A. Implementation Details

This work was implemented in Python 3.10 and the machine learning functionality used PyTorch. The code and a list of required libraries is available at https://github.com/JAEarly/MIL-Multires-EO. The majority of model training was carried out on a remote GPU service using a Volta V100 Enterprise Compute GPU with 16GB of VRAM, which utilised CUDA v11.0 to enable GPU support (IRIDIS 5, University of Southampton). Training each model took a maximum of four hours. Trained models can be found in the code repository. Fixed seeds were used to ensure consistency of dataset splits between training and testing; these are included in the scripts that are used to run the experiments. We used Weights and Biases (Biewald, 2020) to track our experiments, along with Optuna for hyperparameter optimisation (Akiba et al., 2019). During hyperparameter optimisation, we ran 40 trials with pruning using the Tree-structured Parzen Estimator sampler (Bergstra et al., 2011).

B. Datasets

In this section, we give further details on the DeepGlobe (Section B.1) and FloodNet (Section B.2) datasets used in this work.

B.1. DeepGlobe

The DeepGlobe-LCC dataset is openly available and can be acquired from Kaggle. Below we give further details on the dataset, which are adapted from the Kaggle page.[1]

Data

The DeepGlobe-LCC dataset consists of 803 satellite images with 3 channels: red, green, and blue (RGB). Each image is 2448 x 2448 pixels with 50cm pixel resolution. All images were sourced from the WorldView3 satellite, covering regions in Thailand, Indonesia, and India. The Kaggle challenge also has validation and test datasets with 171 and 172 images respectively, but as these datasets do not include segmentation masks, they were not used in this work.

Labels

Each satellite image is paired with a mask image for land cover annotation. Each mask is an image with 7 classes of labels, using colour-coding (RGB) described below. We also give an overview of the class distribution in Figure A1.

0. **Urban land** (0, 255, 255) — Man-made, built-up areas with human artefacts (ignoring roads which are hard to label).
1. **Agriculture land** (255, 255, 0) — Farms, any planned (i.e., regular) plantation, cropland, orchards, vineyards, nurseries, and ornamental horticultural areas.
2. **Rangeland** (255, 0, 255) — Any non-forest, non-farm, green land, grass.
3. **Forest land** (0, 255, 0) — Any land with x% tree crown density plus clearcuts.
4. **Water** (0, 0, 255) — Rivers, oceans, lakes, wetland, ponds.
5. **Barren land** (255, 255, 255) — Mountain, land, rock, desert, beach, no vegetation.
6. **Unknown** (0, 0, 0) — Clouds and others.

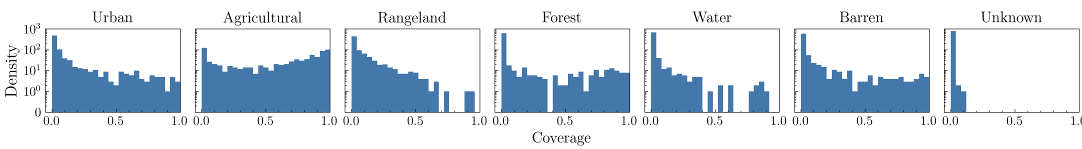

Figure A1. DeepGlobe Class Distribution. *We compute the class coverage for each image in the dataset, then plot a log histogram with 25 bins.*

[1] https://www.kaggle.com/datasets/balraj98/deepglobe-land-cover-classification-dataset

Terms and Conditions

The DeepGlobe Land Cover Classification Challenge and dataset are governed by DeepGlobe Rules, DigitalGlobe's Internal Use License Agreement, and Annotation License Agreement.

Further Details

While the DeepGlobe-LCC dataset provides pixel-level annotations, these segmentation labels are *only* used to generate the regression targets for training and for the evaluation of derived patch segmentation, i.e., they are not used during training. However, we would like to stress that these segmentation labels are not strictly required for our approach, i.e., the scene-level regression targets can be created without having to perform segmentation.

We used 5-fold cross-validation rather than the standard 10-fold due to the limited size of the datasets (only 803 images). With this configuration, each fold had an 80/10/10 split for train/validation/test. We normalised the images by the dataset mean (0.4082, 0.3791, 0.2816) and standard deviation (0.06722, 0.04668, 0.04768). No other data augmentation was used.

