Towards Predicting Wildfires in Polesia

Hamish Campbell [1] **Grace Colverd** [1] **Thomas Højlund-Dodd** [1] **Sofija Stefanović** [1]

1. Introduction

As climate change continues to increase average temperatures globally it is certain that wildfire risk will increase in a variety of different regions internationally [1]. Such changes in ambient temperature directly perturb several cryological and hydrological environmental properties, for example, losses in snow depth/coverage due to reductions in precipitation and earlier snow melting [2]. In Europe, hydrological models have predicted rates of evapotranspiration are likely to increase in-tact with temperature rises [3], which may have the indirect effect of reducing soil and vegetation moisture.

Figure 1. The Polesia region. [4]

One region especially susceptible to such changes are the wetlands of Polesia, a floodplain habitat of 186,000km^2 comprised of peatland, wetlands, and forests [5]. Polesia's susceptibility to wildfire may lie in its peat bogs' sensitivity to reductions in moisture, with drier land more likely to suffer from wildfire [6]. Additionally, reductions in vegetation moisture increase the likelihood of combustion and hence wildfire [7]. Furthermore, direct anthropogenic stresses including land drainage, purposeful fires, and logging have tangible effects upon wildfire frequency and ferocity [5].

Having recognised direct and indirect effects and their potential to exacerbate wildfires in the Polesian region, this project aimed to identify spatial relationships between wildfire distribution and cryological, climatic, and pedological drivers of wildfire. Therewith allowing wildfire prediction which would facilitate the mitigation of destructive wildfire occurrences. This was to be achieved in two steps; firstly, identification of target and predictor geospatial environmental datasets- one predictor, land-cover type, was derived from previous research commissioned by the British Trust for Ornithology (BTO), where a land-cover classification algorithm was specially trained on a sub-region of Polesia [8]. Secondly, a CNN was used to reduce the dimensionality of the datasets sampled from- the architecture was based on a U-net originally described for biomedical image segmentation [9]. This paper will present the target and predictor datasets utilised and their origins, before presenting the practical methods and considerations taken into account when building the CNN. Finally, test results will be presented and discussed before final remarks are made on the challenges yet to be solved.

2. Datasets

2.1. Target data:

1. **MODIS Burned Area** The ground truth for our prediction is provided by NASA, in the form of the Fire_cci Burned Area per pixel product [10]. We processed this into a monthly binary target variable.

2.2. Input data:

1. **Landcover Map** Landcover type is a driver of fire, hence we used a landcover map customised for the Polesia region. This map is a one-off snapshot of the region, generated using 2018 data. The classification separates out 9 classes of landcover, with a resolution of 20m. [8])

2. **Normalised Differential Moisture and Water Index (NDMI & NDWI)**. A key risk factor for wildfires is the moisture content of the local area, hence we included a fine-grained NDMI and NDWI, in the form of 3 spectral bands from Sentinel-2 data, provided by ESA, with a resolution of 50m. [11]

3. **Land Temperature, Snow Cover and Soil Moisture** To account for large scale temporal variation in climatic variables, we include four ERA5 indices, at monthly frequency and a lower resolution of 30km. [12].

   (a) Temperature 2m above land
   (b) Snow Cover
   (c) Soil Water

(d) Snow Depth

Details on data processing and accessing datasets found in Appendix B .

3. Methodology

Here, we describe problem formulation, data alignment, key features of the model used and steps to train it.

3.1. Problem Statement

We frame the problem of fire prediction as a binary semantic segmentation task which aims to classify each pixel as either the positive ('fire') class, or the background ('no fire') class. The goal is to learn a relationship between the high-dimensional dataset combining layers from different data sources outlined above and the pixel class label.

3.2. Data Alignment

To go from our datasets to model input we align three datasets spatiotemporally and downscale the MODIS and Sentinel datasets to the resolution of the landcover dataset (for brevity ERA5 was excluded). This choice in downscaling is made to preserve granularity in the landcover dataset, informed by *a priori* knowledge of its importance for predicting wildfires. We align datasets using TorchGeo, a library designed to allow users to bypass the common pre-processing steps necessary to align geospatial data with imagery and performs this processing on-the-fly [13]. This approach avoids the need for storing intermediate data, and is easily extendable with any further transformations to the raw data.

