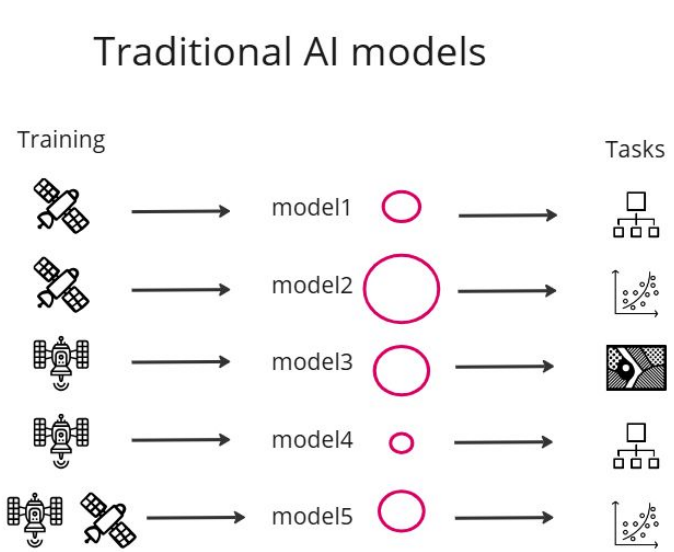
EUMETSAT

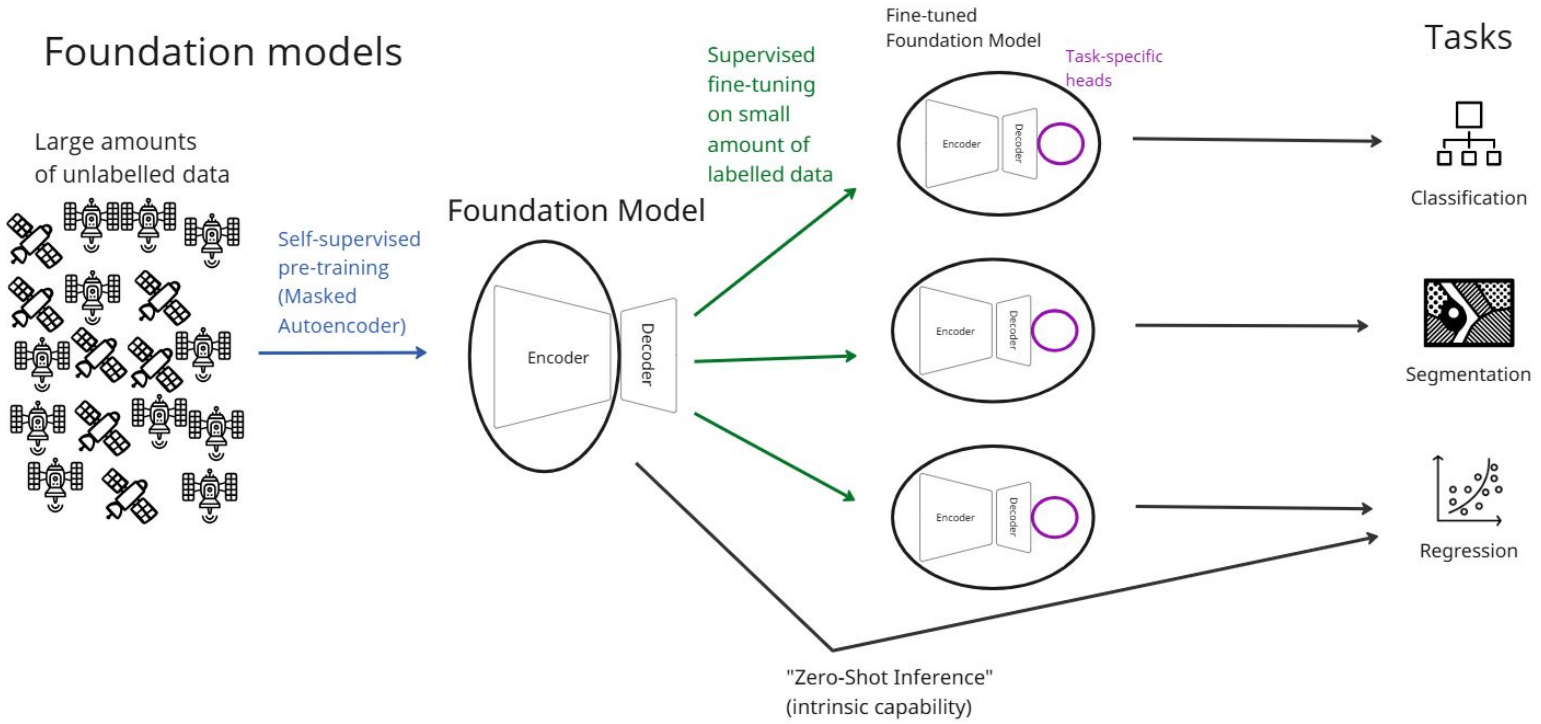
Cloud Masking Sen-3 with Prithvi-EO Foundation Model

Boran Frank
Intern in Copernicus Programme Office

23.04.2025

What are Foundation Models in EO?: Conceptual Definition





Advantages of using FMs:

- Harness large amounts of unlabelled data
- Fine-tuning requires only a small amount of labelled data
- Adaptation to different tasks more efficient than building new standalone models from scratch
    - Supported by Python frameworks like **TerraTorch**

What FMs exist in Geosciences?

Extended Data Table 1: State-of-the-art EO FMs. The slash sign / indicates separate models for each; the range sign - indicates a rough range as the exact numbers are not available; the comma sign , indicates a combined coverage.

Model	Pre-training Dataset	Data Size (# pixels)	Modality	Spatial Resolution	Spatial Coverage (km^2)	Model Size (# params)	Temporal Coverage	Sequence Length
SeCo [33]	SeCo	70B	MS	10m	Global (1.4M)	23M	N/A	1
SatMAE [46]	fMoW	50B	MS / RGB	0.5m / 10m	Global (1.7M)	305M	5 years	3
RVSA [122]	MillionAID	4B	RGB	0.5m–150m	Global (18K)	89M	N/A	1
Scale-MAE [45]	fMoW-RGB	80B	RGB	0.3–10m	Global (1.7M)	305M	N/A	1
CROMA [49]	SSL4EO-S12	140B	MS, SAR	10m	Global (1.8M)	305M	N/A	1
DeCUR [50]	SSL4EO-S12 / GeoNRW	140B	MS, SAR / RGB,DEM	10m / 1m	Global (1.8M)	23M	N/A	1
Presto [47]	Presto	4B	MS, SAR, ERA5, DEM, etc.	10m	Global (2K)	400K	2 years	24
Prithvi [48]	HLS	158B	MS	10m, 30m	US	100M	1+ years	3
SpectralGPT [123]	fMoW+BigEarthNet	12B	MS	10m	Global (1.7M)	86M	N/A	1
SkySense [41]	SkySense	97T	MS, RGB, SAR	0.3m, 10m	Global (8.8M)	2B	?	20
OFA-Net [51]	Satlas.+5B pix. +HyspecNet.	13B	MS, RGB, SAR, HS	1m, 4m, 10m, 30m	Global (260K)	86M	N/A	1
DOFA [52]	Satlas.+5B pix. +HyspecNet.	105B	MS, RGB, SAR, HS	1m, 4m, 10m, 30m	Global (2.6M)	305M	N/A	1
SSL4EO-S12 [34]	SSL4EO-S12	140B	MS, SAR	10m	Global (1.8M)	23M	N/A	1
SSL4EO-L [35]	SSL4EO-L	348B	MS	30m	Global (15.6M)	23M	N/A	1
SatlasPretrain [36]	SatlasPretrain	17T	MS, RGB, SAR	0.5-2m, 10m	Global (21M)	88M	11 years	10
SkyScriptCLIP [54]	SkyScript	130B	MS, RGB, text	0.5m, 1–2m, 10-30m	Global (5M)	305M	N/A	1

Extended Data Table 2: State-of-the-art weather and climate FMs.

Model	Affiliation	Model Architecture	Temporal Resolution	Time Span	Spatial Resolution	Spatial Coverage	Model Size (# params)	Cluster Size	Training Time
KeislerNet [124]	Descartes	GNN	6 hrs	6 days	1°	Global	6.7M	1 A00	5.5 days
FourCastNet [125]	NVIDIA	Transformer/FNO	6 hrs	200 hrs	0.25°	Global	433M–2.19B	64 A100	16 hrs
Pangu-Weather [67]	Huawei	Transformer	1 hr	7 days	0.25°	Global	256M	192 V100	15 days
GraphCast [68]	Google	GNN	6 hrs	10 days	0.25°	Global	36.7M	32 TPU v4	4 wks
ClimaX [63]	UCLA	Transformer	6 hrs	1 month	0.25°	Global	108–111M	80 V100	3 days
FengWu [126]	Shanghai	Transformer	6 hrs	14 days	0.25°	Global	427M	32 A100	17 days
W-MAE [127]	Chengdu	Transformer	6 hrs	9 days	0.25°	Global	96.5M	8 A800	223.4 hrs
FuXi [91]	Fudan	Transformer	6 hrs	15 days	0.25°	Global	1.56B	8 A100	30 hrs
SEEDS [128]	Google	Diffusion	N/A	16 days	2°	Global	114M	16 TPU v4	18 hrs
NeuralGCM [69]	Google	Differential GCM	3.75–12 mins	15 days	0.7–2.8°	Global	115–311M	128 TPU v5e	10 days
Stormer [129]	UCLA	Transformer	6 hrs	14 days	1.4°	Global	100–600M	128 A100	24 hrs
GenCast [92]	Google	Diffusion	12 hrs	15 days	1°	Global	60M	32 TPU v4	5 days
FengWu-GHR [59]	Shanghai	Transformer	6 hrs	10 days	0.09°	Global	4B	16 A100	3 days

Zhu et al. 2024

Earth Observation Foundation Models

- trained on high-resolution data (e.g. Sentinel-2)
- primarily land applications

Weather and Climate Foundation Models

- trained on low-resolution reanalysis data

What about medium-/low-resolution Earth Observation (e.g. Sen-3, Meteosat …)?

Motivation Research question

Research Questions:

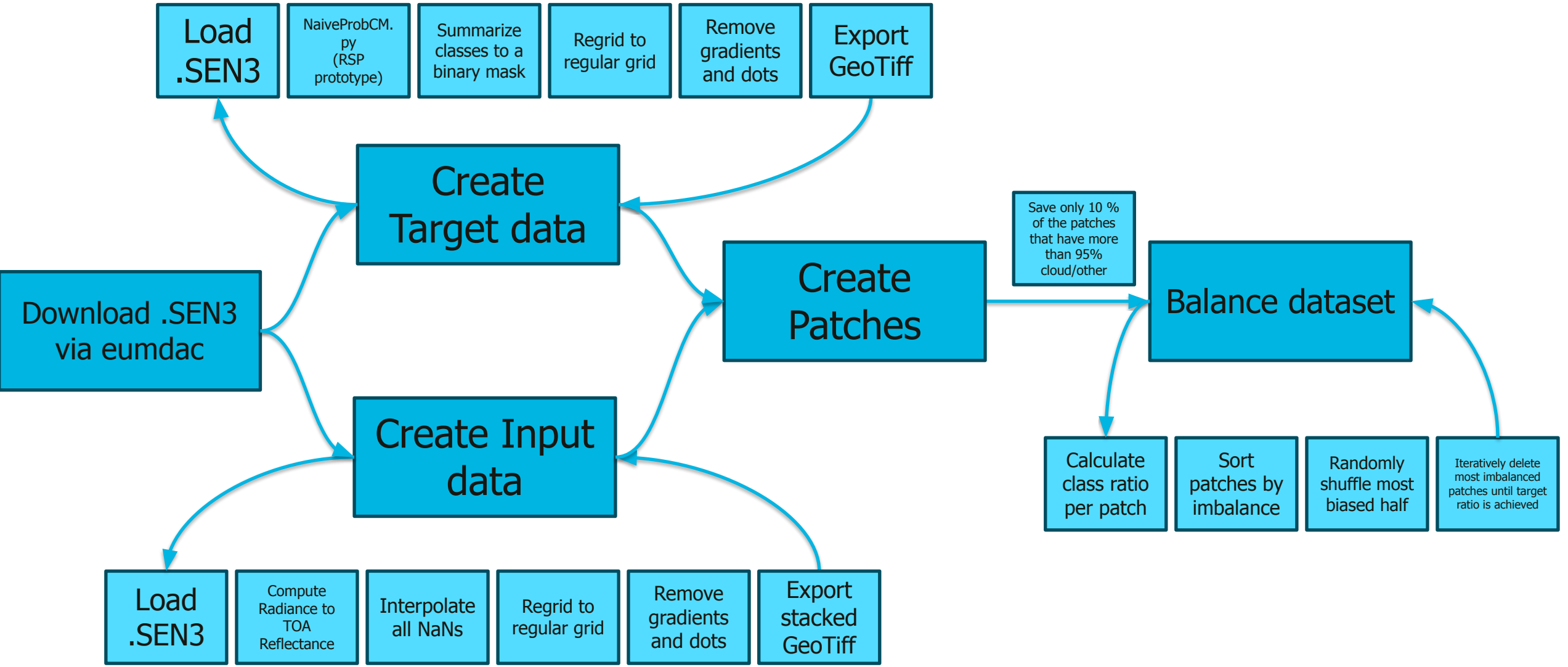
- How does a foundation model perform that was pre-trained with a higher resolution?