B.2. FloodNet

The FloodNet dataset was originally used as a competition dataset as part of EARTHVISION 2021. It was created by Bina Lab (a computer vision and remote sensing laboratory at the University of Maryland, Baltimore County). This work used the openly available data provided on GitHub.[2] Below we give further details on the dataset specifics.

Data

FloodNet consists of 2343 high-resolution (4000 x 3000 px) RGB images of Ford Bend County in Texas, captured between August 30$^{\text{th}}$ to September 4$^{\text{th}}$ 2017 after Hurricane Harvey. Images were taken with DJI Mavic Pro quadcopters at 200 feet above ground level (flown by emergency responders during the disaster response phase). The aim is to capture the post-disaster effects, notably flooding.

Labels

Each aerial image is paired with a mask image. Each mask is a single-channel image with 10 classes of labels, where the pixel values indicate the pixel classes (described below). We also give an overview of the class distribution in Figure A2.

0. **Background** — Regions that do not fall into any of the other classes.
1. **Building Flooded** — Man-made structures where at least one side is touching flood water.
2. **Building Non-flooded** — Man-made structures where no sides are touching flood water.
3. **Road Flooded** — Roads covered by flood water.
4. **Road Non-flooded** — Roads not covered by flood water.
5. **Water** — Natural water bodies (e.g., rivers and lakes); considered distinct from flood water.
6. **Tree** — Vegetation including trees and bushes.
7. **Vehicle** — Car, lorries, trucks, etc.
8. **Pool** — Man-made swimming pools, typically behind houses.
9. **Grass** — Areas covered with grass.

Terms and Conditions

Any research using FloodNet should cite Rahnemoonfar et al. (2021).

Further Details

Similar to the DeepGlobe dataset, FloodNet provides pixel-level annotations, but these segmentation labels are *only* used to generate the regression targets for our training and for the evaluation

[2] https://github.com/BinaLab/FloodNet-Supervised_v1.0

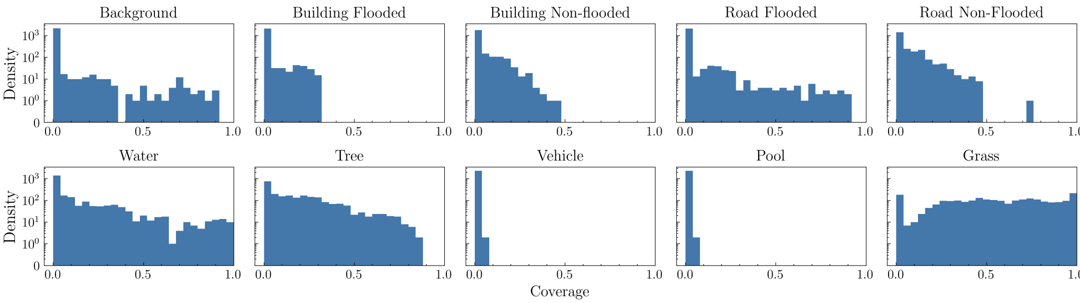

Figure A2. FloodNet Class Distribution. *We compute the class coverage for each image in the dataset, then plot a log histogram with 25 bins.*

of derived patch segmentation, i.e., they are not used during training. The dataset provides fixed train/validation/test splits, so we used these in our work (i.e., no cross-validation; instead five repeats on the same dataset splits). With this configuration, there were 1445/450/448 images (2343 total; ~ 61.7/19.2/19.1) for train/validation/test. We normalised the images by the dataset mean (0.4111, 0.4483, 0.3415) and standard deviation (0.1260, 0.1185, 0.1177). No other data augmentation was used.

C. Models and Training

In this section, we provide further details on our model configurations (Section C.1), training procedure (C.2), and architectures (C.3).

C.1. Model Configurations

We use 14 different model configurations in this work: one ResNet18 fully supervised approach, two U-Net models, and 11 different configurations of our Scene-to-Patch (S2P) approach (nine single-resolution and two multi-resolution). Further details are given below, along with summaries in Tables A1 and A2 for DeepGlobe and FloodNet respectively.