3.3. Model Setup

We use a standard U-Net architecture for semantic image segmentation introduced in [9] with four input channels corresponding to the landcover layer and the three bands of Sentinel. The U-Net is based on a fully convolutional network [14] and consists of similar downsampling and upsampling parts, which yields a u-shape architecture. The U-Net skip connections allow the spatial structure of the input image to be fed in across the net. For the downsampling (encoder) layer we use the ResNet-18.

The network outputs a tensor of probabilities, and each probability p_i of pixel i is converted into a binary mask with $p_i > 0.5$ classified as *fire* and $p_i < 0.5$ classified as *not fire*.

3.4. Training setup

We train the network from scratch for our problem,

We use binary cross entropy implemented in *PyTorch* [15] as the loss function and ignore the background class, effectively creating a weighted loss which assigns zero weight to the background class.

$$\ell = 1\{y_i = 1\} \cdot \sum_{i=0}^{1} y_i \log(\hat{y}_i)$$

where y_i is the prediction probability and $\hat{y}_i$ is the target probability, corresponding to class i.

The model is trained with 256×256 image patches sampled simultaneously from the four channels which were aligned as described in 3.2. The key step here is sampling randomly (i.e. indexing the dataset using a valid spatiotemporal bounding box) across the region in which landcover data is available using the RandomBatchGeoSampler implemented in TorchGeo [13].

We use the Adam optimizer, reduce learning rate on validation loss plateaus, and early stop based on validation loss. To find the best initial learning rate we make use of the auto learning rate finder available in wandb [16], and the sweep functionality to tune the batch size and number of epochs (details in appendix C).

4. Testing, Results and Discussion

4.1. Testing

The constrained sampler was tested to ensure the samples were indeed samples classified as 'burned'. Note that a burned sample is defined as any sample with ≥ 1 burned pixel.

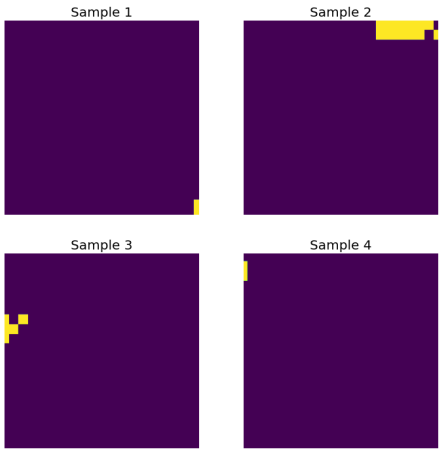

Figure 2. An example of 'burned' samples produced by the constrained sampler (yellow: burned, purple: non-burned).

As an initial model test, the sampler was forced to output the same sample on every call, such that the effect of overfitting the model to a single sample could be observed, seen in 3 (red: burned, blue: non-burned). Clearly the the model is capable of learning to correctly predict the segmentation.

