

# Generating Hard Examples for Pixel-Wise Classification

Hyungtae Lee , *Member, IEEE*, Heesung Kwon, *Senior Member, IEEE*, and Wonkook Kim 

**Abstract**—Pixel-wise classification in remote sensing identifies entities in large-scale satellite-based images at the pixel level. Few fully annotated large-scale datasets for pixel-wise classification exist due to the challenges of annotating individual pixels. Training data scarcity inevitably ensues from the annotation challenge, leading to overfitting classifiers and degraded classification performance. The lack of annotated pixels also necessarily results in few hard examples of various entities critical for generating a robust classification hyperplane. To overcome the problem of the data scarcity and lack of hard examples in training, we introduce a two-step hard example generation (HEG) approach that first generates hard example candidates and then mines actual hard examples. In the first step, a generator that creates hard example candidates is learned via the adversarial learning framework by fooling a discriminator and a pixel-wise classification model at the same time. In the second step, mining is performed to build a fixed number of hard examples from a large pool of real and artificially generated examples. To evaluate the effectiveness of the proposed HEG approach, we design a nine-layer fully convolutional network suitable for pixel-wise classification. Experiments show that using generated hard examples from the proposed HEG approach improves the pixel-wise classification model’s accuracy on red tide detection and hyperspectral image classification tasks.

**Index Terms**—Adversarial learning, hyperspectral image (HSI) classification, online hard example mining (OHEM), pixel-wise classification, red tide detection.

## I. INTRODUCTION

**P**IXEL-WISE classification is the task of identifying entities at the pixel level in remotely sensed images, such as Earth-observing satellite-based images from multi- or hyperspectral imaging (HSI) sensors. The pixel-wise classification has some parallels to image segmentation. Still, there are several limitations to directly using the state-of-the-art image segmentation methods for the pixel-wise classification. Image segmentation methods treat an image as a composition of multiple instances of a scene or object and delineate boundaries between different instances. Current state-of-the-art image segmentation methods

Manuscript received July 8, 2021; revised August 25, 2021; accepted September 11, 2021. Date of publication September 16, 2021; date of current version October 1, 2021. This research was supported by the “Development of the integrated data processing system for GOCI-II” funded by the Ministry of Ocean and Fisheries, Korea. (*Corresponding author: Hyungtae Lee.*)

Hyungtae Lee and Heesung Kwon are with the Intelligent perception branch, the Computational and Information Sciences Division, Army Research Laboratory, Adalphi, MD 20783 USA (e-mail: hyungtae.lee.civ@army.mil; heesung.kwon.civ@army.mil).

Wonkook Kim is with the Department of Civil and Environmental Engineering, Pusan National University, Busan 46241, South Korea (e-mail: wonkook@pusan.ac.kr).

Digital Object Identifier 10.1109/JSTARS.2021.3112924

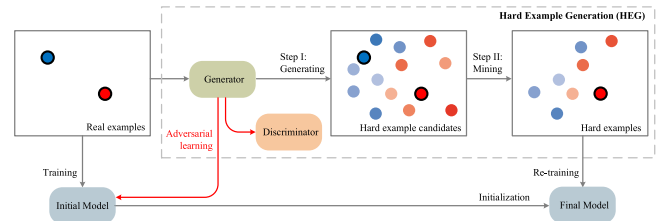


Fig. 1. **HEG approach.** While the initial model for pixel-wise classification is trained using real examples, the final model is initialized from this initial model and retrained with hard examples that are the output of the HEG approach. The HEG approach takes two steps: i) generation of hard example candidates and ii) hard example mining. In the first step, the generator is trained via the adversarial learning framework to fool the initial pixel-wise classification and the discriminator that distinguishes real examples from generated examples. For the red tide detection task, we created hard negatives by applying the generator only to the negative examples, as we aim to solve the problem of the lack of hard negative examples. On the other hand, for the HSI classification task, hard examples are generated for all categories as shown in this figure.

adopt the ability to segment these instances either by using a joint detection and segmentation model [1] or by finetuning a detection model [2]. However, these detection abilities are only useful if the target object or scene provides category-specific contextual or structural information and if each instance covers a relatively large area of the image. Unfortunately, these requirements are not typically met in remotely sensed images; thus, spectral characteristics embedded in each pixel are used as viable pixel-wise classification information.

One of the issues with pixel-wise classification for remote sensing images is the lack of fully annotated large-scale remote sensing datasets. Since it is exceptionally challenging to annotate each pixel of the remote sensing image, frequently, many pixels in the image remain unlabeled, leading to performance degradation. Further performance decrease is caused by sparse training data, including few hard examples necessary to generate a robust classification hyperplane.

To address the lack of hard examples, we introduce a hard example generation approach (HEG) suitable for pixel-wise classification (see Fig. 1). The proposed HEG approach takes two steps: 1) generating hard example candidates that were recognized as false positives for other categories while preserving the properties of the original category (generation step) and 2) processing hard example mining to discover hard examples incorrectly detected with high loss (mining step).

In the first step, we use a variant of the generative adversarial learning (GAN) [3] to train a generator that creates hard example candidates. To prevent the generated examples from losing the

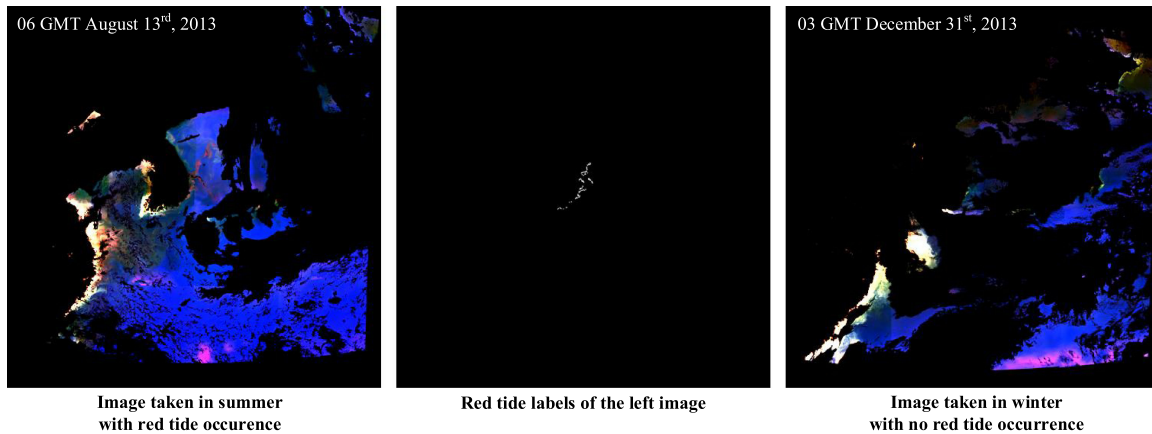


Fig. 2. **GOCI satellite images and red tide labels.** Summer (e.g., an image taken on August) and winter (e.g., an image taken on December) images of the same area are placed on the left and right of the figure, respectively. In the middle figure, the red tide labels of the summer image are marked as white pixels. In the summer image, not all red tide regions are labeled, and not all the regions in red represent red tide. All the red regions that appear in winter images are not red tide regions.

specific property of its corresponding category, we trained a network to distinguish the real examples from the artificially generated examples, which serves as *discriminator* in the GAN framework. In order for the generated example to be a hard example for another category while preserving the original category's properties, a pixel-wise classification model and the discriminator become the counterparts that the generator should deceive.

For the second step, we redesigned the online hard example mining (OHEM) [4] to select hard examples. Shrivastava *et al.* introduced OHEM, which, for every iteration, sampled a small number of examples with a high loss from the overwhelming examples and use them in training CNN. We used a simple variant of OHEM to mine a huge volume of artificially generated examples in a cascaded way. First, we randomly choose a subset of examples that can be either real or artificially generated examples and then apply OHEM to the selected examples to choose hard examples.

To evaluate the proposed HEG approach, we implemented a nine-layer fully convolutional network (FCN) inspired by [5]. The FCN architecture has proved to be suitable for pixel-wise classification [5]–[8]. We validate our approach to red tide detection using the large-scale remote sensing image dataset obtained from multispectral geostationary ocean color imager (GOCI) [9] on a geostationary satellite. We chose this practical task because it clearly presents the ground-truthing problems mentioned earlier.<sup>1</sup> Due to such challenging ground-truthing problems inherent in remote sensing, red tide occurrences are labeled only at a limited number of locations. Moreover, there are no labels about where no red tide was found that could be used as negative examples in training. Therefore, we end up with only a small number of spectral examples from a fraction of areas, where red tide occur in training. In this work, we use

<sup>1</sup>Since the biological properties of red tide are not clearly visible in the image, we used the information on real-world red tide occurrences reported by NIFS (National Institute of Fisheries Science) (<http://www.nifs.go.kr/red/main.red/>) of South Korea. NIFS manually examined red tide occurrence only at a limited number of locations along the southern seashore of South Korea.

the images taken in December as negative examples, where red tides do not occur due to the low water temperature.<sup>2</sup> Fig. 2 shows the GOCI images used for the positive (red tide) and negative (nonred tide) training examples, and the red tide region annotation of the positive image.

From this peculiar GOCI image setting, we found severe issues highlighting the need for the proposed HEG approach. First, the spectral characteristics of the images taken in December are very different from those of the images taken in the summer when red tides mostly occur because the marine environment in summer and winter is very different. Therefore, the negative examples from the December images do not generally represent the nonred tide area from the images collected in the summer. Second, the imbalance between the numbers of positive examples and the negative examples is quite significant. The number of positive examples is in the order of one hundred pixels per image. In comparison, the number of negative examples is about 31 M pixels per image as all the pixels of the GOCI image ( $5567 \times 5685$ ) taken in winter are used as negative examples. Lack of nonred tide examples derived from the property discrepancy between the summer and winter images associated with the first problem is addressed by the first step of HEG (i.e., generation of hard example candidates). The second problem of the data imbalance between the positive and negative examples is effectively alleviated by the second step of HEG (i.e., hard example mining).

We conducted extensive experiments to determine how the proposed HEG addresses the problems that arise in training the pixel-wise classification model on GOCI images. For red tide detection, a one-class classification problem with significantly unbalanced distribution, we use HEG to generate hard negative examples. To show that the proposed HEG can be easily extended to other tasks with multiple categories, we also applied it to several pixel-wise HSI classification tasks. Experiments have confirmed that pixel-wise classification method trained by

<sup>2</sup>In South Korea, summer is in July and August and winter in December. Red tide occurs mainly in summer when the water temperature is high.

adopting the proposed HEG significantly enhances performance for red tide detection and several HSI classification tasks.

