Scene-to-Patch Earth Observation: Multiple Instance Learning for Land Cover Classification

Joseph Early[*] **Ying-Jung Deweese**[†] **Christine Evers**[*] **Sarvapali Ramchurn**[*]

Abstract

Land cover classification (LCC), and monitoring how land use changes over time, is an important process in climate change mitigation and adaptation. Existing approaches that use machine learning with Earth observation data for LCC rely on fully-annotated and segmented datasets. Creating these datasets requires a large amount of effort, and a lack of suitable datasets has become an obstacle in scaling the use of LCC. In this study, we propose Scene-to-Patch models: an alternative LCC approach utilising Multiple Instance Learning (MIL) that requires only high-level scene labels. This enables much faster development of new datasets whilst still providing segmentation through patch-level predictions, ultimately increasing the accessibility of using LCC for different scenarios. On the DeepGlobe-LCC dataset, our approach outperforms non-MIL baselines on both scene- and patch-level prediction. This work provides the foundation for expanding the use of LCC in climate change mitigation methods for technology, government, and academia.

1 Introduction

Land use/land cover (LULC) change has been recognised as a major contributor to the rise of atmospheric carbon dioxide (CO2). Changes in LULC have significant affects on the carbon balance within ecosystem services that contribute to climate change mitigation [9]. LULC remote sensing (RS) techniques are widely used for areas such as sustainable development, crop health/yield monitoring, deforestation, urban planning, and water availability [6, 12]. Due to the recent curation of large volumes of accessible data, machine learning is seeing increased use in RS for Earth Observation (EO). In Land Cover Classification (LCC), the objective of machine learning is to identify and segment regions in satellite images according to a set of classes, e.g., agricultural land, urban land, water, etc. Typically, this requires a segmented dataset — a collection of images that have already been manually annotated with the different class regions. Creating these labelled datasets is a costly process: time and care must be taken to accurately segment the regions. This had lead to bottlenecks and concerns about a lack of datasets for machine learning in EO [1, 11]. In this work, we present an alternative approach that does not require segmentation datasets. Our contributions are as follows:

1. We reframe LCC as a Multiple Instance Learning (MIL) regression problem, reducing the need for segmentation labels whilst also preserving high-resolution data during training and inference.
2. We propose Scene-to-Patch MIL models that transform low-resolution scene labels to high-resolution patch predictions for LCC.
3. We explore how different data and model configurations of our approach affect performance on the popular DeepGlobe LCC dataset [6].[1]

The rest of this paper is laid out as follows. Section 2 provides the background to LCC, and Section 3 details our MIL approach. Our experiments are presented in Section 4, and Section 5 concludes.

[*]University of Southampton, UK; {J.A.Early,C.Evers,sdr1}@soton.ac.uk

[†]Georgia Institute of Technology, USA; yingjungcd@gmail.com

[1]Source code is available at https://github.com/JAEarly/MIL-Land-Cover-Classification

2 Background and Related Work

In this section, we provide the necessary background for our work, detailing LCC and its role in climate change mitigation, existing approaches to LCC, and a brief overview of MIL.

Land Cover Classification Machine learning for EO can be used in monitoring and forecasting of socioeconomic, ecological, agricultural, and hydrological problems [18]. For example, in the domain of monitoring green house gas (GHG) emissions, satellite images can be used to track emission sources and carbon sequestration [22]. It is estimated that human land use contributes about a quarter of global GHG emissions, and that a reduction of around a third of emissions could come from better land management [22]. As such, better understanding about how land is used can contribute towards zero emission targets. Improved policy design, planning, and enforcement can be achieved through real-time monitoring [14]. For example, automated LCC can be used to determine the effect of regulation or incentives to drive better land use practices [22]. LCC can also be used to detect the amount of acreage in use for farmland in order to assess food security and GHG emissions [26].

Existing LCC Approaches LCC is typically approached as an image segmentation problem. The objective is to perform pixel-wise classification of an input image, such that each pixel is assigned a label from a set of classes, in effect creating a new image where different regions in the original image have been separated and classified. This requires the original images to be segmented prior to training, i.e., all pixels must be annotated with a ground-truth class. Popular approaches include Fully Convolutional Networks [19], U-Net [23], and Pyramid Networks [17]. Existing works have applied these or similar approaches to LCC [15, 16, 21, 24, 25, 28]. We refer readers to [11] for a more in-depth review of existing work.

Multiple Instance Learning In MIL, data are grouped into bags of instances [5]. In comparison to conventional supervised learning, where each instance has a label, in MIL, only bag-level labels are provided. This reduces the burden of labelling, as only the bags need to be labelled, not every instance. MIL has seen some existing use with EO observation data, for example fusing panchromatic and multi-spectral images [18], settlement extraction [27], landslide mapping [31], and crop yield prediction [30]. However, to the best of the authors' knowledge, no prior work has studied the use of MIL for generic multi-class LCC. We discuss our approach to this problem in the next section.

3 Methodology

In this section, we propose our MIL approach to LCC. We first explain the process and benefits of reframing LCC as a MIL regression problem, and then detail the particular approach that we use.