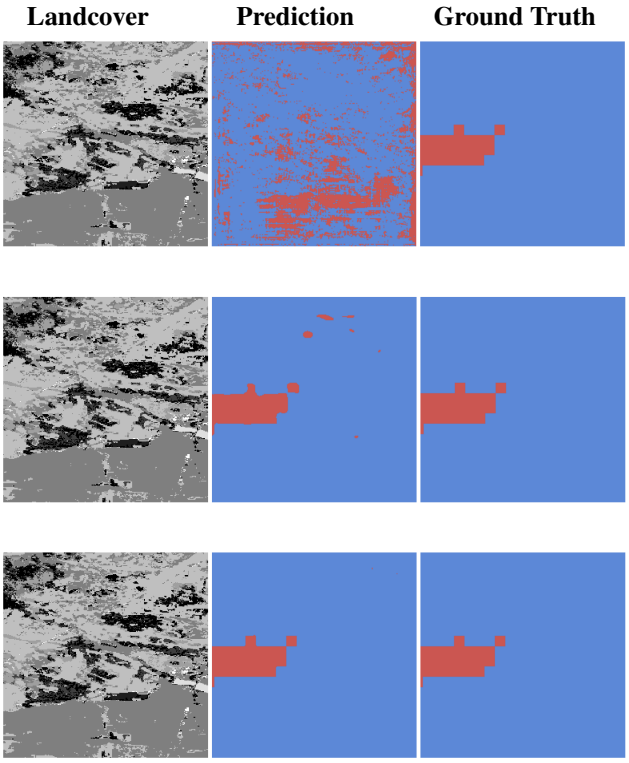

Figure 3. Training and validation using one sample: overfitting_to_one_sample. *Top row:* validation results using untrained model. *Middle Row:* validation results after 1 epoch of training. *Bottom Row:* validation results after 2 epochs of training.

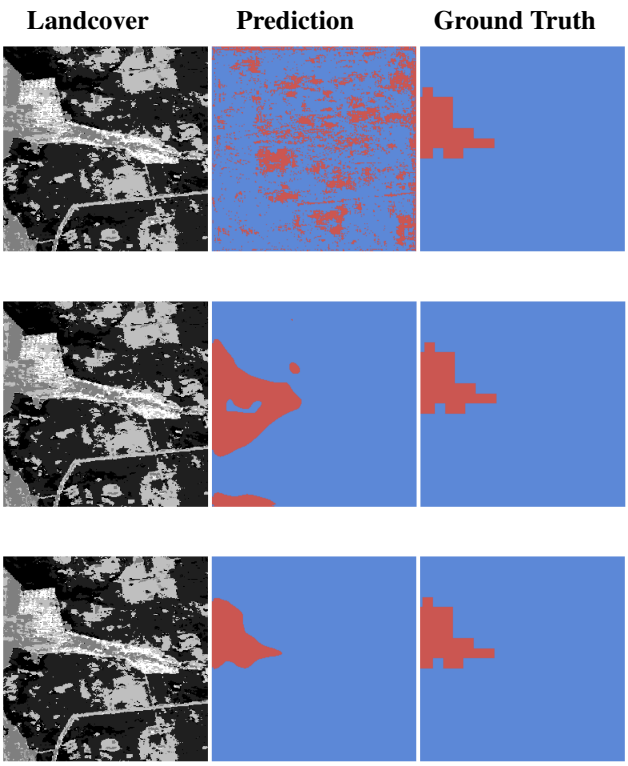

Figure 4. Training using 3 samples and validation with 1 sample from training set: overfitting_to_three_samples. *Top row:* validation results using untrained model. *Middle Row:* validation results after 12 epochs of training. *Bottom Row:* validation results after 50 epochs of training.

As a secondary test, the model was then overfit to a set of three possible input samples. The model was validated against a single sample to test the ability of the model to generalise learning to more than one input sample. The results for this test, shown in 5, illustrates that the model is capable of this. It should be noted that the training time required in this case was higher than for the single sample overfitting case, but the network was capable of accurate prediction at the conclusion of training.

As a final initial test, as previously, the model was trained on a set of three possible samples. However, this time the model observed a new sample during validation. The results from this test can be seen in Figure 5 (the difference in colour is due to a bug in wandb). As expected, the model fails to generalise to an unseen sample due to a lack of variety in the training data.