## II. RELATED WORKS

**Training generator via adversarial learning.** Szegedy *et al.* [10] introduced a method to generate an adversarial image by adding perturbation to be misclassified by a CNN-based recognition approach. These perturbed images become adversarial images to the recognition approach. Goodfellow *et al.* [3] introduced two models: a generator that captures the data distribution and a discriminator that estimates the probability that an example came from the training data rather than the generator. A generator and a discriminator are trained at the same time in a direction to interfere with each other. This is called an adversarial learning framework.

Radford *et al.* [11] devised an image generation approach based on CNN by adopting this adversarial learning framework. Wang *et al.* [12] used the adversarial learning framework to train a network that creates artificial occlusion and deformation on images. An object detection model is trained against this adversary to improve performance. Hughes *et al.* [13] introduced a negative generator based on an autoencoder that takes a positive image as an input. The generator is optimized to make the output acquire the properties of a real image by adopting an adversarial learning framework. This generator is used to augment the set of negative examples, which are not necessarily hard negatives. Choi *et al.* [14] used GAN-based data augmentation for reducing a domain gap from fully annotated synthetic data to unsupervised data. Xie *et al.* [15] also uses adversarial learning to augment training examples for image recognition. In the proposed work, we also use adversarial learning to train a hard example generator (HEG). Unlike [13]–[15], our HEG generates hard negative examples, which are more challenging to be identified as nonred-tide examples by our red tide detector than other real negative examples.

**Hard example mining.** Sung and Poggio [16] first introduced hard negative mining (also known as bootstrapping) that trains the initial model with randomly chosen negatives and adapts the model to hard negatives that consist of false positives of the initial model. Thereafter, hard example mining has been widely used in various applications such as pedestrian detection [17], [18], human pose estimation [19], [20], action recognition [21], [22], event recognition [23], object detection [24]–[27], etc. There are alternative ways to find hard examples using heuristic [28] or other hard example selection algorithms [29], [30], which avoid training multiple times. Kellenberger *et al.* [31] used active learning, which requires human intervention to assign labels to critical examples and address the problem of the small number of positive samples.

Shrivastava *et al.* [4] introduced OHEM, which, for every training iteration, carries out hard example mining that chooses examples with high training loss. However, it is too exhaustive to evaluate all examples on each iteration. Hence, when using an extensive example set like our problem, it is impractical to examine all examples for each iteration. Therefore, we use

OHEM in a cascaded fashion to randomly select a subset of examples and then perform efficient mining on it.

**CNN used for detecting natural phenomena in marine environment.** Since CNN has provided promising performance in image classification [32]–[34], there have been several attempts to use it in the marine environment. CNNs have been effectively used for detection of coral reefs [35], [36], classification of fish [37]–[39], detection of oil from shipwreck [40], [41], etc. However, applying deep neural network to detect objects-of-interest in the marine environment has been quite limited due mainly to difficulties in acquiring large amounts of annotated data, unlike general object detection applications. In this article, we devise a CNN training strategy coupled with an advanced network architecture tailored to red tide detection while minimizing human labeling efforts.

## III. RED TIDE DETECTION APPROACH

### A. Red Tide Detection

In this section, we describe the proposed CNN-based red tide detection approach. This approach takes the GOCI image as input and evaluates whether each pixel in the image belongs to a red tide area or not. Therefore, red tide detection can be considered as pixel-wise classification. The architecture of the proposed approach is built on a model introduced by [5], which is known to be suitable for pixel-wise classification. We apply a sliding window method to deal with limited GPU memory when processing GOCI images. *Pixel-Wise Classification.* Pixel-wise classification has been widely used for multispectral/HSI classification that assigns each pixel vector into a corresponding category by exploiting the spectral characteristics of both the pixel and the neighboring pixels in a local region. Unlike general image segmentation [1], [2], which segments distinctive scene components in an image by primarily leveraging object appearance as well as structural characteristics and attributes of the components (e.g., human anatomy, car with four wheels), pixel-wise classification is a task of predicting each pixel in a region with additional features, such as spectral profiles, and simultaneously little structural attributes available. Therefore, in the proposed work, red tide detection is treated as a pixel-wise classification problem primarily because the red tide is a microscopic alga with no discernible appearances or structures useful for image segmentation.

For recent CNN-based image segmentation, the state-of-the-art approaches have a CNN architecture designed as sequentially stacking multiple layers consisting of filters that capture neighboring information (e.g.,  $3 \times 3$ ,  $5 \times 5$  filters) to leverage information over a large area when predicting each pixel. Furthermore, it adopts multiple downsampling layers such as pooling/convolution layers with stride  $\geq 2$ , which are known to encode the structural characteristics adequately. On the other hand, our approach, described in the next section, also adopts CNN architecture. It is designed by stacking layers made up of  $1 \times 1$  filters except for the initial layers and does not use any downsampling layers. The first layer, consisting of multiscale

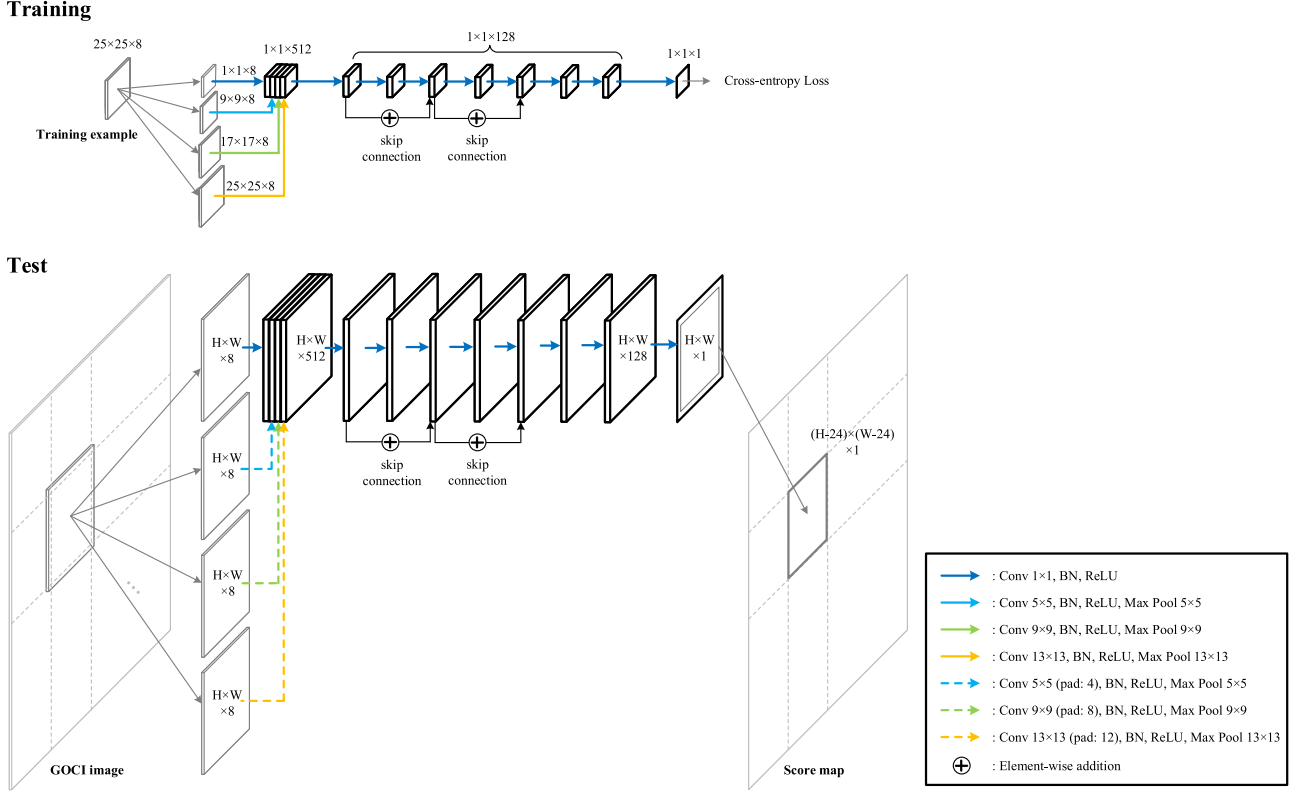


Fig. 3. FCN architecture for red tide detection. The training and test architectures have slightly different initial multiscale filter banks tailored to pixel-wise classification. The weights and dimension of the convolutional layers do not change between training and test. The size of each intermediate feature blob is shown above the blob.

filters, does not capture structural characteristics but rather imposes spatial continuity over neighboring pixels such that they have the same identity.

**Architecture.** The architecture of the proposed red tide detection model is shown in Fig. 3. To cope with the pixel-wise classification of red tide detection, we use a nine-layer FCN, which intakes an image of arbitrary size. The network takes image patches of  $25 \times 25$  as input during training, while an image patch of a certain dimension determined by the maximum size of GPU memory is fed into the network in test time.

The network’s initial module is a multiscale filter bank consisting of convolutional filters with four different sizes ( $1 \times 1$ ,  $5 \times 5$ ,  $9 \times 9$ , and  $13 \times 13$ ). The architecture of the multiscale filter bank is slightly different between training and testing. Given an image patch of  $25 \times 25$  in training, each  $k \times k$  filter is convolved with a patch of  $(2k - 1) \times (2k - 1)$  centered on the  $25 \times 25$  patch. The size of the smaller patch, i.e.,  $(2k - 1) \times (2k - 1)$ , is determined so that each convolution always includes the center of the larger  $25 \times 25$  patch. For example, when applying a  $5 \times 5$  filter to a  $9 \times 9$  patch, the  $5 \times 5$  window always contains the center pixel of the  $25 \times 25$  patch being evaluated. After the initial convolution, a max pooling is applied to the outputs of the convolutional filters so that those pooled feature maps have a size of  $1 \times 1$  except for the  $1 \times 1$  convolution. In the test, four filters are applied to the same patch of the same size. These convolutions use appropriate padding to ensure that the four pooled feature maps have the same size. Four output feature

maps are concatenated for both training and testing and then fed to the second convolutional layer. Accordingly, due to the multiscale filter bank architecture, our network becomes  $25 \times 25$ , and the network’s receptive field uses spatial information based on this receptive field when evaluating each pixel.

In training, dropout layers, which are commonly used to solve the overfitting issue to some extent, are added at the end of the 7th and 8th layers. The rest of the network is the same in training and testing. Specifically, the binary sigmoid classifier that is useful for either single-label or multilabel classification is used for the output layer to identify other natural phenomena (e.g., sea fog, yellow dust, etc.) from the GOCI images later using the same architecture.