3.1 Multiple Instance Learning for Land Cover Classification

For EO images, we identify three distinct tiers of data: *scene level* at the scale of the original images, *patch level* at the scale of small blocks of the original image (typically tens or hundreds of pixels), and *pixel level* at the scale of individual pixels. Segmentation-based models operate at the pixel level, and as such require pixel-level annotations. We propose a MIL-based approach that operates at the scene and patch levels. Instead of using pixel-level annotations, our approach only requires scene-level labels, i.e., for each input image, we only provide the proportion of each land cover class in that image. With these scene-level labels, LCC now becomes a regression problem, with the task of predicting the coverage proportion of each class. The primary motivation for such an approach is that it enables fast acquisition of training labels, as cost- and time-intensive pixel annotations are not required. It also reduces the likelihood of label errors, which often occur in EO datasets [21].

This reframed LCC problem does not necessitate a MIL approach — it can be treated as a purely supervised regression problem. However, as satellite images are often very high resolution, the images would have to be downsampled, and, as such, important data would be lost. With MIL, it is possible to operate at higher resolutions than a supervised learning approach. As depicted in Figure 1, our proposed MIL approach involves extracting patches from the original image using a grid, and then applying a MIL model to process and aggregate patch information. The MIL approach is operating on patch-level data while using scene-level labels, i.e., the images are converted into bags of patches, where each patch is an instance in MIL terms.

3.2 Scene-to-Patch Models for Land Cover Classification

Our proposed MIL approach utilises an Instance Space Neural Network (known as mi-Net in [29]). We call these models Scene-to-Patch (S2P) classifiers as they learn from scene-level labels, but

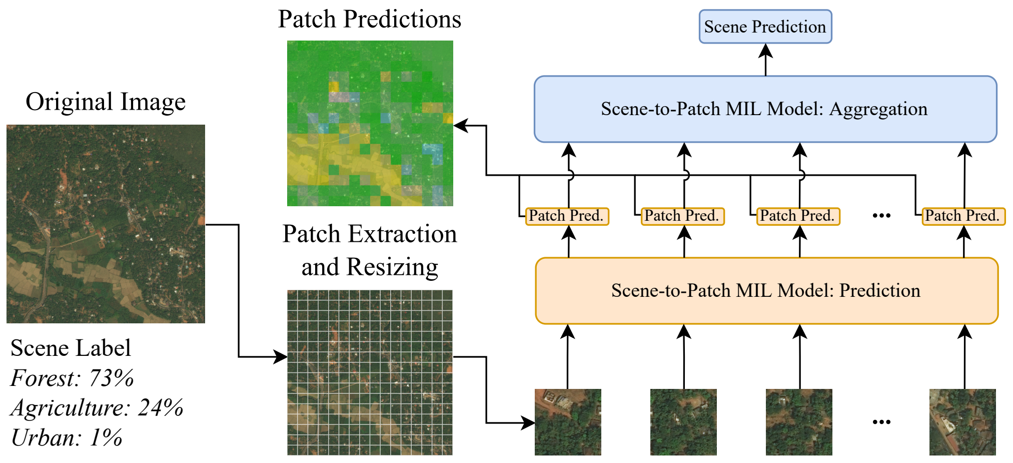

Figure 1: An overview of our MIL Scene-to-Patch approach for LCC. As the model produces both instance (patch) and bag (scene) predictions, scene-level labels are transformed to patch-level outputs.

are able to generate both scene- and patch-level predictions, i.e., they upscale from low-resolution training labels to higher-resolution outputs. For a bag of patches extracted from a satellite image, an S2P model makes a prediction for each patch, and then takes the mean of these patch-level predictions to make an overall scene-level prediction. These S2P models can be relatively small (in comparison to baseline architectures such as ResNet18 and UNet; see Appendix C), which means they are less expensive to train and run. This reduces the GHG emission effect of model development and use, which is an increasingly important consideration for the use of machine learning [14]. Other MIL models (e.g., with attention [13]) could be used, but these would require post-hoc interpretability [7].

4 Experiments

In this section we give detail our experiments. First we introduce the dataset and model configurations used in this work (Section 4.1), then provide our results and a discussion (Section 4.2).

4.1 Dataset and Models

Several datasets exist for using machine learning with EO [11, p. 30]. In this work we use the DeepGlobe-LCC dataset [6] — an image segmentation LULC dataset with data sourced from the WorldView3 satellite (for more details, see Appendix B). When transforming these EO images to MIL bags, there are two parameters to be determined: the size of the grid applied over the image (grid size), and the size that each cell of the grid is resized to (patch size). We experimented with three grid sizes (8, 16, and 24), and three patch sizes (small=28, medium=56, and large=102), resulting in nine different S2P configurations. Each patch size uses a slightly different model architecture. We compared our MIL S2P models to a fine-tuned ResNet18 model [10], and two UNet variations operating on different image sizes (224 x 224 and 448 x 448). These baseline models are trained in the same manner as the S2P models, i.e., using scene-level regression. Although the ResNet model does not produce patch- or pixel-level predictions, we use it as a scene-level baseline as many existing LCC approaches utilise ResNet architectures [11, 12]. For the UNet models, we follow the same procedure as [28] and use class activation maps to recover segmentation outputs. This makes the UNet approach a stronger baseline than ResNet as it can be used for both scene- and pixel-level prediction. For more details on the models and training process, see Appendix C.

4.2 Results

We evaluate performance on scene-, patch-, and pixel-level prediction. For scene-level predictions, we report the root mean square error (RMSE) and mean average error (MAE), where lower values are better. For patch-level predictions, we report the patch-level mean Intersection over Union (mIoU; [8, 20]), where larger values are better. For pixel-level prediction, we compute pixel-level mIoU using the original ground truth segmentation masks (see Figure 2). The pixel segmentation metric is preferred over the patch metric as it is independent of grid size and compares to the highest resolution ground truth segmentation. Strong models should perform well at both scene- and pixel-level prediction, i.e., low scene RMSE and high pixel mIoU. Our results are given in Table 1.

