Technical Report

Team: TheSentinels
Members: Thorben Hellweg, Stefan Oehmcke, Ankit Kariryaa, Fabian Gieseke, and Christian Igel

Overview:

This document provides a brief overview of the approach we eventually took. Our best submission is based on an ensemble of different models. In the following, we briefly present the pre- and post-processing steps underlying all models, as well as an overview of the architectures that we examined and the infrastructure used.

Infrastructure:

To ease development, we followed a standard project structure for data science projects, which he hosted on github. To minimize frictions during development, we chose conda as our environment management system, the environment.yaml is contained in the repository. To minimize setup time, we provide a docker image, in which the conda environment is already resolved (`docker pull thllwg/maya-challenge:develop`). The docker image is automatically generated by Github Actions and pushed to the docker registry. Please check the README file of our repository for further information and instructions on how to set up a working copy of our codebase.

Preprocessing:

First, we transformed the provided raw .tiff images to .npy arrays. For that purpose, we provide a utility script that can be called via `make data`. We only utilized LiDAR and Sentinel II images. Both were normalized between 0 and 1. To reduce the data of the Sentinel II images, we created median composites of cloud free pixels.

Augmentations:

We divided our augmentations into normal and “advanced” augmentations. In contrast to the advanced augmentations, the normal augmentations are considered safe and do not drastically disturb the images. Depending on the ensemble member; only normal or both normal and “advanced” augmentations were used.
Each augmentation has an individual chance of triggering:

Normal**:**

1. Random Cropping (always):
    a. Depending on the model this ranged from 256 to 400 pixel on the LiDAR data

2. Vertical flip (50%)
3. Horizontal flip (50%)

Advanced:

1. Random rotation (75%) between 0 and 360 degrees
2. Only Sentinel II: change of brightness, saturation, and contrast (25%)
3. Gaußian blur on LiDAR (25%)
    a. The change was between 0.1 and 2
4. Random additive noise (25%), half the time either
    a. Uniform noise (strength of 0.1)
    b. Normal noise (strength of 0.03)

Models:

Our starting point was the classic **UNet** model from Ronneberger et. al [6]. We used the Adam optimizer and tried various learning rates in the range 1e-2 and 1e-6 [10].

We proceeded to add changes to the UNet that helped in other scenarios (unet_adv in the code):

- Pixelshuffle as upsampling (as used in superresolution paper [1,2])
- Self-attention layer in the first upsampling layer (**sa layer**)
- Small blurring after each upsampling to avoid artifacts
- ResNet-Block as cross-over layer
- Upsampled and concatenated the Sentinel II at the cross-over part of the unet
    - This showed little effect
- Batch normalization in the encoder and spectral normalization in the decoder
    - Batch norm in decoder reduced performance and was replaced by spectral (inspired by GAN papers)
- We smoothed the learning curve by accumulating the gradient over multiple batches (**effective batch size**)
- For regularization we used dropout and **cosine learning rates** with warm restarts

Then, we also added other encoder architecture to later make our ensemble more heterogeneous:

- ResNet [5]: didn't work better and was omitted in favor of the following two.
- **MNASNet [7]**: A relatively small network
- **SWIN-B [3]**: A nearly fully self-attention network (pretrained on segmentation task)

Activation functions:

- ELU (normal UNet, MNasNet) [4,6,7]
- Mish (MNasNet) [7]
- GELU (SWIN-B) [3]

Loss functions:

- Binary cross-entropy: worked consistently
- (Focal) Tversky: didn't converge at all in this case, omitted in final ensemble

- BCE + VGG loss [9]: using the output of the intermediate layers of a ImageNet pre-trained vgg16 model from both, the target and predicted segmentation to have an abstract spatial loss

We further utilized the **Deeplabv3** architecture [8] with pre-trained weights for the ResNet101 encoder [5].

Some ensemble members were represented at multiple points where the validation error was lowest. For only one model, we excluded aguada, which means that we didn't consider the aguada mask during training as well as prediction. Our final ensemble consisted of 10 model configurations with 18 realized models (through checkpoints):

Architecture	Encoder	Loss	Learning rate	Cosine lr	Batch size	Effective batch size	Exclude Aguada	Number of model checkpoints	sa layer	Random crop size
UNet_adv	swin-b	BCE	6e-05	Yes	52	104	No	3	Yes	256
UNet_adv	swin-b	BCE	6e-05	Yes	52	208	No	4	Yes	256
UNet_adv	swin-b	BCE	6e-05	No	52	208	No	1	Yes	256
UNet_adv	swin-b	VGG	6e-05	No	32	96	No	1	Yes	256
UNet_adv	MNasNet	VGG	6e-05	No	6	12	No	1	Yes	256
UNet_adv	MNasNet	BCE	6e-05	No	52	104	No	1	Yes	256
UNet_adv	MNasNet	BCE	3e-04	No	6	6	No	1	No	400
UNet_adv	swin-b	BCE	3e-04	No	30	90	No	1	No	400
DeepLabv3	Resnet101	BCE	3e-05	No	16	64	Yes	1	No	400
DeepLabv3	Resnet101	BCE	3e-05	No	16	16	No	4	No	400

We first tested the validity of our approach using a train-test split of 0.8 and finally trained different models on the full data set, on the basis of which the ensemble is formed.

Postprocessing:

Test time augmentation

We used test time augmentation where we flipped each test image horizontally, vertically, and rotated it between 0 - 270 degrees with a step of 90 degrees. These operations and their combinations meant we predicted each test image 16 times for each model, and the final output was the average of these predictions after reversing the applied operations.

Unweighted Ensemble

We planned to use weighted ensembles (stacking) of different models but couldn't due to time constraints. In the end, we used an uniformly weighted ensemble with soft voting, i.e predictions on the average of the predicted probabilities of various models.

Morphological cleaning

Removing very small objects; We noticed at times our models predicted building, platform and aguada which were only a couple of pixels wide. Therefore, based upon the size distribution of the buildings, platforms and aguada in the training amks, we removed the predicted objects below a certain minimum size. We used a different threshold when the objects were partially or completely on the boundary of an image. However, the chosen thresholds were probably not ideal and could be improved with some more iterations.

Hole closing; The predicted masks at times had holes inside of predicted objects. In the training masks, we observed that the aguada and platform masks were usually convex without any gaps, and while buildings could take arbitrary shape they did not have holes either. Therefore, we decided to fill the holes in the predicted polygons during post-processing.

References:

1. Karras, Tero, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.
2. Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.
3. Liu, Ze, et al. "Swin transformer: Hierarchical vision transformer using shifted windows." *arXiv preprint arXiv:2103.14030* (2021).
4. Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)." *arXiv preprint arXiv:1511.07289* (2015).
5. He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
6. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *International Conference on Medical image computing and computer-assisted intervention*. Springer, Cham, 2015.
7. Tan, Mingxing, et al. "Mnasnet: Platform-aware neural architecture search for mobile." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.

8. Chen, Liang-Chieh, et al. "Rethinking atrous convolution for semantic image segmentation." *arXiv preprint arXiv:1706.05587* (2017).
9. Johnson, Justin, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution." *European conference on computer vision*. Springer, Cham, 2016.
10. Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).