ResNet18. For the ResNet18 model, we treat our regression problem in a fully supervised manner, i.e., without using a MIL approach. Instead, the entire image is resized to 224 x 224 px (the size that ResNet18 expects), and the model makes (only) a scene-level prediction. Conceptually, this is equivalent to using a grid size of one and a patch size of 224 x 224 px (see Tables A1 and A2). We used a pre-trained ResNet18 model, with weights sourced from TorchVision.[3] We replaced the final classifier layer of the network with a new linear layer of the correct size (7 for DeepGlobe and 10 for FloodNet), and then re-trained the entire network, i.e., no weights were frozen during re-training.

U-Net Models. We used two different U-Net configurations — one using entire image inputs resized to 224 x 224 px, and the other 448 x 448 px. The model makes scene-level predictions using global average pooling over F (the output of the U-Net's final convolutional layer) followed by a single classification layer L. Using class activation maps, it is possible to recover pixel-level segmentation outputs: $M_c = W_c F + B_c$, where M_c is the class activation map for class c, W_c are the weights in L for class c, and B_c is the bias for class c in L. Note, for the U-Net upsampling process, we experimented with fixed bilinear or learnt convolutional upsampling; the latter increases the number of model parameters. This was included as a hyperparameter during tuning on the DeepGlobe dataset, and it was found that fixed upsampling was best for the U-Net 224 architecture, but learnt upsampling was best for U-Net 448 architecture, leading to an increase in the number of parameters for the U-Net 448 model.

Single Resolution S2P Models. We tested nine different configurations of our single resolution S2P models. Two parameters were changed: the grid size and the patch size. The grid size determines the number of cells extracted from the original image. The patch size is the dimension that each extracted cell is resized to before being input to the model and also determines the model architecture that is

[3] https://pytorch.org/vision/stable/models/generated/torchvision.models.resnet18.html#torchvision.models.resnet18

used, i.e., we used three different patch sizes and thus designed three different model architectures. This means models with different grid sizes but the same patch size used the same architecture, e.g., the S2P Large 8, S2P Large 16, and S2P Large 32 models all used the same model architecture (hence having the same number of model parameters in Tables A1 and A2).

Multi-Resolution S2P Models. We used two different configurations of the multi-resolution models, see Section 3.3. These models use the large single-resolution architecture as their backbone (as this was found to be most effective for single resolutions, see Section 5.3). Both multi-resolution models make predictions at the same resolution as the grid size = 32 single-resolution models ($s = m$), and the MRMO model also makes independent predictions at grid sizes 8, 16, and 32.

Table A1. DeepGlobe Configurations. *The grid size determines the number of cells and the size of each cell. Each cell is then resized (patch size), leading to a reduction in the overall image size (effective resolution and scale). # Params is the number of parameters in each model. Note, when using a grid size of 32, a patch size of 102 x 102 px is greater than the maximum possible cell size (i.e., the extracted cells would need to be upsampled and the effective resolution would be greater than 100%). Therefore, for grid size 32, we use a patch size of 76 x 76 px for the large model configurations.*