- How does a foundation model perform that was pre-trained on other spectral bands?

- How easy is it to apply foundation models? What computational and intellectual resources are required?

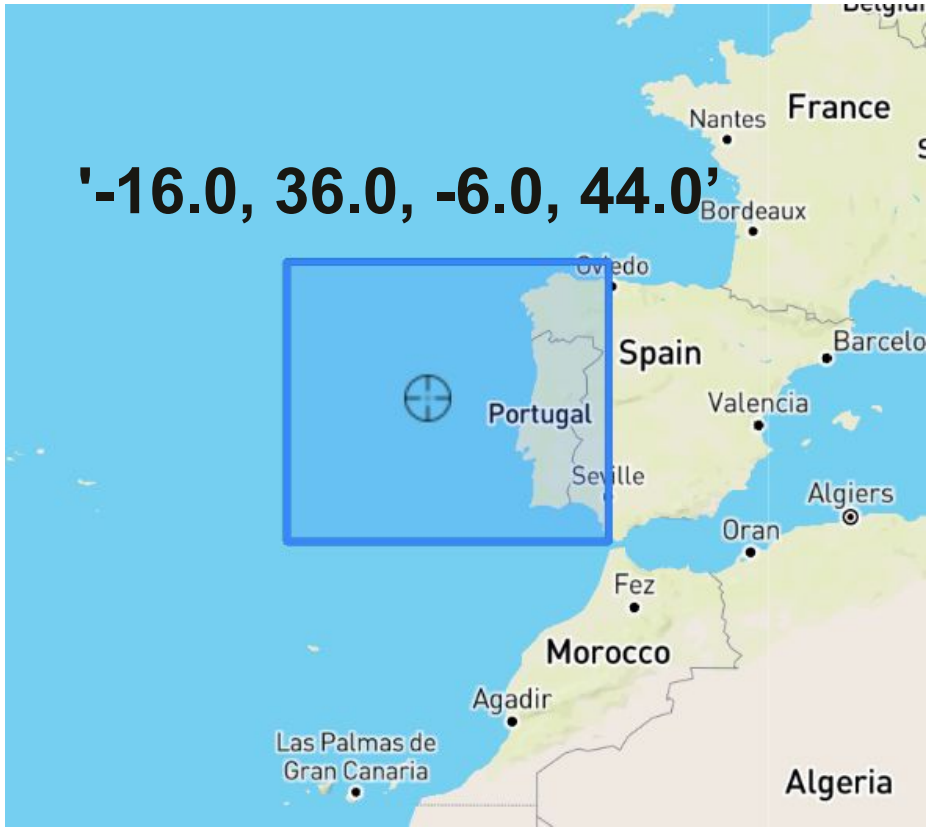
 Do this with the Prithvi-EO 2.0 foundation model

Methodology: Pre-processing script (~12 mins per tile)



Methodology: Download .SEN3 via eumdac

Randomly downloading tiles from summer 2024, 2023 that overlap this bounding box



Methodology: Choose bands

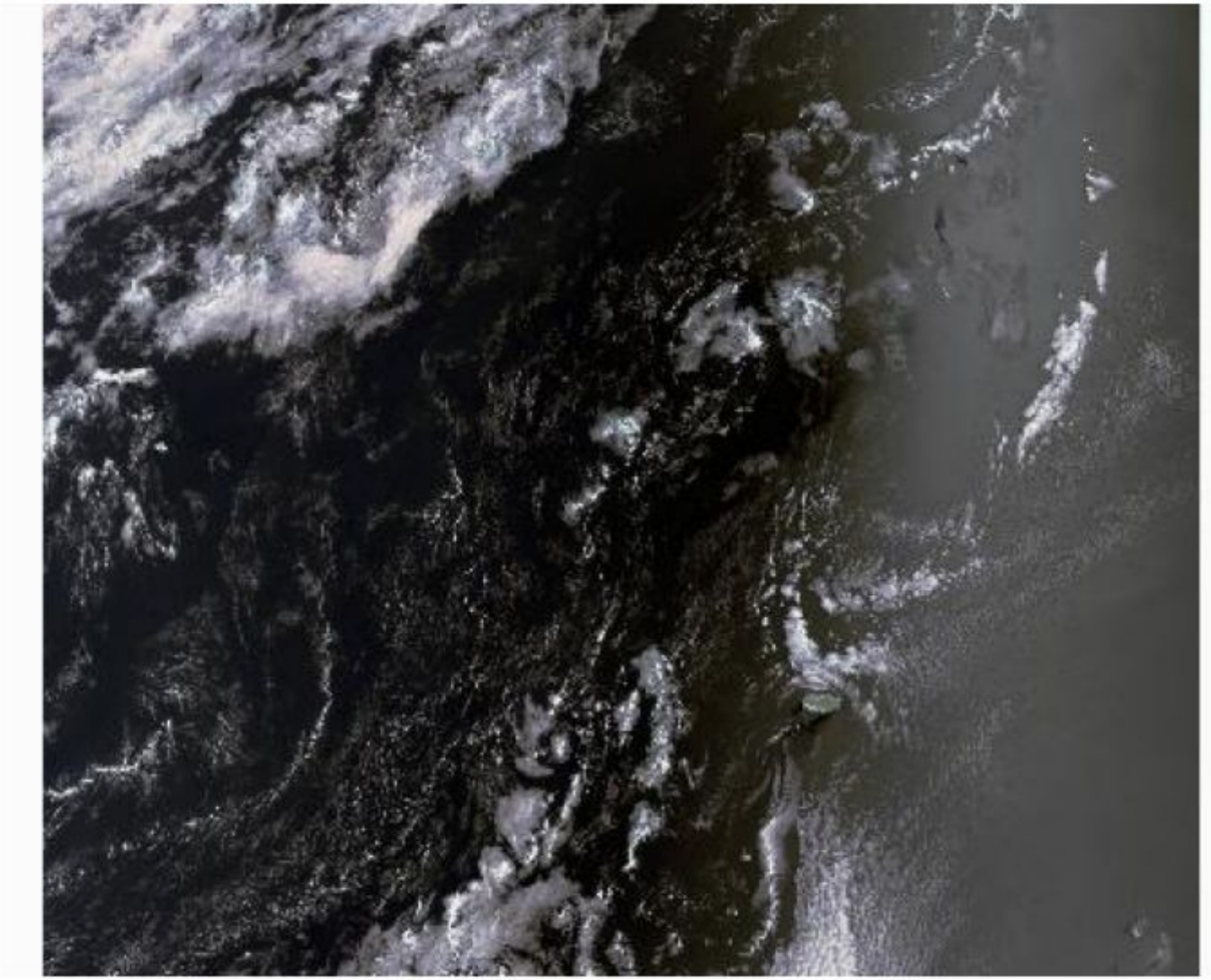
S2 band number used in Prithvi-EO	S2 corresponding wavelength	S3 SLSTR band number	S3 SLSTR corresponding wavelength	Used in Prithvi_CM
2	0.49 (Blue)	None, but OLCI 4	None, but OLCI4: 0.49	2 (0.66)
3	0.56 (Green)	1	0.55	2 (0.66)
4	0.665 (Red)	2	0.66	3 (0.87)
8A	0.865 (NIR)	3	0.87	5 (1.61)
11	1.610 (SWIR1)	5	1.61	5 (1.61)
12	2.190 (SWIR2)	6	2.25	6 (2.25)

BIG OOPS!

Extended research question:
How does the FM perform with suboptimally chosen spectral bands?

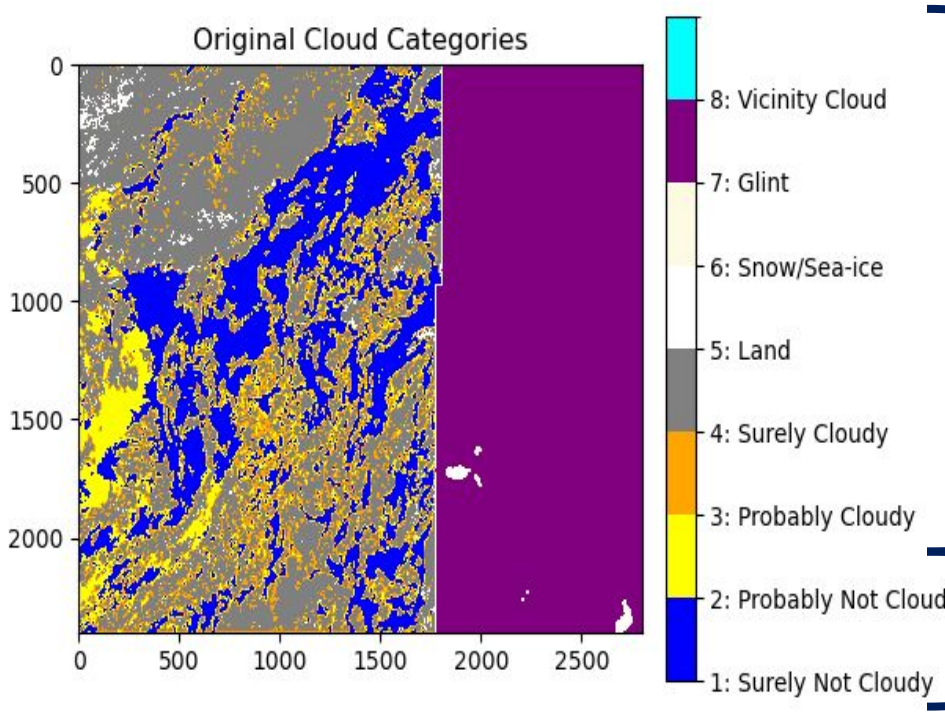
Methodology: Example data: 20230721T104815

Browse Image



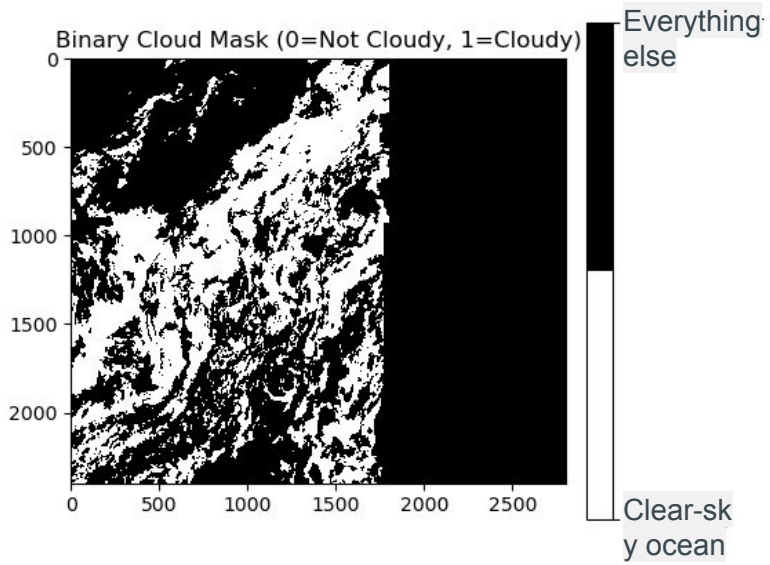
Nadir view!