**Sliding window strategy.** The proposed method cannot process a huge GOCI image at once during inference because of GPU memory limitations, where the size of the image is  $5567 \times 5685$ . To overcome this issue, we use a sliding-window-based strategy, where each window size is  $H \times W$ . In our experiment, we set  $H$  and  $W$  to 600. Considering our network’s receptive field, we use only the scores corresponding to the central  $(H - 24) \times (W - 24)$  region as the final output.

### B. Training: Adopting HEG

To meet the need for hard examples in devising an accurate hyperplane with a small number of examples that can be adequately applied to test examples, we introduce HEG approach. It takes

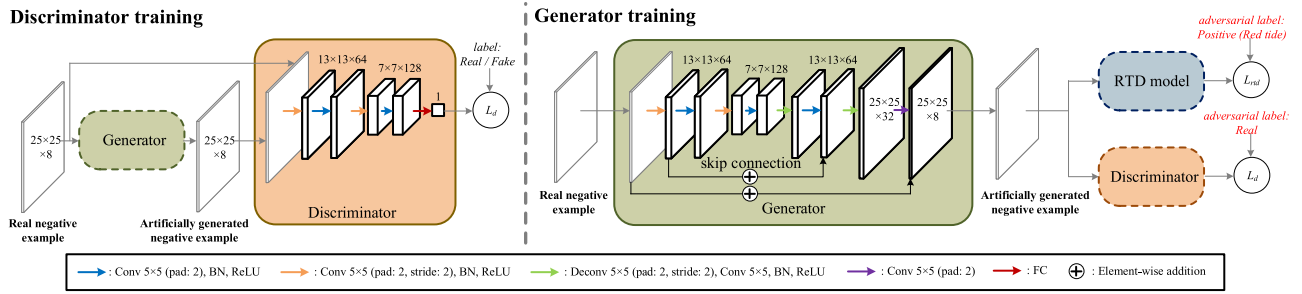


Fig. 4. *Training hard example candidates generator.* For each iteration in training, the discriminator and the generator are trained in order. Components not updated during training is indicated with a dashed box.

two steps: 1) generation of hard example candidates (*generation* step) and 2) hard example mining (*mining* step). This section provides details for each step and our training strategy to jointly train the red tide detection model with HEG. Note that the details given are primarily focused on the red tide detection task but can easily be extended to other pixel-based classifications.

**Generation step.** This step develops a generator that creates artificial examples that the red tide detection model likely classifies as false positives. The proposed generator creates hard negative example candidates only for single-category red tide detection. However, it can still be applied to other multicategory pixel-based classification tasks for generating hard example candidates for multicategory positives and negatives. This extension will be presented with HSI classification in Section V. The generator is designed as a 10-layered conv–deconv network consisting of eight convolutional layers and two deconvolutional layers inspired by U-Net with high image generation capability [42]–[46], as shown in Fig. 4.

We aim to achieve two goals in the training of the generator. First, the generator must be able to fool the red tide detection model so that the generated examples are incorrectly classified as red tides. Second, generated examples should have typical nonred-tide spectral characteristics. To achieve the goals, we introduce a discriminator that distinguishes real examples from artificially generated ones. The generator is trained to deceive the discriminator as in the typical GAN framework [3]. The discriminator consists of four convolutional layers and one fully connected layer, as shown in Fig. 4. Generated examples become hard example candidates that are designed to maximize the losses of the red tide detection model and the discriminator, which conflicts with the two models' objectives. The training process of the generator and the discriminator is shown in Fig. 4.

To mathematically formulate the process of generating hard negative examples, the red tide detection model and its loss are represented by  $\mathcal{F}_{\text{rtid}}$  and  $\mathcal{L}_{\text{rtid}}$ , respectively. The red tide detection model is trained by minimizing its loss expressed as

$$\mathcal{L}_{\text{rtid}}(E, L_{\text{rtid}}) = \mathcal{H}(\mathcal{F}_{\text{rtid}}(E), L_{\text{rtid}}) \quad (1)$$

where  $E$  and  $L_{\text{rtid}}$  are training examples and their associated labels, respectively. For each example  $e \in E$ , its red tide labels  $l_{\text{rtid}} \in L_{\text{rtid}}$  can be either 1 (red tide) or 0 (nonred tide).  $\mathcal{H}(p, q)$  is the cross-entropy for the distributions  $p$  and  $q$ .

The discriminator and its loss are denoted as  $\mathcal{F}_d$  and  $\mathcal{L}_d$ , respectively, in (2). The discriminator is optimized by minimizing

its loss, which is expressed as

$$\mathcal{L}_d(N) = \mathcal{H}(\mathcal{F}_d(N), \mathbf{1}) + \mathcal{H}(\mathcal{F}_d(\mathcal{F}_g(N)), \mathbf{0}) \quad (2)$$

where  $N$  is a set of real negative examples. The discriminator's labels can be either 1 (real example) or 0 (artificially generated example).  $\mathcal{F}_g$  denotes the generator. The randomly selected 256 negative examples and the generated examples are used for every iteration when training the discriminator. The generated examples are the output of the generator taking the randomly selected real negative examples as input.

The generator's objective is to generate negative examples incorrectly classified as red tide by the red tide detection model and as real negative examples by the discriminator. Accordingly, the generator loss ( $\mathcal{L}_g$ ) can be expressed as

$$\mathcal{L}_g(N) = \mathcal{H}(\mathcal{F}_{\text{rtid}}(\mathcal{F}_g(N)), \mathbf{1}) + \mathcal{H}(\mathcal{F}_d(\mathcal{F}_g(N)), \mathbf{1}) \quad (3)$$

where  $\mathbf{1}$  indicates that labels associated with the generated negative examples are red tide for  $\mathcal{F}_{\text{rtid}}$  or real negatives for  $\mathcal{F}_d$ . The generator can be trained by minimizing  $\mathcal{L}_g(N)$ , where  $\mathcal{F}_g(N)$  becomes adversarial examples for the red tide detection model and the discriminator.

**Mining step.** An additional mining step is necessary because the generated examples may not necessarily be hard examples. We use OHEM introduced by Shrivastava *et al.* [4] to build batches by collecting hard examples to minimize objective loss throughout the entire examples collectively. However, in problems like ours with many examples, it takes quite long time to find hard examples. In the red tide detection problem, the number of original negative examples in the GOCI images is huge, and even larger-scale generated examples are also considered in training. Therefore, we used OHEM in a cascaded fashion (cOHEM) to first build a pool of randomly chosen negative examples and then carry out OHEM with all the positive examples along with the negative examples in this pool (see Fig. 5).

For every pixel labeled with red tide, a  $25 \times 25$  area around each pixel is collected and used as a positive example. In *negative random sampler*, we randomly select 100 regions of a size  $37 \times 37$  from a negative image, equivalent to 16,900 examples of size  $25 \times 25$ . In our experiments, we found that it is crucial to randomly select multiple regions instead of one wide region for negative examples, to improve the accuracy of red tide detection using cOHEM effectively. The red tide detection losses for all selected examples are calculated by feeding the examples to the model. Note that this loss represents the extent to which the

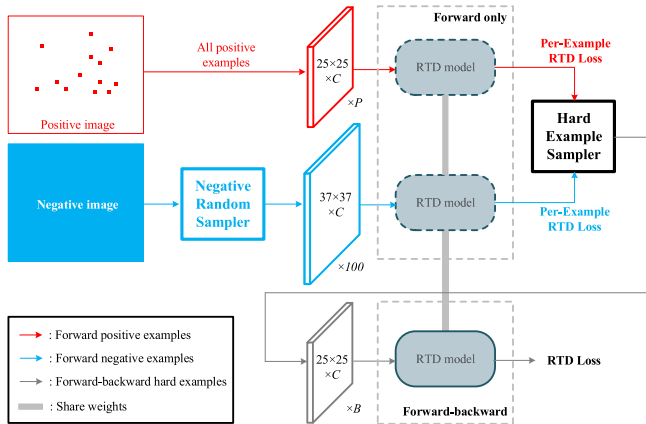


Fig. 5. **Cascaded OHEM** finds hard examples through two-stage sampling (*negative random sampler* and *hard example sampler*) for every iteration. (RTD Model: the proposed red tide detection model,  $P$ : the size of the positive example set,  $B$ : batch size,  $C$ : the spectral dimension of the input example.).

current model correctly classifies each example. *Hard example sampler* randomly selects the high loss examples with predetermined batch size. Then, the proposed model is trained with these batches consisting of hard examples. **Three-stage training strategy.** For training the red tide detection model using the proposed two-step HEG approach, we adopt a three-stage training strategy. The first stage is to train the initial red tide detection model using cOHEM. In the second stage, the generator and the discriminator are trained, as shown in Fig. 4. In this stage, the discriminator is first trained with generator weights unchanged and then the generator is trained while keeping the discriminator and the red tide detection model fixed. In the last stage, the red tide detection model is updated by using hard examples that are the output of the proposed HEG approach. Hard examples are mined from real positives, real negatives, and artificially generated negatives via cOHEM. In the third stage, generator weights are fixed.

For the first and third stage, all the  $25 \times 25$  positive regions and 100 randomly sampled examples of the  $37 \times 37$  negative regions are used for training. For the second stage, 256 of the  $25 \times 25$  negative regions are used. Note that our FCN can intake an input image of arbitrary size. The flowchart of the three-stage training algorithm is shown in Fig. 6. The intermediate output size of the proposed model in each training stage and test are shown in Appendix A to understand our proposed model and training strategy better.

#### IV. EXPERIMENTS: RED TIDE DETECTION

##### A. GOCI Satellite Images

GOCI acquires multispectral images from a large area surrounding the Korean peninsula. The GOCI image [9] has eight channels consisting of six visible and two near infrared (NIR) frequency bands<sup>3</sup> and 500 m spatial resolution. The size of

<sup>3</sup>While band extension [47]–[49] can be considered because of the small number of bands in the GOCI image, the nine-layer CNN shows sufficient performance even without band extension.

