SRIP

Graphics Visualisation and Computing Lab

Robust Semantic Segmentation of Sentinel Images with UNet and Adversarial Uncertainty

---

Authors:

M Srinivasan — IMT2021058
May-July, 2024

Siddharth Kothari — IMT2021019
May-July, 2024

Sankalp Kothari — IMT2021028
May-July, 2024

Contents

Project Report - SRIP

M Srinivasan

August 8, 2024

Contents

1 Overview

During my tenure as a research intern at the GVCL lab under the guidance of Professor Jaya S. Nair, We focused on leveraging Sentinel-2 satellite images to train advanced computer vision models. Our primary objective was to develop and refine models capable of accurately segmenting these images into land and water regions.

In addition to training the segmentation models, we also worked on making the trained model robust by applying adversarial attacks to test and improve the model's resilience against external perturbations. The ultimate goal was to ensure the models could withstand and effectively respond to various types of adversarial threats, therefore improving their reliability and robustness in real-world applications.

2 Work

2.1 Dataset Curation

The initial weeks focused more on obtaining the datasets for the Sentinel 2 images. We had previously trained a UNet model using Sentinel-1 images(the results are shown in Table 2) but then the ground truth (masks) that we used for the same were not accurate. So the final prediction made by the earlier Unet model was not up to the mark. So we had to find a correct ground truth (mask) and then leverage it to train correct vision models. But getting the masks wasn't easy as there was no direct source from where We could download the water masks required for the amazon basin.

To obtain ground truth masks for custom locations, we utilized Google Earth Engine. We first used pre-stored latitude and longitude coordinates to download Sentinel-2 images from the Copernicus website for the specified locations. Using Google Earth Engine, we manually generated the masks by applying the NDWI (Normalized Difference Water Index) and subsequently downloaded these masks.

The first 2 weeks were devoted to getting the correct masks and then downloading them onto our lab system. There were 1263 images for which the masks had to be generated and then downloaded.