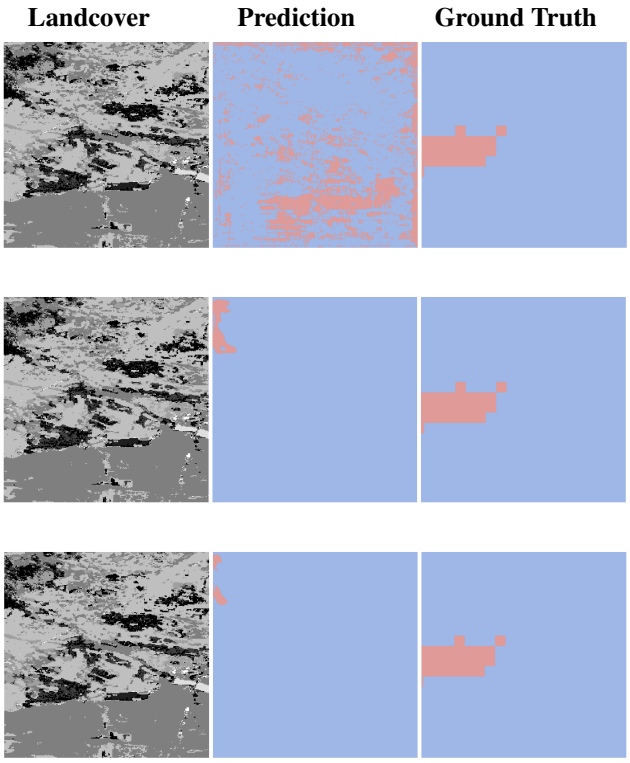

Figure 5. Training using 3 samples and validation with new sample: overfitting_to_three_new_val. *Top row:* validation results using untrained model. *Middle Row:* validation results after 9 epochs of training. *Bottom Row:* validation results after 30 epochs of training.

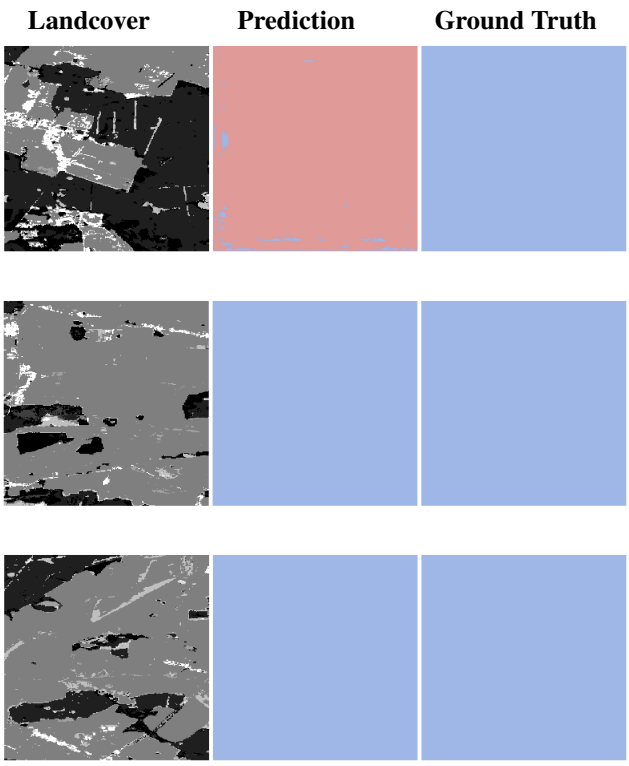

Figure 6. Training and validation with random samples: Non-constrained_samples_output. *Top row:* validation results using untrained model. *Middle Row:* validation results after 1 epoch of training. *Bottom Row:* validation results after 9 epochs of training.

4.2. Results and Discussion

4.2.1. Results

We ran the complete model with patch size = 256, length = 2048, batch size = 16 for a maximum of 100 epochs. In the first instance, the standard random sampler was utilised for the interest of run time. However, due to the low proportion of burn pixels present within the dataset (approx 0.0015%), the model quickly learned to output no burned pixels for the whole sample area, with an accuracy of ≈ 0.998 and Jaccard index of ≈ 0.49. This is illustrated by the predictions shown in Figure 6.

To mitigate this issue, the model must use the constrained sampler. However, due to the additional complexity involved in using constrained samples, this run will take significantly longer to complete. Therefore, the output of this run will be provided at a later date.