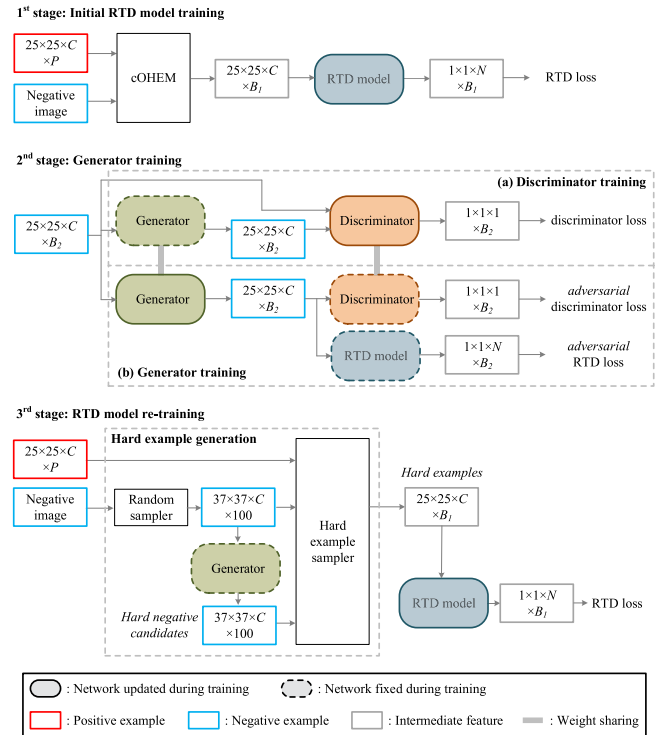


Fig. 6. **Three-stage training algorithm.** In the second stage, discriminator training (a) and generator training (b) are performed for each iteration. Data dimension is shown in the data blob in order of height, width, channel, and batch size.  $C$ ,  $P$ ,  $N$ ,  $B_1$ , and  $B_2$  are the spectral dimensions (e.g., 8 for GOCI image), the number of positive examples, the number of categories (e.g., 2 for red tide detection), batch sizes for training RTD model, and batch sizes for training generator, respectively.

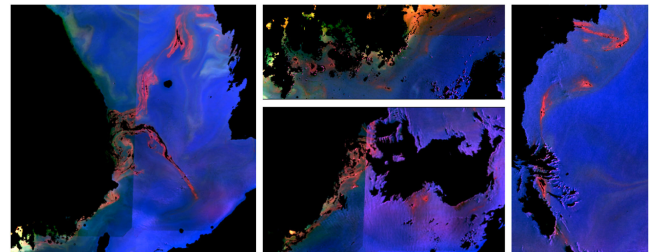


Fig. 7. **Red tide examples shown on GOCI images.** In the above figure, the red tide appears as elongated red bands. The images are false color images by combining the 6th, 4th and 1st band of the GOCI multispectral image representing the red, green, and blue colors, respectively.

the GOCI image is  $5567 \times 5685$ . Some examples of GOCI images are shown in Fig. 2. Several red tide examples on GOCI multispectral images are also shown in Fig. 7.

In this article, we use GOCI images taken in July, August, and December of 2013 to evaluate our red tide detection model. Images from July and August where red tide occurred are used as positive images, and images from December are used as negative images. Based on some conditions such as the atmosphere, we chose eight images in July and August and four images in December. Half of them were used for training and the other half for testing.

To label red tide pixels, we used the red tide information reported by National Institute of Fisheries (NIFS) of South Korea,

which directly tested seawater from a ship. NIFS examined red tide occurrence only at a limited number of locations; so it is impossible to cover the entire red tide areas. Furthermore, the red tide positions indicated in the reports were not very accurate due to the error-prone manual process that included mapping geo-coordinates of red tide locations onto GOCI images. Hence, we have extended potential red tide regions up to 25 km (50 pixel distance) in all directions from the red tide location indicated in the report and then labeled red tide with experts' help. Approximately 100 pixels from each training image were sparsely labeled as a red tide area. We used pixels labeled as red tide as positive examples and all the pixels of the December images as negative examples.

### B. Evaluation Settings

**Evaluation metrics.** We used two different metrics to evaluate the proposed model: the receiver operating characteristic (ROC) curve and the ROC variation curve. The ROC variation curve describes changes in the detection rate based on varying numbers of (true or false) detections per image (NDPI) instead of the false positive rate. This metric is beneficial when there are numerous unlabeled examples whose identity is unknown. Note that in a GOCI image only a fraction of red tide pixels are labeled and the rest of the image remains unlabeled. For quantitative analysis, we calculate the AUC (the area under the ROC curve) and  $\text{ndpi}@dr = 0.25$ ,  $\text{ndpi}@dr = 0.5$  and  $\text{ndpi}@dr = 0.75$  indicating the NDPI values when the detection rate reaches 0.25, 0.5, and 0.75, respectively.

**Model training.** The proposed models are trained from scratch. When HEG is used, we used a three-stage training strategy and trained the model with 1250 iterations for each stage. A base learning rate is 0.01 for the red tide detection model and generator and 0.0001 for the discriminator. The base learning rate drops to a factor of 10 for every 500 iterations. When the three stage training strategy is not used to train the model (i.e., artificially generated examples are not used for training), we trained the model with 2500 iterations. A base learning rate is 0.01 and drops by a factor of 10 for every 1 K iterations.

The proposed models are optimized by using a minibatch stochastic gradient descent (SGD) approach with a batch size of 256 examples, the momentum of 0.9, and weight decay of 0.0005. The red tide detection model's training objective is to minimize the cross entropy losses between the red tide labels and the final output scores. Each batch consists of examples extracted from one positive image with red tide occurrence and one negative image with no red tide occurrence. The positive-to-negative ratio in each batch is set to 1:3.

To reduce overfitting in training, data augmentation is carried out. Since a GOCI image is taken from a top view, training examples are augmented by mirroring across the horizontal, vertical, and diagonal axes. This mirroring can be performed in one direction or in multiple directions. This will increase the number of examples by eight times.

When training the red tide detection model, all learnable layers except for the layers of residual modules (3rd, 4th, 5th, and 6th layers) are initialized according to Gaussian distribution with

zero mean and 0.01 standard deviation. The layers of the residual modules are initialized according to Gaussian distribution with a mean of zero and a standard deviation of 0.005. All layers of the generator except for the last layer are initialized according to Gaussian distribution with a mean of zero and a standard deviation of 0.02. The last layer is initialized according to Gaussian distribution with a mean of zero and a standard deviation of 50.

### C. Architecture Design

In this section, we use a twofold cross validation that splits the training set into two subsets, alternating one for training and another for testing. AUC reported in the tables is the average over two validations.

**Finding the model specification.** To find the optimal specification of the red tide detection model, we evaluate the model by changing various types of model parameters, such as the number of filters and residual modules, and the types of filters used in the multiscale filter bank. The proposed model specifications are determined by evaluating detection accuracy (AUC), training time, and test time on various model parameters. The final model specification used in the proposed work is shown in Table I. Table I also indicates if a more extensive network is used by increasing the depth and breadth of the network, network overfitting on the GOCI dataset starts to occur. We use one GPU (NVIDIA Titan XP) that affects training and test times to conduct our experiments.

We also optimize the generator by changing the number of filters. As shown in Fig. 4, the generator is designed as a conv-deconv network consisting of eight convolutional layers and two deconvolutional layers. In this architecture, the number of filters in the first layer  $n$  is doubled in the third layer and then reduced by half in the sixth and again in the ninth layer. The last layer has eight filters so that its output has the same eight channels as those of the GOCI image's spectral signal. We evaluate detection accuracy and training time to find optimal architecture while  $n$  is varied among 16, 32, 64, and 128, as shown in Table II. Based on the results in Table II, we used the third architecture that adopts 64, 64, 128, 128, 128, 64, 64, 64, 32, and 8 filters for all ten layers, respectively. The fourth architecture, which employs the largest number of filters, may overfit the GOCI dataset.

**Finding sampling strategy of mining.** We compare various negative example sampling strategies of cOHEM with respect to detection accuracy and training time. Negative example sampling can change based on two factors: window size and the number of windows. In Table III, we compare four different sampling strategies with various factors. The first three strategies are chosen to maintain a similar number of negative examples as used in training. The last strategy (192 windows of  $25 \times 25$ ) is training without using cOHEM. Accordingly, its training time is the shortest.

In Table III, the third strategy gives the best performance in terms of AUC and training time. This observation indicates that increasing spatial diversity of sampling is essential in providing competitive performance. Therefore, even though the third strategy requires a large amount of memory due to the large pixels, it is adopted in our training approach.

TABLE I  
OPTIMUM SPECIFICATION OF THE RED TIDE DETECTION MODEL

|               | 64   | 96   | <b>128</b>  | 192  |               | 1    | <b>2</b>    | 3    |               | $1 \times 1$ | $\sim 5 \times 5$ | $\sim 9 \times 9$ | $\sim 13 \times 13$ |
|---------------|------|------|-------------|------|---------------|------|-------------|------|---------------|--------------|-------------------|-------------------|---------------------|
| AUC (%)       | 79.6 | 83.5 | <b>87.1</b> | 87.0 | AUC (%)       | 81.0 | <b>87.1</b> | 86.6 | AUC (%)       | 85.0         | 85.3              | 86.7              | <b>87.1</b>         |
| Train (min)   | 42.0 | 49.1 | <b>56.4</b> | 62.5 | Train (min)   | 47.5 | <b>56.4</b> | 63.1 | Train (min)   | 37.5         | 40.5              | 47.5              | <b>56.4</b>         |
| Test (sec/im) | 13.4 | 16.2 | <b>18.3</b> | 20.5 | Test (sec/im) | 16.0 | <b>18.3</b> | 21.4 | Test (sec/im) | 11.3         | 12.5              | 14.3              | <b>18.3</b>         |

(a) # of filters of each layer

(b) # of residual module

(c) Multi-scale filter bank.

The model specification is empirically determined based on three measures: detection accuracy (AUC), training time, and test time. Titan XP is used as a GPU that affects training and test time. *Numbers in bold* indicate the specifications used in our model.  $\sim 13 \times 13$  means the multiscale filter bank consisting of  $1 \times 1$ ,  $5 \times 5$ ,  $9 \times 9$ , and  $13 \times 13$  convolutional filters.

TABLE II  
OPTIMUM SPECIFICATION OF THE GENERATOR

| Architecture  | AUC (%)     | Train (sec/iter) |
|---|-------------|------------------|
| $16 \rightarrow 16 \rightarrow 32 \rightarrow 32(\rightarrow 32) \rightarrow 16 \rightarrow 16(\rightarrow 16) \rightarrow 8 \rightarrow 8$                       | 84.3        | 42.5             |
| $32 \rightarrow 32 \rightarrow 64 \rightarrow 64(\rightarrow 64) \rightarrow 32 \rightarrow 32(\rightarrow 32) \rightarrow 16 \rightarrow 8$                      | 88.6        | 47.6             |
| <b><math>64 \rightarrow 64 \rightarrow 128 \rightarrow 128(\rightarrow 128) \rightarrow 64 \rightarrow 64(\rightarrow 64) \rightarrow 32 \rightarrow 8</math></b> | <b>92.0</b> | <b>51.8</b>      |
| $128 \rightarrow 128 \rightarrow 256 \rightarrow 256(\rightarrow 256) \rightarrow 128 \rightarrow 128(\rightarrow 128) \rightarrow 64 \rightarrow 8$              | 91.5        | 55.3             |

For each architecture, the numbers in parentheses indicate the number of filters in the deconvolutional layers. The discriminator architecture is designed by adding one fully connected layer to the first four layers of the generator.

