Deployment Plan for Automatic Solar Panel Detection

1. Objective

The goal of this deployment is to operationalize a deep learning-based semantic segmentation model (U-Net) for detecting solar panels in Sentinel-2 satellite images. The model will be deployed using AWS to enable scalable processing of large satellite images. The system will be accessible via a web-based interface for users to input regions of interest and retrieve detection results.

---

2. Deployment Strategy

2.1 Main Workflow Sequence

The deployment consists of a modular workflow with four key steps:

1. Image Acquisition:

- Fetch Sentinel-2 satellite images from Brazil Data Cube (BDC) based on user-inputted bounding box coordinates.
- Retrieve and tile images into 256x256 pixel patches covering the requested area.

2. Image Preprocessing & Enhancement:

- Apply contrast adjustments and bicubic interpolation (x2) to improve model performance.
- As proved in previous project steps, the image preprocessing leads to better inference results (IoU and F1-score improvements).

3. Model Inference:

- Perform semantic segmentation using the U-Net model deployed in AWS.
- Use AWS Batch Jobs with Docker Containers for inference over large areas.
- Generate binary masks identifying solar panels.

4. Report Generation & Storage:

- Mosaic input images and predictions for a complete visualization.
- Overlay detection results on input images.
- Compute summary statistics (detected area, panel count, etc.).
- Store outputs in AWS S3 and provide access via a web interface.

Then, to orchestrate the sequence with the four modules we will use AWS StepFunction.

3. Technical Implementation

3.1 Model Testing & Optimization

- First, develop and test modular components in Jupyter Notebooks on AWS SageMaker.
- Validate model inference using the best pre-trained weights.
- Ensure robust testing of Acquisition, Enhancement, Prediction, and Reporting Modules.
- Validate inputs and outputs of each module.

3.2 AWS Infrastructure Selection

- AWS Lambda: Lightweight operations (image acquisition, metadata processing).
- AWS Batch + Docker Containers: Scalable model inference for large-area processing.
- AWS S3: Storage of satellite images, processed data, and reports.
- Amazon API Gateway & Lambda: Backend to handle user requests.
- Amazon CloudWatch: Monitor resource usage and optimize cost efficiency.

3.3 Web Interface & API

- Frontend: Simple, user-friendly web app for region selection and report retrieval (HTML, JavaScript).
- Backend: Flask-based API hosted on AWS Lambda deployed using Zappa.
- User Inputs: Bounding box (central coordinates and 2D extensions).
- Outputs: Detection overlay, statistics, and downloadable report.

3.4 Cost Optimization Strategies

- Use SageMaker's instances for cost-effective inference (for example: ml.m3.large).
- Enable automatic instance shutdown after batch processing.
- Optimize AWS Batch resource allocation based on image size and compute demand.
- Use AWS Lambda for lightweight tasks to minimize costs.

---

4. Main Challenges & Considerations

4.1 Handling Large-Scale Inference

- Large satellite images are split into 256x256 pixel tiles for processing.
- Individual tiles are stitched back to form the final prediction (mosaicking).
- Optimize processing pipeline to prevent memory overflows. Key libraries for geoprocessing like ***rasterio*** and ***pillow*** don't write either read images from AWS S3, then we need to work those intermediate images using **BytesIO** lib (managing in-memory files).

4.2 Image Resolution Considerations

- Preprocessing methods like super-resolution increase computational cost.
- Trade-off between image quality improvement vs. processing time.

4.3 Workflow Efficiency

- Introduce a transaction-based workflow to handle multiple requests efficiently.
- Assign a unique ***transaction_id*** to track user requests throughout processing.

4.4 Web-Based Interaction

- Users access the system via a web portal.
- The backend validates user inputs, triggers AWS jobs, and fetches results.
- Final reports (with overlays and statistics) are accessible via AWS S3 links.

---

5. Monitoring & Maintenance

5.1 Logging & Performance Tracking

- CloudWatch Logs: Monitor AWS Lambda, Batch Jobs, and API Gateway requests.
- Error Handling: Implement detailed logging for error debugging.

5.2 Resource Scaling

- Autoscaling AWS Batch jobs based on image size and processing demand.
- Adjust Lambda memory and timeout settings dynamically.

5.3 Security Measures

- Restrict access to AWS S3 buckets using IAM policies.
- Implement HTTPS for secure communication.
- Use API Gateway authentication to prevent unauthorized access.

---

6. Conclusion

This deployment plan ensures a scalable, cost-efficient, and accessible deep learning pipeline for solar panel detection in Sentinel-2 imagery. By leveraging AWS services and best practices, we optimize the system for real-time processing, secure access, and effective resource utilization. The modular approach allows future improvements and easy integration of new features.