Configuration	Grid Size	Cell Size	Patch Size	Eff. Resolution	Scale	# Params
ResNet18	1 x 1	2448 x 2448 px	224 x 224 px	224 x 224 px	0.8%	11.18M
U-Net 224	1 x 1	2448 x 2448 px	224 x 224 px	224 x 224 px	0.8%	4.31M
U-Net 448	1 x 1	2448 x 2448 px	448 x 448 px	448 x 448 px	3.3%	7.76M
S2P SR Small 8	8 x 8	306 x 306 px	28 x 28 px	224 x 224 px	0.8%	707K
S2P SR Medium 8	8 x 8	306 x 306 px	56 x 56 px	448 x 448 px	3.3%	3.63M
S2P SR Large 8	8 x 8	306 x 306 px	102 x 102 px	816 x 816 px	11.1%	2.98M
S2P SR Small 16	16 x 16	153 x 153 px	28 x 28 px	448 x 448 px	3.3%	707K
S2P SR Medium 16	16 x 16	153 x 153 px	56 x 56 px	896 x 896 px	13.4%	3.63M
S2P SR Large 16	16 x 16	153 x 153 px	102 x 102 px	1632 x 1632 px	44.4%	2.98M
S2P SR Small 32	32 x 32	76 x 76 px	28 x 28 px	896 x 896 px	13.4%	707K
S2P SR Medium 32	32 x 32	76 x 76 px	56 x 56 px	1792 x 1792 px	53.6%	3.63M
S2P SR Large 32	32 x 32	76 x 76 px	76 x 76 px	2432 x 2432 px	98.7%	1.52M
S2P MRSO $s = 0$	8 x 8	306 x 306 px	76 x 76 px	608 x 608 px	6.2%	4.56M
S2P MRSO $s = 1$	16 x 16	153 x 153 px	76 x 76 px	1216 x 1216 px	24.7%	
S2P MRSO $s = 2$	32 x 32	76 x 76 px	76 x 76 px	2432 x 2432 px	98.7%	
S2P MRMO $s = 0$	8 x 8	306 x 306 px	76 x 76 px	608 x 608 px	6.2%	4.59M
S2P MRMO $s = 1$	16 x 16	153 x 153 px	76 x 76 px	1216 x 1216 px	24.7%	
S2P MRMO $s = 2$	32 x 32	76 x 76 px	76 x 76 px	2432 x 2432 px	98.7%	

Despite using larger patches, the S2P Large architectures have fewer parameters than the S2P Medium architectures as they use an additional convolutional and pooling layer, leading to smaller embedding sizes and thus fewer parameters in the fully connected layers (see Appendix C.3 for further details on the model architectures). Furthermore, while the DeepGlobe multi-resolution MIL models are larger (more parameters) than the DeepGlobe single-resolution MIL models, the DeepGlobe multi-resolution models have fewer parameters than the total number of parameters for the equivalent single-resolution models (a total of 7.48M parameters). This is because the DeepGlobe multi-resolution models use a patch size of 76 x 76 px rather than 102 x 102 px. For the FloodNet multi-resolution models, as the patch size is the same as the single-resolution models, the number of parameters is only slightly higher than the equivalent total for the single-resolution models (a total of 8.94M parameters). The implication of having fewer parameters (in the case of DeepGlobe), or only slightly more (in the case of FloodNet) is that using a single MRMO model is mostly equivalent to training three separate single-resolution models, with the added benefit of improved performance and faster training.

Table A2. FloodNet Configurations.

Configuration		Grid Size	Cell Size	Patch Size	Eff. Resolution	Scale	# Params
ResNet18		1 x 1	4000 x 3000 px	224 x 224 px	224 x 224 px	0.4%	11.18M
U-Net 224		1 x 1	4000 x 3000 px	224 x 224 px	224 x 224 px	0.4%	4.31M
U-Net 448		1 x 1	4000 x 3000 px	448 x 448 px	448 x 448 px	1.7%	7.76M
S2P SR Small 8		8 x 6	500 x 500 px	28 x 28 px	224 x 168 px	0.3%	707K
S2P SR Medium 8		8 x 6	500 x 500 px	56 x 56 px	448 x 336 px	1.3%	3.63M
S2P SR Large 8		8 x 6	500 x 500 px	102 x 102 px	816 x 612 px	4.2%	2.98M
S2P SR Small 16		16 x 12	250 x 250 px	28 x 28 px	448 x 336 px	1.3%	707K
S2P SR Medium 16		16 x 12	250 x 250 px	56 x 56 px	896 x 672 px	5.0%	3.63M
S2P SR Large 16		16 x 12	250 x 250 px	102 x 102 px	1632 x 1224 px	16.6%	2.98M
S2P SR Small 32		32 x 24	125 x 125 px	28 x 28 px	896 x 672 px	5.0%	707K
S2P SR Medium 32		32 x 24	125 x 125 px	56 x 56 px	1792 x 1344 px	20.1%	3.63M
S2P SR Large 32		32 x 24	125 x 125 px	102 x 102 px	3264 x 2448 px	66.6%	2.98M
	$s = 0$	8 x 6	500 x 500 px	102 x 102 px	816 x 612 px	4.2%	
S2P MRSO	$s = 1$	16 x 12	250 x 250 px	102 x 102 px	1632 x 1224 px	16.6%	8.95M
	$s = 2$	32 x 24	125 x 125 px	102 x 102 px	3264 x 2448 px	66.6%	
	$s = 0$	8 x 6	500 x 500 px	102 x 102 px	816 x 612 px	4.2%	
S2P MRMO	$s = 1$	16 x 12	250 x 250 px	102 x 102 px	1632 x 1224 px	16.6%	8.98M
	$s = 2$	32 x 24	125 x 125 px	102 x 102 px	3264 x 2448 px	66.6%	