Methodology: Example data: 20230721T104815

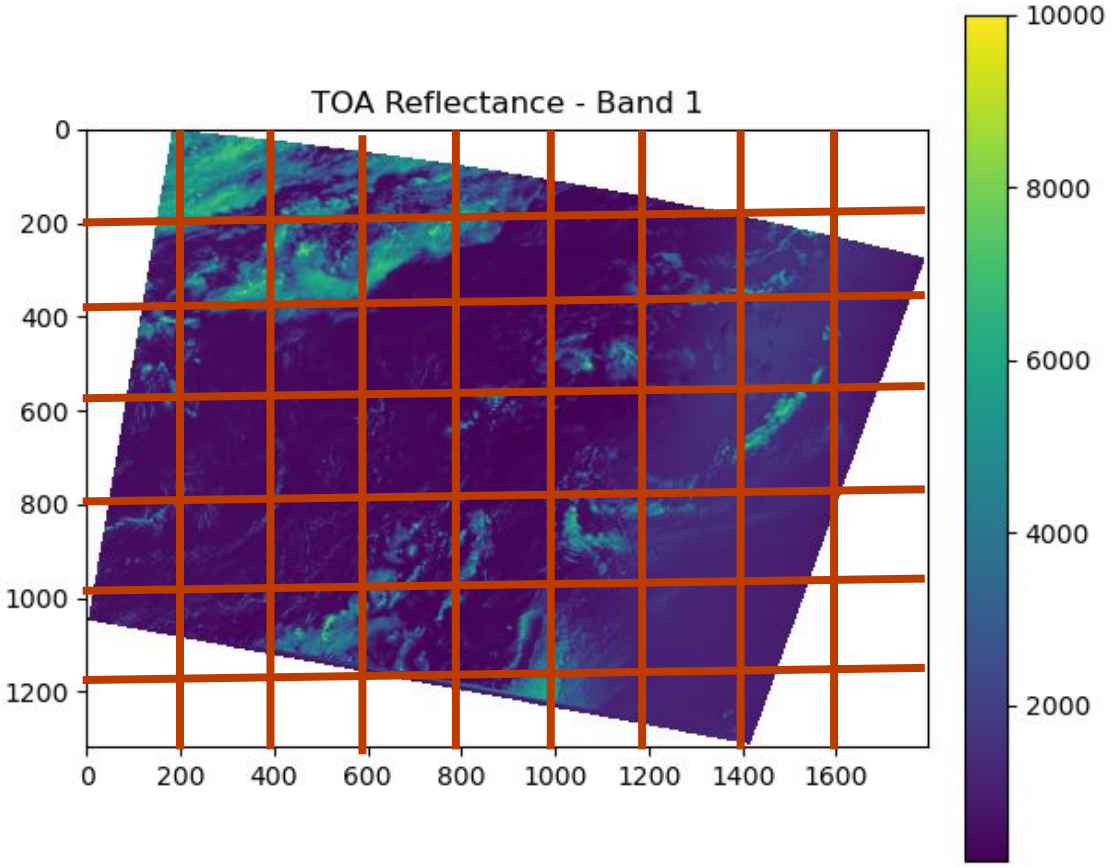


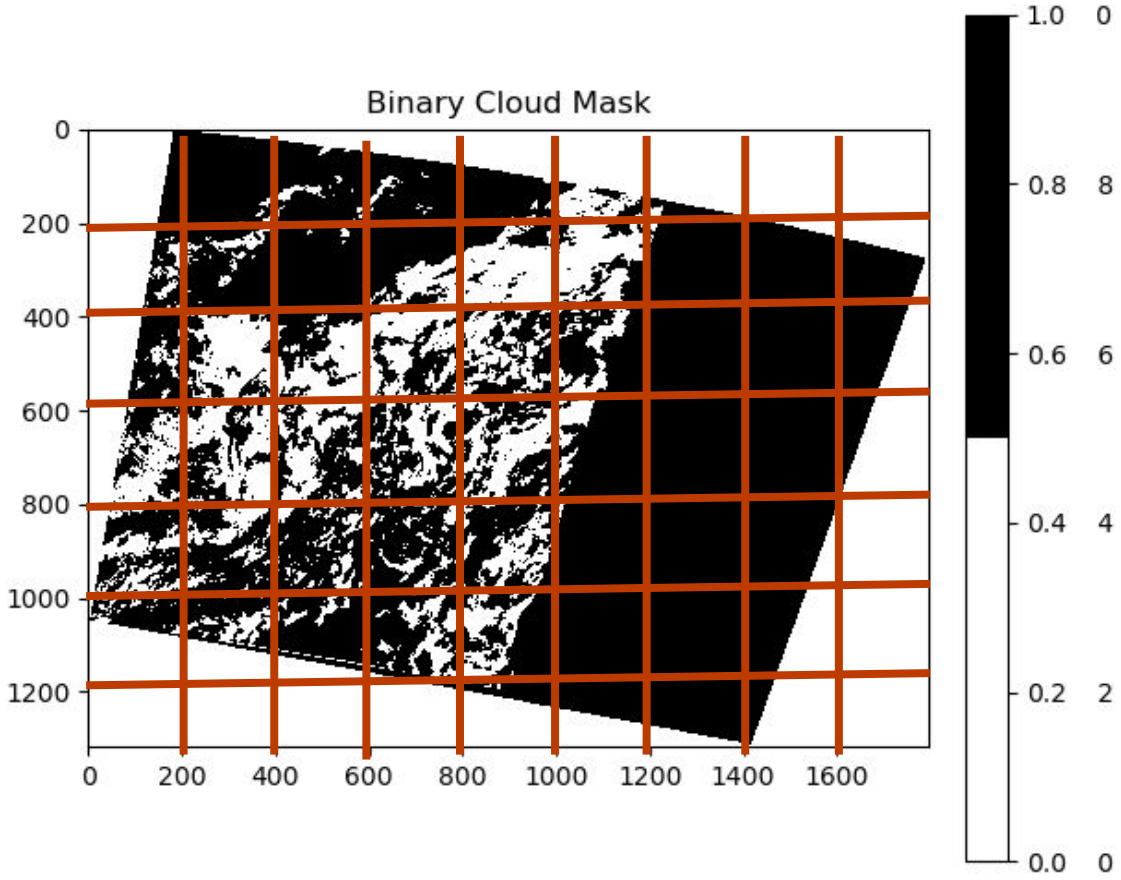
“Else”

“Clear-sky Ocean”



Methodology: Example data: 20230721T104815



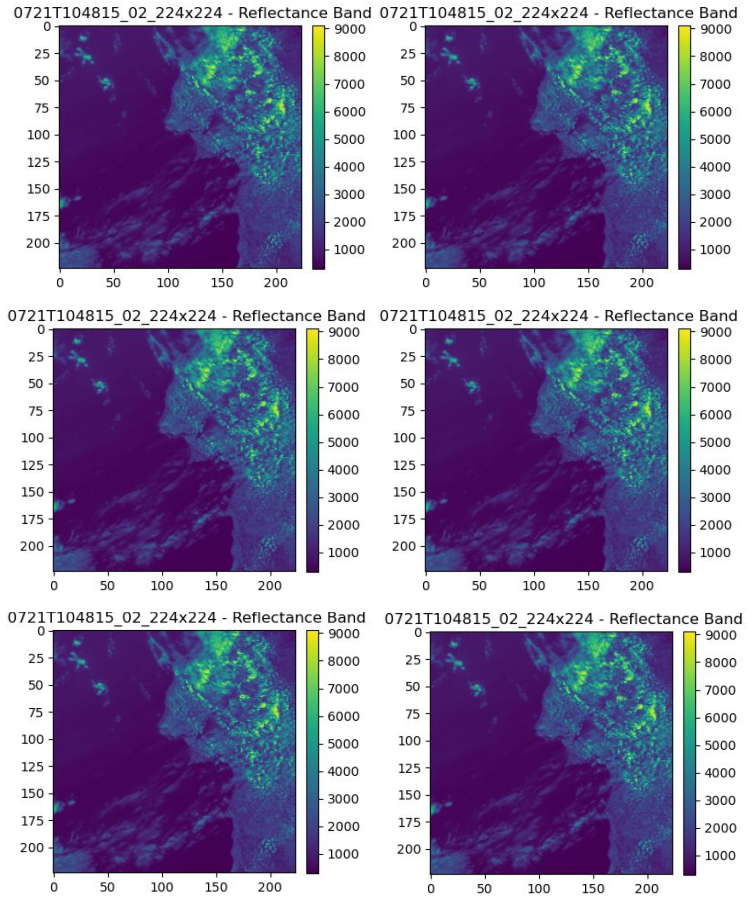


224 x 224 patches are cut out

re-gridding introduces NAN borders only patches are used that do not include any NAN border not optimal

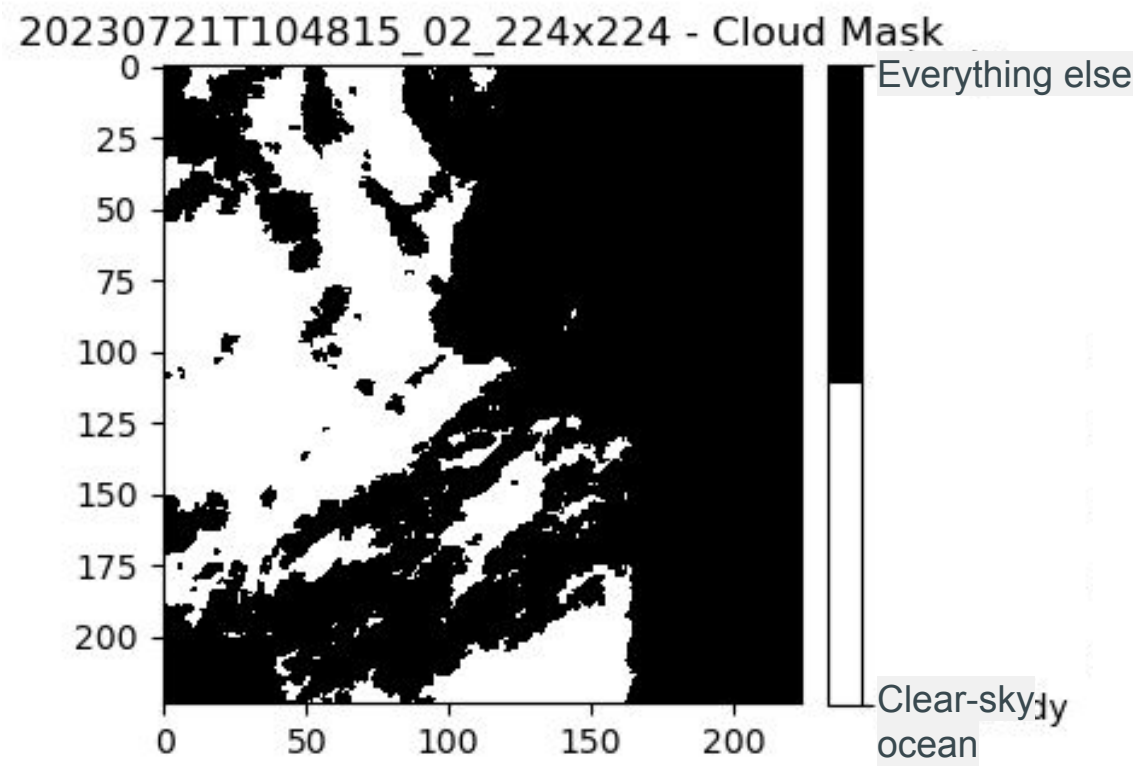
Methodology: Example data: 20230721T104815

