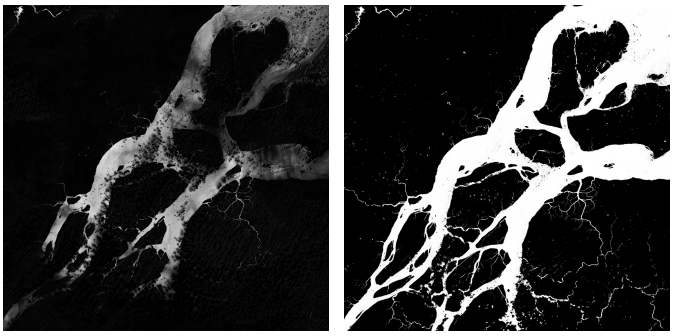
Seeing Through the Clouds With DeepWaterMap

Leo F. Isikdogan, Alan Bovik, and Paola Passalacqua

***Abstract*— We present our next-generation surface water mapping model, DeepWaterMapV2, which uses improved model architecture, data set, and a training setup to create surface water maps at lower cost, with higher precision and recall. We designed DeepWaterMapV2 to be memory efficient for large inputs. Unlike earlier models, our new model is able to process a full Landsat scene in one-shot and without dividing the input into tiles. DeepWaterMapV2 is robust against a variety of natural and artificial perturbations in the input, such as noise, different sensor characteristics, and small clouds. Our model can even "see" through the clouds without relying on any active sensor data, in cases where the clouds do not fully obstruct the scene. Although we trained the model on Landsat-8 images only, it also supports data from a variety of other Earth observing satellites, including Landsat-5, Landsat-7, and Sentinel-2, without any further training or calibration. Our code and trained model are available at https://github.com/isikdogan/deepwatermap.**

***Index Terms*— Computer vision, image processing, machine learning, oceans, remote sensing, water resources.**



(a) (b)

Fig. 1. Comparison of surface water detection results in a scene that is partially occluded by clouds. Higher pixel intensity denotes the higher probability of water. Although (a) former model (DeepWaterMap) is correct about cloud pixels not being water, (b) cloud-tolerant model (DeepWaterMapV2), which classifies cloud pixels as either water or land depending on the context, is more practical in many cases.

I. Introduction

ACCURATE surface water maps are needed for many aspects of Earth science, such as monitoring changes in wetlands and rivers in response to climate change and anthropogenic modifications. There has long been a deep interest in mapping surface water [1] using satellite imagery. However, the ability to generate surface water maps quickly, accurately, and fully automatically over any specified period of time, without requiring a plethora of data sets, remains a highly desirable but unreached goal. In particular, creating detailed records of changes in surface water over time and space require fast and robust models that can operate on easy-to-obtain, uncalibrated, and possibly incomplete data.

In response to those needs, earlier, we created a fully automatic, deep learning-based surface water segmentation model, named DeepWaterMap [2]. Our model was able to learn both spatial and spectral characteristics of water bodies on Landsat-7 imagery using data having relatively noisy labels. The training data set for DeepWaterMap was created by matching the Landsat-7 images in the GLS2000 collection [3] with surface water maps in the GLCF inland water data set [4]. Although the model learned to generalize well, its accuracy was limited by the accuracy of the labels in the training set. Furthermore, the model expected only Landsat-7 ETM+ images as input and needed to be fine-tuned to work on data from other sensors.

In this letter, we propose DeepWaterMapV2, our next-generation surface water mapping model. Our new model significantly improves upon its predecessor, DeepWaterMap, in terms of efficiency, robustness, and accuracy.

We carefully redesigned the network architecture of DeepWaterMap to further reduce the memory footprint needed during inference while improving the accuracy. The macroarchitecture of our model (see Fig. 2) resembles well-known image segmentation models, such as fully convolutional networks [5] and U-Net [6] architectures. However, it has several key differences that adapt our model to the targeted application, including a greatly reduced memory footprint for very large input images. Earlier, to compute water maps on a full-sized Landsat scene, we needed to divide it into tiles, run inference on each tile separately, and stitch the resulting maps. This process resulted in some stitching artifacts when nonoverlapping tiles were used. Although using overlapping tiles reduced those artifacts, it increased the total number of pixels that needed to be processed, which slowed down the process. DeepWaterMapV2 is designed to use constant memory throughout the network; therefore, increasing the input size does not disproportionally increase the memory consumption. As a result, our new model can run inference on a full Landsat image ($\sim 7500 \times 7500 \times 6$ pixels) in one-shot on a typical workstation having over $\sim$10 GB of available memory, in less than a minute.