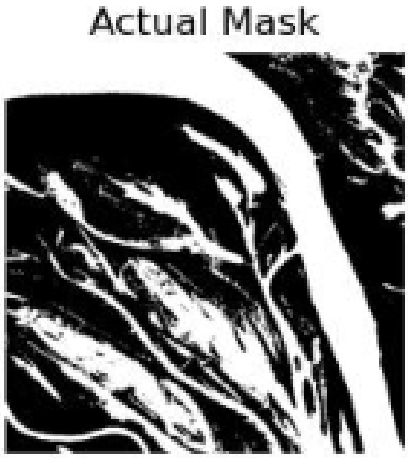


Figure 1: Sample mask generated using the Google Earth Engine and NDWI

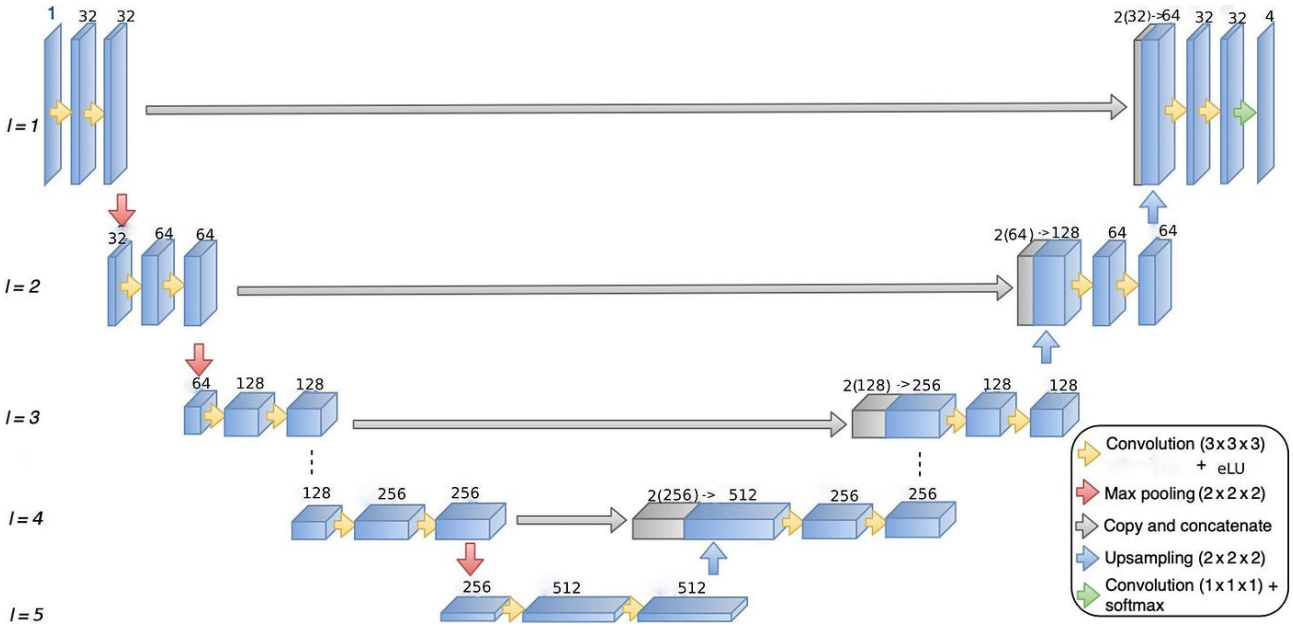


Figure 2: Unet model architecture

2.2 Training Unet Model

2.2.1 Training Data

The masks that were generated earlier(using Google Earth Engine) were then used as the new ground truth for the UNet model.

2.2.2 Training Period

The model was then put to train in the lab system. The training process took for about 16hrs to complete. The validation accuracy gradually increased per epoch from 88 % to 97%.

2.2.3 Results

The results we obtained from the model were much better than what we had obtained through the previous model(trained with wrong masks). Tables 2 and 1 show the results and the comparison between the models (the newly trained model and the previously trained model using Diva-GIS).

Metric	Value
Average IoU	0.8067
Average Pixel Accuracy	0.9530
Average Precision	0.8978
Average Recall	0.8495
Average F1 Score	0.8530
Average Specificity	0.9799

Table 1: Performance metrics of the evaluated model trained using correct ground truth masks from Sentinel-2.

Metric	**Value**
Average IoU	0.5950
Average Pixel Accuracy	0.9011
Average Precision	0.6445
Average Recall	0.5950
Average F1 Score	0.6143
Average Specificity	0.9011

Table 2: Peformance metrics of the evaluated model trained using incorrect ground truth from Diva-GIS.

3 GVCL-Seminar Presentation

During our summer internship at GVCL Lab, each of us had to present papers related to our work and future plans. Since I was working on training Unet models and making them robust to adversarial attacks (mentioned in section 4), I presented the following two papers :

- Single-Step Adversarial Training for Semantic Segmentation [6].
- Label Noise Types and Their Effects on Deep Learning [1].

In the end, we had an open Q&A session, during which I had insightful discussions with Ph.D. students who gave valuable suggestions on my presentation and showed great interest in my work.

4 Adversarial Attacks

4.1 Introduction

Adversarial attacks on neural networks are deliberate attempts to deceive and destroy models. They can range from simple bit flip of pixels to large morphological operations over the input images. These modifications cannot be seen by the naked eye but can have a significant impact on the predictions done by the model. In the context of Semantic Segmentation using UNet models, adversarial attacks can have several specific impacts:

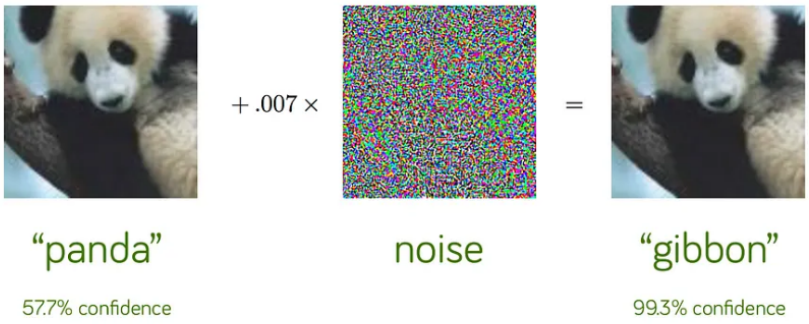
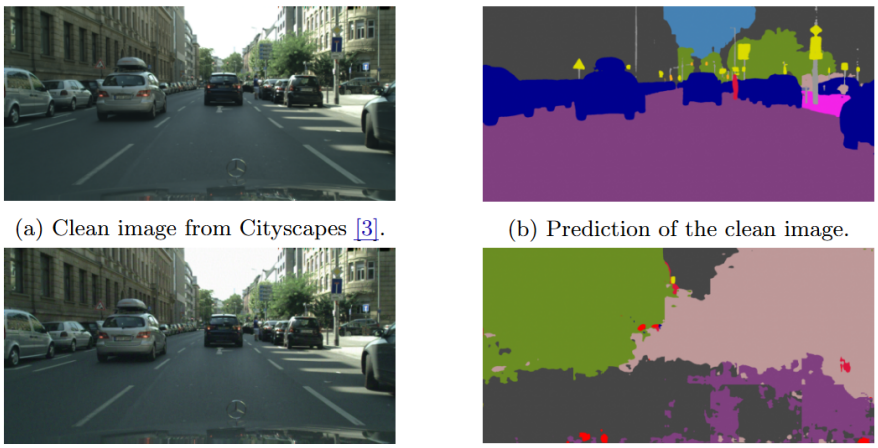


Figure 3: Simple demonstration of what adversarial attack is. A Panda image which was being classified with 57.7 % confidence, after an adversarial attack is now being classified by the same model as Gibbon with 99.3 % confidence.

1. **Misclassification of Pixles** - Adversarial attacks can cause the UNet model to misclassify pixels, leading to incorrect segmentation maps. This can result in the model assigning incorrect labels to regions of an image, which can be particularly problematic in critical applications.
2. **Boundary Errors** - Since UNet models are sensitive to edges and boundaries in images, adversarial attacks can distort these boundaries causing errors in the precise identification of different segment regions.
3. **Degraded Performance Metrics** - Adversarial attacks can significantly degrade performance metrics such as Dice coefficient, Intersection over Union (IoU), pixel accuracy, precision, recall, F1 score and specificity.



(a) Clean image from Cityscapes [3].

(b) Prediction of the clean image.

(c) Adversarial example created with the Basic Iterative Method ($\varepsilon = 0.03$).

(d) Prediction of the adversarial example.

Figure 4: Figure taken from the work of Wiens et al. [6], it shows how the neural networks segments the image after a small perturbation (invisible to the naked eye) is added to the image.

4.2 Reading Material

There were a lot of reading materials that were provided to me to go through and understand about how to implement adversarial attacks. Here is a small overview of them:

In [2], the authors propose a novel defensive transformation for maintaining high classification accuracy under both clean images and adversarial examples. A block-wise preprocessing technique with a secret key is used, enhancing adversarial robustness by obfuscating gradients. Three algorithms were developed: Pixel Shuffling, Bit Flipping, and FFX Encryption. Experiments on CIFAR-10 and ImageNet datasets show that high accuracy is achieved even under adaptive attacks.

In [5], the authors give a short review on the data poisoning angle of adversarial attacks. In such attacks, it is assumed that the attacker can manipulate the training by controlling the train set or by adding data points to the train set. These attacks can be executed by flipping the labels of the data in the dataset or by adding new data with correct labels (clean-label poisoning).

In [3], the authors introduce *MetaPoision*, a first-order method, approximates the bilevel problem by metalearning to craft poisons that fool neural networks. The authors claim that MetaPoison significantly outperforms previous clean-label poisoning methods, is robust across various victim models with unknown training settings, and is versatile, working for both fine-tuning and end-to-end training from scratch. It can achieve arbitrary adversarial goals, such as using poisons from one class to reclassify a target image into another class.

In this work [4], a novel DNN weight attack methodology called Bit-Flip Attack (BFA) is proposed. This method can severely degrade a neural network's performance by flipping a very small number of bits in its weight storage memory system, such as DRAM, using the Row-Hammer attack. An algorithm is developed to identify the most vulnerable bits in DNN weight parameters to maximize accuracy degradation with minimal bit-flips. The Progressive Bit Search (PBS) method, which combines gradient ranking and progressive search, is utilized to identify these vulnerable bits.