TABLE III  
ACCURACY OF VARIOUS NEGATIVE RANDOM SAMPLING STRATEGIES OF COHEM

| Window size           | $153 \times 153$ | $65 \times 65$ | <b><math>37 \times 37</math></b> | $25 \times 25$ |
|-----------------------|------------------|----------------|----------------------------------|----------------|
| # of window           | 1                | 10             | <b>100</b>                       | 192            |
| # of pixels           | 22.9K            | 41.3K          | <b>133.7K</b>                    | 120.0K         |
| # of examples         | 16.3K            | 16.4K          | <b>16.5K</b>                     | 192            |
| AUC (%)               | 87.7             | 89.1           | <b>90.4</b>                      | 88.5           |
| Train time (sec/iter) | 0.521            | 0.534          | <b>0.540</b>                     | 0.482          |

*Numbers in bold* indicate our cOHEM sampling strategy.

#### D. Experimental Results

**Baselines.** We implement three baselines: SVM and two CNN-based HSI classification approach [5], [50]. In SVM, a  $25 \times 25$  region centered on the pixel in test is used as a feature representing the center pixel. To know the advantages of adopting hard example mining, we applied conventional hard negative mining [16] to SVM training. Lee and Kwon [5], the CNN-based approach by which our model has been inspired. Another CNN-based baseline, diverse region-based CNN (DR-CNN) [50], inputs a set of diverse regions consisting of six different regions (i.e., global, right, left, top, bottom, and local regions) to encode semantic context-aware representation. In this experiment, we use a  $25 \times 25$  image patch as a global region compatible with the input dimensions of our approach.  $13 \times 25$  subpatch at top and bottom of the patch are used as the top and bottom regions. Similarly,  $25 \times 13$  subpatch at the left and right of the global patch are the left and right regions, respectively. The  $3 \times 3$  region at the center of the global patch is used as a local region.

**Performance comparison.** Table IV shows that our model trained using hard examples via HEG provides the highest accuracy in all four metrics. The proposed HEG was effective in improving the performance of our model and two CNN-based baselines. It is also observed that adopting a hard example mining approach consistently improves the accuracy of all four

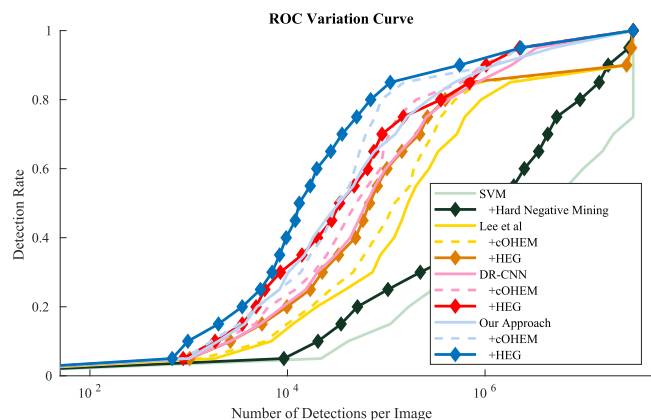


Fig. 8. **ROC variation curve.** To better view changes in detection rate and NDPI in detail, the  $x$  axis is shown with a logarithmic scale.

methods as it efficiently eases the significant imbalance between red tide and nonred tide examples.

Fig. 8 shows the ROC variation curves for our model and baselines. From Fig. 8, we can confirm that our model provides significantly enhanced detection performance compared to the baselines over the most range of NDPI. Fig. 9 shows red tide detection results from our approach.

#### E. Analyzing Hard Negative Candidates

In Fig. 10, we observe that the generator and discriminator converge successfully in the second training stage. This shows that generator optimization overcomes the interference of the red tide detection model and the discriminator.

Some examples of generated hard negative candidates, the successfully trained generator's output, are presented in Fig. 11. In Fig. 11, it can be observed that activated pixel regions (the third column of each set) tend to have different colors—i.e., different intensity values for some spectral bands—for certain



TABLE IV  
RED TIDE DETECTION ACCURACY

| Method                 | AUC (%)     | ndpi@dr=0.25 | ndpi@dr=0.5  | ndpi@dr=0.75 |
|------------------------|-------------|--------------|--------------|--------------|
| SVM                    | 81.0        | 295732       | 4641404      | 31648395     |
| + Hard Negative Mining | 82.4        | 114067       | 1259805      | 5313930      |
| [5]                    | 84.9        | 38642        | 165402       | 623806       |
| + cOHEM                | 87.5        | 27320        | 121269       | 356460       |
| + HEG                  | 88.9        | 16990        | 67792        | 261054       |
| DR-CNN [50]            | 89.4        | 15381        | 62933        | 254869       |
| + cOHEM                | 91.1        | 12057        | 49507        | 129733       |
| + HEG                  | 92.9        | 5890         | 33696        | 149099       |
| Ours                   | 90.6        | 8307         | 33080        | 131789       |
| + cOHEM                | 93.2        | 8298         | 30917        | 77196        |
| + HEG                  | <b>95.0</b> | <b>5722</b>  | <b>13168</b> | <b>50220</b> |

For each metric, *numbers in bold* indicate the best accuracy. Note that the higher the AUC value, the better the performance, and the smaller the value of NDPI, the better the performance.

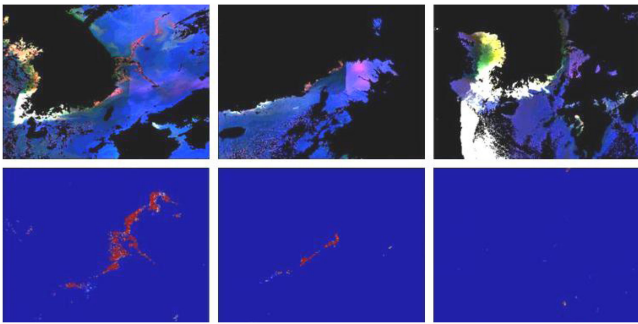


Fig. 9. **Red tide detection results.** The top images and the bottom images are input images and the corresponding red tide detection results from our approach, respectively. The first and second input images have red tide in them and the last image is a negative image without red tide.

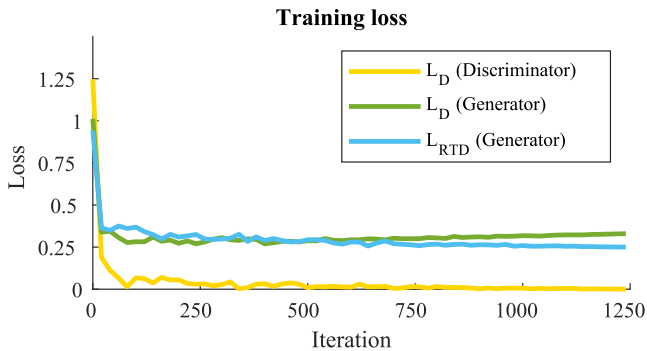


Fig. 10. **Training curve.** Losses changed during training generator and discriminator in the second training stage are shown.

areas than the ones covered by real nonred tide examples. Note that the activated regions in Fig. 11 do not show typical characteristics of red tide regions—narrow elongated bands with sharp boundaries as shown in Fig. 7. Furthermore, the activated regions by artificial nonred examples appear to be pink while the real red tide is generally red.

Fig. 12 shows an evolution of the ratio of training examples between real negative examples and generated negative examples within one batch as the third training stage progresses. Overall, the proportion of generated negative examples within

a batch gradually decreases as the training iteration increases. In Fig. 12, the generated examples appear to be heavily used for optimization in the first few iterations. After about 350 iterations, the positive examples are successfully separated from the generated examples by the optimized network, providing higher accuracy than the case in which the generated examples are not used for training.

To analyze the relationship among three different types of training examples—i.e., positive, real negative, and artificially generated negative examples—after training the generator at the second training stage, we use a visualization technique called *t*-distributed stochastic neighborhood embedding (*t*-SNE) [51], as shown in Fig. 13. For the visualization, 500 examples are randomly chosen separately from real positive and negative examples. The artificially generated examples are the output of the generator, which takes real negative examples as input. This selection of only a small fraction of real negative examples—i.e., about 500 from 400 M examples—is due to the significant imbalance between the number of positive and negative examples. To address this issue, we provide multiple instances of the *t*SNE-based visualization using different sets of randomly selected training examples to illustrate the general relationship, as shown in Fig. 13. We use the output of the 8th convolutional layer (after ReLU layer) as the selected examples' features. As shown on the top rows of Fig. 13, the artificially generated negative examples are placed between the positive and real negative examples. This indicates that the generated negative examples can be considered hard example candidates when retraining the red tide detection model. When training the generator without using the discriminator, the artificially generated negative examples are no longer hard examples as they are placed far to the right of the positive and real negative examples, as shown on the bottom rows of Fig. 13.

#### F. Ablation Study

**Training with unlabeled examples.** The main problem with using GOCI images is the labeling of red tide pixels. It is quite challenging to label every pixel, where a red tide appears on GOCI due to practical issues. Therefore, there is no guaranty

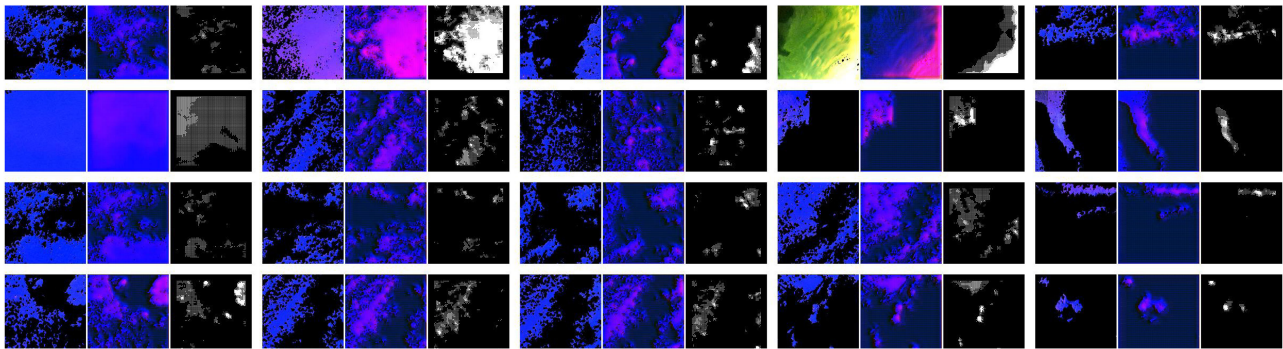


Fig. 11. **Generated training examples (after second training stage)**. Each set consists of three images: real negative example (left), generated negative example (center), and red tide detection model activation by the generated example (right). White pixels in the activation map indicate pixels with a red tide score estimated by the red tide detection model greater than 0.5. For every set in this figure, there was no activation for a real negative example.