To train DeepWaterMapV2, we created a new global-scale data set that contains more challenging samples having higher

Manuscript received July 30, 2019; revised September 1, 2019; accepted October 12, 2019. This work was supported in part by the Microsoft AI for Earth Program and in part by the National Science Foundation under Grant CAREER/EAR-1350336, Grant EAR-1719670, and Grant SEES/OCE-1600222. *(Corresponding author: Leo F. Isikdogan.)*

L. F. Isikdogan and A. Bovik are with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78112 USA (e-mail: isikdogan@utexas.edu; bovik@ece.utexas.edu).

P. Passalacqua is with the Department of Civil, Architectural and Environmental Engineering, The University of Texas at Austin, Austin, TX 78112 USA (e-mail:paola@austin.utexas.edu).

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LGRS.2019.2953261

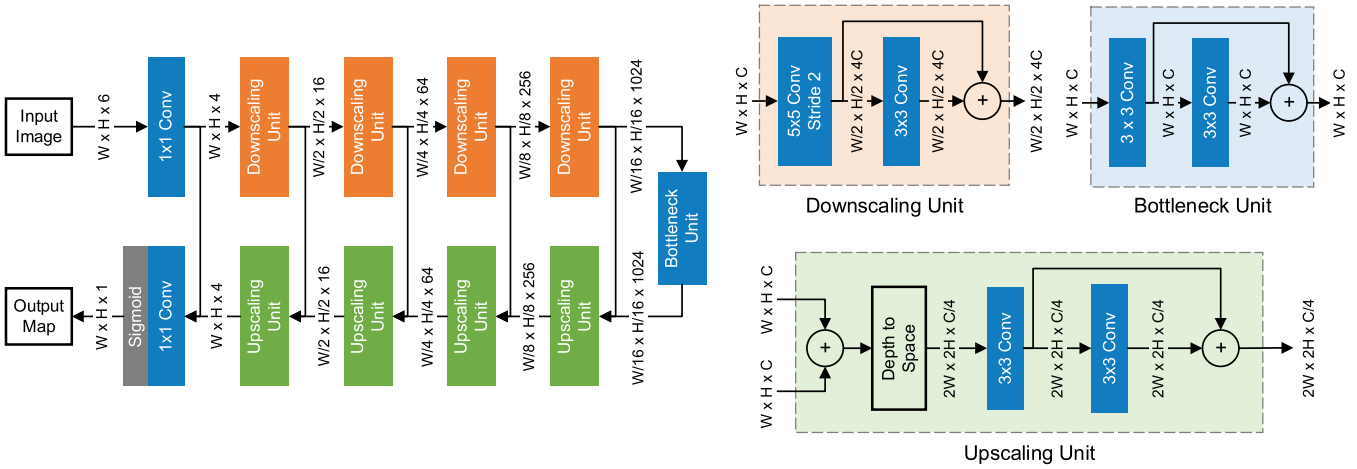


Fig. 2. Network topology of DeepWaterMapV2 consists of three primary building blocks: downscaling units, a bottleneck unit, and upscaling units. The downscaling unit uses striding to spatially downscale the input. The bottleneck unit outputs a feature map that has the same dimensions as its input. The upscaling unit uses a depth-to-space transformation followed by convolutions to upsample the input. All three types of units preserve the size of the feature maps throughout the network.

quality labels compared to the data used in the previous version. This new data set and our improved training setup helped DeepWaterMapV2 better utilize its capacity and achieve higher accuracy at a lower cost. Using massive amounts of high quality, well-balanced data in our data set helped the model learn more granular characteristics of water bodies in context across the globe. These shape, texture, and spectral characteristics helped distinguish water from nonwater pixels, without the need for any ancillary data, human intervention, or calibration.

We trained DeepWaterMapV2 to be robust against changes in the input using a set of data preprocessing and augmentation methods. The model tolerates both spectral and spatial perturbances caused by different sensors and atmospheric conditions. For example, it classifies a cloud pixel either as water or nonwater depending on the context. A cloud surrounded by water would be likely classified as water (see Fig. 1).

Once the model is trained, it runs out-of-the-box on input images having different characteristics. For example, DeepWaterMapV2 accepts both raw and radiometrically calibrated images as inputs and can segment water on images collected from different sensors, having overlapping but not identical spectral bands. Although the model is trained using Landsat-8 images only, it also supports images acquired by Landsat-5, Landsat-7, Sentinel-2, and possibly other Earth-observing missions. This capability allows for using the same model for a variety of images without collecting separate training sets for each use case and further training the model for calibration.