INPUT: SLSTR patches from bands: 2,2,3,5,5,6

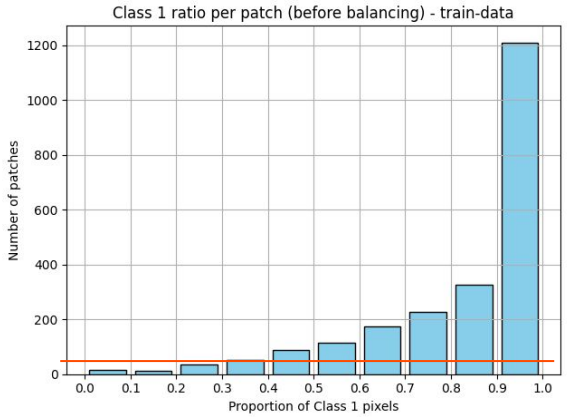


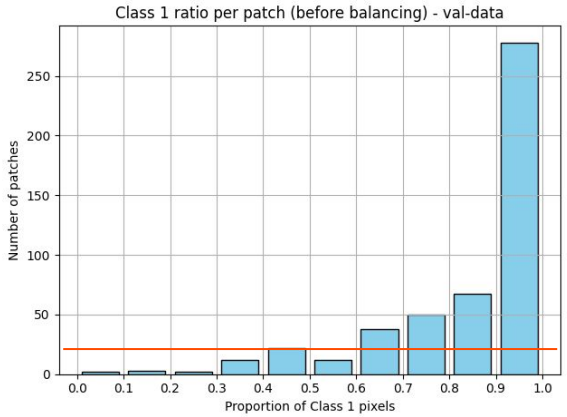
! Here only band 1 is displayed, should be 2,2,3,5,5,6

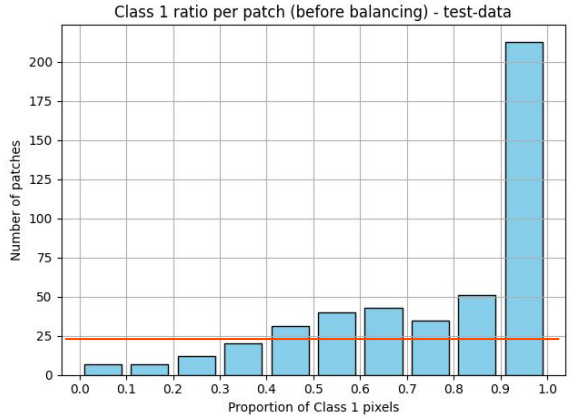
TARGET: Binary Cloud Mask from NaiveProbCM.py

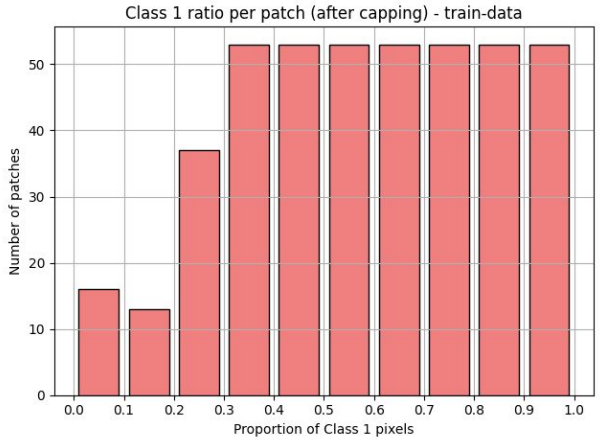


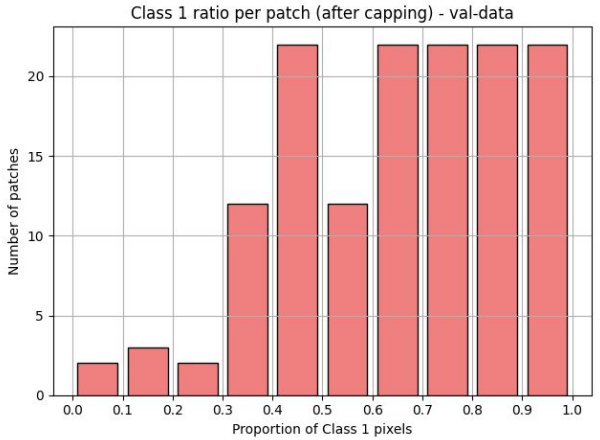
Methodology: Balance Dataset

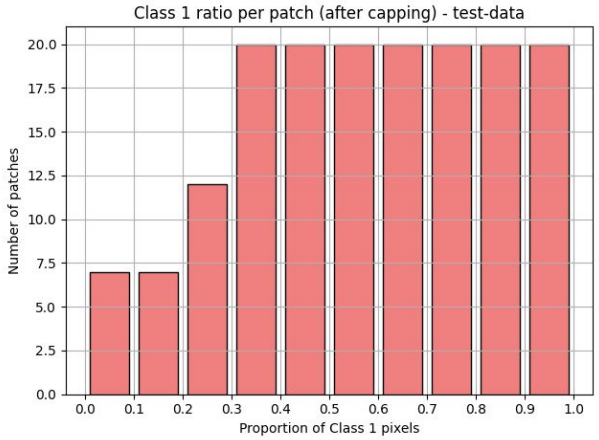












Final dataset

Train: 437
Val: 141
Test: 166

📊 Class distribution in 'train':
Class 0: 9096526 pixels (41.49%)
Class 1: 12830386 pixels (58.51%)

📊 Class distribution in 'val':
Class 0: 2384375 pixels (33.70%)
Class 1: 4690441 pixels (66.30%)

📊 Class distribution in 'test':
Class 0: 3489298 pixels (41.89%)
Class 1: 4839918 pixels (58.11%)

Methodology: Prithvi-EO 2.0 and TerraTorch

Prithvi-EO:

- Developed by IBM x NASA
- trained on 4.2m 256 x 256 samples of HLS (Harmonized Landsat Sentinel-2) with **30m** resolution
- Vision Transformer with 300M or 600M parameters
  - Trained with Masked Autoencoder strategy
- trained for ~ 58k GPU hours (NVIDIA A100)
- **[2412.02732] Prithvi-EO-2.0: A Versatile Multi-Temporal Foundation Model for Earth Observation Applications**

TerraTorch:

- flexible fine-tuning framework for Geospatial Foundation Models based on **TorchGeo** (torchgeo) and **PyTorch Lightning** (PyTorch Lightning)
- open-source code and weights
- easily access pre-trained and fine-tuned models, datasets, configurations
- **IBM/terratorch: A Python toolkit for fine-tuning Geospatial Foundation Models (GFMs).**

Zhu et al. 2024

Methodology: Workflow with TerraTorch

Define datamodule

```
datamodule = terratorch.datamodules.GenericNonGeoSegmentationDataModule(
    batch_size=8,
    num_workers=2,
    num_classes=2,

    # Define dataset paths
    train_data_root=dataset_path / 'data/',
    train_label_data_root=dataset_path / 'data/',
    val_data_root=dataset_path / 'data/',
    val_label_data_root=dataset_path / 'data/',
    test_data_root=dataset_path / 'data/',
    test_label_data_root=dataset_path / 'data/',

    # Define splits
    train_split=dataset_path / 'splits/train.txt',
    val_split=dataset_path / 'splits/val.txt',
    test_split=dataset_path / 'splits/test.txt',

    img_grep='*_reflectance.tif',
    label_grep='*_binary.tif',

    train_transform=[
        albumentations.D4(), # Random flips and rotation
        albumentations.pytorch.transforms.ToTensorV2(),
    ],
    val_transform=None,  # Using ToTensor() by default
    test_transform=None,

    # Define standardization values
    means=[
      stats[0][0],
      stats[1][0],
      stats[2][0],
      stats[3][0],
      stats[4][0],
      stats[5][0],
    ],
    stds=[
      stats[0][1],
      stats[1][1],
      stats[2][1],
      stats[3][1],
      stats[4][1],
      stats[5][1],
    ],
    no_data_replace = 0,
    no_label_replace = -1,
    # We use all six bands of the data, so we don't need to define dataset_bands
)
```

Define model

```
# Model
model = terratorch.tasks.SemanticSegmentationTask(
    model_factory="EncoderDecoderFactory",
    model_args={
        # Backbone
        "backbone": "prithvi_eo_v2_300_tl", # Model can be either prithvi_eo_v1_100, prithvi_eo_v2_300
        "backbone_pretrained": True,
        "backbone_num_frames": 1, # 1 is the default value
        "backbone_bands": ["BLUE", "GREEN", "RED", "NIR_NARROW", "SWIR_1", "SWIR_2"],
        "backbone_coords_encoding": ['time', 'location'], # use ["time", "location"] for time and loca

        # Necks
        "necks": [
            {
                "name": "SelectIndices",
                # "indices": [2, 5, 8, 11] # indices for prithvi_eo_v1_100
                "indices": [5, 11, 17, 23] # indices for prithvi_eo_v2_300
                # "indices": [7, 15, 23, 31] # indices for prithvi_eo_v2_600
            },
            {"name": "ReshapeTokensToImage",},
            {"name": "LearnedInterpolateToPyramidal"}
        ],

        # Decoder
        "decoder": "UNetDecoder",
        "decoder_channels": [512, 256, 128, 64],

        # Head
        "head_dropout": 0.1,
        "num_classes": 2,
    },

    loss="dice",
    optimizer="AdamW",
    lr=1e-4,
    ignore_index=-1,
    freeze_backbone=True, # Only to speed up fine-tuning
    freeze_decoder=False,
    plot_on_val=True,
    class_names=['no cloud', 'cloud o.o.']  # optionally define class names
)
```

Define trainer

```
# Lightning Trainer
trainer = pl.Trainer(
    accelerator="auto",
    strategy="auto",
    devices=1, # Deactivate multi-gpu because it often fails in notebooks
    precision='16-mixed',  # Speed up training
    num_nodes=1,
    logger=True,  # Uses TensorBoard by default
    max_epochs=15,
    log_every_n_steps=1,
    enable_checkpointing=True,
    callbacks=[checkpoint_callback, pl.callbacks.RichProgressBar()],
    default_root_dir="output/sen3_cm",
)
```

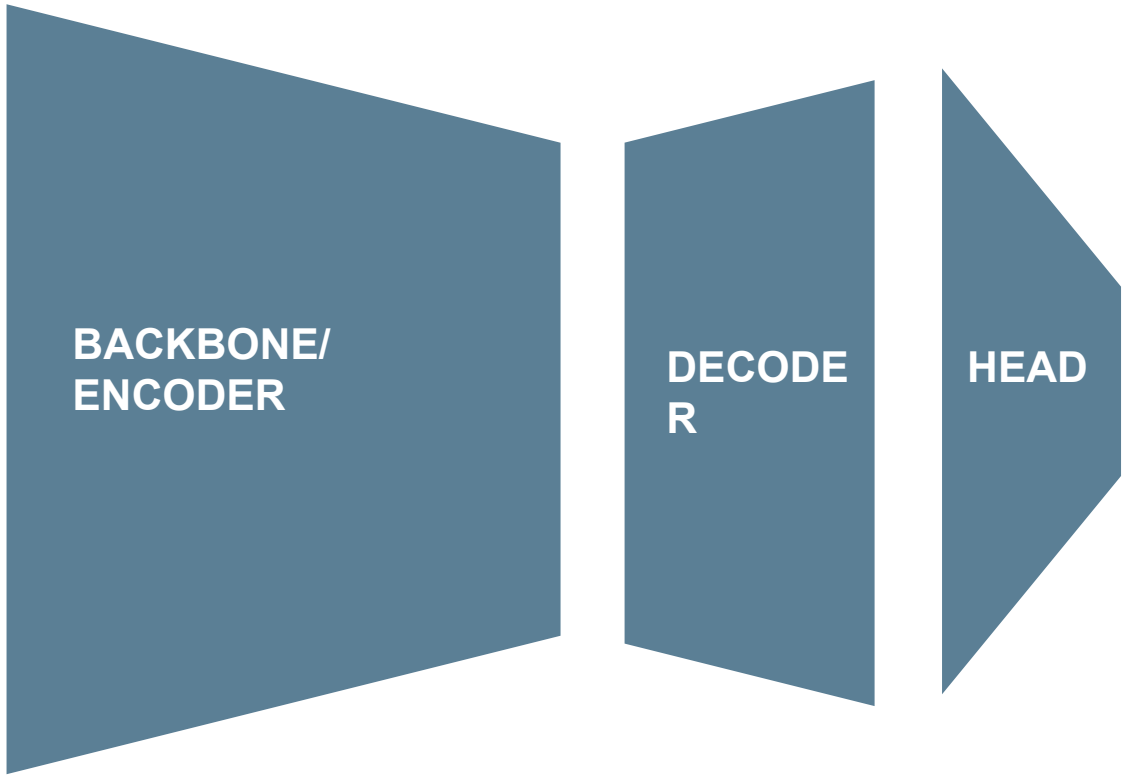
Methodology: Model Architecture and Training strategy

