Evaluating Self and Semi-Supervised Methods for Remote Sensing Segmentation Tasks

Chaitanya Patel*[1] Shashank Sharma*[2] and Varun Gulshan[2]
Google Research India†

***Abstract*—We perform a rigorous evaluation of recent self and semi-supervised ML techniques that leverage unlabeled data for improving downstream task performance, on three remote sensing tasks of riverbed segmentation, land cover mapping and flood mapping. These methods are especially valuable for remote sensing tasks since there is easy access to unlabeled imagery and getting ground truth labels can often be expensive. We quantify performance improvements one can expect on these remote sensing segmentation tasks when unlabeled imagery (outside of the labeled dataset) is made available for training. We also design experiments to test the effectiveness of these techniques when the test set has a domain shift relative to the training and validation sets.**

***Index Terms*—Deep Learning, Remote Sensing, Semi-supervised Learning, Self-supervised Learning, Segmentation.**

I. Introduction

OVER the recent years, many machine learning methods have been proposed that leverage unlabeled data in order to build high performing models using only small labeled datasets [1]–[3]. This is both important for advancing our understanding of how humans learn (e.g. we do not need millions of labels to learn new concepts) as well as practical applications, where labeling data is expensive and time consuming. Remote sensing tasks are especially good candidates for leveraging unlabeled data for multiple reasons: (i) Labeling can be expensive for certain tasks, requiring ground surveying (e.g. crop type labeling [4]), (ii) Generalization issues/domain shifts are common as the geographical context changes and often needs additional labels when deploying models to a new region of interest, and (iii) There is easy access to unlabeled data that is well sampled across all domains. Our goal in this work is to evaluate recently proposed ML techniques that show promising and robust results in learning from small labeled datasets, on remote sensing tasks. We specifically test these methods on their ability to improve generalization across domain shifts, and also augment all the supervised remote sensing datasets with unlabeled images sampled from the same satellite collections.

A. Review of ML Methods

There are two broad ML techniques that have shown promising results using unlabeled image data in recent years: Self-supervised learning and Semi-supervised learning.

Self-supervised learning aims at learning image representations from unlabeled data by solving a pretext task which doesn't require manual labels. Solving such pretext tasks forces the models to learn representations that can encode prior knowledge of the input domain (i.e. satellite imagery in our case). These representations can be used as feature extractors or can be finetuned for a specific downstream task with small amounts of labeled data. One category of self-supervised pretext tasks drops a part of the input signal and trains the network to predict it from the remaining information. Such tasks include image-inpainting (i.e. predicting an artificially cut out part of the image) [5], colorizing grayscale images [6], predicting rotation angle [7], solving jigsaw puzzles (by predicting the correct order of shuffled patches) [8] etc. More recently, contrastive self-supervised methods have gained popularity where the network is trained to embed the augmented versions of the same sample (positive pair) closer to each other while pushing away the embeddings of other samples (negative samples). Various contrastive self-supervised learning methods (SimCLR [3], MoCo [9], BYOL [10], SwAV [11]) have a similar setup and mainly differ in how they generate negative samples. Representations learned by these contrastive models perform on par with the representations learned using purely supervised baselines on ImageNet classification and also show SOTA (state-of-the-art) results for other vision tasks via transfer learning [9] [11]. Self-supervised models can also improve the robustness of the model against uncertainty and outliers [12].

Semi-supervised learning methods work by encouraging the model to better generalize to unseen data, usually done by adding an extra loss function. The loss function is often one of entropy minimization or consistency regularization. These loss functions improve model generalization by regularizing the model outputs on unlabeled data. Entropy minimization encourages the model to output more confident predictions for the unlabeled data: Grandvalet *et al.* [13] achieve this by explicitly training the model to reduce entropy of the predicted distribution on unlabeled data, while Sajjadi *et al.* [14] enforce this via guiding the decision boundary to lie on the low density space between the manifolds. Consistency regularization, on the other hand, encourages the network to output the same predictions for perturbations of the same input data point (MixMatch [15], ReMixMatch [16], FixMatch [2], [17]). With these recent advancements of FixMatch [2] (and others, e.g. DivideMix [18]), semi-supervised learning has established itself as a promising technique for utilizing unlabeled data (especially in low label data regimes).