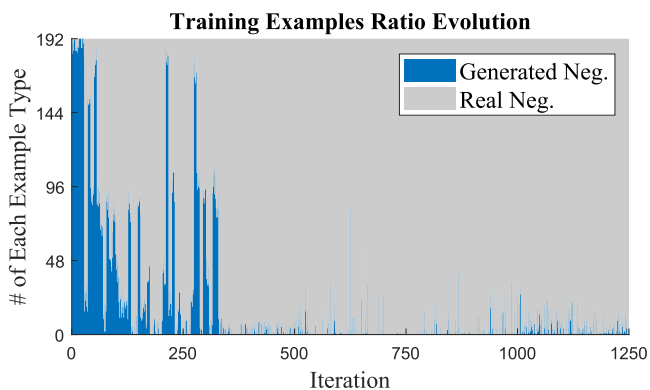


Fig. 12. **Negative example ratio evolution (in third training stage)**. This evolution shows the change in training negative example ratio at the third stage of the training strategy.

TABLE V  
ACCURACY WITH AND WITHOUT UNLABELED EXAMPLES

| Training Examples | AUC         | npdi<br>@dr=0.25 | npdi<br>@dr=0.5 | npdi<br>@dr=0.75 |
|-------------------|-------------|------------------|-----------------|------------------|
| w/ Unlabeled      | 82.8        | 105472           | 952403          | 1608326          |
| w/o Unlabeled     | <b>90.6</b> | <b>8307</b>      | <b>33080</b>    | <b>131789</b>    |

Numbers in bold indicate the best accuracy for each evaluation metric.

that pixels not labeled as the red tide in positive images (i.e., red tide images) are nonred tide pixels.

We carried out ablation experiments to validate our claim that pixels not labeled as red tide in positive images should not be used to train the proposed model. When unlabeled pixels are used, we also used all pixels of negative images for training. Table V provides accuracy with and without the use of unlabeled examples for training with regard to the four evaluation metrics. Table V shows that using unlabeled examples significantly reduces the accuracy for all the evaluation metrics. The unlabeled examples including red tide pixels, adversely affect the proposed model training, supporting the reason why unlabeled pixels should not be used for training.

#### Generating negative examples with the discriminator only.

To demonstrate the effectiveness of fooling red tide detector for generating hard negative example candidates, we also test

TABLE VI  
COMPARISON WITH GAN-BASED NEGATIVE GENERATOR

| HEG     | AUC         | npdi<br>@dr=0.25 | npdi<br>@dr=0.5 | npdi<br>@dr=0.75 |
|---------|-------------|------------------|-----------------|------------------|
| w/o HEG | 90.6        | 8307             | 33080           | 131789           |
| w/ GAN  | 90.3        | 8797             | 36217           | 129536           |
| w/ Ours | <b>95.0</b> | <b>5722</b>      | <b>13168</b>    | <b>50220</b>     |

W/o HEG indicates the case where generated negative examples are not used for training. W/GAN is the case using generated examples via GAN where the generator is trained only to fool the discriminator. W/Ours uses hard negative examples generated by our approach. Numbers in bold indicate the best accuracy for each evaluation metric.

a generator trained by fooling the discriminator only. The generator training approach becomes GAN [3] in that it consists of a discriminator and a generator, and these networks learn to restrain each other. Augmenting the training data by GAN-based image generation increased accuracy in many approaches [13], [52]–[54]. However, Table VI shows that the GAN-based data augmentation approach is significantly less performed than our approach. Moreover, it was even worse than the case without using generated examples. These observations indicate that deceiving the red tide detector in training the generator is essential for improving accuracy.

## V. EXPERIMENTS: HSI CLASSIFICATION

Our approach can also be easily generalized to HSI classification, requiring pixel-wise classification. For the HSI classification, we have used three benchmark datasets: Indian pines, Salinas, and PaviaU. For each HSI dataset, 200 pixels randomly selected from each category are used for training and all the remaining pixels are used for testing.

**Evaluation setting.** For HSI classification, we carried out experiments on three hyperspectral datasets: Indian Pines, Salinas, and PaviaU. The Indian Pines dataset is an image consisting of  $145 \times 145$  pixels with 200 spectral reflectance bands covering the spectral range from 0.4 to  $2.5 \mu\text{m}$  with a spatial resolution of 20 m. There are 16 material categories in the Indian Pines dataset, but only eight materials with relatively large samples are used for evaluation. The Salinas dataset includes 16 classes with  $512 \times 217$  pixels, 204 spectral bands, and a high spatial resolution of

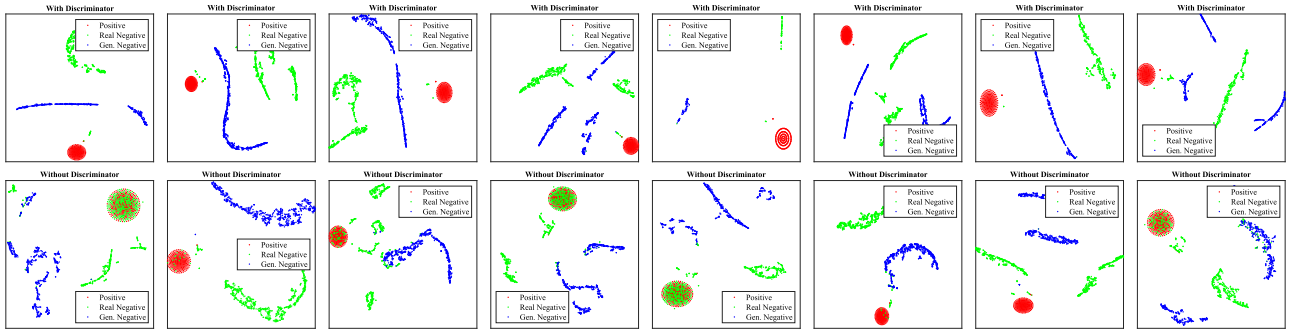


Fig. 13. **t-SNE visualization of three different types of training examples.** The top row cases indicate when the discriminator was used for training the generator, and the bottom row cases show when the discriminator was not used. Data points close to each other in the embedded space are likely to have similar spectral characteristics.

TABLE VII  
SELECTED CLASSES FOR EVALUATION AND THE NUMBERS OF TRAINING AND TEST EXAMPLES

| Class           | Training    | Test        |
|-----------------|-------------|-------------|
| Corn-notill     | 200         | 1228        |
| Corn-mintill    | 200         | 630         |
| Grass-pasture   | 200         | 283         |
| Hay-windrowed   | 200         | 278         |
| Soybean-notill  | 200         | 772         |
| Soybean-mintill | 200         | 2255        |
| Soybean-clean   | 200         | 393         |
| Woods           | 200         | 1065        |
| <b>Total</b>    | <b>1600</b> | <b>6904</b> |

(a) Indian Pines

| Class                     | Training    | Test         |
|---------------------------|-------------|--------------|
| Broccoli green weeds 1    | 200         | 1809         |
| Broccoli green weeds 2    | 200         | 3526         |
| Fallow                    | 200         | 1776         |
| Fallow rough plow         | 200         | 1194         |
| Fallow smooth             | 200         | 2478         |
| Stubble                   | 200         | 3759         |
| Celery                    | 200         | 3379         |
| Grapes untrained          | 200         | 11071        |
| Soil vineyard develop     | 200         | 6003         |
| Corn senesced green weeds | 200         | 3078         |
| Lettuce romaines, 4 wk    | 200         | 868          |
| Lettuce romaines, 5 wk    | 200         | 1727         |
| Lettuce romaines, 6 wk    | 200         | 716          |
| Lettuce romaines, 7 wk    | 200         | 870          |
| Vineyard untrained        | 200         | 7068         |
| Vineyard vertical trellis | 200         | 1607         |
| <b>Total</b>              | <b>3200</b> | <b>50929</b> |

(b) Salinas

| Class        | Training    | Test         |
|--------------|-------------|--------------|
| Asphalt      | 200         | 6431         |
| Meadows      | 200         | 18449        |
| Gravel       | 200         | 1899         |
| Trees        | 200         | 2864         |
| Sheets       | 200         | 1145         |
| Bare soils   | 200         | 4829         |
| Bitumen      | 200         | 1130         |
| Bricks       | 200         | 2482         |
| Shadows      | 200         | 747          |
| <b>Total</b> | <b>1800</b> | <b>40976</b> |

(c) PaviaU

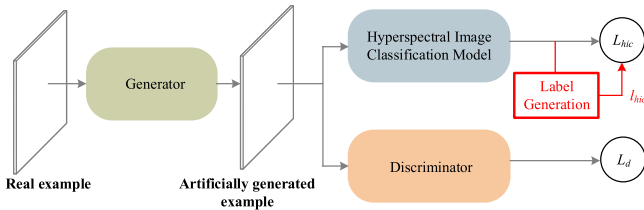


Fig. 14. **Training multicategory generator** follows the same process as hard negative candidate generator training shown in Fig. 4. The only difference is to add the “Label Generation” module, which calculates the adversarial label for each example, at the end of the HSI classification model.

TABLE VIII  
STEP SIZE AND THE NUMBER OF ITERATION FOR DIFFERENT TRAINING SETS

| Dataset Stage | Indian Pines |          | Salinas |        | PaviaU |        |
|---------------|--------------|----------|---------|--------|--------|--------|
|               | 1 & 3        | 2        | 1 & 3   | 2      | 1 & 3  | 2      |
| SS/Iter.      | 500/1250     | 500/1250 | 2K/5K   | 800/3K | 2K/5K  | 800/3K |

3.7 m. The Salinas and the Indian Pines datasets have the same frequency characteristics because they are acquired by the same

TABLE IX  
HSI CLASSIFICATION ACCURACY (MEAN AND STANDARD DEVIATION)

| Dataset      | [5]          | Ours         | +HEG                |
|--------------|--------------|--------------|---------------------|
| Indian pines | 93.61 ± 0.56 | 95.17 ± 0.48 | <b>96.43 ± 0.42</b> |
| Salinas      | 95.07 ± 0.23 | 96.09 ± 0.40 | <b>96.53 ± 0.48</b> |
| PaviaU       | 95.97 ± 0.46 | 96.27 ± 0.51 | <b>97.33 ± 0.37</b> |

When calculating the accuracy for our model, training and evaluation protocols follow those in [5].