Table 1: Results for our nine MIL Scene-To-Patch (S2P) models and ResNet18 baseline. All results are given on the test dataset, and five repeats were conducted using 5-fold cross validation.

Configuration	Scene RMSE	Scene MAE	Patch mIoU	Pixel mIoU
ResNet18	0.218 ± 0.008	0.128 ± 0.004	N/A	N/A
UNet 224	0.136 ± 0.008	0.075 ± 0.004	N/A	0.245 ± 0.008
UNet 448	0.114 ± 0.006	0.064 ± 0.002	N/A	0.290 ± 0.010
S2P Small 8	0.107 ± 0.003	0.058 ± 0.002	0.366 ± 0.015	0.337 ± 0.014
S2P Medium 8	0.098 ± 0.004	0.054 ± 0.002	0.414 ± 0.014	0.375 ± 0.013
S2P Large 8	**0.090 ± 0.005**	**0.047 ± 0.002**	**0.439 ± 0.014**	**0.397 ± 0.014**
S2P Small 16	0.112 ± 0.007	0.059 ± 0.004	0.317 ± 0.007	0.305 ± 0.007
S2P Medium 16	0.093 ± 0.005	0.051 ± 0.003	0.407 ± 0.013	0.388 ± 0.012
S2P Large 16	0.097 ± 0.003	0.051 ± 0.001	0.404 ± 0.016	0.384 ± 0.014
S2P Small 24	0.099 ± 0.004	0.052 ± 0.002	0.371 ± 0.012	0.360 ± 0.011
S2P Medium 24	0.098 ± 0.004	0.053 ± 0.002	0.379 ± 0.015	0.367 ± 0.015
S2P Large 24	0.106 ± 0.009	0.056 ± 0.005	0.353 ± 0.006	0.343 ± 0.006

From these results, we observe that the S2P Large model with a grid size of 8 is the most effective across all metrics. The ResNet18 model performs significantly worse than all of the S2P models on scene-level prediction. The S2P models all decrease in performance from patch to pixel mIoU, which is to be expected as they are unable to produce the fine details required in pixel-level segmentation. Despite this, they still outperform the UNet models. For the most part, smaller grid sizes lead to better performance. However, larger grid sizes give higher-resolution image segmentation outputs, so there is a trade-off. We suspect that smaller grid sizes are more effective as they are an inherent form of regularisation — larger grid sizes mean larger bags, which allow the model to more easily overfit. There is also the consideration that grid sizes that are too large may not capture sufficient information in each cell to facilitate accurate classification. We give an example of the outputs in Figure 2.

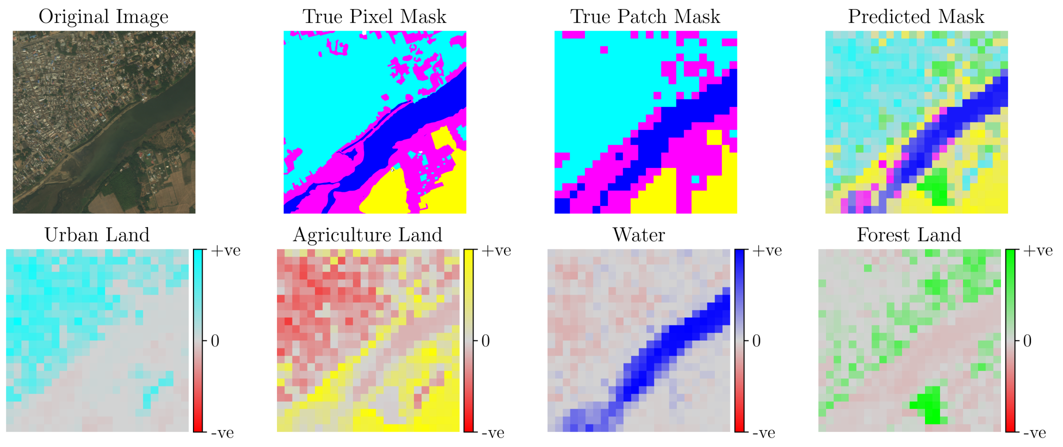

Figure 2: Example output of our S2P model (Medium 24). *Top*: Original image, ground truth masks, and overall predicted mask. *Bottom*: A per-class breakdown of the model predictions. Note how the model is able to both support (+ve) and refute (-ve) different classes for different regions of the image. There are also trees in the original image that have not been labelled as Forest land, but the model has been able to identify them. We provide further examples in Appendix D.

5 Conclusions and Future Work

In this work, we proposed a new approach to LCC. Our method involves using MIL models as Scene-to-Patch classifiers that transform low-resolution training labels into high-resolution predictions. This overcomes the need for segmentation labels and will help accelerate the development of further EO datasets by reducing the need for costly and time-consuming labelling of EO data. Our approach can be improved in several ways: further optimising model architectures, using multiple grid sizes in a single model, and utilising spatial relationships to improve predictive performance. Ultimately, our work is a baseline exploratory approach. By enabling faster and easier curation of larger and more diverse datasets, it is hoped this foundational work will lead to further developments in the use of LCC for climate change mitigation in technology, government, and academia.

Acknowledgments and Disclosure of Funding

This work was funded by the AXA Research Fund and the UKRI Trustworthy Autonomous Systems Hub (EP/V00784X/1). We would like to thank the University of Southampton and the Alan Turing Institute for their support. The authors acknowledge the use of the IRIDIS high-performance computing facilities and associated support services at the University of Southampton in the completion of this work. Finally, this work was supported by the Climate Change AI (CCAI) Mentorship Program.