* C. Patel and S. Sharma contributed equally as first authors.
[1] Email: chaitanya100100@gmail.com
[2] Email: {sharshashank, varungulshan}@google.com
† This work was done while C. Patel was with Google Research India.

B. Self-supervised and semi-supervised methods in Remote Sensing

Self-supervised methods have been adopted for various remote sensing tasks to learn rich image features from unlabeled data and improve performance on specific downstream applications like change-detection [19], hyperspectral image super-resolution [20], scene classification via pretext multitask learning [21], and land-cover classification using the pretext task of colorization [22]. Lin *et al.* [23] use GANs (Generative Adversarial Network) with mid and high level feature fusion to learn image representations for several classification tasks. Lu *et al.* [24] use unsupervised feature learning for scene classification from high spatial resolution data. Jean *et al.* [25] use triplet loss to enforce similarity in the representations of neighboring tiles as their learning task. More recently, Jung *et al.* [26], Stojnic *et al.* [27], Ayush *et al.* [28], and Kang *et al.* [29] have explored and adopted state-of-the-art contrastive self supervised methods for remote sensing to learn better representations and improve transfer learning performance. However, these papers have largely focused on image-level classification tasks. Leenstra *et al.* [30] train a self-supervised network to identify overlapping patches and make their representations similar to each other, which is then finetuned for change detection - a segmentation task. Li *et al.* [31] train a multi-task self-supervised network with three pretext tasks (inpainting occluded patch, predicting relative transformation between two patches and contrastive loss) and show better performance than ImageNet pretrained baseline for finetuning on segmentation datasets.

Semi-supervised techniques have been used in remote sensing tasks that use classical ML methods, e.g. SVMs for image classification ([32], [33], [34]) and segmentation ([35], [36]). Semi-supervised learning has also been used for Image retrieval: Chaudhuri *et al.* [37] use a graph-theoretic method, and, Hu *et al.* [38] use semi-supervised manifold alignment (SSMA), a multi-modal data fusion algorithm for Manifold alignment while combining optical image and polarimetric SAR Data. Semi-supervised methods have also helped in the problem of dimensionality reduction of hyperspectral data: Hong *et al.* [39] utilize iterative multitask regression framework and Wu *et al.* [40] propose semi-supervised based Local Fisher Discriminant Analysis. More recently, due to the success of deep neural networks, deep learning based semi-supervised methods have been used for a variety of remote sensing tasks as well. These include adversarial hashing based large-scale image retrieval [41], boundary-aware semantic segmentation of very-high resolution images (BASNet) [42], individual tree canopy detection [43] and high-resolution scene classification [44]. A recent work by Hong *et al.* [45] explores semi-supervised cross-modal learning between multi-spectral data (MSI), synthetic aperture radar (SAR) data and small-scale-hyperspectral data (HSI). This enables their model to generalize well and it achieves better performance than the state-of-the-art models at the task of classification.

Given rapidly evolving self and semi-supervised ML techniques (especially on camera imagery), there is a critical need to have a systematic study of these methods on Earth observation datasets, specifically with respect to label scarcity and geographical domain shifts. Much of the existing work either uses a fraction of the labeled data as unlabeled, or uses small unlabeled datasets. In our work, we imitate a more real world setting, where we augment labeled datasets with additional unlabeled imagery from the same satellite and domain. In addition, segmentation tasks have received less attention compared to image-level classification tasks for such techniques. In this paper, we test two popular approaches - SimCLR [3] (one of the recent SOTA self-supervised methods) and FixMatch [2] (one of the SOTA semi-supervised methods) at the task of segmentation using the DeepLabv3+ neural network architecture [46] (a recent SOTA semantic segmentation model). With extensive experimentation, we analyze the improvements that these methods can provide by using large unlabeled data sampled from the same satellites as the labeled data, and studying the impact of label scarcity and domain-shifts. We compare these methods against supervised finetuning from ImageNet pretrained weights - a strong baseline not only for consumer camera image based vision problems [47] [48], but also for remote sensing tasks [49]. We also show that the remote sensing tasks not only benefit from semi and self-supervised training individually, combining them allows us to reap even more benefits.