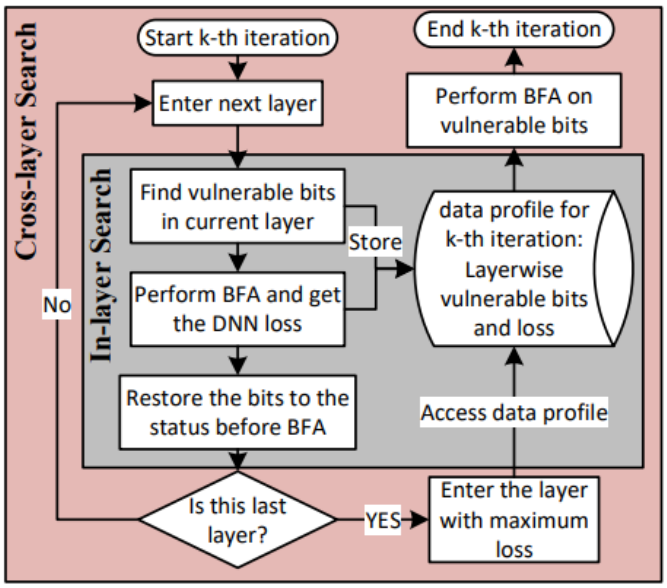


Figure 5: Image taken from Rakin et al. [4]. Flowchart of PBS with in-layer and cross layer search.

Figure 5 shows the flowchart to perform Progressive Bit Search (PBS) with in-layer and cross layer search.

4.3 Implementation

In this project We implemented two types of adversarial morphological attacks namely *Dilation* and *Erosion*.

1. **Erosion** - The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero). So what happens is that, all the pixels near boundary will be discarded depending upon the size of kernel. So the thickness or size of the foreground object decreases or simply white region decreases in the image. It is useful for removing small white noises (as we have seen in colorspace chapter), detach two connected objects etc.

```
import cv2 as cv
import numpy as np

img = cv.imread('j.png', cv.IMREAD_GRAYSCALE)
assert img is not None, "file could not be read, check with os.path.
    exists()"
kernel = np.ones((5,5),np.uint8)
erosion = cv.erode(img,kernel,iterations = 1)

```

Listing 1: OpenCV erosion function used for erosion operation.

2. **Dilation** - It is just opposite of erosion. Here, a pixel element is '1' if at least one pixel under the kernel is '1'. So it increases the white region in the image or size of foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So dilation is done. It is also useful in joining broken parts of an object.

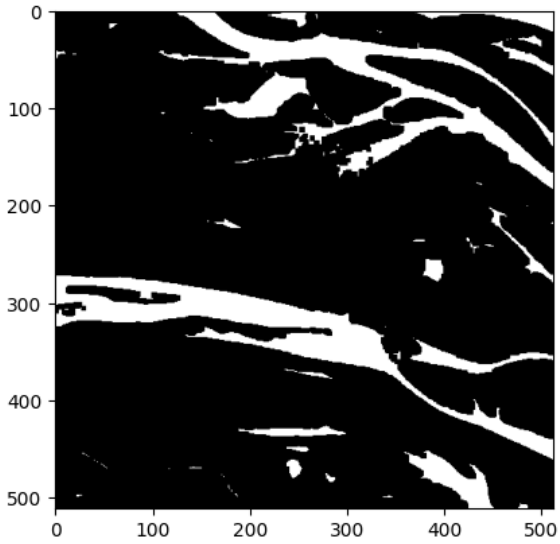
```
import cv2 as cv
import numpy as np

img = cv.imread('j.png', cv.IMREAD_GRAYSCALE)
assert img is not None, "file could not be read, check with os.path.
    exists()"
kernel = np.ones((5,5),np.uint8)
dilation = cv.dilate(img,kernel,iterations = 1)

