy [8].
  - **F1-Score**: The harmonic mean of precision and recall, used to evaluate the balance between false positives and false negatives [8][1].

- **Hyperparameters** Based on the project's training log file [7]:
  - **Learning Rate**: 1×10^{-4}
  - **Optimizer**: SGD with momentum (0.9)
  - **Batch Size**: 1
  - **Number of Epochs**: 25
  - **Image Size**: 512×512 pixels
  - **Encoder Backbone**: ResNet-50 & ResNet-101 pretrained on MS COCO Weights

- **Why This Strategy?**

  - **Stability and Convergence**: Starting with pre-trained weights provides a strong baseline, while a small learning rate (1×10^{-4}) allows for stable and fine-grained adjustments to the model's weights during fine-tuning.
  - **Comprehensive Evaluation**: Using IoU and F1-score alongside AP ensures that segmentation quality is measured from multiple perspectives, capturing both pixel-level accuracy and the model's ability to correctly balance precision and recall.
  - **Resource Optimization**: The hyperparameters, such as an image size of 512×512, a batch size of 1, and the on-the-fly mask generation, are tuned to balance performance with the memory constraints of consumer-grade GPUs, as highlighted in the project's documentation and training logs [1, 7].

- **Evaluation Metrics** To ensure fair and comprehensive comparison across different models, the following evaluation metrics were considered:
  - **Intersection over Union (IoU):** Measures the overlap between predicted and ground-truth segmentation masks.
  - **F1 Score:** Balances precision and recall, useful when classes are imbalanced.
  - **Inference Time:** Average time taken to process one image (ms per image).
  - **Training Time:** Total time required for training (measured in hours).
  - **GPU Memory Usage:** Peak VRAM consumption during training and inference.

7 Results and Discussion

7.1 Quantitative Results

Table 1 summarizes the benchmark results.

Model	IoU	F1	Inference (ms)	VRAM (GB)
YOLOv8-seg	0.745	0.854	50	4.21
U-Net	0.742	0.841	52	5.65
Mask-RCNN	0.687	0.354	130	3.27

Table 1: Comparison of models

7.2 Qualitative Results

Figure 1 shows the training curves for U-Net, illustrating the decrease in training loss and the corresponding improvement in validation IoU and F1-score over 20 epochs.

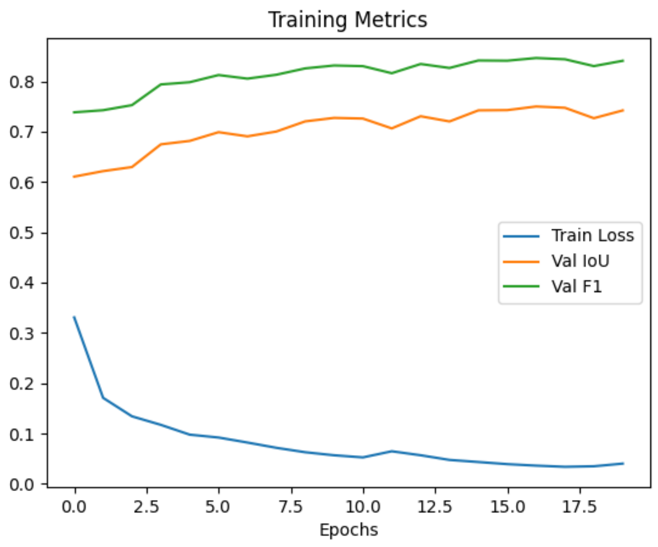

Figure 1: U-Net Training Metrics

Figure 2 shows the training curves for Mask-RCNN, illustrating the decrease in training loss and the corresponding improvement in validation IoU and F1-score over 25 epochs on a Resnet-50 backbone.

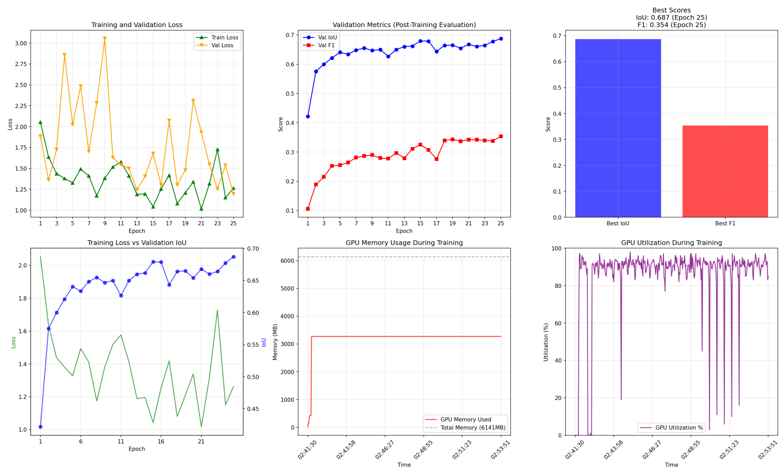

Figure 2: Mask-RCNN Training Metrics

Figure 3 and Figure 4 shows example segmentation outputs for U-Net compared to other models.