- Prithvi backbone (Vision Transformer): 303M or 631M parameters
- U-Net decoder: 20.3M or 25.7M
- Segmentation head (ConvNN): 4.8M

- Total size: 1.3 GB or 2.6 GB

-Epochs = 20
-Batch size = 8
-Lr = 1e-4
-AdamW
-Loss = dice
-vRAM needed: 8025MB for Prithvi_300 when all parameters are trained
-Time per epoch: ~30s

Methodology: Backbone – Decoder - Head



Pretrained_encoder = True Foundation model/ pre-trained weights are used
Pretrained_encoder = False Foundation model/ pre-trained weights are NOT used
 randomly initialized

Frozen_encoder = True Encoder weights are NOT changed during fine-tuning
Frozen_encoder = False Encoder weights are changed during fine-tuning

Frozen_decoder = True Decoder weights are NOT changed during fine-tuning
Frozen_decoder = False Decoder weights are changed during fine-tuning

! Decoder and Head always start with random weights

Methodology: Segmentation Task Metrics

Dice:

- 0 = no overlap
- 1 = perfect overlap
- Dice loss = 1 / Dice Coefficient

$$\text{Dice Coefficient} = \frac{2 \times \text{Intersection}}{\text{Size of Predicted} + \text{Size of Actual}}$$

Multiclass Jaccard Index:

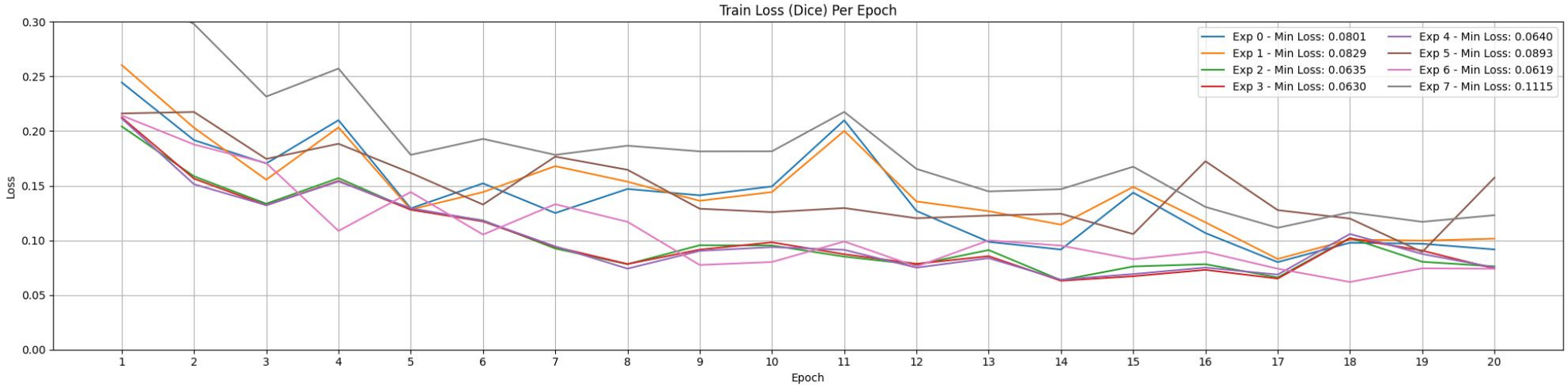
- 0 = no overlap
- 1 = perfect overlap

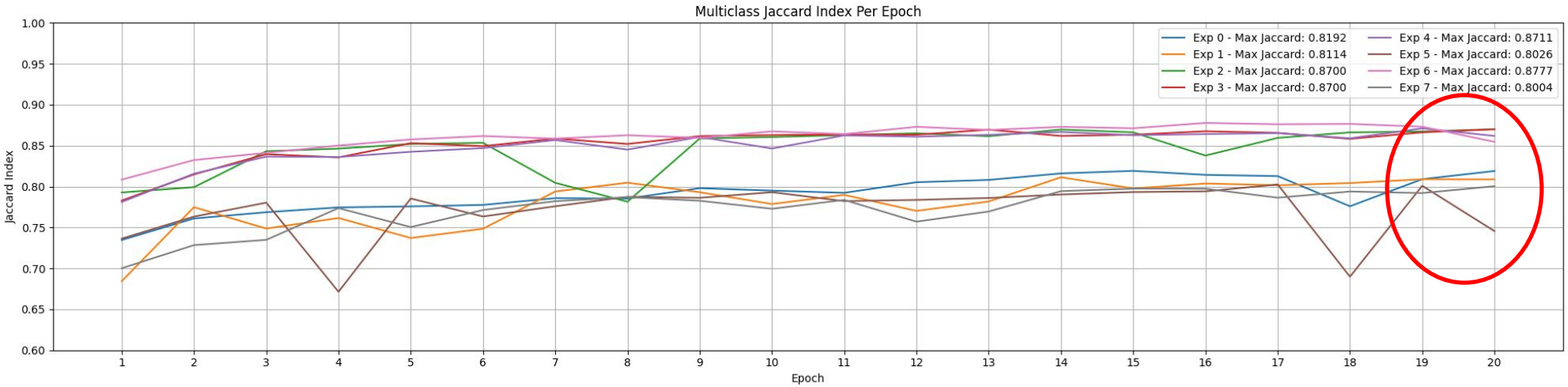
$$\text{Jaccard Index} = \frac{\text{Intersection of predicted and ground truth}}{\text{Union of predicted and ground truth}}$$

Results: Ablation studies

experiment	backbone	backbone_pretrained	backbone_encoding	freeze_backbone	freeze_decoder	Test Loss (ranking)	Test Multiclass Jaccard Index (ranking)
exp_0	v2_300	FALSE	[]	FALSE	FALSE	0.1088 (6.)	0.8226 (6.)
exp_1	v2_300	FALSE	[]	TRUE	FALSE	0.1191 (7.)	0.8103 (7.)
exp_2	v2_300	TRUE	[]	FALSE	FALSE	0.0774 (3.)	0.8697 (3.)
exp_3	v2_300	TRUE	[]	TRUE	FALSE	0.0901 (4.)	0.8576 (4.)
exp_4	v2_300_tl	TRUE	['time', 'location']	TRUE	FALSE	0.0725 (2.)	0.8798 (2.)
exp_5	v2_600	FALSE	[]	FALSE	FALSE	0.1001 (5.)	0.8313 (5.)
exp_6	v2_600	TRUE	[]	TRUE	FALSE	**0. 0711 (1.)**	**0.8836 (1.)**
exp_7	v2_300	FALSE	[]	TRUE	TRUE	0.1386 (8.)	0.8041 (8.)

Results: Loss





Two groups:
0,1,5,7, - worse without pre-trained backbone
2,3,4,6 – better with pre-trained backbone

Main Results: Pre-trained backbone

These comparisons are the key results, as they evaluate whether using pre-trained weights—essentially the foundation model—adds value.

experiment	backbone	backbone_pretrained	backbone_encoding	freeze_backbone	freeze_decoder	Test Loss (ranking)	Test Multiclass Jaccard Index (ranking)
exp_0	v2_300	FALSE	[]	FALSE	FALSE	0.1088 (6.)	0.8226 (6.)
exp_2	v2_300	TRUE	[]	FALSE	FALSE	0.0774 (3.)	0.8697 (3.)
exp_1	v2_300	FALSE	[]	TRUE	FALSE	0.1191 (7.)	0.8103 (7.)
exp_3	v2_300	TRUE	[]	TRUE	FALSE	0.0901 (4.)	0.8576 (4.)