AVIRIS sensor. The PaviaU dataset acquired by ROSIS sensor has  $610 \times 340$  pixels with nine material categories and 103 spectral bands covering 0.43 to 0.86  $\mu\text{m}$  spectral range with a 1.3 m spatial resolution. For the Salinas dataset and the PaviaU dataset, unlike the Indian Pines dataset, all available material categories are used.

For each dataset, randomly selected 200 examples from each category are used for training and the remaining examples are used for test, as shown in Table VII. We perform this partitioning 20 times and present the mean and standard deviation as overall classification accuracy.

TABLE X  
SIZE OF THE INTERMEDIATE OUTPUT

| Training: Stage 1 & 3 (Training Red Tide Detection Network by Using cOHEM) |               |                        |                        |                        |                        |                           |                           |                   |                 |                 |             |           |                |           |  |     |
|--|---------------|------------------------|------------------------|------------------------|------------------------|---------------------------|---------------------------|-------------------|-----------------|-----------------|-------------|-----------|----------------|-----------|--|-----|
| Before Hard Example Sampling   |               |                        |                        |                        |                        |                           |                           |                   |                 |                 |             |           |                |           |  |     |
|  | Input         | $C1 \times 1$          | $C5 \times 5$ (pad: 4) | $MAX5 \times 5$        | $C9 \times 9$ (pad: 8) | $MAX9 \times 9$           | $C13 \times 13$ (pad: 12) | $MAX13 \times 13$ | Concat.         | $C2, \dots, C8$ | $C9$        | Output    |                |           |  |     |
| Positive   | (25,25,8)     | (25,25,128)            | (29,29,128)            | (25,25,128)            | (33,33,128)            | (25,25,128)               | (37,37,128)               | (25,25,128)       | (25,25,512)     | (25,25,128)     | (25,25,1)   | (1,1,1)   |                |           |  |     |
| Negative   | (37,37,8)     | (37,37,128)            | (41,41,128)            | (37,37,128)            | (45,45,128)            | (37,37,128)               | (49,49,128)               | (37,37,128)       | (37,37,512)     | (37,37,128)     | (37,37,1)   | (13,13,1) |                |           |  |     |
| After Hard Example Sampling  |               |                        |                        |                        |                        |                           |                           |                   |                 |                 |             |           |                |           |  |     |
|  | Input         | $C1 \times 1$          | $C5 \times 5$          | $MAX5 \times 5$        | $C9 \times 9$          | $MAX9 \times 9$           | $C13 \times 13$           | $MAX13 \times 13$ | Concat.         | $C2, \dots, C8$ | $C9$        | Output    |                |           |  |     |
| Hard Example   | (1,1,8)       | (1,1,8)                | .                      | .                      | .                      | .                         | .                         | .                 | .               | .               | .           | .         |                |           |  |     |
|  | (9,9,8)       | .                      | (5,5,128)              | (1,1,128)              | .                      | .                         | .                         | .                 | (1,1,512)       | (1,1,128)       | (1,1,1)     | (1,1,1)   |                |           |  |     |
|  | (17,17,8)     | .                      | .                      | .                      | (9,9,128)              | (1,1,128)                 | .                         | .                 | .               | .               | .           | .         |                |           |  |     |
|  | (25,25,8)     | .                      | .                      | .                      | .                      | .                         | (13,13,128)               | (1,1,128)         | .               | .               | .           | .         |                |           |  |     |
| Training: Stage 2 (Training HNG/Discriminator)                             |               |                        |                        |                        |                        |                           |                           |                   |                 |                 |             |           |                |           |  |     |
| HNG  |               |                        |                        |                        |                        |                           |                           |                   |                 | Discriminator   |             |           | RT Detect Net. |           |  |     |
| Input  | $C1, C2$      | $C3, C4$               | $D5$                   | $C6, C7$               | $D8$                   | $C9$                      | $C10$                     | Output            | Input           | $C1, C2$        | $C3, C4$    | $FC$      | Output         | Input     | $C1 \times 1 \dots$                          |     |
| (25,25,8)  | (13,13,64)    | (7,7,128)              | (13,13,128)            | (13,13,64)             | (25,25,64)             | (25,25,32)                | (25,25,8)                 | (25,25,8)         | (25,25,8)       | (13,13,64)      | (7,7,128)   | (1)       | (1)            | (25,25,8) | (1,1,8)<br>(9,9,8)<br>(17,17,8)<br>(25,25,8) | ... |
| Training: Stage 3 (Artificially Generating Hard Examples)                  |               |                        |                        |                        |                        |                           |                           |                   |                 |                 |             |           |                |           |  |     |
| HNG  |               |                        |                        |                        |                        |                           |                           |                   |                 |                 |             |           |                |           |  |     |
| Input  | $C1, C2$      | $C3, C4$               | $D5$                   | $C6, C7$               | $D8$                   | $C9$                      | $C10$                     | Output            |                 |                 |             |           |                |           |  |     |
| (37,37,8)  | (19,19,64)    | (10,10,128)            | (19,19,128)            | (19,19,64)             | (37,37,64)             | (37,37,32)                | (37,37,8)                 | (37,37,8)         |                 |                 |             |           |                |           |  |     |
| Test   |               |                        |                        |                        |                        |                           |                           |                   |                 |                 |             |           |                |           |  |     |
| Input  | $C1 \times 1$ | $C5 \times 5$ (pad: 4) | $MAX5 \times 5$        | $C9 \times 9$ (pad: 8) | $MAX9 \times 9$        | $C13 \times 13$ (pad: 12) | $MAX13 \times 13$         | Concat.           | $C2, \dots, C8$ | $C9$            | Output      |           |                |           |  |     |
| (600,600,8)  | (600,600,128) | (604,604,128)          | (600,600,128)          | (608,608,128)          | (600,600,128)          | (612,612,128)             | (600,600,128)             | (600,600,512)     | (600,600,128)   | (600,600,1)     | (576,576,1) |           |                |           |  |     |

The numbers in parentheses indicate the width, height, and channels of the intermediate output, respectively.  $C$ ,  $D$ , and  $FC$  denote the convolutional layer, the deconvolutional layer, and the fully connected layer, respectively.

**Hard example candidate generation.** When applying the proposed training strategy to HSI classification, we replace single-category hard negative generator with a multicategory hard example candidate generator that generates hard examples of individual material categories instead of hard negatives. The generator is trained to create examples misclassified by the HSI classification model. For each example  $e$ , its adversarial training label ( $l_{\text{hic}}^a$ ) is the category with the largest loss among all categories except for the class the example belongs to

$$l_{\text{hic}}^a = \arg \max_{l \in C, l \neq l_{\text{hic}}} \mathcal{L}_{\text{hic}}(e, l) \quad (4)$$

where  $l_{\text{hic}}$  is a label associated with the example  $e$  and  $C$  is the set of all categories. Generator optimization is carried out with (3) by replacing  $L_{\text{rd}}$  (denoted as  $\mathbf{1}$  in the Equation) with  $L_{\text{hic}}^a = \{l_{\text{hic}}^a\}$ .

**Architecture.** Unlike the red tide detection model that outputs one-dimensional (1-D) sigmoid probabilities, the HSI classification model uses the softmax layer to calculate multiple categories' final probabilities. We also use the softmax loss to train the model while the red tide detection model is optimized by minimizing the cross-entropy loss.

When training the generator in the second training stage, we add the module ("Label Generation" in Fig. 14) to the end of the network. This module finds an adversarial label for each example using (4). The example becomes a hard example for its adversarial label when training the model. While the generator used in red tide detection only intakes negative examples to

generate hard negative examples, the generator for HSI classification takes examples of any material category as an input.

**Model training.** A SGD approach is used to train the model for HSI classification. For each iteration in training, a batch includes 256 examples. The base learning rate is set to 0.001 and reduced by a factor of 10 for every step size. For the other optimization parameters, we set the momentum to 0.9, gamma to 0.1, and weight decay to 0.0005. Table VIII shows the step size and the number of iterations for each training stage.

**Results.** Table IX shows the classification accuracy for the three datasets. For all the three datasets, our nine-layer CNN outperforms the baseline introduced in [5]. Our strategy of generating HEG further improves performance by at least 0.44%.

Note that the problems we encountered with GOCI images (i.e. extreme sparsity of red tide samples, significant imbalanced distribution, difficulties in accurate groundtruthing, etc.) are not normally observed in other HSIs. Enhanced performance for HSI classification verifies that the proposed HEG approach is also applicable to other related problem domains, such as HSI analysis.

## VI. CONCLUSION

In this article, we have developed a novel nine-layer FCN suitable for pixel-wise classification. Due to the challenges of annotating Earth-observing remotely sensed images to the pixel level, there are very few fully annotated satellite-based remote sensing data. To avoid the performance degradation caused by

significantly insufficient and imbalanced training data, we introduce a novel approach based on HEG. The proposed HEG approach takes two steps, first generating hard example candidates and mining hard examples from real and generated examples. In the first step, the generator that creates hard example candidates is learned via the adversarial learning framework by fooling a discriminator and a pixel-wise classification model at the same time. In the second step, OHEM is used in a cascaded fashion to mine hard examples from a large pool of real and artificially generated examples. The proposed FCN jointly trained with HEG approach provides state-of-the-art accuracy for red tide detection. We also show that the proposed approach can be easily extended to other tasks, such as HSI classification.

## APPENDIX A

### INTERMEDIATE OUTPUT SIZE OF THE PROPOSED MODELS

Our FCN architecture is designed to be slightly different between training and test to make it suitable for pixel-wise classification. The size of the input is also different for individual training stages. Furthermore, in the first and third training stages, the sizes of positive and negative examples are different for the same stages, making it easy to understand the architecture. Accordingly, to get an accurate understanding of the architecture, we provide the size of the intermediate output of the model in Table X. The red tide detection model weights are transferred between training and testing and during different training phases.