C.2. Training Procedure

Models were trained to minimise scene-level RMSE using the Adam optimiser. Hyperparameter details are given in Table A3. We utilised early stopping based on validation performance — if the validation RMSE had not decreased for 5 epochs (or the epoch limit was reached, which very rarely happened), we terminated the training procedure and reset the model to the point at which it caused the last decrease in validation loss. Hyperparameter tuning was only carried out on the DeepGlobe dataset, i.e., the same hyperparameters were used for the FloodNet models. This demonstrates that the hyperparameters found through tuning are robust, which is also supported by consistent values across the S2P models.

Table A3. Model training hyperparameters.

Configuration	Max Epochs	Learning Rate	Weight Decay	Dropout
ResNet18	30	0.05	0.10	N/A
U-Net 224	30	5×10^{-4}	1×10^{-5}	0.25
U-Net 448	30	5×10^{-4}	1×10^{-6}	0.2
S2P SR Small 8	30	1×10^{-4}	1×10^{-6}	0.05
S2P SR Medium 8	30	1×10^{-4}	1×10^{-5}	0.35
S2P SR Large 8	30	1×10^{-4}	1×10^{-5}	0.25
S2P SR Small 16	30	5×10^{-4}	1×10^{-6}	0.10
S2P SR Medium 16	30	1×10^{-4}	1×10^{-6}	0.05
S2P SR Large 16	30	1×10^{-4}	1×10^{-5}	0.35
S2P SR Small 32	30	1×10^{-4}	1×10^{-6}	0.25
S2P SR Medium 32	30	1×10^{-4}	1×10^{-4}	0.00
S2P SR Large 32	30	1×10^{-4}	1×10^{-6}	0.40
S2P MRSO	40	1×10^{-4}	1×10^{-6}	0.20
S2P MRMO	40	1×10^{-4}	1×10^{-6}	0.10

C.3. Model Architectures

The single-resolution S2P models all use a consistent architecture: a feature extractor (convolutional and pooling layers), followed by a patch classifier (fully connected layers), and finally, a MIL mean aggregator. The output of the classifier is a c-dimensional vector, which represents the prediction for the c classes ($c = 7$ for DeepGlobe and $c = 10$ for FloodNet). Each patch is passed independently through the feature extractor + patch classifier to produce a prediction for each patch, and then MIL mean aggregation is used to produce a scene-level prediction. For the multi-resolution models, there are three feature extractors, one for each input resolution, and a combined patch classifier that utilises embeddings from each input resolution (see Section 3.3). The MRMO model also has independent patch classifiers for each input resolution. Both datasets use the same model architectures for each configuration. Below, in Tables A4 to A11, we give the exact architectures used.

Table A4. S2P Single-Resolution Small Architecture; patch size 28. *For the Conv2d and MaxPool2d layers, the numbers in the brackets are the kernel size, stride, and padding. b is the bag size (number of patches), and c is the number of classes. The final three rows represent the aggregation and classification; the other rows are for the feature extractor.*

Layer	Type	Input	Output
Conv1	Conv2d(4, 1, 0) + ReLu	b x 3 x 28 x 28	b x 36 x 25 x 25
	MaxPool2d(2, 2, 0)	b x 36 x 25 x 25	b x 36 x 12 x 12
Conv2	Conv2d(3, 1, 0) + ReLu	b x 36 x 12 x 12	b x 48 x 10 x 10
	MaxPool2d(2, 2, 0)	b x 48 x 10 x 10	b x 48 x 5 x 5
	Flatten	b x 48 x 5 x 5	1 x b x 1200
FC1	FC + ReLU + Dropout	1 x b x 1200	1 x b x 512
FC2	FC	1 x b x 512	1 x b x 128
FC3	FC + ReLU + Dropout	1 x b x 128	1 x b x 64
FC4	FC	1 x b x 64	1 x b x c
-	MIL Mean Pooling	1 x b x c	1 x 1 x c

Table A5. S2P Single-Resolution Medium Architecture; patch size 56.

