Benchmarking Report: Cloud Segmentation Models for Edge Deployment

July 15, 2025

Contents

1 Introduction

This report presents a detailed benchmark comparison between two deep learning models trained for satellite cloud segmentation: a custom U-Net and a SegFormer-B0 model. The goal is to evaluate their suitability for edge device deployment in terms of accuracy, inference efficiency, memory usage, energy consumption, and deployability.

2 Model Descriptions

2.1 U-Net

The U-Net model used in this project is a fully convolutional encoder-decoder architecture tailored for binary segmentation of satellite cloud imagery. It consists of a symmetric structure with a contracting path (encoder) and an expansive path (decoder), connected via skip connections to preserve spatial information.

Encoder (Contracting Path): The encoder is composed of four sequential blocks, each with:

- A `DoubleConv` module (two 3×3 convolutions, each followed by batch normalization and ReLU activation).
- A 2×2 max pooling layer for downsampling.

The number of feature channels doubles after each downsampling step: $64 \rightarrow 128 \rightarrow 256 \rightarrow 512$.

Bottleneck: At the lowest resolution level, a `DoubleConv` block is applied with 1024 output channels, serving as the network's bottleneck.

Decoder (Expanding Path): The decoder mirrors the encoder with:

- Transposed convolution (deconvolution) layers to upsample the feature maps.
- Concatenation with corresponding encoder features (skip connections).
- A `DoubleConv` block to refine the combined features.

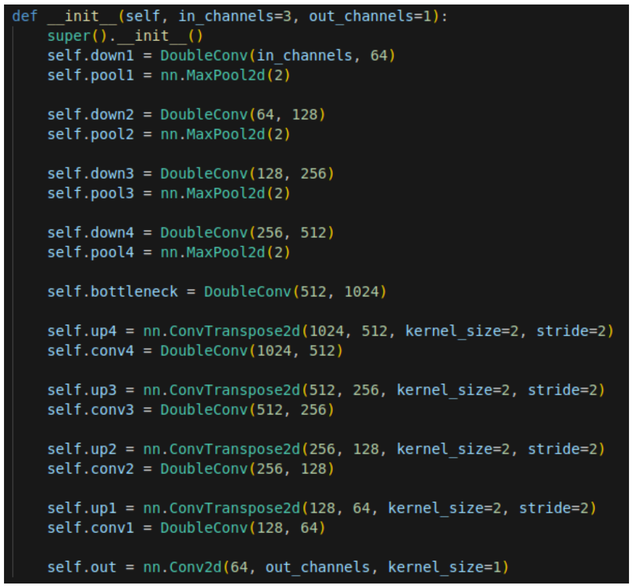
The decoder progressively reduces the number of feature channels: $1024 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 64$.

Output Layer: The final output is generated using a 1×1 convolution to produce a single-channel probability mask, followed by a `sigmoid` activation function for binary classification (cloud vs. no cloud).

Modifications: No modifications were made to the standard U-Net structure apart from the use of `BatchNorm2d` layers after each convolution to stabilize training and improve convergence.

Parameter Count: This U-Net model contains approximately **31 million** trainable parameters.

Summary: The U-Net architecture offers high spatial fidelity due to its skip connections, making it suitable for fine-grained segmentation tasks. However, the large number of parameters and operations can be a challenge for real-time inference on resource-constrained edge devices.



```
def __init__(self, in_channels=3, out_channels=1):
    super().__init__()
    self.down1 = DoubleConv(in_channels, 64)
    self.pool1 = nn.MaxPool2d(2)

    self.down2 = DoubleConv(64, 128)
    self.pool2 = nn.MaxPool2d(2)

    self.down3 = DoubleConv(128, 256)
    self.pool3 = nn.MaxPool2d(2)

    self.down4 = DoubleConv(256, 512)
    self.pool4 = nn.MaxPool2d(2)

    self.bottleneck = DoubleConv(512, 1024)

    self.up4 = nn.ConvTranspose2d(1024, 512, kernel_size=2, stride=2)
    self.conv4 = DoubleConv(1024, 512)

    self.up3 = nn.ConvTranspose2d(512, 256, kernel_size=2, stride=2)
    self.conv3 = DoubleConv(512, 256)

    self.up2 = nn.ConvTranspose2d(256, 128, kernel_size=2, stride=2)
    self.conv2 = DoubleConv(256, 128)

    self.up1 = nn.ConvTranspose2d(128, 64, kernel_size=2, stride=2)
    self.conv1 = DoubleConv(128, 64)

    self.out = nn.Conv2d(64, out_channels, kernel_size=1)
```