References

[1] Climate Change AI Dataset Wishlist, 2022. URL https://www.climatechange.ai/dataset-wishlist.pdf. Accessed 2022-09-08.

[2] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[3] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[4] L. Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

[5] M.-A. Carbonneau, V. Cheplygina, E. Granger, and G. Gagnon. Multiple instance learning: A survey of problem characteristics and applications. *Pattern Recognition*, 77:329–353, 2018.

[6] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, and R. Raskar. Deepglobe 2018: A challenge to parse the earth through satellite images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 172–181, 2018.

[7] J. Early, C. Evers, and S. Ramchurn. Model Agnostic Interpretability for Multiple Instance Learning. In *International Conference on Learning Representations*, 2022.

[8] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[9] P. Friedlingstein, M. O'sullivan, M. W. Jones, R. M. Andrew, J. Hauck, A. Olsen, G. P. Peters, W. Peters, J. Pongratz, S. Sitch, et al. Global carbon budget 2020. *Earth System Science Data*, 12 (4):3269–3340, 2020.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] T. Hoeser and C. Kuenzer. Object detection and image segmentation with deep learning on earth observation data: A review-part i: Evolution and recent trends. *Remote Sensing*, 12(10): 1667, 2020.

[12] T. Hoeser, F. Bachofer, and C. Kuenzer. Object detection and image segmentation with deep learning on earth observation data: A review—part ii: Applications. *Remote Sensing*, 12(18): 3053, 2020.

[13] M. Ilse, J. Tomczak, and M. Welling. Attention-based deep multiple instance learning. In *International conference on machine learning*, pages 2127–2136. PMLR, 2018.

[14] L. H. Kaack, P. L. Donti, E. Strubell, G. Kamiya, F. Creutzig, and D. Rolnick. Aligning artificial intelligence with climate change mitigation. *Nature Climate Change*, pages 1–10, 2022.

[15] K. Karra, C. Kontgis, Z. Statman-Weil, J. C. Mazzariello, M. Mathis, and S. P. Brumby. Global land use/land cover with sentinel 2 and deep learning. In *2021 IEEE international geoscience and remote sensing symposium IGARSS*, pages 4704–4707. IEEE, 2021.

[16] T.-S. Kuo, K.-S. Tseng, J.-W. Yan, Y.-C. Liu, and Y.-C. Frank Wang. Deep aggregation net for land cover classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 252–256, 2018.

[17] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[18] X. Liu, L. Jiao, J. Zhao, J. Zhao, D. Zhang, F. Liu, S. Yang, and X. Tang. Deep multiple instance learning-based spatial–spectral classification for pan and ms imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 56(1):461–473, 2017.

[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[20] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[21] A. Rakhlin, A. Davydow, and S. Nikolenko. Land cover classification from satellite imagery with u-net and lovász-softmax loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 262–266, 2018.

[22] D. Rolnick, P. L. Donti, L. H. Kaack, K. Kochanski, A. Lacoste, K. Sankaran, A. S. Ross, N. Milojevic-Dupont, N. Jaques, A. Waldman-Brown, et al. Tackling climate change with machine learning. *ACM Computing Surveys (CSUR)*, 55(2):1–96, 2022.

[23] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[24] S. Seferbekov, V. Iglovikov, A. Buslaev, and A. Shvets. Feature pyramid network for multi-class land segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 272–275, 2018.

[25] X.-Y. Tong, G.-S. Xia, Q. Lu, H. Shen, S. Li, S. You, and L. Zhang. Land-cover classification with high-resolution remote sensing images using transferable deep models. *Remote Sensing of Environment*, 237:111322, 2020.

[26] F. Ullah, J. Liu, M. Shafique, S. Ullah, M. N. Rajpar, A. Ahmad, and M. Shahzad. Quantifying the influence of chashma right bank canal on land-use/land-cover and cropping pattern using remote sensing. *Ecological Indicators*, 143:109341, 2022.

[27] R. R. Vatsavai, B. Bhaduri, and J. Graesser. Complex settlement pattern extraction with multi-instance learning. In *Joint Urban Remote Sensing Event 2013*, pages 246–249. IEEE, 2013.

[28] S. Wang, W. Chen, S. M. Xie, G. Azzari, and D. B. Lobell. Weakly supervised deep learning for segmentation of remote sensing imagery. *Remote Sensing*, 12(2):207, 2020.

[29] X. Wang, Y. Yan, P. Tang, X. Bai, and W. Liu. Revisiting multiple instance neural networks. *Pattern Recognition*, 74:15–24, 2018.

[30] Z. Wang, L. Lan, and S. Vucetic. Mixture model for multiple instance regression and applications in remote sensing. *IEEE Transactions on Geoscience and Remote Sensing*, 50(6):2226–2237, 2011.

[31] M. Zhang, W. Shi, S. Chen, Z. Zhan, and Z. Shi. Deep multiple instance learning for landslide mapping. *IEEE Geoscience and Remote Sensing Letters*, 18(10):1711–1715, 2020.

A Implementation and Resource Details

This work was implemented in Python 3.10 and the machine learning functionality used PyTorch. All the libraries used are detailed in the Github repository for this work, which can be found at https://github.com/JAEarly/MIL-Land-Cover-Classification. 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. Training each model took a maximum of three and a half hours. Trained models can be found alongside the code in the Github 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 [4] to track our experiments, along with Optuna for hyperparameter optimisation [2]. During hyperparamater optimisation, we ran 40 trials with pruning using the Tree-structured Parzen Estimator sampler [3].

