III. Model documentation and write-up

Information included in this section may be shared publicly with challenge results. You can respond to these questions in an e-mail or as an attached file. Please number your responses.

1. Who are you (mini-bio) and what do you do professionally? If you are on a team, please complete this block for each member of the team.

   I am a senior data scientist at Nodalpoint Systems in Athens, Greece. I am a geologist and an oceanographer by education turned to data science through online courses and by taking part in numerous machine learning competitions.

2. What motivated you to compete in this challenge?

   I consider myself as a machine learning engineer who enjoys taking part in various machine learning competitions. Furthermore, using machine learning for Earth Observation is a combination of my educational background and my professional occupation.

3. High level summary of your approach: what did you do and why?

   Blending models with different encoders and in various image sizes. Final models use the SWIR, NIR and Green channels. Unet models with mit_b1, mit_b2, mit_b3 and mit_b4, encoders from segmentation models pytorch library as well as uppernet models with convnext_tiny and convenxt_base encoders from the transformers library were used. Image sizes of 512, 640, 768, 896 and 1024 were selected for the final modells. As a post processing step, a 0.43 threshold turned probabilities to masks and a land-sea mask from last image channel was applied to remove predicted masks on land.

4. Do you have any useful charts, graphs, or visualizations from the process?

   -

5. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

   Segmentation_models_pytorch library is widely used for segmentation tasks. Including a different model from a different library helped the final ensemble, while not expected to have been used from many other competitors.

   ```
   from transformers import UperNetForSemanticSegmentation
   model = UperNetForSemanticSegmentation.from_pretrained("openmmlab/upernet-convnext-tiny")
   model.decode_head.classifier = torch.nn.Conv2d(512, 1, kernel_size=(1, 1), stride=(1, 1))
   ```

Clipping image values and selecting 3 channels. The first 2 channels, SWIR and NIR were the most important, for 3rd channel the Green band was used as it gave marginally better results in the validation set. Null values was set to 0.

```
channels=[0,1,3]
CLIPMIN=6000
CLIPMAX=24000
image=np.clip(image, CLIPMIN, CLIPMAX)
image[original_image<=-32000]=0
image =image/ CLIPMAX
```

Post processing, turn probabilities to masks and remove pixels over land

```
THRESH=0.43
preds = np.array([(x>THRESH).astype('uint8') for x in preds])
#POST PROCESS - remove masks in altitude>0
preds= preds*(ALTIMETRY<0.5).astype('uint8')
```

6. Please provide the machine specs and time you used to run your model.
   - CPU (model): Intel(R) Xeon(R) CPU X5650 @ 2.67GHz
   - GPU (model or N/A): GeForce GTX 1080
   - Memory (GB): 70 GB
   - OS: Ubuntu 18.04
   - Train duration: About 4 days
   - Inference duration: less than 1 hour

   Except the above server, Google Colab pro was used for training 1 unet model with mit_b1 encoder at 1024 image size and 1 uppernet model with convnext_tiny encoder at 896 image size.

7. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

   -

8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

   -

9. How did you evaluate performance of the model other than the provided metric, if at all?

   -

10. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

    Many other encoders. Out of many tried, especially maxvit and hrnet encoders despite improving local validation score, didn't help the public LB score and as suspected neither helped the private LB score. Also, training with other than soft dice loss didn't seem to help in my experiments, as well as color, mixup, cutmix and a lot of other augmentation.

11. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

    As dataset labels comes from a platform users annotations, it contains many misslabelled images and this has passed to trained models. For a real use case a curated updated dataset followed by a model retraining and parameters retuning should be used. If I continued working on this problem for the next year I would prefer to work in a curated dataset for the models to be able to be used in a real project but If I had to work on the exact same dataset I would first reconsider my image and mask preprocessing, mask smoothing and boundary loss.

12. What simplifications could be made to run your solution faster without sacrificing significant accuracy?

    Removing large image size (>=896) models. Also, depending on the hardware I would consider to increase the batch size.