```

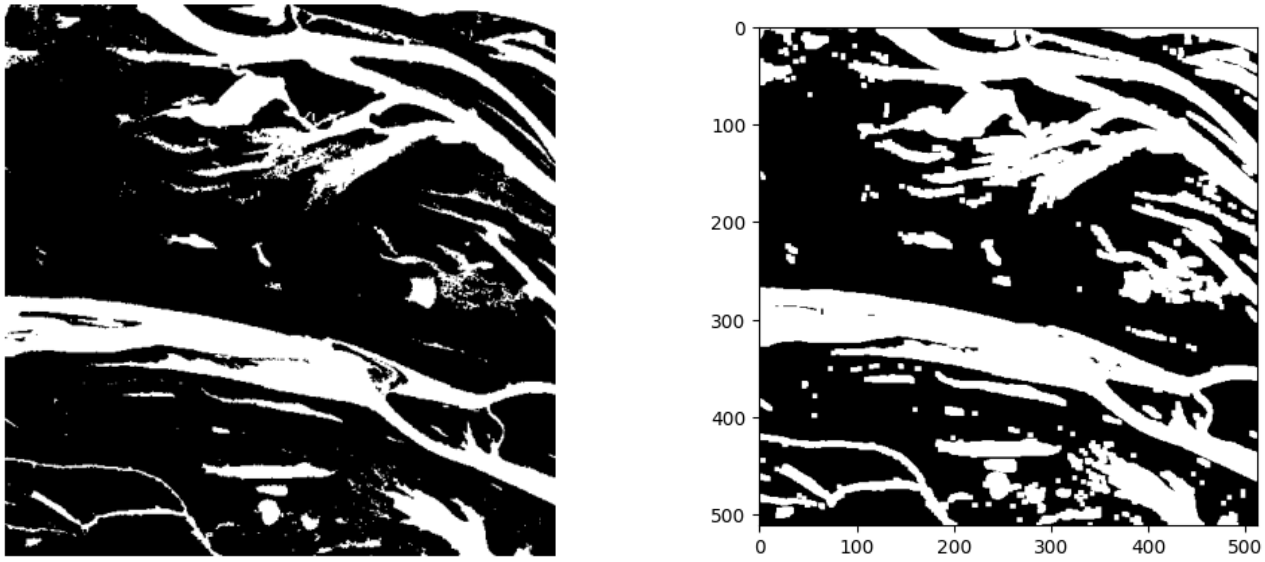
Listing 2: OpenCV dilation function used for dilation operation.

(a) Original Mask



(b) Same mask after erosion operation.

Figure 6: Figure 7a is the actual mask that we obtained from the Sentinel-2 GEE, and Figure 6b is the result when erosion code in Listing 1 is applied on it.



(a) Original Mask **(b)** Same mask after dilation operation.

Figure 7: Figure 7a is the actual mask that we obtained from the Sentinel-2 GEE, and Figure 7b is the result when dilation code in Listing 2 is applied on it.

4.3.1 Dataset curation

One of the first and foremost necessity for training any ML models is the availability of datasets. The corruptions had to be done synthetically with the pre-existing datasets. The existing data was split into two almost equal parts, one for training and the other for testing. The training part was further divided into three almost equal parts. Maks corresponding to *Part 1* were put into erosion operation, masks for *Part 2* with put into dilation operation and then last set was left untouched(no attacks). So in total about 630 images were present in the total train dataset, out of which *erosion, dilation* and the other set contained approximately 210 images.

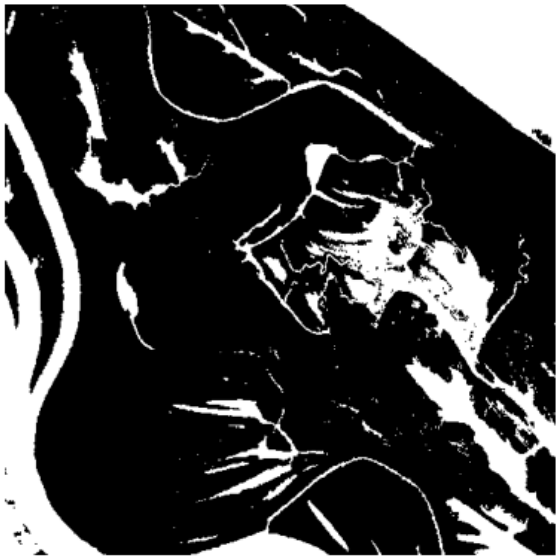
4.3.2 Model training

Then a new *dense Unet* model was trained from scratch for 10epochs with the training set. It took approximately 370 minutes using the Nvidia A4000 GPU. Once the training was completed, the inferences were run on the test set that was created earlier, the infereneces we got from the model were quite good. The Table 3 shows the evaluation metrics, and the Figures 8b and 9b shows the predicted images.