Like its predecessor, DeepWaterMapV2 is able to automatically produce high-quality maps of surface water, independent of the geographic region being imaged and the atmospheric conditions. The model is straightforward to implement and is fast in application. As we show, the trained model delivers remarkable water mapping results. Our code, trained model, and scripts to obtain the training data are publicly available on our project repository.

II. Model

General-purpose pixelwise prediction models work well for a wide variety of applications, such as semantic [5], [7] and biomedical [6] image segmentation. However, remotely sensed images are typically much larger than what those models use. For example, a Landsat scene can be orders of magnitude larger (over 300 million pixels in one image) than the images in semantic segmentation data sets.

In DeepWaterMapV2, we made several design choices to create a model that can process very large input images. As the input sizes grow, the impact of the size of the intermediate feature maps on memory becomes much larger compared to the number of parameters. Therefore, we decreased the number of output channels in the first layers and moved most of the trainable parameters to the layers that process lower resolution feature maps. It is a common practice to increase the number of channels in the feature maps in many convolutional neural network architectures. However, early layer feature maps still make up a large portion of the memory footprint. For example, the first layer feature maps in U-Net [6] are twice as large as the activations of the layer after first downscaling.

We designed DeepWaterMapV2 to have a constant feature map size throughout the network. For example, the first layer in our model has a feature map of $W \times H \times 4$, whereas the bottleneck layer has a feature map of $W/16 \times H/16 \times 1024$, both having the same pixel count. We also replaced all channelwise feature concatenations with residual sums to avoid increasing the dimensionality of the intermediate feature maps.

Our new network topology consists of three primary building blocks (see Fig. 2): downscaling units, a bottleneck unit, and upscaling units. The downscaling unit uses a 5×5 convolutional layer with a stride of two followed by a 3×3 residual convolutional layer. The first layer in each downscaling unit outputs four times as many channels as the input channels to preserve the total feature map size. Using striding instead of max pooling to spatially downscale the input further saves memory. The bottleneck unit consists

Fig. 3. Our data set includes tiles of 0.2° height and width that has at least 1% water presence by pixel area. We excluded those tiles where more than 99% of the pixels had the same class label to create a more balanced data set. The tiles that are included in our data set are shown in blue regions.

of two residually connected 3×3 convolutional layers that do not change the spatial resolution. The upscaling unit uses a depth-to-space transformation to rearrange its input pixels from $W \times H \times C$ to $2W \times 2H \times C/4$. This transformation is followed by two 3×3 residual convolutional layers that act as subpixel convolutions. This type of upscaling has been shown to be effective for super-resolution applications [8].

All convolutional layers are followed by batch normalization [9] and use ReLU activation except the first and last layers. The input layer is a 1×1 convolutional layer that acts as a linear channelwise compression layer. Not using a ReLU at the end of this layer prevents too much information loss due to the reduced number of channels early in the network. The output layer is also a 1×1 convolutional layer but has a sigmoid activation that outputs values that correspond to the probabilities of each pixel in the input being a water pixel.

DeepWaterMapV2 has a much larger capacity than its previous version despite being computationally cheaper and more memory efficient to compute. DeepWaterMapV2 has over 37M trainable parameters, whereas the largest variant of DeepWaterMap had 1.5M. Despite having over $24\times$ parameters, DeepWaterMapV2 uses 60% less memory than its predecessor for an input size of 512×512 pixels. The memory savings go up to 83% for full-sized Landsat images and approach 84% as the input size grows further.

III. Data Preparation and Training

We curated a data set that consists of multispectral image tiles and their corresponding pixelwise labels, covering the entire globe, using the Google Earth Engine [10] platform. We used Landsat-8 composites as inputs and the water labels in the Global Surface Water data set [1] as ground-truth outputs. We generated the composites by merely computing the pixelwise median of the images in the collection over the same time period as the ground-truth labels. We deliberately did not mask the clouds before computing the median images so that a model trained using the data set can learn to handle the clouds. Therefore, the composites in our data set are not completely cloud-free although computing the temporal median filtered out some of the sporadic clouds.

To maximize cross-sensor compatibility, we excluded those bands not commonly provided by satellites other than Landsat-8. Namely, we used the Landsat-8 bands 2, 3, 4, 5, 6, and 7 as inputs that have their equivalents in images acquired by Sentinel-2 and earlier Landsat missions.

