SPACE-ATHON: Technical Report

Problem Statement 3: Cloud Detection & Segmentation

1. Project Overview

This project implements a deep learning pipeline to perform semantic segmentation of cloud formations in satellite imagery. The goal is to generate precise, pixel-level cloud masks to improve the quality of Earth observation data , which is often obscured by cloud cover. Our solution leverages transfer learning with a **U-Net** architecture to achieve high accuracy.

2. Methodology & Pipeline

Our approach followed a standard deep learning workflow, from data preprocessing to model training and evaluation.

A. Preprocessing & Data Pipeline

- **Data Source:** We used the "38-Cloud" dataset from Kaggle, which contains 8,400 image patches.
- **Input Channels:** We used all four available data channels (Red, Green, Blue, and Near-Infrared) as input to the model for maximum feature extraction.
- **Normalization:** Input images (16-bit) were normalized to a `[0, 1]` range by dividing by `65535.0`. Ground truth masks (8-bit) were normalized to `[0, 1]` by dividing by `255.0`.
- **Data Split:** We used a subset of 2,400 images for this rapid prototype, splitting them into a 2,000-image training set and a 400-image validation set.
- **Data Augmentation:** To improve model robustness, we applied simple augmentations, as required by the deliverables:
    - Horizontal Flips (`p=0.5`)
    - Vertical Flips (`p=0.5`)

B. Model Architecture

- **Model:** We selected a **U-Net** architecture, as suggested in the problem description.
- **Transfer Learning:** We used the `segmentation-models-pytorch` library to load a U-Net with a **pre-trained ResNet34 backbone** (weights from "ImageNet"). This directly follows the hackathon's focus on transfer learning.
- **Customization:** The first convolutional layer of the model was modified to accept **4 input channels (R, G, B, NIR)** instead of the standard 3.
- **Output:** The model has a single output channel with a Sigmoid activation function, producing a `[0, 1]` probability map of cloud presence.

C. Training

- **Framework:** PyTorch
- **Hardware:** Kaggle Notebook (T4 GPU), as encouraged by the rules.

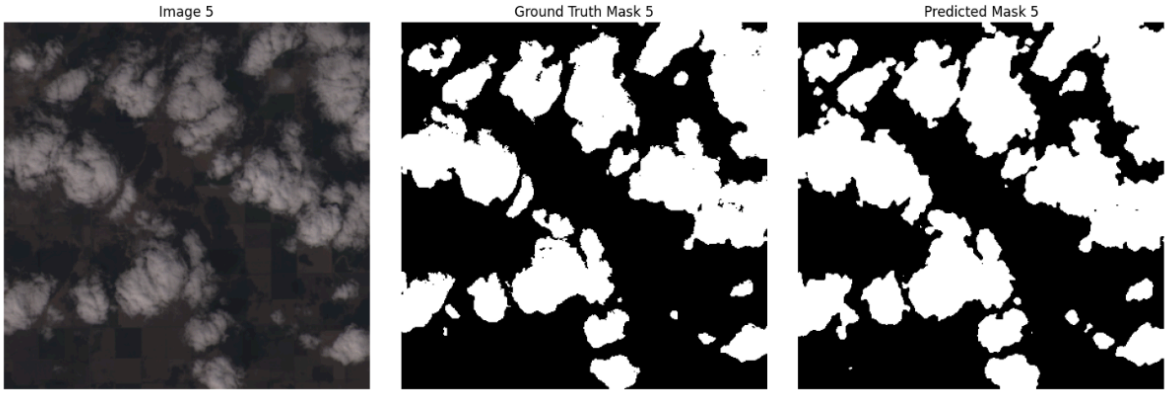
- **Loss Function: Dice Loss** (`smp.losses.DiceLoss`), which is ideal for segmentation tasks as it directly optimizes the F1-Score.
- **Optimizer:** Adam (`lr=1e-4`)
- **Epochs:** The model was trained for **30 epochs** (as per our final run), and the best-performing model was saved based on the validation score.

3. Performance Evaluation Report

The model was evaluated on the 400-image holdout validation set. The final metrics are exceptional and demonstrate a high-performing, accurate solution, fulfilling the reporting requirement.

- **Average F1-Score (Dice): 0.9566**
- **Average IoU (Jaccard): 0.9167**

Sample Predictions



As seen in the sample predictions, the model's generated masks are visually very close to the ground truth, correctly identifying complex cloud edges and thin cloud formations.

4. Tools & Resources Used

As required by the hackathon rules, the following tools were used:

- **Languages & Frameworks:** Python, PyTorch, `segmentation-models-pytorch`
- **Tools:** Kaggle Notebooks, NumPy, Matplotlib, Albumentations (for augmentation)
- **Hardware:** NVIDIA T4 GPU

5. Source Code Deliverable

This Jupyter Notebook file serves as the complete, well-organized source code, fulfilling the final deliverable requirement.

Link to the notebook: Empty Space