## REFERENCES

- [1] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.
- [2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.
- [3] I. Goodfellow *et al.*, "Generative adversarial Nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, vol. 27, pp. 2672–2680.
- [4] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 761–769.
- [5] H. Lee and H. Kwon, "Going deeper with contextual CNN for hyperspectral image classification," *IEEE Trans. Image Process.*, vol. 26, no. 10, pp. 4843–4855, Oct. 2017.
- [6] H. Lee, S. Eum, and H. Kwon, "Cross-domain CNN for hyperspectral image classification," in *Proc. Int. Geosci. Remote Sens. Symp.*, 2018, pp. 3627–3630.
- [7] H. Lee and H. Kwon, "Contextual deep CNN based hyperspectral classification," in *Proc. Int. Geosci. Remote Sens. Symp.*, 2016, pp. 3322–3325.
- [8] H. Lee, S. Eum, and H. Kwon, "Is pretraining necessary for hyperspectral image classification?" *Proc. Int. Geosci. Remote Sens. Symp.*, 2019, pp. 3321–3324.
- [9] S. Cho, Y. Ahn, J. Ryu, G. Kang, and H. Youn, "Development of geostationary ocean color imager (GOCI)," *Korean J. Remote Sens.*, vol. 26, no. 2, pp. 157–165, 2010.
- [10] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Representations*, 2014, pp. 1–10.
- [11] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. Int. Conf. Learn. Representations*, 2016, pp. 1–16.
- [12] X. Wang, A. Shrivastava, and A. Gupta, "A-Fast-RCNN: Hard positive generation via adversary for object detection," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2606–2615.
- [13] L. H. Hughes, M. Schmitt, and X. X. Zhu, "Mining hard negative samples for SAR-optical image matching using generative adversarial networks," *Remote Sens.*, vol. 10, no. 10, pp. 1–17, Sep. 2018.
- [14] J. Choi, T. Kim, and C. Kim, "Self-ensembling with gan-based data augmentation for domain adaptation in semantic segmentation," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 6830–6840.
- [15] C. Xie, M. Tan, B. Gong, J. Wang, A. Yuille, and Q. V. L., "Adversarial examples improve image recognition," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 819–828.
- [16] K. Sung and T. Poggio, "Learning and example selection for object and pattern detection," A.I. Technical Report, no. 1521, Jan. 1996, pp. 1–195.
- [17] L. Bourdev, S. Maji, T. Brox, and J. Malik, "Detecting people using mutually consistent poselet activations," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 168–181.
- [18] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2005, vol. 1, pp. 886–893.
- [19] L. Bourdev and J. Malik, "Poselets: Body part detectors trained using 3D human pose annotations," in *Proc. Int. Conf. Comput. Vis.*, 2009, pp. 1365–1372.
- [20] H. Lee, V. Morariu, and L. Davis, "Qualitative pose estimation by discriminative deformable part models," in *Proc. Asian Conf. Comput. Vis.*, 2012, pp. 734–747.
- [21] S. Maji, L. Bourdev, and J. Malik, "Action recognition from a distributed representation of pose and appearance," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 3177–3184.
- [22] H. Lee, V. Morariu, and L. Davis, "Robust pose features for action recognition," in *Proc. Conf. Comput. Vis. Pattern Recognit. Workshops*, 2014, pp. 359–366.
- [23] H. Lee, V. Morariu, and L. Davis, "Clauselets: Leveraging Temporally Related Actions for Video Event Analysis," in *Proc. Winter Conf. Appl. Comput. Vis.*, 2015, pp. 1161–1168.
- [24] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.
- [25] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, Jan. 2016.
- [26] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.
- [27] T. Malisiewicz, A. Gupta, and A. Efros, "Ensemble of exemplar-SVMs for Object Detection and Beyond," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 89–96.
- [28] R. Girshick, "Fast R-CNN," in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.
- [29] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
- [30] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, 2013.
- [31] B. Kellenberger, D. Marcos, S. Lobry, and D. Tuia, "Half a percent of labels is enough: Efficient animal detection in UAV imagery using deep CNNs and active learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 12, pp. 9524–9533, Dec. 2019.
- [32] G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han, "When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative CNNs," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 5, pp. 2811–2821, May 2018.
- [33] X. Feng, J. Han, X. Yao, and G. Cheng, "Progressive contextual instance refinement for weakly supervised object detection in remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 11, pp. 8002–8012, Nov. 2020.
- [34] X. Yao, X. Feng, J. Han, G. Cheng, and L. Guo, "Automatic weakly supervised object detection from high spatial resolution remote sensing images via dynamic curriculum learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 1, pp. 675–685, Jan. 2021.
- [35] A. King, S. Bhandarkar, and B. Hopkinson, "A comparison of deep learning methods for semantic segmentation of coral reef survey images," in *Proc. Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 1394–1402.
- [36] A. Mahmood *et al.*, "Coral classification with hybrid feature representations," in *Proc. Int. Conf. Image Process.*, 2016, pp. 519–523.
- [37] G. Ding *et al.*, "Fish recognition using convolutional neural network," in *Proc. OCEANS Anchorage*, 2017, pp. 1–4.
- [38] Z. Ge, C. McCool, C. Sanderson, and P. Corke, "Modelling local deep convolutional neural network features to improve fine-grained image classification," in *Proc. Int. Conf. Image Process.*, 2015, pp. 4112–4116.

- [39] R. Mandal, R. Connolly, T. Schlacher, and B. Stantic, "Assessing fish abundance from underwater video using deep neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, 2018, pp. 1–6.
- [40] M. Nieto-Hidalgo, A.-J. Gallego, P. Gil, and A. Pertusa, "Two-stage convolutional neural network for ship and spill detection using SAR images," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 9, pp. 5217–5230, Sep. 2018.
- [41] J. E. Zini, Y. Rizk, and M. Awad, "A deep transfer learning framework for seismic data analysis: A case study on bright spot detection," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 5, pp. 3202–3212, May 2020.
- [42] H. Cai, C. Bai, Y.-W. Tai, and C.-K. Tang, "Deep video generation, prediction and completion of human action sequences," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 366–382.
- [43] D. Fourure, R. Emonet, E. Fromont, D. Muselet, A. Tremeau, and C. Wolf, "Residual conv-deconv grid network for semantic segmentation," in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 181.1–181.13.
- [44] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection - a new baseline," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6536–6545.
- [45] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 1520–1528.
- [46] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Interv.*, 2015, pp. 234–241.
- [47] C. Kwan *et al.*, "Deep learning for land cover classification using only a few bands," *Remote Sens.*, vol. 12, no. 12, pp. 1–17, 2020, Art. no. 2000.
- [48] C. Kwan, D. Gribben, B. Ayhan, J. Li, S. Bernabe, and A. Plaza, "An accurate vegetation and non-vegetation differentiation approach based on land cover classification," *Remote Sens.*, vol. 12, no. 23, pp. 1–29, 2020, Art. no. 3880.
- [49] C. Kwan, D. Gribben, B. Ayhan, S. Bernabe, A. Plaza, and M. Selva, "Improving land cover classification using extended multi-attribute profiles (EMAP) enhanced color, near infrared, and LiDAR data," *Remote Sens.*, vol. 12, no. 9, pp. 1–28, 2020, Art. no. 1392.
- [50] M. Zhang, W. Li, and Q. Du, "Diverse region-based CNN for hyperspectral image classification," *IEEE Trans. Image Process.*, vol. 27, no. 6, pp. 2623–2634, Jun. 2018.
- [51] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. Nov, pp. 2579–2605, 2008.
- [52] A. Dosovitskiy and T. Brox, "Generating images with perceptual similarity metrics based on deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 658–666.
- [53] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, "Plug & play generative networks: Conditional iterative generation of images in latent space," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4467–4477.
- [54] A. Diba, V. Sharma, R. Stiefelhagen, and L. V. Gool, "Weakly supervised object discovery by generative adversarial & ranking networks," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 601–610.



**Hyungtae Lee** (Member, IEEE) received the B.E. degree in electrical engineering and mechanical engineering from Sogang University, Seoul, South Korea, in 2006, the master's degree from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2008, and the Ph.D. degree from the University of Maryland, College Park, MD, USA, in 2014.

He was at Booz Allen Hamilton Inc and US Army Research Lab from 2014 to 2021. In 2021, he joined with the U.S. Army Research Laboratory as a Civilian

Government Employee and Research Scientist. He was a Senior Program Committee for AAAI-22, a Guest Associated Editor for the remote sensing special issue, and a Program Committee Member/Reviewer for tens of top-tier conferences and journals. His current research interests are computer vision, machine learning, deep learning, and AI.



**Heesung Kwon** (Senior Member, IEEE) received the B.Sc. degree in electronic engineering from Sogang University, Seoul, South Korea, in 1984, and the M.S. and Ph.D. degrees in Electrical engineering from the State University of New York at Buffalo, NY, USA, in 1995 and 1999, respectively.

From 1983 to 1993, he was with Samsung Electronics Corp., where he worked as a Senior Research Engineer. He was with the U.S. Army Research Laboratory (ARL), Adelphi, MD from 1996 to 2006 working on automatic target detection and hyperspectral signal processing applications. From 2006 to 2007, he was with Johns Hopkins University Applied Physics Laboratory (JHU/APL) working on biological standoff detection problems. He rejoined ARL in August, 2007 as a Senior Electronics Engineer, leading hyperspectral research efforts in the Image Processing Branch. He has authored or coauthored more than 130 journal, book chapters, and conference papers on these topics. His current research interests include AI & ML, ATR, image/video analytics, human-autonomy interaction, hyperspectral signal processing, and statistical learning.

Dr. Kwon was Associate Editor for the IEEE TRANSACTIONS ON AEROSPACE AND ELECTRONIC SYSTEMS. He also served as one of government leads on the Internet of Battlefield Things (IoBT) Collaborative Research Alliance (CRA) with ARL, leading the research area of intelligent battlefield services that aims to provide advanced analytic services in IoBT environments.



**Wonkook Kim** received the master's and Ph.D. degree in civil engineering from Purdue University, West Lafayette, IN, USA, in 2005 and 2011, respectively.

He was a Research Associate with the Department of Geographical Sciences, University of Maryland, College Park, MD, USA, and a Visiting Scholar with the National Environmental Satellite, Data, and Information Service (NESDIS) in National Oceanic and Atmospheric Administration (NOAA), where he worked on the vicarious calibration and the BRDF

characterization for satellite products. He performed research on Geostationary Ocean Color Imager with Korea Institute of Ocean Science and Technology (KIOST), as a Senior Researcher between 2013 and 2019. He joined Department of Civil and Environmental Engineering, Pusan National University, South Korea, as an Assistant Professor in 2019, where he teaches remote sensing and photogrammetry. His research interests are on the quantitative remote sensing that encompasses the use of hyperspectral sensors, UAVs, and satellite data, for the retrieval of geophysical variables of water, atmosphere, and land surface.