5.7 % improvement

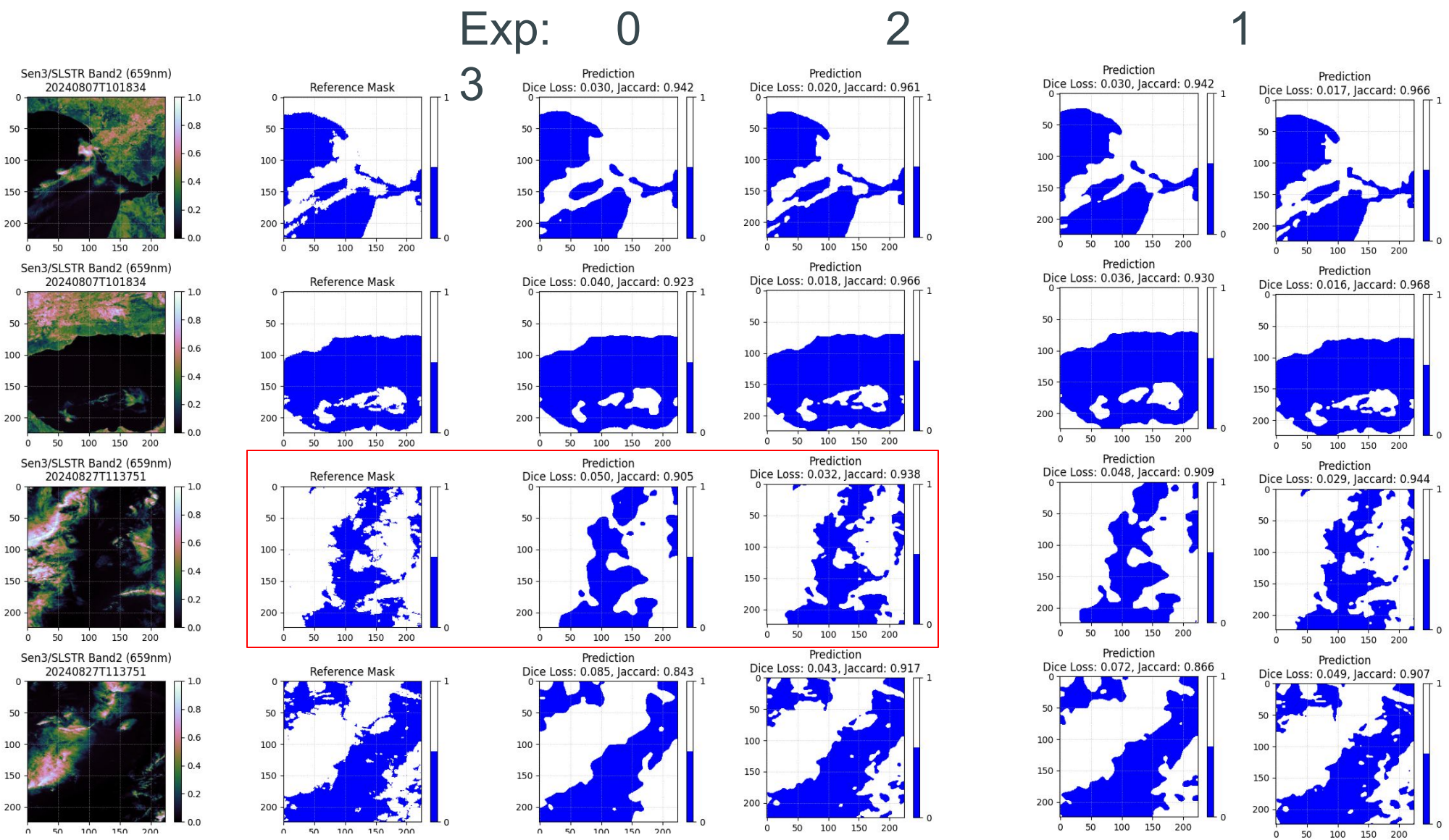
5.8 % improvement

Main Results: Pre-trained backbone

0,1 : without pre-trained backbone

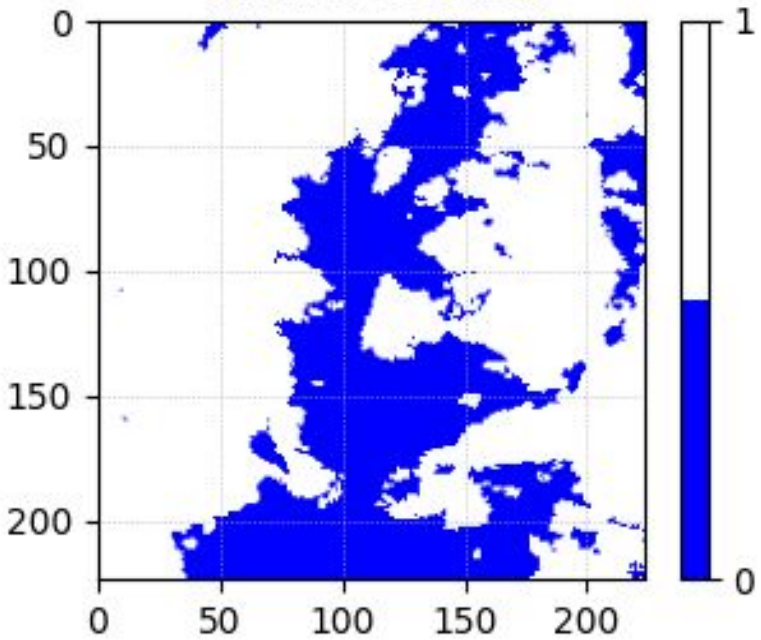
2,3: with pre-trained backbone

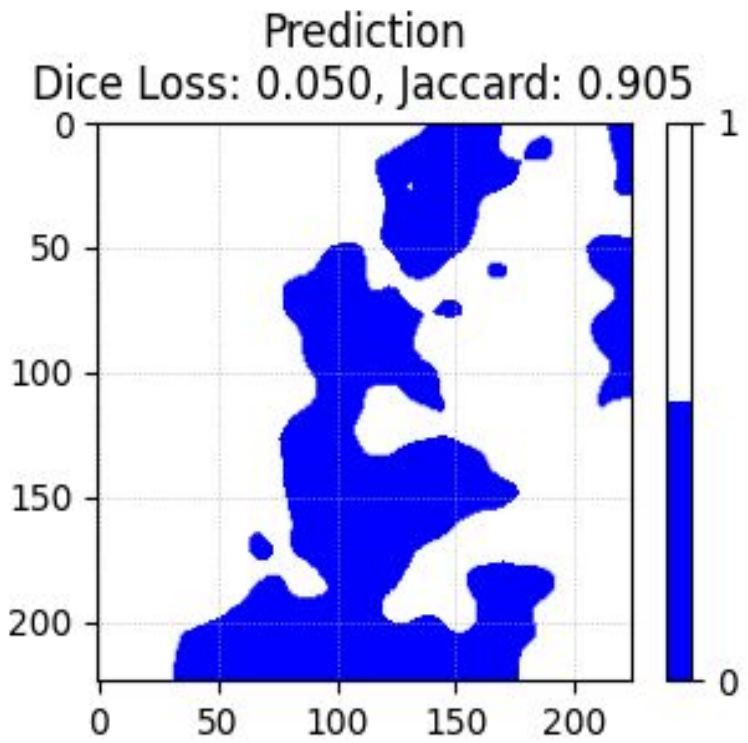
- 2,3 more detailed and recognizes small structures better



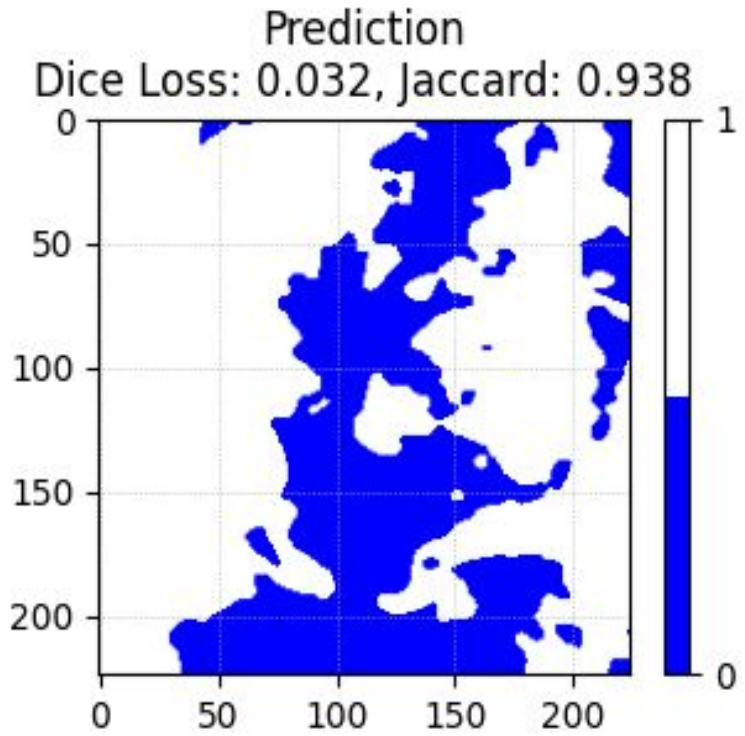
Main Results: Pre-trained backbone

without pre-trained backbone





with pre-trained backbone



Results: Trainable backbone

experiment	backbone	backbone_pretrained	backbone_encoding	freeze_backbone	freeze_decoder	Test Loss (ranking)	Test Multiclass Jaccard Index (ranking)
exp_0	v2_300	FALSE	[]	FALSE	FALSE	0.1088 (6.)	0.8226 (6.)
exp_1	v2_300	FALSE	[]	TRUE	FALSE	0.1191 (7.)	0.8103 (7.)

- Non-frozen backbone without pre-trained weights improves MJI by 1,5 %

Results: Trainable backbone

experiment	backbone	backbone_pretrained	backbone_encoding	freeze_backbone	freeze_decoder	Test Loss (ranking)	Test Multiclass Jaccard Index (ranking)
exp_2	v2_300	TRUE	[]	FALSE	FALSE	0.0774 (3.)	0.8697 (3.)
exp_3	v2_300	TRUE	[]	TRUE	FALSE	0.0901 (4.)	0.8576 (4.)

- Non-frozen backbone with pre-trained weights improves MJI by 1.4 %

Results: Using time and location embeddings

experiment	backbone	backbone_pretrained	backbone_encoding	freeze_backbone	freeze_decoder	Test Loss (ranking)	Test Multiclass Jaccard Index (ranking)
exp_3	v2_300	TRUE	[]	TRUE	FALSE	0.0901 (4.)	0.8576 (4.)
exp_4	v2_300_tl	TRUE	['time', 'location']	TRUE	FALSE	0.0725 (2.)	0.8798 (2.)

- Using time and location embedding improves MJI by 2,5 %

Results: More parameters

experiment	backbone	backbone_pretrained	backbone_encoding	freeze_backbone	freeze_decoder	Test Loss (ranking)	Test Multiclass Jaccard Index (ranking)
exp_0	v2_300	FALSE	[]	FALSE	FALSE	0.1088 (6.)	0.8226 (6.)
exp_5	v2_600	FALSE	[]	FALSE	FALSE	0.1001 (5.)	0.8313 (5.)

- Using 600M parameter version without pre-trained weights improves MJI by 1.0 %

Results: More parameters

experiment	backbone	backbone_pretrained	backbone_encoding	freeze_backbone	freeze_decoder	Test Loss (ranking)	Test Multiclass Jaccard Index (ranking)
exp_3	v2_300	TRUE	[]	TRUE	FALSE	0.0901 (4.)	0.8576 (4.)
exp_6	v2_600	TRUE	[]	TRUE	FALSE	**0. 0711 (1.)**	**0.8836 (1.)**

- Using 600M parameter version with pre-trained weights improves MJI by 3.0 %

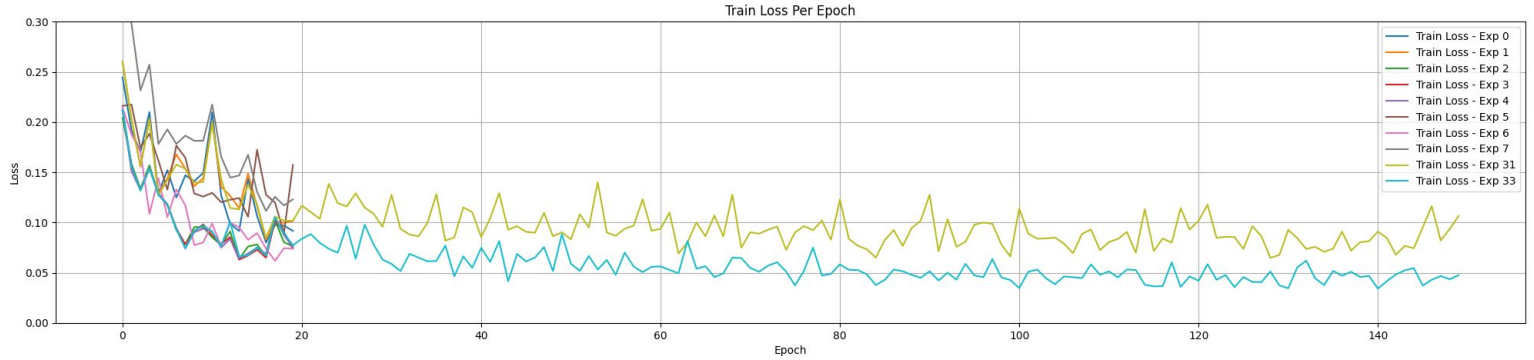
Results: Train only the head

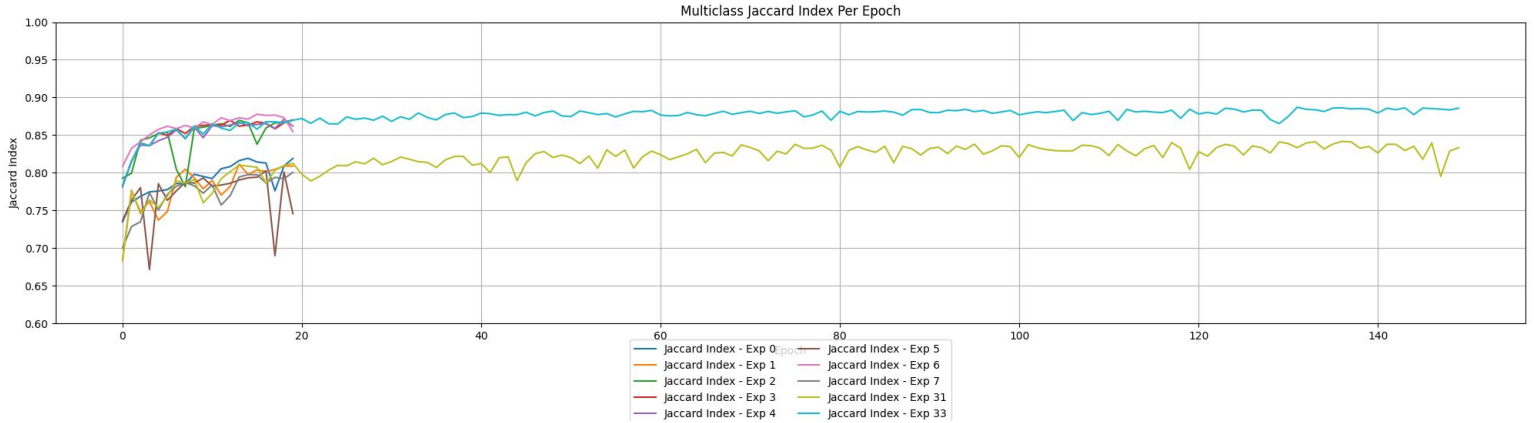
experiment	backbone	backbone_pretrained	backbone_encoding	freeze_backbone	freeze_decoder	Test Loss (ranking)	Test Multiclass Jaccard Index (ranking)
exp_1	v2_300	FALSE	[]	TRUE	FALSE	0.1191 (7.)	0.8103 (7.)
exp_7	v2_300	FALSE	[]	TRUE	TRUE	0.1386 (8.)	0.8041 (8.)