B Dataset

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.[2]

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 original challenge also had validation and test datasets with 171 and 172 images respectively, but as these datasets did not include 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) as follows:

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

Terms and Conditions

The DeepGlobe Land Cover Classification Challenge and dataset are governed by DeepGlobe Rules, The 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 our 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.

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

C Models and Training

In this section we give a formal definition of the MIL S2P models (C.1), then detail the model configurations (C.2), architectures (C.3), and training procedures (C.4).

C.1 Formal Model Definition

For a collection of n satellite images $\mathcal{X} = \{X_1, \dots, X_n\}$, each image $X_i \in \mathcal{X}$ has corresponding label $Y_i \in \mathcal{Y}$, where $Y_i = \{Y_i^1, \dots, Y_i^C\}$. C is the number of land cover classes, Y_i^c is the coverage proportion for class c in image X_i, and $\sum_{c=1}^{C} Y_i^c = 1$. For an input image $X_i = \{x_i^1, \dots, x_i^k\}$, where $x_i^j \in X_i$ is a patch extracted from the original image, our proposed Scene-to-Patch models make a prediction $\hat{y}_i^j$ for each patch, and then take the mean of these patch-level predictions to make an overall scene-level prediction $\hat{Y}_i = \frac{1}{k} \sum_{j=1}^{k} \hat{y}_i^j$. Note these models are trained entirely end-to-end and only use scene-level labels — no patch-level labels are used during training.

C.2 Model Configurations

We use twelve different model configurations in this work: one ResNet18 fully supervised approach, two UNet models, and nine different configurations of our Scene-to-Patch (S2P) approach. A summary of these configurations is given in Table A1, and further details are given below.

Table A1: The ten different configurations used in this work. 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; see C.3 for more details regarding the # Params in our S2P models.

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%	11.2M
UNet 224	1 x 1	2448 x 2448 px	224 x 224 px	224 x 224 px	0.8%	4.3M
UNet 448	1 x 1	2448 x 2448 px	448 x 448 px	448 x 448 px	3.3%	7.8M
S2P Small 8	8 x 8	306 x 306 px	28 x 28 px	224 x 224 px	0.8%	707K
S2P Medium 8	8 x 8	306 x 306 px	56 x 56 px	448 x 448 px	3.3%	3.6M
S2P Large 8	8 x 8	306 x 306 px	102 x 102 px	816 x 816 px	11.1%	3.0M
S2P Small 16	16 x 16	153 x 153 px	28 x 28 px	448 x 448 px	3.3%	707K
S2P Medium 16	16 x 16	153 x 153 px	56 x 56 px	896 x 896 px	13.4%	3.6M
S2P Large 16	16 x 16	153 x 153 px	102 x 102 px	1632 x 1632 px	44.4%	3.0M
S2P Small 24	24 x 24	102 x 102 px	28 x 28 px	672 x 672 px	7.5%	707K
S2P Medium 24	24 x 24	102 x 102 px	56 x 56 px	1344 x 1344 px	30.1%	3.6M
S2P Large 24	24 x 24	102 x 102 px	102 x 102 px	2448 x 2448 px	100.0%	3.0M

ResNet18 For the ResNet18 model, we treat our LCC regression problem in a fully supervised manner, i.e., without using a MIL approach. Instead, the entire satellite 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 Table A1). 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 seven outputs, and then re-trained the entire network, i.e., no weights were frozen during re-training.

UNet Models We used two different UNet 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 UNet'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 UNet upsampling process, we experimented with fixed bilinear or learnt convolutions upsampling; the latter increases the number of model parameters. This was included as a hyperparameter during tuning, and it was found that fixed upsampling was best for the UNet 224 architecture, but learnt upsampling was best for UNet 448 architecture, leading to an increase in the number of parameters for the UNet 448 model.

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

S2P Models We tested nine different configurations of our S2P models. Two parameters were changed: the grid size (8, 16, or 24), and the patch size (small=28, medium=56, or large=102). The grid size determines the number of extracted patches. The patch size determines the model architecture that was used, i.e., we designed three different architectures, one for each patch size. 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 24 models all used the same model architecture (hence having the same number of model parameters in Table A1).

C.3 S2P Model Architectures

The S2P models all used 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 7-dimensional vector, which represents the prediction for each class. Each patch is passed independently through the feature extractor + patch classifier to produce a prediction for each class, and then MIL mean aggregation is used to produce a scene-level prediction. The different architectures are given in Tables A2 - A4. Note that, despite using larger patches, the S2P large architecture has fewer parameters than the S2P Medium architecture (see Table A1) as it has an additional convolutional and pooling layer, leading to a smaller embedding size (5600 rather than 6912).

Table A2: S2P 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).

Layer	Type	Input	Output
1	Conv2d(4, 1, 0) + ReLu	b x 3 x 28 x 28	b x 36 x 25 x 25
2	MaxPool2d(2, 2, 0)	b x 36 x 25 x 25	b x 36 x 12 x 12
3	Conv2d(3, 1, 0) + ReLu	b x 36 x 12 x 12	b x 48 x 10 x 10
4	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	b x 1200
5	FC + ReLU + Dropout	b x 1200	b x 512
6	FC + ReLU + Dropout	b x 512	b x 128
7	FC + ReLU + Dropout	b x 128	b x 64
8	FC + ReLU + Dropout	b x 64	b x 7
-	MIL Mean Aggregation	b x 7	7

Table A3: S2P Small Architecture; patch size 56. 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).