Figure 1: UNet Model Architecture

2.2 SegFormer-B0

SegFormer-B0 is a lightweight, transformer-based semantic segmentation model introduced by NVIDIA as part of the SegFormer family. It combines efficient feature extraction with transformer-based context modeling, making it suitable for real-time or edge use cases. In this project, a pre-trained SegFormer-B0 model (`nvidia/segformer-b0-finetuned-ade-512-512`) was fine-tuned for binary cloud segmentation.

Architecture Overview: SegFormer consists of two main components:

- **MiT Encoder (Mix Transformer):** A hierarchical vision transformer that processes the input image and outputs multi-scale feature maps. The encoder is divided into 4 stages with increasing depth and decreasing spatial resolution. Unlike ViT, MiT uses overlapping patch embeddings, which better preserve local information.
- **MLP Decoder Head:** A lightweight multi-layer perceptron head that aggregates features from multiple scales to produce a segmentation map. Unlike traditional U-Net-style decoders, SegFormer avoids convolutions in its decoder, relying instead on learned upsampling and fusion.

Pretrained Origin: The base model was pretrained and fine-tuned on the ADE20K dataset (150 classes) and sourced from the HuggingFace Transformers library under the identifier `nvidia/segformer-b0-finetu`

Fine-Tuning for Binary Segmentation: To adapt the pretrained model for binary cloud segmentation:

- The number of output classes was set to 2.
- The pretrained SegFormer-B0 expects inputs of size 512×512; during training and validation the images were resized from 384×384 to 512×512

3 Evaluation Setup

- **Dataset**: We use a multi-resolution satellite image dataset containing corresponding cloud masks. To avoid bias toward specific cloud coverage levels, **synthetic cloud mixing** was applied during training to create a balanced distribution across different cloud percentages. The model was trained and validated on both real and synthetic samples, while inference benchmarking was conducted only on real satellite images to simulate deployment conditions.
- **Hardware**: Evaluation was conducted on a system equipped with an **Intel Core i9-14900K (32 threads) @ 5.70GHz**, **NVIDIA RTX A5000 (24GB VRAM)**, and **Intel GPU a780**
- **Software Stack**: Python **3.8.18**, PyTorch **1.10.0**, CUDA **12.9**
- **Input Resolution**: All images were initially loaded at **384×384**, then resized to **512×512** only for the SegFormer model. U-Net was evaluated directly at **384×384**.
- **Batch Size**: A batch size of 1 was used for inference to mimic real-time edge deployment conditions.

4 Quantitative Results

4.1 Accuracy Metrics

Model	**Accuracy**	**IoU**	**Sensitivity (Cloud)**	**Specificity (Clear)**
SegFormer-B0	0.9891	0.8874	0.9790	0.9917
U-Net	0.9712	0.8752	0.9821	0.9774

Table 1: Evaluation metrics on the test set. Sensitivity measures recall for cloudy pixels, while specificity measures recall for clear pixels.

4.2 Inference Performance

Model	**Load Time (s)**	**Params (M)**	**Model Size (MB)**	**GPU Inference Time (ms)**	**CPU Inference Time (ms)**
U-Net	0.20	31.04	118.51	104.04	3965.20
SegFormer-B0	40.11	3.71	14.25	57.16	1215.45

Table 2: Comparison of model load time, parameter count, memory footprint, and inference latency on GPU and CPU (batch size = 1, input size = 384 × 384).