A large portion of images in the Landsat collection consists of samples that are "not interesting" from a machine learning perspective. Many samples consist of nearly completely land or water. Labels in those samples are very easy to learn for a machine learning model, even using only linear or almost-linear transformations. Models that are trained on such samples are likely to learn a shortcut for easy cases but fail in more challenging cases. To balance the distribution of easy and hard examples in the data set, we excluded the tiles where less than 1% of the pixels were labeled as water or land.

The resulting data set contained over 140 000 labeled tiles of 0.2° height and width (see Fig. 3). We randomly selected 2500 of those tiles for validation, 2500 for test, and used the rest for training. We also used a separate test set that consists of Landsat-7 images to evaluate the cross-data set generalization of our model and to rule out the possibility of any information leakage from the validation set.

To improve the cross-data set performance of our model, we perturbed and normalized the images in our data set on the fly during training. First, we randomly cropped a 512×512 patch from each sample in a given batch. Then, to emulate spectral responses of sensors on different satellite missions, we "leaked" information between consecutive bands in random amounts by multiplying the bands of the patches with smoothed identity matrices as $X_{512\times512\times6} \cdot g_{1\times3}(I_6)$, where X is the input and g is a Gaussian filter. We also distorted those patches with additive Gaussian noise in random magnitude. Finally, we applied min–max scaling to the inputs as $X' = (X - \min(X))/\max(\max(X) - \min(X), 1)$. Defining a lower bound for the denominator as 1 stabilized the normalization for uniform inputs, such as all-water scenes. Those preprocessing steps helped make our model robust against differences between sensors, atmospheric conditions, and calibration methods.

Our training set has labels only for water and nonwater pixels. Therefore, certain types of pixels, such as snow, ice,