Layer	Type	Input	Output
Conv1	Conv2d(4, 1, 0) + ReLu	b x 3 x 56 x 56	b x 36 x 53 x 53
	MaxPool2d(2, 2, 0)	b x 36 x 53 x 53	b x 36 x 26 x 26
Conv2	Conv2d(3, 1, 0) + ReLu	b x 36 x 26 x 26	b x 48 x 24 x 24
	MaxPool2d(2, 2, 0)	b x 48 x 24 x 24	b x 48 x 12 x 12
	Flatten	b x 48 x 12 x 12	1 x b x 6912
FC1	FC + ReLU + Dropout	1 x b x 6912	1 x b x 512
FC2	FC	1 x b x 512	1 x b x 128
FC3	FC + ReLU + Dropout	1 x b x 128	1 x b x 64
FC4	FC	1 x b x 64	1 x b x c
-	MIL Mean Pooling	1 x b x c	1 x 1 x c

Table A6. S2P Single-Resolution Large Architecture; patch size 76. *Visualised in Figure 2.*

Layer	Type	Input	Output
Conv1	Conv2d(4, 1, 0) + ReLu	b x 3 x 76 x 76	b x 36 x 73 x 73
	MaxPool2d(2, 2, 0)	b x 36 x 73 x 73	b x 36 x 36 x 36
Conv2	Conv2d(3, 1, 0) + ReLu	b x 36 x 36 x 36	b x 48 x 34 x 34
	MaxPool2d(2, 2, 0)	b x 48 x 34 x 34	b x 48 x 17 x 17
Conv3	Conv2d(3, 1, 0) + ReLu	b x 48 x 17 x 17	b x 56 x 14 x 14
	MaxPool2d(2, 2, 0)	b x 56 x 14 x 14	b x 56 x 7 x 7
	Flatten	b x 56 x 7 x 7	1 x b x 2744
FC1	FC + ReLU + Dropout	1 x b x 2744	1 x b x 512
FC2	FC	1 x b x 512	1 x b x 128
FC3	FC + ReLU + Dropout	1 x b x 128	1 x b x 64
FC4	FC	1 x b x 64	1 x b x c
-	MIL Mean Pooling	1 x b x c	1 x 1 x c

Table A7. S2P Single-Resolution Large Architecture; patch size 102.

Layer	Type	Input	Output
Conv1	Conv2d(4, 1, 0) + ReLu	b x 3 x 102 x 102	b x 36 x 99 x 99
	MaxPool2d(2, 2, 0)	b x 36 x 99 x 99	b x 36 x 49 x 49
Conv2	Conv2d(3, 1, 0) + ReLu	b x 36 x 49 x 49	b x 48 x 47 x 47
	MaxPool2d(2, 2, 0)	b x 48 x 47 x 47	b x 48 x 23 x 23
Conv3	Conv2d(3, 1, 0) + ReLu	b x 48 x 23 x 23	b x 56 x 21 x 21
	MaxPool2d(2, 2, 0)	b x 56 x 21 x 21	b x 56 x 10 x 10
	Flatten	b x 56 x 10 x 10	1 x b x 5600
FC1	FC + ReLU + Dropout	1 x b x 5600	1 x b x 512
FC2	FC	1 x b x 512	1 x b x 128
FC3	FC + ReLU + Dropout	1 x b x 128	1 x b x 64
FC4	FC	1 x b x 64	1 x b x c
-	MIL Mean Pooling	1 x b x c	1 x 1 x c

