

Satellite Image Classification for COTS-Based Satellites

Project Overview

This project develops an efficient image classification system designed to be deployed on Commercial Off-The-Shelf (COTS) based satellites equipped with cameras. The primary objective is to **optimize satellite-to-Earth communication bandwidth** by intelligently determining which captured images are worth transmitting back to Earth and which can be safely dismissed on-board.

By running this lightweight classifier directly on satellite hardware, we can:


- Reduce unnecessary data transmission costs
- Preserve bandwidth for high-value imagery
- Enable autonomous decision-making in space-based imaging systems
- Maximize the scientific and commercial value of satellite missions

The classifier categorizes images into three main classes: **cloudy**, **land**, and **water**, with an additional **space** classification for dark/empty regions.

Architecture Selection

After extensive experimentation with various deep learning architectures, we determined that a **custom CNN** provides the optimal balance of accuracy and computational efficiency for our specific use case.

Architectures Tested:

- **MobileNetV2**: While efficient, showed suboptimal performance on our specific satellite imagery dataset
- **EfficientNet-B0**: Good accuracy but higher computational overhead than desired for satellite deployment
- **Custom CNN** : Best performance with minimal resource requirements, specifically designed for 64x64 pixel tiles

Our final architecture features:

- 4 convolutional blocks with batch normalization and ReLU activation
- Progressive channel expansion (32→64→128→256)
- MaxPooling for spatial dimension reduction
- Fully connected classifier with dropout for regularization
- Optimized for 64x64 input images

Data Augmentation Strategy

We employed **Albumentations** library for robust data augmentation to improve model generalization:

```
train_transform = A.Compose([
```

```
A.Resize(height=64, width=64),  
  
A.HorizontalFlip(p=0.5),  
  
A.Rotate(limit=15, p=0.3),  
  
A.RandomBrightnessContrast(),  
  
A.HueSaturationValue(),  
  
A.RGBShift(r_shift_limit=0, g_shift_limit=0, b_shift_limit=90, p=0.8),  
  
A.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),  
  
ToTensorV2()
```

These augmentations apply transformations such as Horizontal Flips, Rotations, Random Brightness & Contrast, Hue and Saturations, RGB Shift (with focus on the blue channel) and finally normalization.

Dataset Challenges and Validation

Initial Dataset Limitations

Our project faced significant challenges due to the **domain gap** between our training data and real-world satellite imagery:

- **Training Data:** Primarily composed of Google Maps imagery and some high-resolution satellite images
- **Target Domain:** Actual satellite imagery captured by COTS cameras (low resolution) in Low Earth Orbit (LEO)

This mismatch initially raised concerns about model generalization and real-world performance.

Real-World Validation

To validate our model's effectiveness, we conducted extensive testing using **actual satellite images captured in space**:

- **Hardware:** Arducam cameras mounted on COTS satellites
- **Orbit:** Low Earth Orbit (LEO) conditions
- **Dataset:** [COTS-based LEO Satellite Images](#)

The validation results demonstrated that despite the domain gap, our model mostly generalizes to real satellite imagery, confirming viability of such a system for deployment on actual satellite missions.

Dataset Setup

Required Dataset

Download the satellite image classification dataset from: [Kaggle: Satellite Image Classification](#)

Data Preprocessing Steps

1. Download and extract the dataset
2. **Important:** Combine the `desert` and `green_area` classes into a single directory called `land`
3. Ensure your final directory structure looks like:

```
data/
├── cloudy/
│   ├── image1.jpg
│   ├── image2.jpg
│   └── ...
├── land/          # Combined desert + green_area
│   ├── image1.jpg
│   ├── image2.jpg
│   └── ...
└── water/
    ├── image1.jpg
    ├── image2.jpg
    └── ...
```

This consolidation reflects the practical reality that satellites primarily need to distinguish between cloud-covered areas, land masses, and water bodies, regardless of specific terrain types.

Installation

```
# Clone the repository via git clone

# Install required dependencies
pip install torch torchvision opencv-python numpy matplotlib scikit-learn seaborn tqdm
albumations
```

Usage

Training

```
python .py
```

Single Image Classification

```
from main import test_image

# Test a single image
prediction, confidence = test_image('path/to/image.jpg', 'image_classifier_64x64.pth')
print(f"Predicted: {prediction} (Confidence: {confidence:.3f})")
```

Large Image Analysis

```
from main import test_large_image

# Analyze large satellite images using tile-based approach

results, confidence, image_type = test_large_image(
```

```

image_path=custom_image_path,

tile_size=64,                # Size of each tile (64x64 pixels)

overlap=0.2,                 # 20% overlap between tiles

confidence_threshold=0.5,    # Only count tiles with >50% confidence

model_path='image_classifier_64x64.pth', # Path to your trained model

visualize=True,              # Show visualization plots

darkness_threshold=30,

)