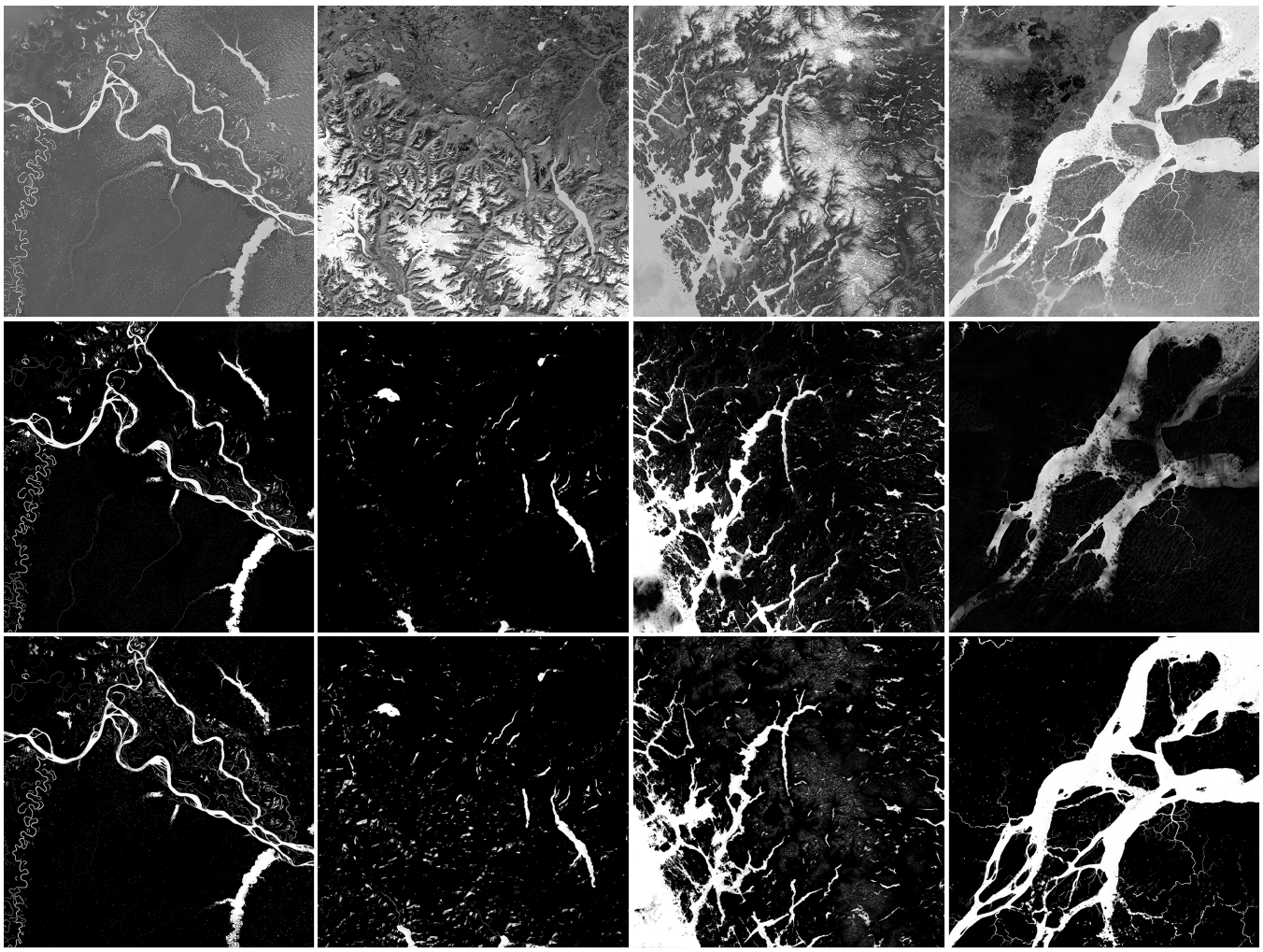


Fig. 4. Comparison of surface water probability maps generated by (Bottom Row) DeepWaterMapV2 against (Top Row) MNDWI and (Center Row) DeepWaterMap on Landsat-7 scenes from Global Land Survey 2000 [3]. Overall, the maps generated by both versions of DeepWaterMap accurately captured water bodies, where the newer version handled the clouds better. Images are contrast stretched for visualization.

shadow, and clouds, are harder to classify than others. Those pixels are classified either as water or nonwater depending on their context. For example, cloud pixels should be classified as water when they are surrounded by water pixels and as nonwater otherwise. It is hard for a model to learn such complex cases using a loss function that treats those harder-to-learn pixels the same as the easier ones. We used a combination of a max-pooled adaptive loss [11] and focal loss [12] functions to assign a higher cost to harder-to-learn pixels. The max-pooled loss spatially shifts the attention toward those pixels having the highest loss, whereas the focal loss acts as a pointwise weighting scheme that places the focus on harder examples.

We optimized the parameters of our model using stochastic gradient descent (SGD) with momentum algorithm. Although adaptive gradient methods, such as Adam, usually converge faster, we were able to achieve better results with SGD, confirming the results in [13].

IV. Results

We validated DeepWaterMapV2 both quantitatively and qualitatively on two different test sets. The first set consisted of randomly selected Landsat-8 tiles and their corresponding labels from the Global Surface Water data set. We used this test set to quantitatively evaluate DeepWaterMapV2 using precision, recall, and F1-score as performance metrics. We compared DeepWaterMapV2 against its predecessor DeepWaterMap and two other benchmark methods: a thresholded modified normalized difference water index (MNDWI) classifier [14] and a traditional multilayer perceptron [2]. DeepWaterMapV2 outperformed the other methods in all three metrics, except that the simple MNDWI classifier had a higher recall rate. However, MNDWI thresholding yielded many false

TABLE I

Comparison of DeepWaterMapV2 to Earlier Surface Water Classification Methods

Method	Precision	Recall	F1-score
MNDWI [14]	0.57	0.98	0.72
MLP (retrained) [2]	0.84	0.68	0.75
DeepWaterMap [2]	0.91	0.88	0.90
DeepWaterMapV2	**0.97**	0.90	**0.93**

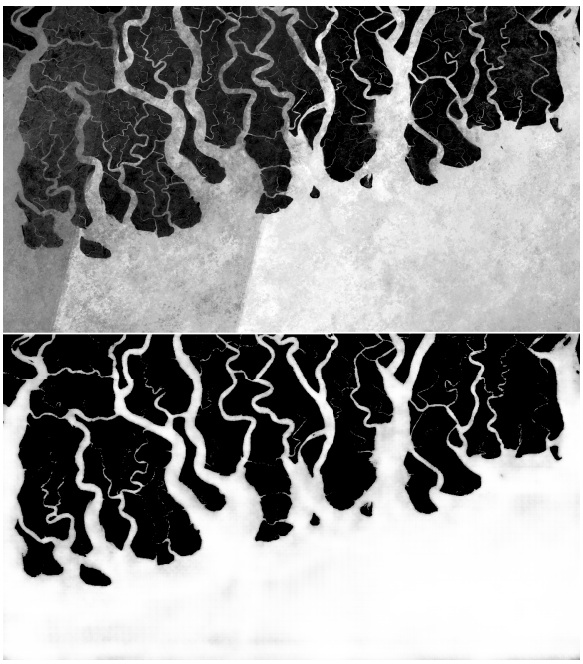


Fig. 5. DeepWaterMapV2 on Sentinel-2 images. The input mosaic is created by stitching Sentinel-2 images near the Ganges river delta. The figure compares (Bottom) DeepWaterMapV2 water probability map to (Top) MNDWI response.

positives whereas DeepWaterMapV2 delivered significantly better overall performance in F1-score (see Table I).