4.2.2. Challenges and Extensions

This project presented a number of challenges:

1. Our planned loss functions (Jaccard and Focal Tver-

sky) were incompatible with wandb tracking, and were found to be reporting incorrect results. Our solution was to use weighted-cross-entropy as an alternative, and use Jaccard as a metric instead. This greatly limited the ability of our model to learn from the few burned samples it received. We believe that fixing the integration of these loss functions will reduce the frequency of completely non-burned predictions.

2. The TorchGeo package is still under development, and our use of it to stretched the bounds of its tested capabilities. In particular, our custom constrained sampler is not yet working in full training mode. We are working with the developers to overcome this, and when fixed, will be a valuable contribution from the team to the TorchGeo package. This challenge slowed our ability to present the model with enough burned samples for it to learn. Once fixed, we will be able to better select samples with fire, such that the model can better learn to predict fires.

3. We are training our model weights from scratch, so getting meaningful predictions requires a lot of compute power and time. A potential improvement would be to use pre-trained Imagenet weights for three channels and initialise the others randomly.

We see a number of paths forward to extend this project:

1. **ERA5 Variables** We have already created a ERA5 TorchGeo class, but would need to update the datamodule to union this dataset with our current dataset. This would allow us to see the impacts in segmentation ability of including the long-range temperature variability.

2. **One-hot Encoding** We have already built this functionality into the Landcover TorchGeo class, so would just need to re-run the model training with this switched on to see the impact on predictions.

3. **Expand Training Data Period** For brevity, we have run the model using only 4 years of Sentinel 2 data. An extension would be to use Landsat data to fill in years back to 2000 and then train over the whole 20 year time period.

4. **Explore MODIS Confidence Data** The MODIS data set includes a confidence level for burned areas. It would be interesting to see if using this data impacted model's predictions.

Acknowledgements

We are thankful for sponsorship received from the European Space Agency (ESA) and the British Trust for Ornithology (BTO), and for mentorship from Martin Rogers (British Arctic Survey), Adham Ashton-Butt (BTO), James Wheeler (ESA) and Stephen Briggs.

5. Report Authors

Report

1. **Introduction:** Thomas Højlund-Dodd
2. **Datasets:** Grace Colverd
3. **Methodology:** Sofija Stefanović
4. **Testing:** Hamish Campbell
5. **Results:** Sofija Stefanović
6. **Challenges and Extensions:** Grace Colverd

Appendices

1. **Code Overview:** Hamish Campbell
2. **Dataset Access and Preprocessing:** Grace Colverd
3. **Hyperparameter Tuning:** Sofija Stefanović

References

[1] V. MassonDelmotte. 'summary for policymakers. in: Climate change 2021: The physical science basis. contribution of working group i to the sixth assessment report of the intergovernmental panel on climate change'. Technical report, IPCC (Intergovernmental Panel on Climate Change, 2021.

[2] A. L. Westerling and B. P. Bryant. Climate change and wildfire in california. *Climatic Change*, 87(1):231–249, 2008.

[3] Chunyu Dong and Lucas Menzel. Recent snow cover changes over central european low mountain ranges. *Hydrological Processes*, 34(2):321–338, 2020.

[4] SP (Save Polesia). 'polesia - unique, wild, and untouched'. https://savepolesia.org/polesia/. Accessed: 2022-03-18.

[5] SP (Save Polesia). 'polesia under threat; how a new waterway could destroy polesia's natural environment'. https://file.ejatlas.org/docs/5049/SavePolesia_Factsheet_Polesia-under-threat.pdf. Accessed: 2022-03-18.

[6] FZS (Frankfurt Zoological Society). 'achieving protected status for polesia europe's largest wetland wilderness'. https://fzs.org/en/news/achieving-protected-status-for-polesia-europes-largest-wetland-wilderness/. Accessed: 2022-03-16.

[7] M. Follette-Cook. 'satellite observations and tools for fire risk, detection, and analysis' nasa applied remote sensing training program (arset). https://appliedsciences.nasa.gov/join-mission/training/english/arset-satellite-observations-and-tools-fire-risk-detection-and. Accessed: 2022-03-13.

[8] Thomas Dowling and mdj4. tpfd/polesia-landcover: Polesia mapping v1, December 2021.