Layer	Type	Input	Output
1	Conv2d(4, 1, 0) + ReLu	b x 3 x 56 x 56	b x 36 x 53 x 53
2	MaxPool2d(2, 2, 0)	b x 36 x 53 x 53	b x 36 x 26 x 26
3	Conv2d(3, 1, 0) + ReLu	b x 36 x 26 x 26	b x 48 x 24 x 24
4	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	b x 6912
5	FC + ReLU + Dropout	b x 6912	b x 512
6	FC + ReLU + Dropout	b x 512	b x 128
7	FC + ReLU + Dropout	b x 128	b x 64
8	FC + ReLU + Dropout	b x 64	b x 7
-	MIL Mean Aggregation	b x 7	7

Table A4: S2P Small Architecture; patch size 102. 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).

Layer	Type	Input	Output
1	Conv2d(4, 1, 0) + ReLu	b x 3 x 102 x 102	b x 36 x 99 x 99
2	MaxPool2d(2, 2, 0)	b x 36 x 99 x 99	b x 36 x 49 x 49
3	Conv2d(3, 1, 0) + ReLu	b x 36 x 49 x 49	b x 48 x 47 x 47
4	MaxPool2d(2, 2, 0)	b x 48 x 47 x 47	b x 48 x 23 x 23
5	Conv2d(3, 1, 0) + ReLu	b x 48 x 23 x 23	b x 56 x 21 x 21
6	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	b x 5600
7	FC + ReLU + Dropout	b x 5600	b x 512
8	FC + ReLU + Dropout	b x 512	b x 128
9	FC + ReLU + Dropout	b x 128	b x 64
10	FC + ReLU + Dropout	b x 64	b x 7
-	MIL Mean Aggregation	b x 7	7

C.4 Training Procedure

All of our models were trained to minimise scene-level root mean square error (RMSE) using the Adam optimiser. The hyperparamater details for learning rate, weight decay, and dropout are given in Table A5. We utilised early stopping based on validation performance — if the validation RMSE had not decreased for 5 epochs then we terminated the training procedure and reset the model to the point at which it caused the last decrease in validation loss. Otherwise, training terminated after 30 epochs.

Table A5: Model training hyperparameters.

Configuration	Learning Rate	Weight Decay	Dropout
ResNet18	0.05	0.1	N/A
UNet 224	5×10^{-4}	1×10^{-5}	0.25
UNet 448	5×10^{-4}	1×10^{-6}	0.2
S2P Small 8	1×10^{-4}	1×10^{-6}	0.05
S2P Medium 8	1×10^{-4}	1×10^{-5}	0.35
S2P Large 8	1×10^{-4}	1×10^{-5}	0.25
S2P Small 16	5×10^{-4}	1×10^{-6}	0.1
S2P Medium 16	1×10^{-4}	1×10^{-6}	0.05
S2P Large 16	1×10^{-4}	1×10^{-5}	0.35
S2P Small 24	1×10^{-4}	1×10^{-5}	0.05
S2P Medium 24	1×10^{-4}	1×10^{-6}	0.2
S2P Large 24	5×10^{-4}	1×10^{-5}	0.3

D Additional Interpretability Outputs

Below we provide further outputs of our S2P models. In Figure A1, we compare the predictions for the different S2P and UNet models. In Figures A2 - A6 we examine edge cases involving potential mislabelling and confusion in model prediction.