The second set consisted of Landsat-7 images without ground-truth labels. We used this image set to compare the results visually over regions that were previously used to validate DeepWaterMap [2]. To evaluate the cross-data set generalization of our model, we used the trained DeepWaterMapV2 model as is without any further training or fine-tuning. The input images in the test set were top-of-atmosphere calibrated Landsat-7 images, whereas the DeepWaterMapV2 was trained using raw Landsat-8 images. Despite being trained on different data sets, both versions of DeepWaterMap produced visually similar results (see Fig. 4). Overall, DeepWaterMapV2 handled the clouds better by making an educated guess about what might be behind the clouds, given their surroundings. Although we have not systematically evaluated our model on other sensors, our preliminary results on Sentinel-2 images (see Fig. 5) showed that the cross-sensor generalization capabilities of DeepWaterMapV2 extend beyond Landsat sensors.

V. Conclusion

We described a memory-efficient convolutional neural network topology that is tailored for segmenting large, remotely sensed images. The model has a larger capacity and smaller memory footprint than its predecessor.

We derived a global-scale surface water data set from publicly available data sets to train an accurate water segmentation model. We presented a data preprocessing and training scheme that allowed for training a model that can process inputs from different sensors as is.

Using the presented model, data set, and the training setup, we built the DeepWaterMapV2, our next-generation surface water mapping model. Our new model is robust against clouds, noise, and changes in the input sensor characteristics. Although DeepWaterMapV2 cannot actually see through the clouds since that information is not available in the input, it filters out small clouds as a form of inpainting.

Our model opens up new possibilities for a variety of hydrography applications. It is technically possible to create a high-resolution surface water map of the entire Earth every eight days, as soon as new Earth imaging data from the Landsat 8 mission become available. Those surface maps created by our model can be used as inputs to the task-specific models for further analysis. For example, DeepWaterMapV2 can be used as a drop-in replacement for a water index in RivaMap [15] to automatically derive river networks from the given surface water maps.

References

[1] J.-F. Pekel, A. Cottam, N. Gorelick, and A. S. Belward, "High-resolution mapping of global surface water and its long-term changes," *Nature*, vol. 540, pp. 418–422, Dec. 2016.

[2] F. Isikdogan, A. C. Bovik, and P. Passalacqua, "Surface water mapping by deep learning," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 11, pp. 4909–4918, Nov. 2017.

[3] G. Gutman *et al.*, "Towards monitoring land-cover and land-use changes at global scale: The global land use survey," *Photogramm. Eng. Remote Sens.*, vol. 74, no. 1, pp. 6–10, 2008.

[4] M. Feng, J. O. Sexton, S. Channan, and J. R. Townshend, "A global, high-resolution (30-m) inland water body dataset for 2000: First results of a topographic–spectral classification algorithm," *Int. J. Digit. Earth*, vol. 9, no. 2, pp. 113–133, 2016.

[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3431–3440.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Cham, Switzerland: Springer, 2015, pp. 234–241.

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.

[8] W. Shi *et al.*, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1874–1883.

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," Feb. 2015, *arXiv:1502.03167*. [Online]. Available: https://arxiv.org/abs/1502.03167

[10] (2016). *Google Earth Engine: A Planetary-Scale GEO-Spatial Analysis Platform*. [Online]. Available: https://earthengine.google.com

[11] F. Isikdogan, A. Bovik, and P. Passalacqua, "Learning a river network extractor using an adaptive loss function," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 6, pp. 813–817, Jun. 2018.

[12] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2999–3007.

[13] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4148–4158.

[14] H. Xu, "Modification of normalised difference water index (NDWI) to enhance open water features in remotely sensed imagery," *Int. J. Remote Sens.*, vol. 27, no. 14, pp. 3025–3033, 2006.

[15] F. Isikdogan, A. Bovik, and P. Passalacqua, "RivaMap: An automated river analysis and mapping engine," *Remote Sens. Environ.*, vol. 202, pp. 88–97, Dec. 2017.