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[10] Chris Justice, NASA. Accessed Dec, 2021.

[11] Sentinel -2 Spectral Reflectance, Level-2A. Accessed Mar, 2021.

[12] European Centre for Medium-Range Weather Forecasts (ECMWF) ReAnalysis v5 ERA5. Accessed Dec, 2021.

[13] Adam J. Stewart, Caleb Robinson, Isaac A. Corley, Anthony Ortiz, Juan M. Lavista Ferres, and Arindam Banerjee. TorchGeo: deep learning with geospatial data. *arXiv preprint arXiv:2111.08872*, 11 2021.

[14] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651, 2017.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. 2019.

[16] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[17] T.P.F Dowling and De Jong M. C. Technical report: Mapping Polesia. November 2021.

[18] NDMI for Sentinel 2, Sentinel Hub. Accessed Mar, 2021.

[19] NDWI for Sentinel 2, Sentinel Hub. Accessed Mar, 2021.

Appendices

A. Code

All code can be found in our GitHub repository: https://github.com/ai4er-cdt/WildfireDistribution

A.1. Overview

The high level training script is *train.py*. This script makes use of a PyTorch Lightning wrapper to easily complete each stage of setup and training. Within this script, any source code referenced can be found within the *src* folder. This folder contains all the code necessary to access the data and train a model for the binary semantic segmentation task:

1. ***/datamodules*** This is where the main data object can be found. The datamodule object combines the Landcover, Sentinel and MODIS dataset classes into a single, unified dataset and creates dataloaders for training, validation and testing.

2. ***/samplers*** Within the training dataloader object, a constrained sampler is utilised to ensure only samples that contain ≥ 1 burned pixel are observed. This constraint is necessary due to the relative abundance of non-burned pixels, compared to burned ones (mean proportion of burn pixels across all years = 0.0015%). However, the standard random sampler used for validation and testing is contained within the TorchGeo library.

3. ***/evaluation*** Contains the callback object required to plot input and segmentation mask outputs within wandb.

A.2. Running an Experiment

To run an experiment, follow the steps:

1. **Data** Complete data download as detailed below

2. **Environment Setup** Use the *requirements/environment.yml* file to create a conda environment. Activate this environment once it has been created.

3. **Alter File Paths** Within the *train.py* script, file paths to input data should be changed.

4. **Parameter Choices** All (hyper)parameter choices and the name of the output directory can be specified in a configuration file that should be placed in the *config/* directory.

5. **Run train.py** To run the training script pass the path to the file with configurations from the command line. Example usage can be found in the repository.

B. Accessing Datasets and Data Processing

B.1. Data download

To download MODIS and Sentinel 2 datasets, create a conda environment with data-env.yml and run download-datasets.py

For Landcover and ERA5, use the download links: Polesia Landcover Classified Tiles Download ERA5

B.2. Dataset Information and Processing

1. **MODIS Burned Area** is provided by ESA, and contains burned area and confidence level information on a per-pixel basis (7, 8), derived from MODIS. The dataset is comprised of separate monthly files.

MODIS Burned Area per Pixel

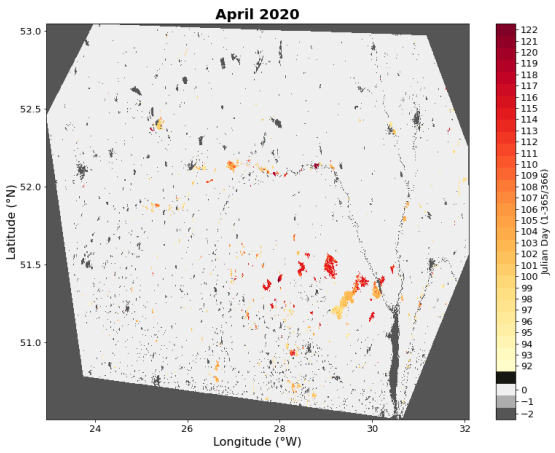

Figure 7. An example of the MODIS burned area information for April 2020, showing Julian Day of burn. '-1' corresponds to patches that were covered in cloud the whole month, and '-2' indicates pixel is non-burnable i.e. water.