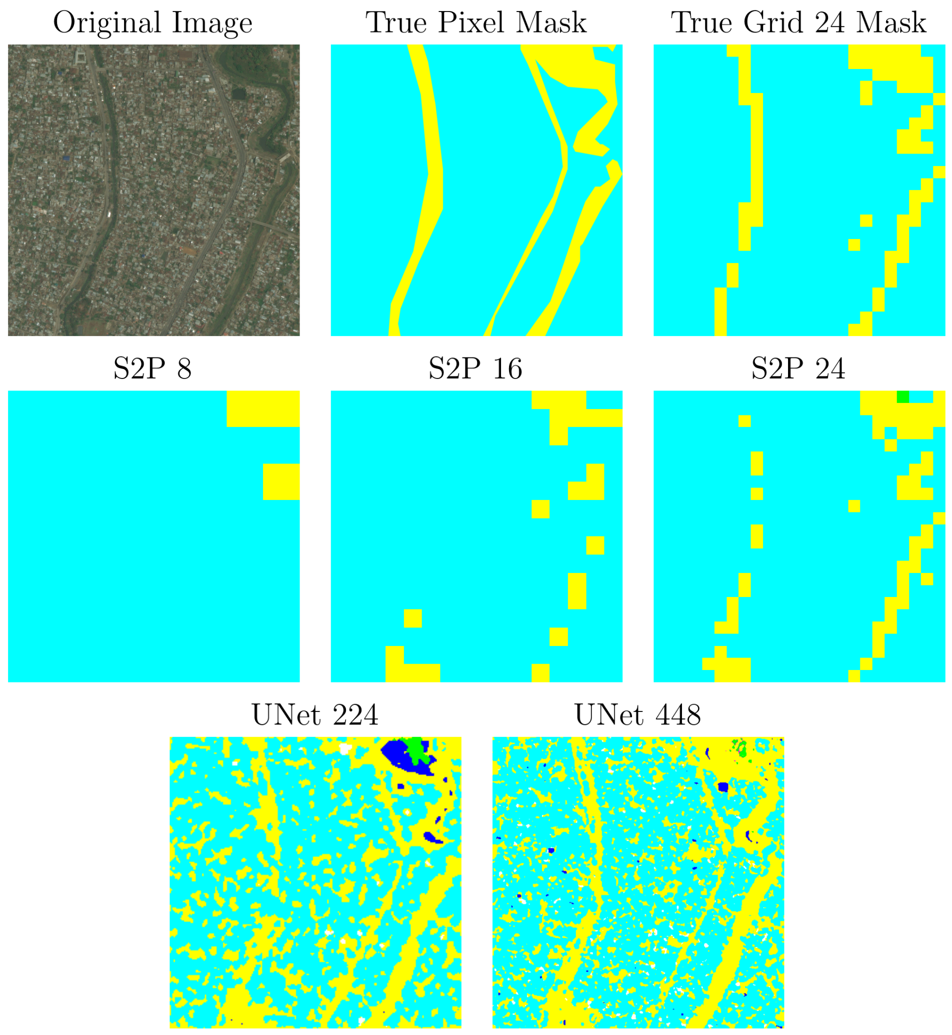

Figure A1: A comparison of patch predictions at different grid sizes. With a grid size of 8, the cells are too large to be able to resolve the fine detail of the agricultural land (yellow) — only the large areas of agricultural land and can be separated from the urban land (blue). Grid sizes of 16 and 24 provide smaller cells, allowing the model to correctly identify more of the agricultural regions. The UNet models are able to resolve the agricultural regions in the true pixel mask, but also pick up on lots of other agricultural regions not labelled in the ground truth. They also make misclassify other areas in the image, identifying water and forest regions. Note this is using unweighted model outputs.

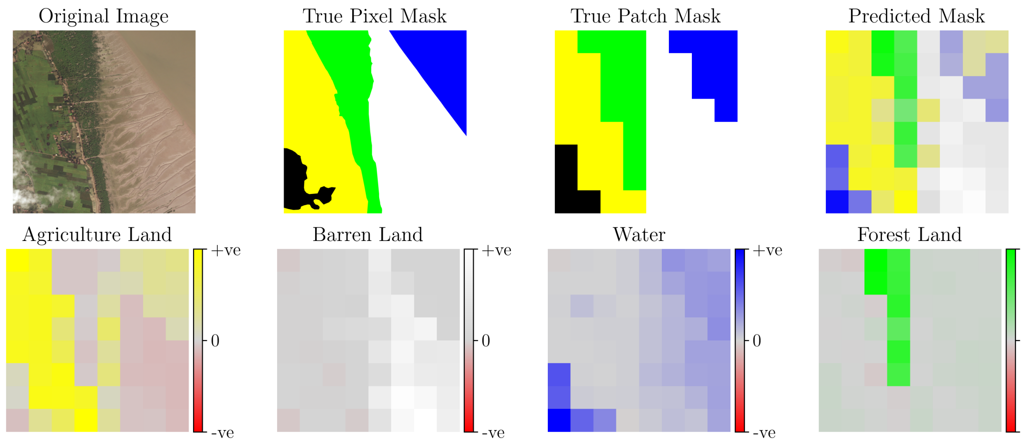

Figure A2: An output from an S2P Large 8 model. Cloud covers the bottom left of the original image, so that area has been marked as unknown (black). However, the model labels it water, i.e., it has been unable to correctly learn the unknown class as it occurs very infrequently in this dataset.

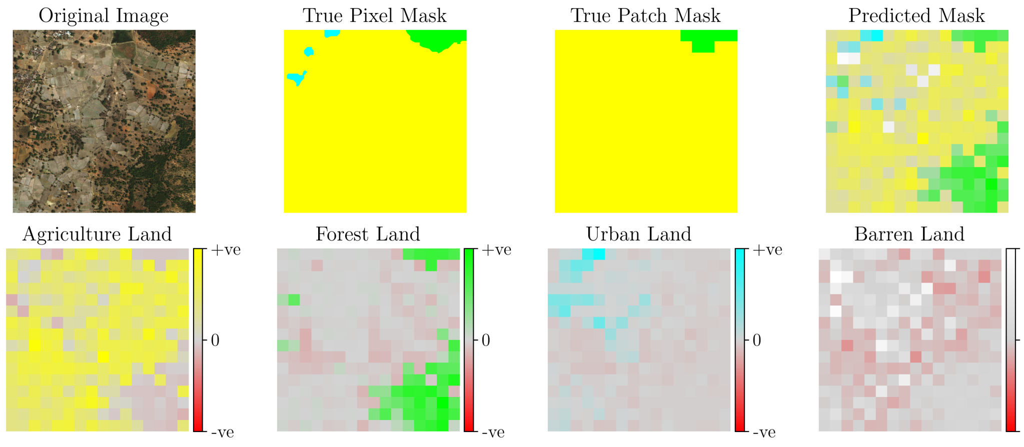

Figure A3: An output from an S2P Medium 16 model. This gives an example of potential mislabelling in the ground truth, where the trees in the lower right have not been labelled, but the ones in the upper right have. However, this could be intentional as the two regions have different tree densities. The model predicts both regions as forest, but will be penalised for doing so. Also note that the urban areas disappear in the true patch mask as they are not the majority class in their respective patches. However, the model still predicts those regions as urban, matching the pixel mask but not the patch mask.