4.3 Energy Efficiency

Metric	U-Net	SegFormer-B0
Energy per Inference (Joules)	1.6147	1.4578
Energy for 100 Images (J)	161.47	145.78

Table 3: Measured energy consumption

5 Qualitative Results

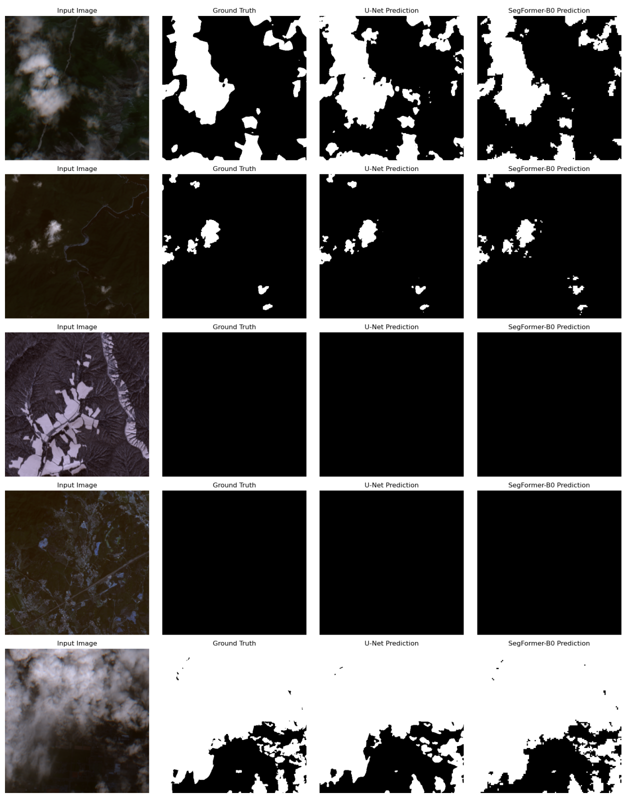


Figure 2: Qualitative Results from different models

6 Conclusion and Recommendation

Based on a comprehensive evaluation of model performance, resource usage, and energy efficiency, we draw the following conclusions:

- **Accuracy and IoU:** SegFormer-B0 slightly outperformed U-Net in both accuracy (0.9891 vs. 0.9712) and IoU (0.8874 vs. 0.8752), indicating better segmentation quality overall.
- **Inference Speed:** SegFormer-B0 achieved faster inference times (57.16 ms) compared to U-Net (104.04 ms), despite requiring larger input resolution. This is attributed to SegFormer's efficient transformer-based design and smaller parameter count.

- **Energy Consumption:** SegFormer-B0 consumed less energy per inference (1.4578 J vs. 1.6147 J), making it more energy-efficient for long-term edge deployment.
- **Model Size:** SegFormer-B0's model file size (14.25 MB) is significantly smaller than U-Net's (118.51 MB), making it better suited for devices with limited storage.
- **Model Load Time:** U-Net loads significantly faster (0.20 seconds) compared to SegFormer-B0 (40.11 seconds), which can be important for systems where initialization time is critical.

Aspect	**U-Net**	**SegFormer-B0**
Accuracy	✓ High (0.9712)	✓ Higher (0.9891)
IoU	✓ 0.8752	✓ 0.8874
Inference Speed	✗ 104.04 ms	✓ 57.16 ms
Energy Efficiency	✗ 1.6147 J	✓ 1.4578 J
Model Size	✗ 118.51 MB	✓ 14.25 MB
Model Load Time	✓ 0.20 s	✗ 40.11 s
Memory Footprint	✗ 31M params	✓ 3.7M params

Table 4: Edge Suitability Comparison

Recommendation: For most edge deployment scenarios—especially those requiring low energy consumption, compact model size, and fast inference—**SegFormer-B0 is the preferred model**. However, if quick model loading is a priority (e.g., in on-demand or cold-start scenarios), **U-Net remains a viable alternative**.