MODIS data is available for download from the CEDA Archive, or directly via JASMIN HPC - /neodc/esacci/fire/data/burned_area/MODIS/.

The pre-processing of the dataset is done within our custom TorchGeo 'MODIS' Class and consists of the following:

(a) **Binarize** We covert the numeric Julian day of burn into a binary value of: '0' for no burn or '1' for burn observed within a given month.

2. **Polesia Landcover Mapping** was generated using the following open-access mapping repository:

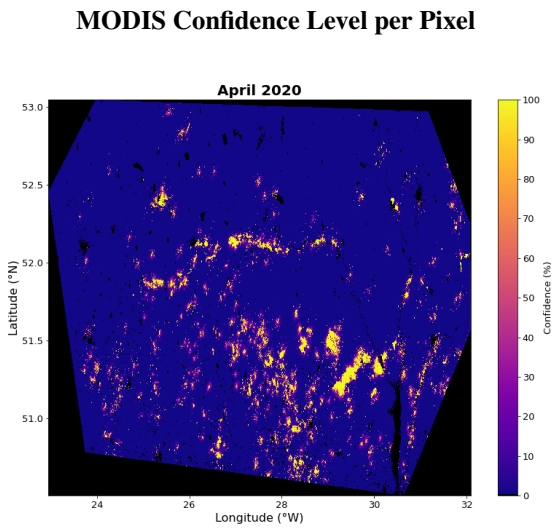

Figure 8. An example of the MODIS burn confidence level for April 2020.

https://github.com/tpfd/Polesia-Landcover

This generates both a 'simple' and 'complex' landcover mapping, as seen in 9 with the former splitting out 9 categories or landcover, and the latter 13. In this report we used the simple land cover mapping; an extension would be to explore the impact on our results of using the complex landcover mapping.

We are grateful to Thomas Dowling for his support with generating our version of the Landcover mapping.

The pre-processing of the Landcover data is done within our custom TorchGeo 'Polesia Landcover' Class and consists of the following:

(a) One Hot Encoding - this optional step can be applied for feed in to a CNN to detach the class numbers from the class labels.

3. **NDMI and NDWI** were created from Bands 3, 8 and 11 of Sentinel 2 data using the following definitions:

$NDMI = \frac{B8-B11}{B8+B11}$ [18] $NDWI = \frac{B3-B8}{B3+B8}$ [19]

The spectral bands of these indices were input into the CNN in their uncombined state, due to the hierarchical nature of the CNN architecture.

We plot the indices in 10 for illustrative purposes. NDWI distinguishes water content of bodies, while NDMI distinguishes water content of vegetation.

Sentinel 2 data is available for download from Google Earth Engine, via web or API, or via Google BigQuery.

The pre-processing we implemented involves:

(a) **Cloud Removal** Clouds, cloud shadows and dark pixels were removed, integrating the cloudless masking method detailed

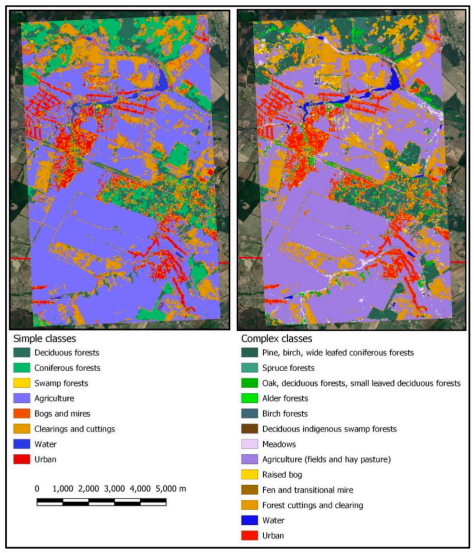

Figure 9. Left: an example of simple landcover mapping. Right: an example of complex landcover mapping, both 20m resolution [17].