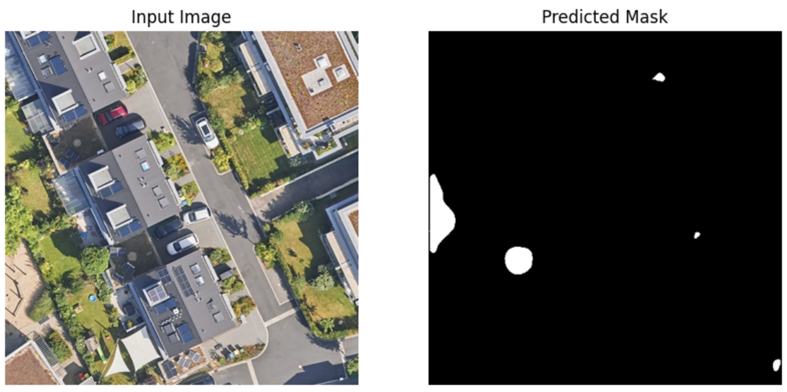

Figure 3: Example segmentation masks U-Net

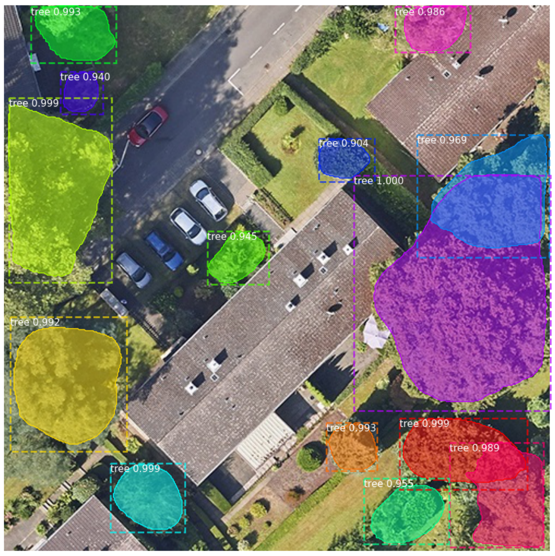

Figure 4: Example segmentation masks Mask-RCNN

7.3 Discussion

- **U-Net:** U-Net achieved a validation IoU of 0.742 and F1-score of 0.841, demonstrating strong pixel-level segmentation performance. The model converged steadily within 20 epochs, with training loss decreasing from 0.33 to 0.04 and validation metrics stabilizing around epoch 15. Inference was efficient at approximately 52 ms per image, with a VRAM footprint of 5.65 GB on the RTX 4050 Laptop GPU. This indicates that U-Net provides a solid balance between performance and computational resource usage.

- **Mask R-CNN:** Mask R-CNN achieved a validation IoU of 0.687 and F1-score of 0.354 after 25 epochs on a Resnet-50 backbone, demonstrating reasonable instance segmentation performance for aerial tree detection. The model showed gradual convergence with training loss decreasing from 2.05 to 1.26, though validation loss exhibited variability indicating some training instability.

Training required approximately 29.5 seconds per epoch with a total time of 12.3 minutes for 25 epochs. GPU memory usage stabilized at 3.27 GB on the RTX 4050 Laptop GPU with 85-97% utilization. While IoU performance was competitive, the lower F1-score suggests challenges in precise instance boundary detection. The instance-based approach provides detailed object-level information but requires higher computational resources compared to semantic segmentation methods.

- **Challenges:** Despite good performance, shadow occlusion and variable sunlight conditions remain challenging for all models. Fine-tuning data augmentation and loss functions may further improve robustness.

8 Documentation and Reproducibility

8.1 Minimum Working Example

Listing 1: Running PrithviVision

```
git clone https://github.com/ItsShriks/
   Shadow_Casting_Object_Segmentation.git
cd PrithviVision
conda env create -f Essentials/dlrv.yml

conda activate dlrv



# Training
python src/train_unet.py

# Inference
python inference_unet.py
```

8.2 Minimum Working Example

Listing 2: Running Shadow Casting Object Segmentation

```
git clone https://github.com/saiga006/
   Shadow_Casting_Object_Segmentation.git
cd Shadow_Casting_Object_Segmentation

# Setup environment for Mask R-CNN
conda env create -f Essentials/maskrcnn_gpu.yml
conda activate maskrcnn_gpu

# Install Mask R-CNN dependencies
cd mrcnn_lib
python setup.py install
wget https://github.com/matterport/Mask_RCNN/releases/
   download/v2.0/mask_rcnn_coco.h5

# Mask R-CNN Training
python samples/aerial_segmentation.py train --dataset ../
   dataset/yolo_dataset --weights coco

# Mask R-CNN Inference
python samples/aerial_segmentation.py test --weights logs/
   mask_rcnn_tree_0025.h5 --image /path/to/test_image.jpg

# Evaluation
python samples/eval_ap.py --dataset ../dataset/yolo_dataset
   --subset valid --weights logs/mask_rcnn_tree_0025.h5
```