- Training ONLY the head decreases MJI by 0.7 %

Results: More epochs

experiment	epochs	backbone	backbone_pretrained	backbone_encoding	freeze_backbone	freeze_decoder	Test Loss	Test Multiclass Jaccard Index
exp_1	25	v2_300	FALSE	[]	TRUE	FALSE	0.1191	0.8103
exp_31	150	v2_300	FALSE	[]	TRUE	FALSE	0.0899	0.8450
exp_3	25	v2_300	TRUE	[]	TRUE	FALSE	0.0901	0.8576
exp_33	150	v2_300	TRUE	[]	TRUE	FALSE	0.0633	0.8876

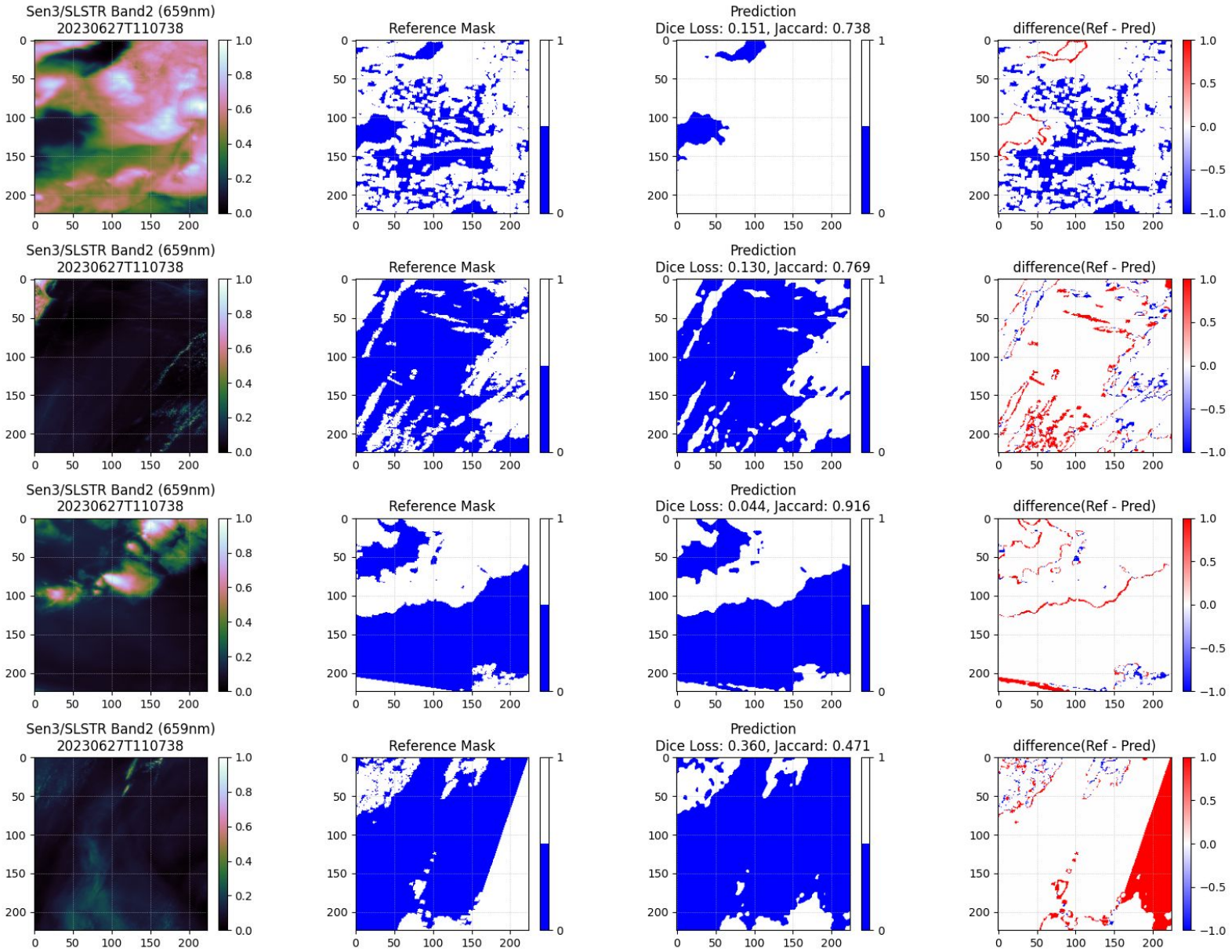




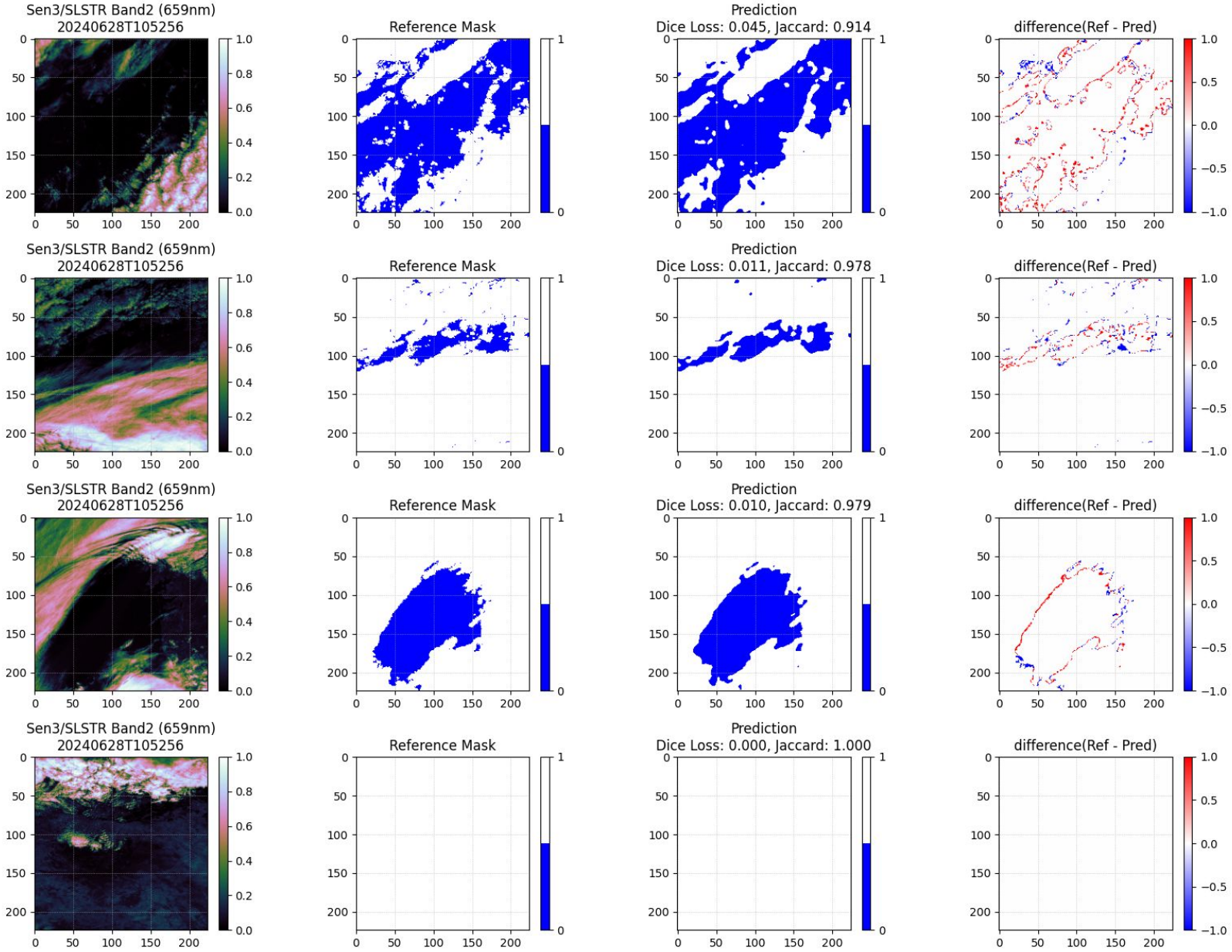
Effect of longer training (more epochs):
Without pre-trained backbone (exp 1 vs 31):
4 % improvement
BUT: still worse than with pre-trained

With pre-trained backbone (exp 3 vs 33):
2 % improvement

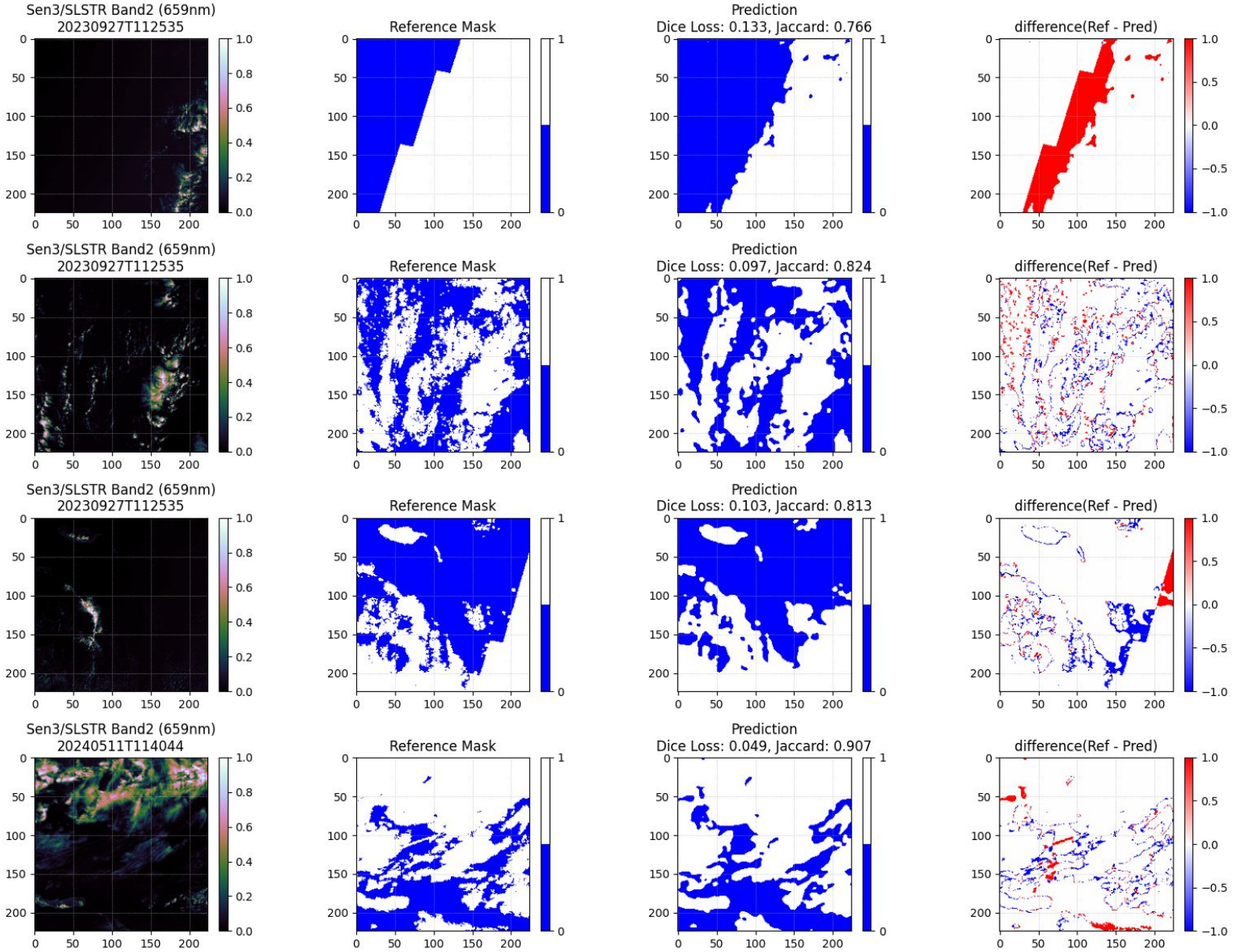
Results: Visual examples from best model (6)



Results: Visual examples from best model (6)



Results: Visual examples from best model (6)



Results

Ablation Studies (exact improvements also depend on other settings):

- **pre-trained weights, aka the foundation model itself improves accuracy ~ 5-6 %**
- Training all weights improves accuracy ~1 - 2 %
- Using time and location embeddings improves accuracy ~ 2- 3 %
- Using more parameters improves MJI ~1 – 3 %
- Training only the head (without pre-trained weights) is only 9.8 % worse than the best model
- More epochs improve MJI ~2 – 4 %

Workflow:

- Not only improvement of performance through pre-trained weights, but also python framework (terratorch) makes it easy and quick to use
- As always in ML: large part of the task is pre-processing the dataset

Other advantages:

- Computation probably much faster, especially if a GPU is available

Results

Research Questions and Answers

- How does a foundation model perform that was pre-trained with a higher resolution?
  - It can be used and improves the accuracy!