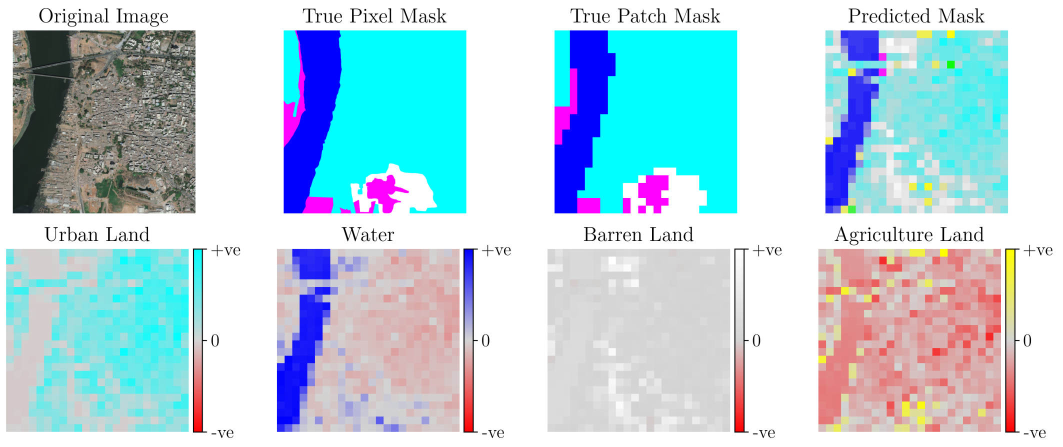

Figure A4: An output from an S2P Medium 24 model. Due to the large grid size, the model is able to segment the bridge crossing the water and label it as urban (blue). However, the ground truth labelling omits this, meaning segmenting the bridge in this way is incorrect (according to the original segmentation labels).

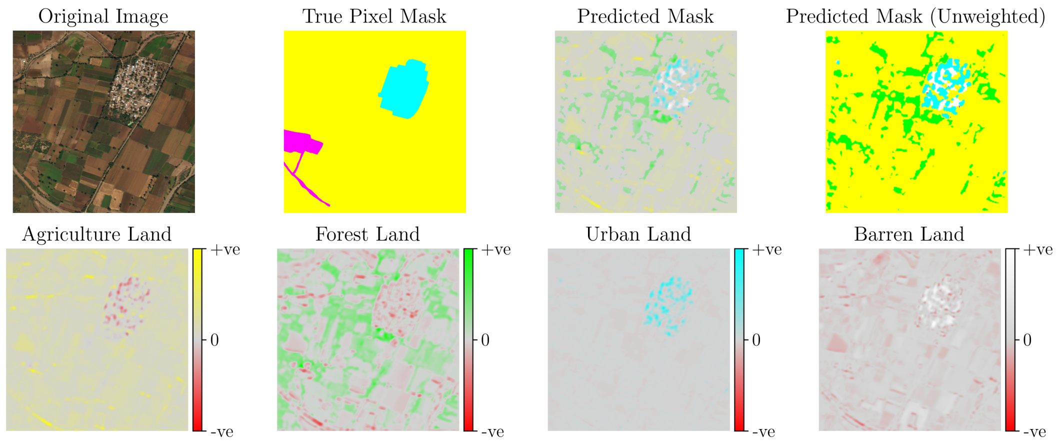

Figure A5: An output from a UNet 224 model. The model is able to separate features such as individual buildings in the urban area, and small clusters of trees (that weren't labelled in the true pixel mask). However, it misclassifies the rangeland regions in the bottom left of the image. As the model often has relatively low weighted predictions for some areas, we also provide the unweighted prediction mask.

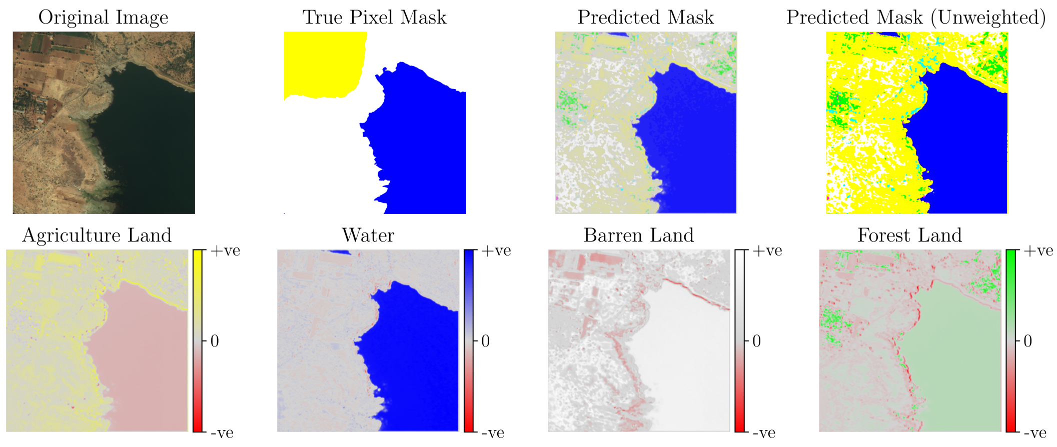

Figure A6: An output from a UNet 448 model. The model has been able to segment the area of water with high fidelity, but produces different segmented regions to the true pixel mask for the land. Note on the bottom row, the model also predicts the lower right region as barren and forest, but not as strongly as it (correctly) predicts it as water. This highlights potential model confusion, with smooth blue/green/grey areas leading to misclassification. However, for agricultural land, the model correctly labels the lower right as negative (red), i.e., refuting the prediction of agricultural land for the area of water.