Since the number of pixels that can be corrupted in each of the images vary, instead of keeping a constant fixed value We kept an upper bound upto which the number of pixels can be corrupted or flipped in a particular image.

4.3.3 Results

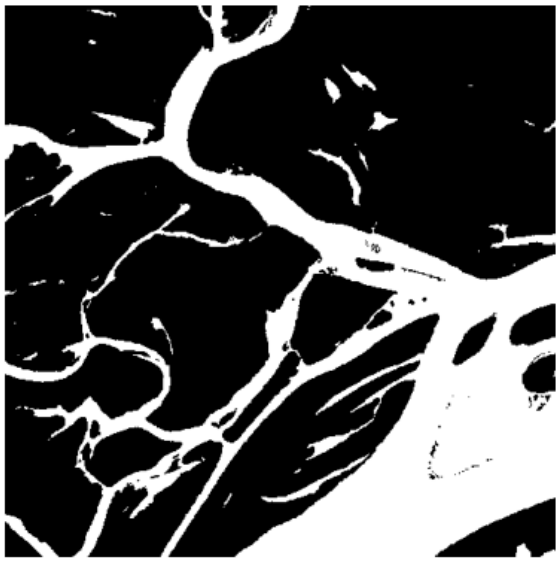
Figures 8 show the samples from the inferences that we got from the model and compares them with the original mask for the same area.



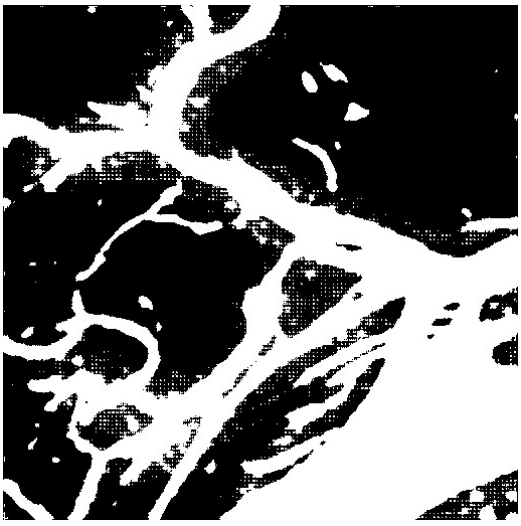
(a) Original Mask

(b) The mask predicted by the model for the same area.

Figure 8: Figure 8a is the actual mask that we obtained from the Sentinel-2 GEE, and Figure 8b is the output the trained adversarial gives for the same area.



(a) Original Mask



(b) The mask predicted by the model for the same area.

Figure 9: Figure 9a is the actual mask that we obtained from the Sentinel-2 GEE, and Figure 9b is the output the trained adversarial gives for the same area.

Table 3 shows the evaluation metrics of the final model that was trained with adversarial attacks.

Metric	Value
Average Dice Coefficient	0.6784
Average IoU	0.7527
Average Pixel Accuracy	0.9357
Average Precision	0.8111
Average Recall	0.8437
Average F1 Score	0.8133
Average Specificity	0.9345
Model's Overall Accuracy	0.9357

Table 3: Summary of the trained adversarial model.

References

[1] Görkem Algan and Ilkay Ulusoy. "Label noise types and their effects on deep learning". In: *arXiv preprint arXiv:2003.10471* (2020).

[2] MaungMaung AprilPyone and Hitoshi Kiya. "Block-wise image transformation with secret key for adversarially robust defense". In: *IEEE Transactions on Information Forensics and Security* 16 (2021), pp. 2709–2723.

[3] W Ronny Huang et al. "Metapoison: Practical general-purpose clean-label data poisoning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 12080–12091.

[4] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. "Bit-Flip Attack: Crushing Neural Network With Progressive Bit Search". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[5] Pooya Tavllali and Vahid Behzadan. "Short Review on Supervised Learning Under Adversarial Label Poisoning and Clean-Label Data Poisoning". In: *2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)*. IEEE. 2023, pp. 378–382.

[6] Daniel Wiens and Barbara Hammer. "Single-Step Adversarial Training for Semantic Segmentation". In: *arXiv preprint arXiv:2106.15998* (2021).