```

Model Features

- **Lightweight Architecture:** Optimized for resource-constrained satellite hardware
- **Tile-Based Processing:** Handles large satellite images by processing 64x64 tiles.
- **Space Detection:** Automatically identifies and classifies dark/empty space regions
- **Confidence Scoring:** Provides reliability metrics for each classification
- **Mixed Image Analysis:** Generates comprehensive statistics for complex scenes

Contributing

This project is designed for satellite mission deployment. Contributions should focus on:

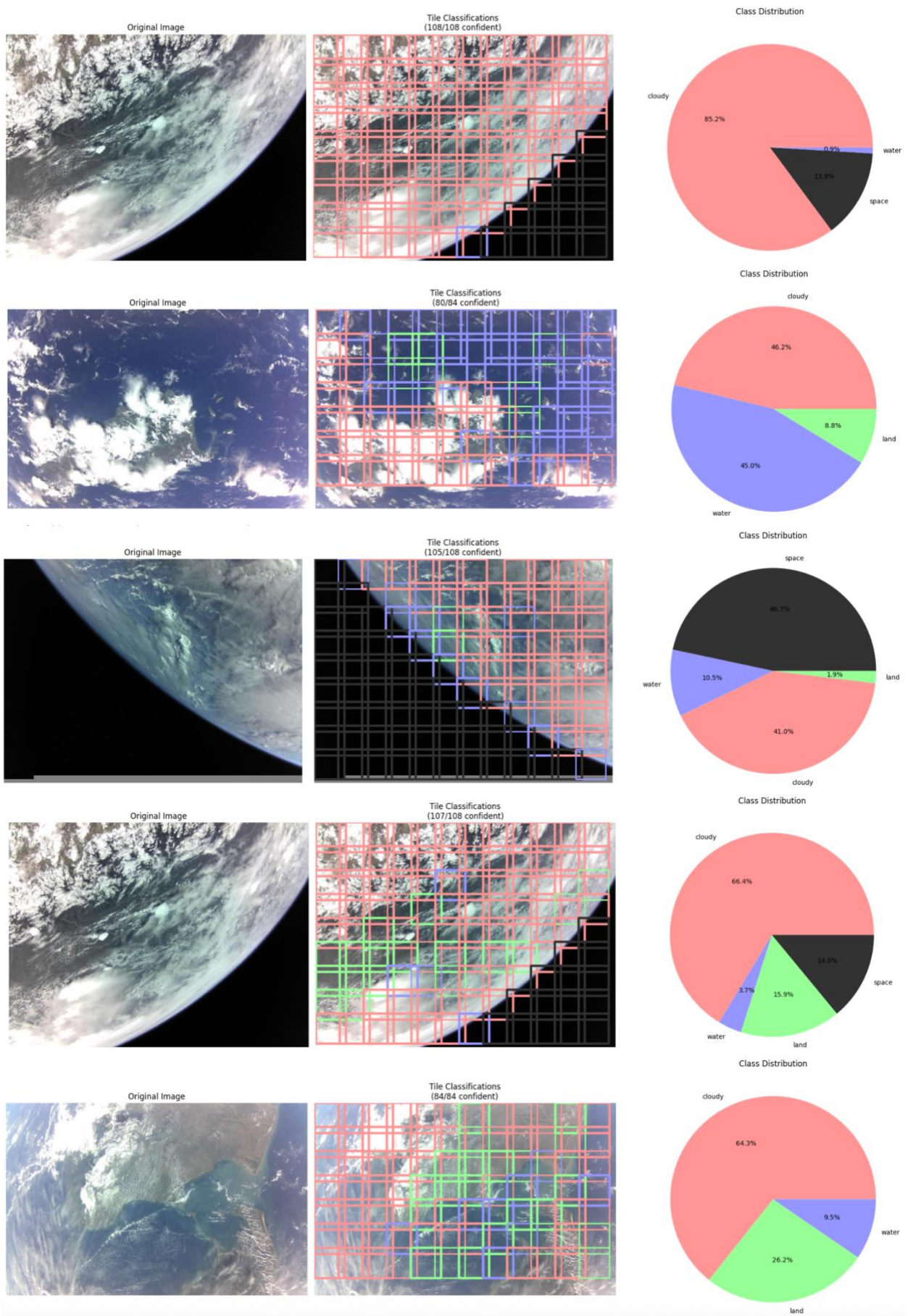
- Improving computational efficiency
- Enhancing generalization to new satellite platforms
- Optimizing for specific COTS hardware configurations
- Extending classification capabilities