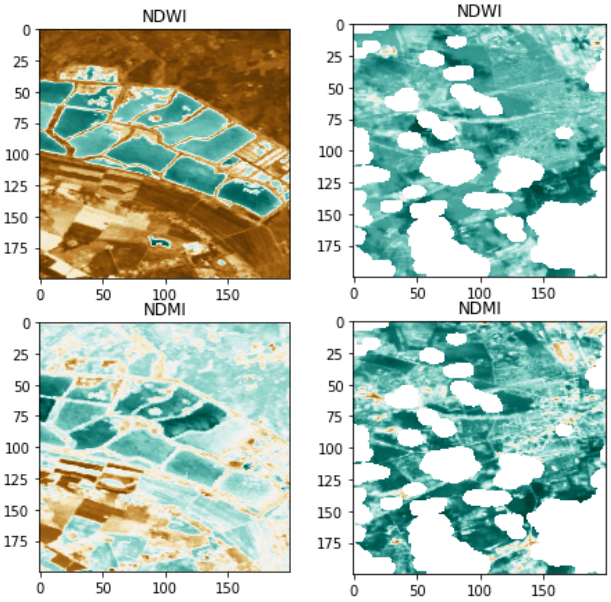

Figure 10. Examples of Sentinel 2 Data. Blue tone indicates higher water content. Left: A cloud-free image of a dry May 2018. Note how NDMI picks up water in the vegetation as well as bodies of water. Right: A partially cloudy image, from a damp Feb 2018.

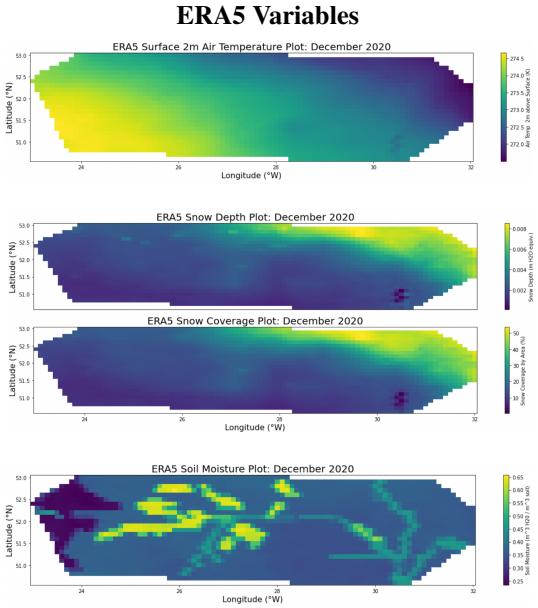

Figure 11. Plots of ERA5 Variables for December 2020.

here: https://developers.google.com/earth-engine/tutorials/community/sentinel-2-s2cloudless.

(b) **Normalising** each spectral band to within 0-1, before input into CNN. The following method of normalisation was used: $\frac{value-min}{max-min}$ using the minimum and maximum values of each band across the whole time period of interest.

An extension of this work would be to improve the cloud removal process further. In particular, over some winter months there is cloud covering some pixels across the whole month and therefore gaps in the data. An extension of this work would be to fill in these gaps, and fine tune the cloud removal parameters used in the download script.

The Sentinel-2 data was integrated into our CNN using a customised versions of the TorchGeo Sentinel2 class, updated to deal with the monthly nature of the data. An extension of this work would be to integrate the data download and pre-processing into this custom class and push it to the TorchGeo package.

4. **ERA5** data in monthly format was provided to us by Martin Rogers (BAS), for the following indices:

   (a) Temperature 2m above land surface

   (b) Snow Cover

   (c) Snow Depth

   (d) Volumetric Soil Water

B.3. TorchGeo Classes

The custom TorchGeo classes for Landcover and MODIS and the associated methods will shortly be pushed to the TorchGeo package, and in the meantime can be found in */src/datasets*.

We are grateful to Adam J. Stewart and the rest of the TorchGeo developers for their support with our use of their package.

C. Hyperparameter Tuning