Table A8. S2P Multi-Resolution Single Out Architecture; patch size 76. *This architecture utilises the feature extractor from the single-resolution large architecture (Table A6; layers Conv1 to FC2) to create embeddings at each input resolution. The embeddings are then concatenated (see Section 3.4), and finally classified using the same classifier and MIL pooling approach as in the other models. Note, the patch predictions of size 1 x b x c are still produced by this model but are omitted from the table for simplicity. Visualised in Figure 3.*

Layer	Input	Output
s = 0 Feature Extractor	b x 3 x 76 x 76	1 x b x 128
s = 1 Feature Extractor	4b x 3 x 76 x 76	1 x 4b x 128
s = 2 Feature Extractor	16b x 3 x 76 x 76	1 x 16b x 128
Multi-resolution Concatenation	1 x $\{b, 4b, 16b\}$ x 128	1 x 16b x 384
s = m Classifier	1 x 16b x 384	1 x 1 x c

Table A9. S2P Multi-Resolution Single Out Architecture; patch size 102. *This architecture utilises the feature extractor from the single-resolution large architecture (Table A7; layers Conv1 to FC2).*

Layer	Input	Output
s = 0 Feature Extractor	b x 3 x 102 x 102	1 x b x 128
s = 1 Feature Extractor	4b x 3 x 102 x 102	1 x 4b x 128
s = 2 Feature Extractor	16b x 3 x 102 x 102	1 x 16b x 128
Multi-resolution Concatenation	1 x $\{b, 4b, 16b\}$ x 128	1 x 16b x 384
s = m Classifier	1 x 16b x 384	1 x 1 x c

Table A10. S2P Multi-Resolution Multi-Out Architecture; patch size 76. *This architecture utilises the feature extractor from the single-resolution large architecture (Table A6; layers Conv1 to FC2) to create embeddings at each input resolution. The embeddings are then concatenated (see Section 3.4), and finally classified using the same classifier and MIL pooling approach as in the other models. In addition, each set of embeddings is also independently classified. Patch predictions are produced for each independent resolution as well as the combined resolution, but are omitted from the table for simplicity. Visualised in Figure 3.*

Layer	Input	Output
s = 0 Feature Extractor	b x 3 x 76 x 76	1 x b x 128
s = 1 Feature Extractor	4b x 3 x 76 x 76	1 x 4b x 128
s = 2 Feature Extractor	16b x 3 x 76 x 76	1 x 16b x 128
s = 0 Classifier	1 x b x 128	1 x 1 x c
s = 1 Classifier	1 x 4b x 128	1 x 1 x c
s = 2 Classifier	1 x 16b x 128	1 x 1 x c
Multi-resolution Concatenation	1 x $\{b, 4b, 16b\}$ x 128	1 x 16b x 384
s = m Classifier	1 x 16b x 384	1 x 1 x c

Table A11. S2P Multi-Resolution Multi-Out Architecture; patch size 102. *This architecture utilises the feature extractor from the single-resolution large architecture (Table A7; layers Conv1 to FC2).*

Layer	Input	Output
s = 0 Feature Extractor	b x 3 x 102 x 102	1 x b x 128
s = 1 Feature Extractor	4b x 3 x 102 x 102	1 x 4b x 128
s = 2 Feature Extractor	16b x 3 x 102 x 102	1 x 16b x 128
s = 0 Classifier	1 x b x 128	1 x 1 x c
s = 1 Classifier	1 x 4b x 128	1 x 1 x c
s = 2 Classifier	1 x 16b x 128	1 x 1 x c
Multi-resolution Concatenation	1 x $\{b, 4b, 16b\}$ x 128	1 x 16b x 384
s = m Classifier	1 x 16b x 384	1 x 1 x c

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyper-parameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. https://doi.org/10.1145/3292500.3330701

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*, *24*.

Biewald, L. (2020). Experiment tracking with Weights and Biases [Software available from wandb.com]. https://www.wandb.com/

Rahnemoonfar, M., Chowdhury, T., Sarkar, A., Varshney, D., Yari, M., & Murphy, R. R. (2021). Flood-Net: A high resolution aerial imagery dataset for post flood scene understanding. *IEEE Access*, *9*, 89644–89654. https://doi.org/10.1109/access.2021.3090981