- How does a foundation model perform that was pre-trained on other spectral bands?
  - It can be used and improves the accuracy, even when choosing suboptimal corresponding spectral bands!

- How easy is it to apply foundation models? What computational and intellectual resources are required?
  - One 20GB GPU is enough, if the individual training data points are small (here: 224x224x6).

Mistakes/ Limitations

Pre-Processing/ Data curation:

- Maybe tested if the conversion Radiance to Reflectance is necessary for the model – it doesn't seem like it.
- Wrong corresponding SLSTR bands in dataset
- Can't cut out what I want/ unflexible cut out because of regridding
- Too much Land data involved very easy to predict

Model experiments:

- Test with data from different tiles

Evaluation:

- Predictions are overly smoothed common ML problem!

- No comparison to "traditional" ML
  - Only comparison between encoder with pre-trained weights and encoder with random weights, which basically is the comparison between using the foundation model or without, but not a comparison to a different baseline ML model

Future steps

- Improve model accuracy even further by:
  - More data
  - Find ‘best’ configuration

- Do more ablation studies to understand the behaviour:
  - How does it perform with less data?
  - How does it perform on other locations and times?
  - How important is the class balance?
  - How flexible is the model with the used spectral bands?
  - Test it with other, even coarser instruments like SEVIRI (Meteosat)

- Adapt training data by hand and test against numerical algorithm for smoke cases

References

Bommasani et al. 2022 – On the Opportunities and Risks of Foundation Models

Jakubik et al. 2023 – Foundation Models for Generalist Geospatial Artificial Intelligence (Prithvi 1.0)

Szwarcman et al. 2025 - Prithvi-EO-2.0: A Versatile Multi-Temporal Foundation Model for Earth Observation Applications

Zhu et al. 2024 – On the Foundations of Earth and Climate Foundation Models

Thank you!

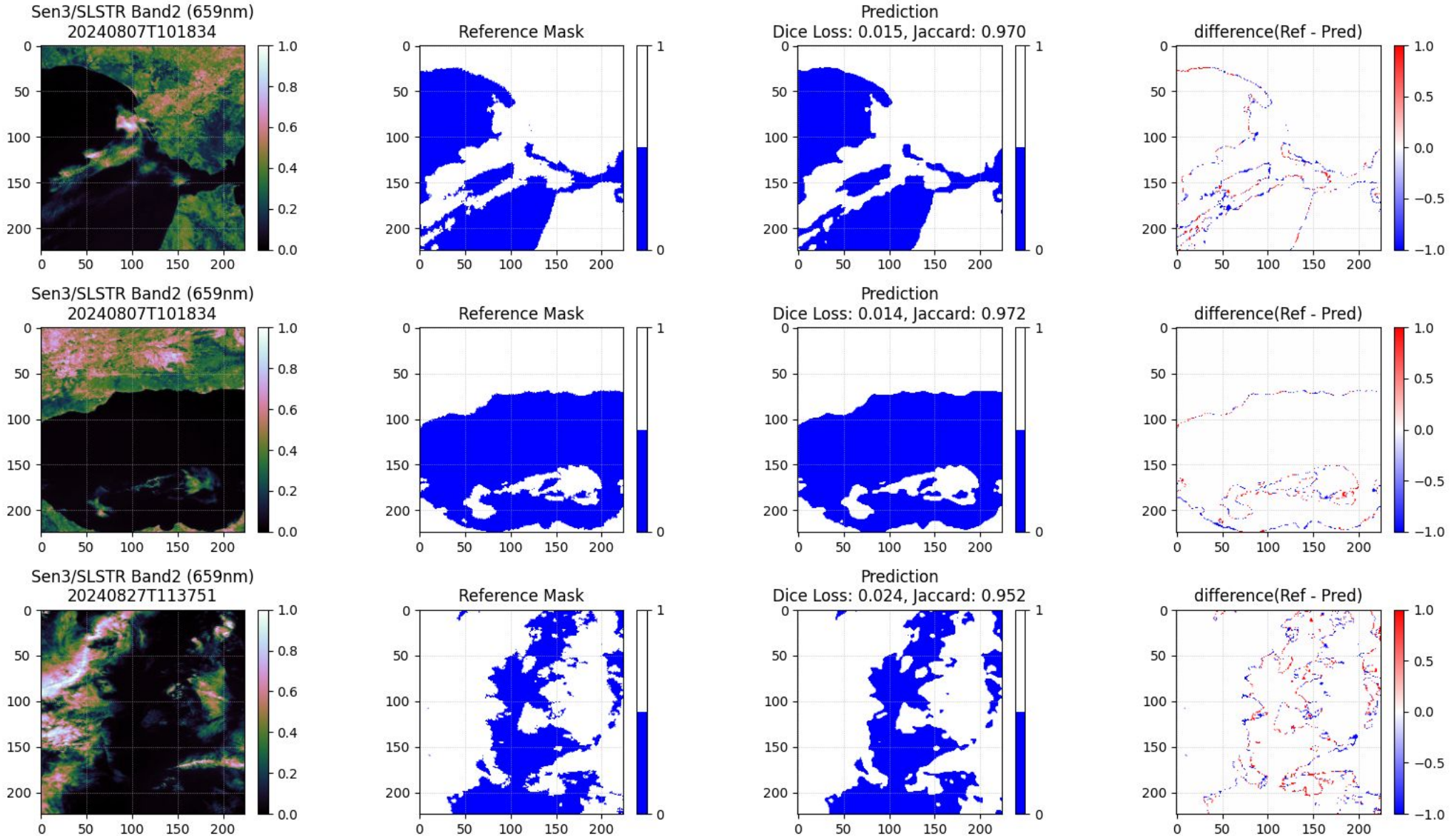
Questions are welcome.

Contact:

Boran Frank, Anna-Lena Erdmann (LinkedIn)

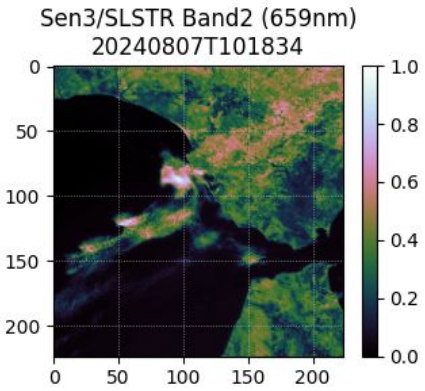
Results: Visual comparisons

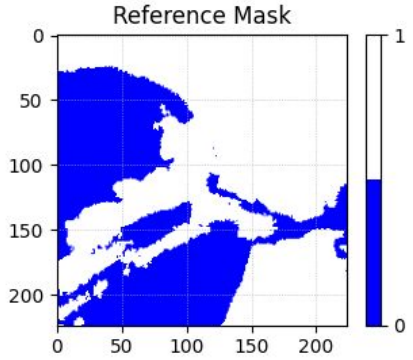
Exp_33:

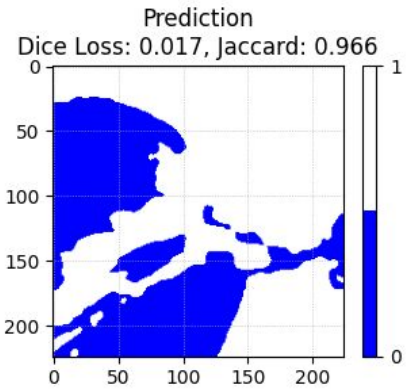


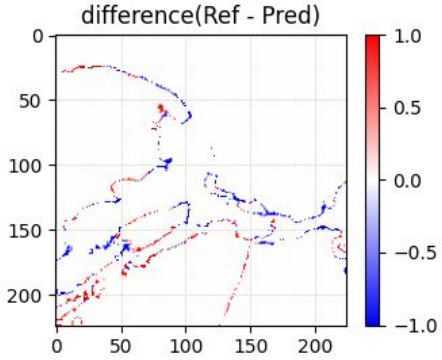
Results: Visual comparisons

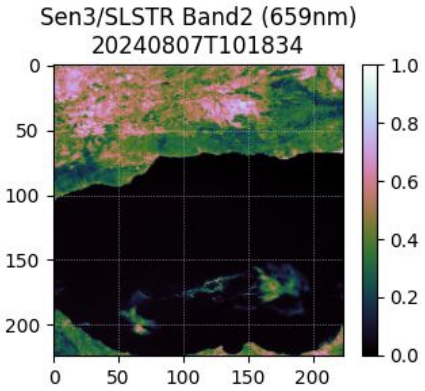
Exp_6:

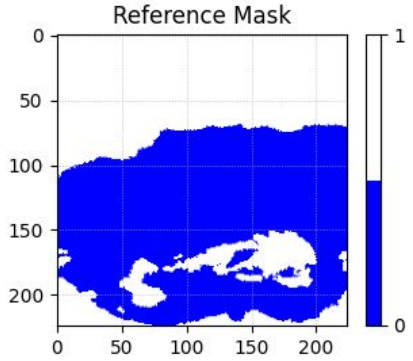


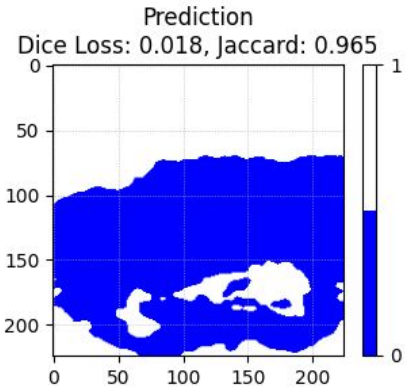


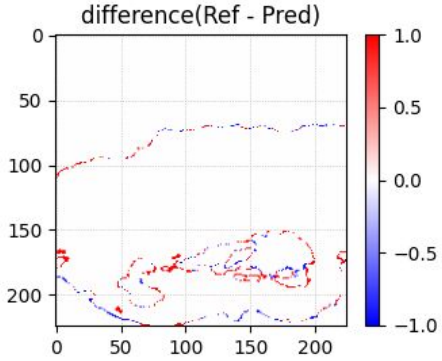


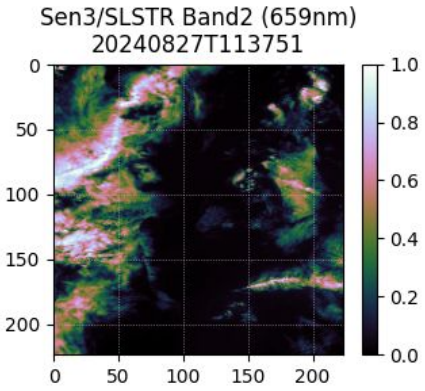


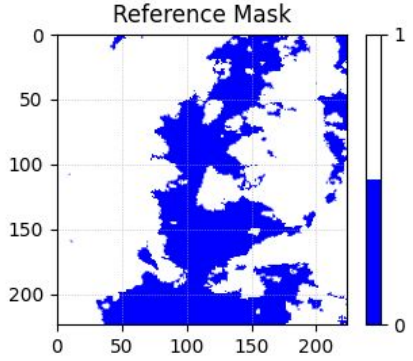


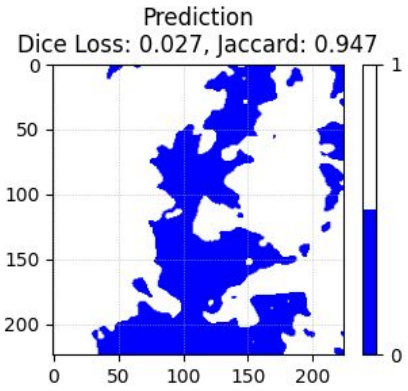


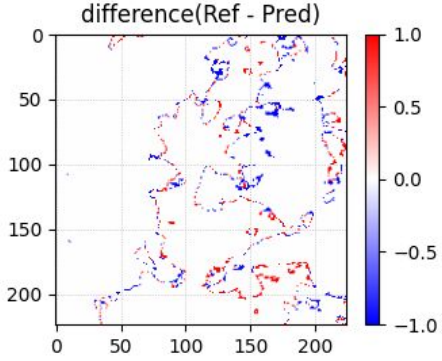












Results: Visual comparisons

Exp_7: Only head, no pre-trained backbone