8.3 Repository Structure

```
PrithviVision/
├── Essentials/   (environment & setup files)
│   └── dlrv.yml   (Conda environment file)
├── Proposal/   (project proposal documents)
├── dataset/   (training/validation data)
│   ├── unet_dataset/   (for U-Net segmentation)
│   │   ├── images/   (input images)
│   │   └── masks/   (ground truth masks)
│   └── yolo_dataset/   (for YOLO object detection)
│       ├── train/   (training images & labels)
│       └── valid/   (validation images & labels)
├── outputs/ plots/   (results, segmentation outputs)
├── src/   (source code)
│   ├── train_unet.py   (U-Net training script)
│   └── inference_unet.py   (U-Net inference script)
├── .gitignore   (Git ignore configuration)
└── README.md   (project overview & usage guide)
```

[4][5]

8.4 Repository Structure - Mask-RCNN Impl[7] - forked from PrithviVision[4] & Matterport MaskRCNN implementation[1]

```
Shadow_Casting_Object_Segmentation/
├── Essentials/   (environment & setup files)
│   └── maskrcnn_gpu.yml   (Mask R-CNN Conda GPU environment)
├── Proposal/   (project proposal documents)
│   └── DLRV_Project_Proposal.pdf   (final proposal)
├── dataset/   (training/validation data)
│   └── yolo_dataset/   (for Mask R-CNN training)
│       ├── train/   (training images & annotations)
│       ├── valid/   (validation images & annotations)
│       └── dataset.yaml   (dataset configuration)
├── mrcnn_lib/   (Mask R-CNN implementation)
│   ├── mrcnn/   (core Mask R-CNN modules)
│   ├── samples/   (training & evaluation scripts)
│   │   ├── aerial_segmentation.py   (main training script)
│   │   └── eval_ap.py   (evaluation script)
│   ├── logs/   (training logs & saved models)
│   └── mask_rcnn_coco.h5   (pre-trained weights)
├── outputs/   (results & analysis)
│   └── maskrcnn_output/   (Mask R-CNN results)
│       ├── training_metrics.csv   (performance metrics)
│       ├── training_timing.csv   (timing analysis)
│       └── gpu_memory.csv   (GPU usage logs)
└── README.md   (project overview & usage guide)
```

9 Conclusion and Future Work

Conclusion

- Developed and benchmarked a segmentation pipeline for shadow-casting objects in aerial imagery of Bonn City.
- Evaluated three models — U-Net, Mask R-CNN, and YOLOv8-seg — on accuracy, inference time, and GPU memory usage.
- U-Net achieved a strong balance (IoU: 0.742, F1: 0.841) with stable training and efficient inference.
- YOLOv8-seg was the fastest and most resource-efficient (IoU: 0.745, F1: 0.854), suitable for large-scale deployment.
- Mask R-CNN provided instance-level masks but underperformed quantitatively in this setup, though useful for tasks like tree counting.
- Results highlight trade-offs between accuracy, efficiency, and instance-level detail for real-world applications.

Future Work

- Apply model quantization and pruning for deployment on edge devices (e.g., NVIDIA Jetson).
- Extend dataset size and diversity to improve robustness across environments.
- Enhance training with advanced augmentation and loss functions to handle shadows and illumination variations.
- Incorporate multimodal data (LiDAR, NIR) for improved segmentation quality beyond RGB-only inputs.

References

[1] Waleed Abdulla. Mask r-cnn for object detection and instance segmentation on keras and tensorflow. `https://github.com/matterport/Mask_RCNN`, 2017.

[2] Waleed Abdulla. Splash of color: Instance segmentation with mask r-cnn and tensorflow. `https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b`, 2018.

[3] Kai Glassenhap and City of Bonn. Bonn hackathon udp platform, 2025. URL `https://github.com/MrZinken/Hackathon-Bonn`.

[4] Sai Mukkundan Ramamoorthy and Shrikar Nakhye. Prithvivision: Shadow casting object segmentation. `https://github.com/ItsShriks/PrithviVision`, 2025.

[5] Sai Mukkundan Ramamoorthy and Shrikar Nakhye. Prithvivision: Shadow casting object segmentation. `https://git.inf.h-brs.de/snakhy2s/PrithviVision`, 2025.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, volume 9351, pages 234–241. Springer International Publishing. ISBN 978-3-319-24573-7 978-3-319-24574-4. doi: 10.1007/978-3-319-24574-4_28. URL `http://link.springer.com/10.1007/978-3-319-24574-4_28`. Series Title: Lecture Notes in Computer Science.

[7] Sai Mukkundan Ramamoorthy and Shrikar Nakhye. Shadow casting object segmentation, 2025. URL `https://github.com/saiga006/Shadow_Casting_Object_Segmentation/tree/maskrcnn_model`.

[8] Frederik Østerby Hansen, Mikkel Vestergaard Hem, and Matias Dahlin Holst. Mask r-cnn for segmentation of aerial data with edge aware loss, 2020.