Examples: Good Classification

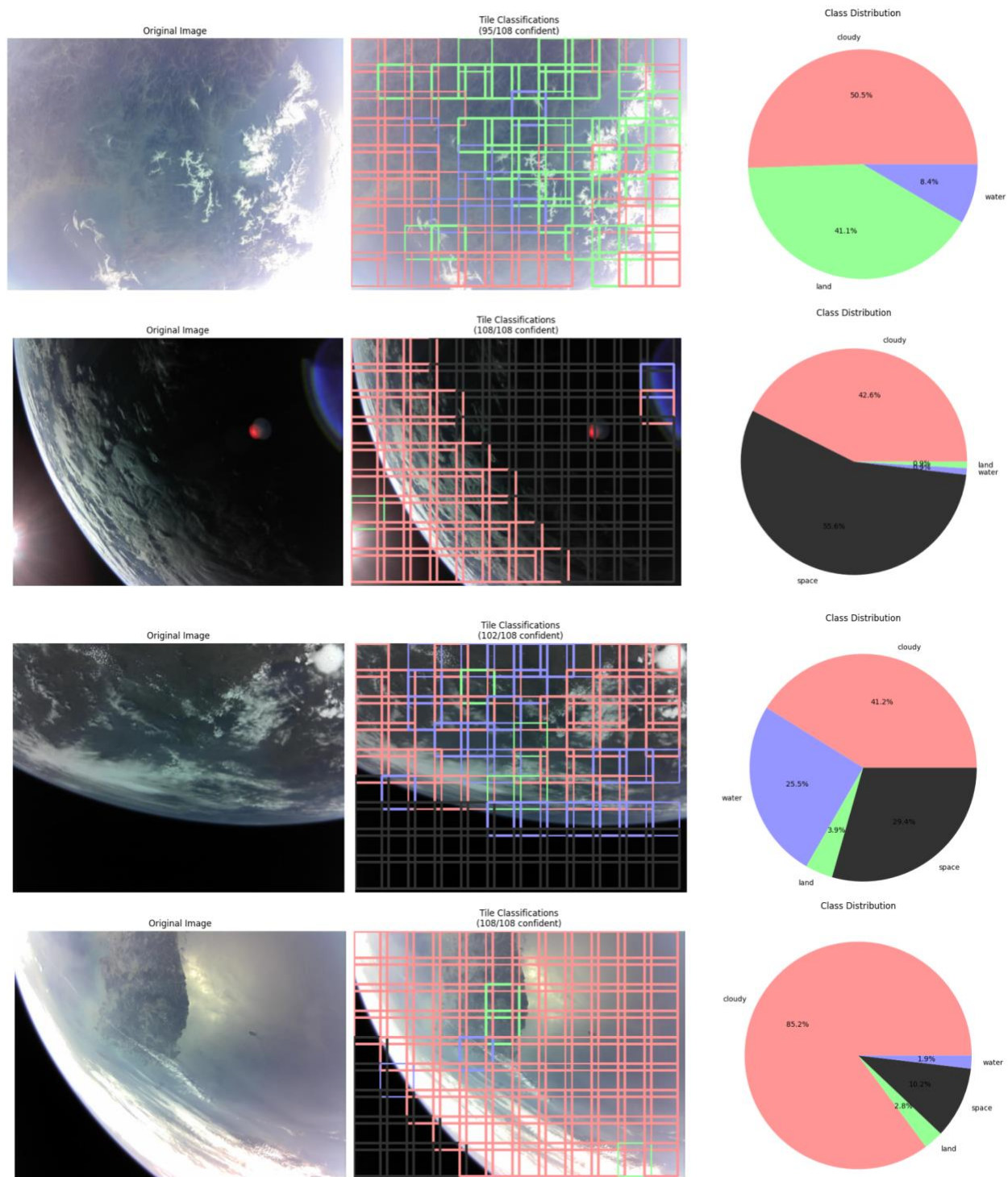
We will now present results of the classifier tested on real images taken by COTS-based camera in Low Earth Orbit.

Overlap is set to 0.2 (20% overlap of each tile). Class Distribution on the right.





Examples: Bad Classification



Conclusion & deployment on future SATLLA iterations

Although the classifier performs well in some cases, it struggles to distinguish between land and water with high accuracy—primarily due to atmospheric blue tinting that causes ground surfaces to resemble water. This visual ambiguity can be challenging even for the human eye. We've experimented with several color transformation techniques to address the issue, but the limitations persist, largely due to dataset constraints.

Despite extensive searches, this is the only suitable dataset we have found. Unfortunately, it lacks adequate representation of both land and water under varying conditions.

For example, here is one image from the "water" class and one from the "land" class. Could you tell which is which?



Spoiler: left is classified as "land" and the right is classified as "water".

For future development—particularly for deployment on real COTS-based, Arduino-compatible, LEO satellite cameras—we recommend building a new dataset via the data available at <https://data.mendeley.com/datasets/5kygfmfdmr/2>, which we have used in this project for determining how the model generalizes to real examples. We advise breaking down each big image into smaller tiles and labeling each by its dominant class; this hypothetical dataset is likely to yield better results.