II. Methods

We evaluate our unlabeled data techniques on the task of semantic image segmentation (i.e. providing a class label to each pixel in the image). The next section describes the supervised DeepLabv3+ [46] segmentation model that is used across all our experiments, followed by an overview of the self-supervised representation learning model SimCLR [3] and semi-supervised learning model FixMatch [2].

A. Supervised DeepLabv3+

DeepLabv3+ [3] is a widely used neural network architecture for semantic image segmentation which has shown state-of-the-art results on several computer vision segmentation benchmark datasets (MSCOCO [50], PASCAL VOC [51]). In traditional encoder-decoder based segmentation models like SegNet [52], max-pooling and/or striding in encoder CNN reduces the spatial resolution of feature maps, which are then upsampled by deconvolution layers in the decoder. DeepLabv3+ uses a similar architecture but utilizes atrous convolutions [46] in its backbone encoder to learn deep features without losing spatial resolution. These feature maps are further processed by ASPP (atrous spatial pyramid pooling) to learn features at different scales. The decoder consists of an upsampling+convolutional layer with a skip connection to the corresponding encoder layer. Figure 1 illustrates the high level model architecture for Deeplab. Final feature maps are upsampled to the original input size and projected to output logits, and cross entropy loss is computed for each pixel against the ground truth pixel label. For more technical details, please refer [46].

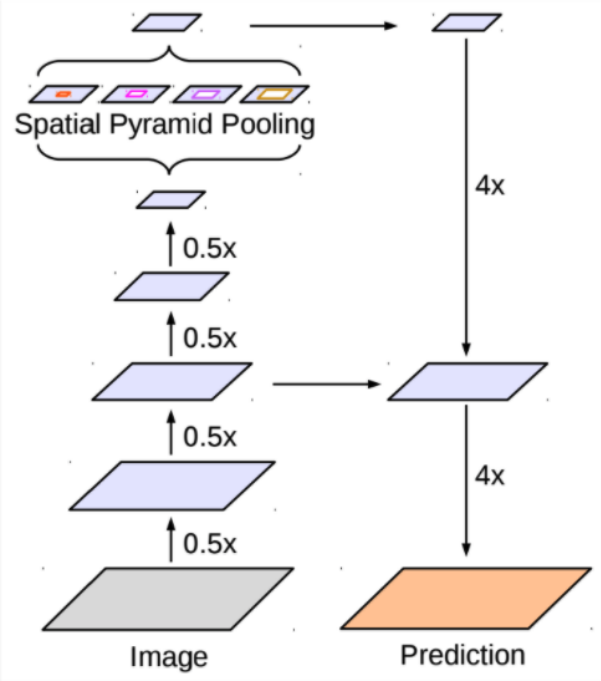


Fig. 1: Encoder-decoder architecture of DeepLabv3+(Courtesy of *Chen et al.* [46]). A CNN feature encoder is followed by an Atrous Spatial Pyramid Pooling layer to learn features at different scales. A decoder with one skip connection upsamples the learned features before predicting the segmentation class probabilities of each pixel.

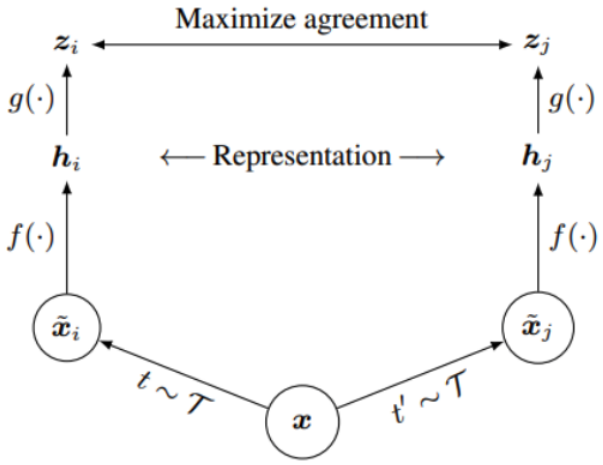


Fig. 2: SimCLR framework (Courtesy of [3]).

Augmentations: Input augmentation is a standard practice to prevent overfitting, regularize models and improve generalization [53] [54]. Many semi and self supervised methods rely on augmentations of input image as part of their learning process as well. Hence it was important to ensure that the augmentations used in the baseline model are thoroughly optimized, to avoid conflating the benefits of self/semi-supervised techniques with the augmentation improvements associated with them. The final set of optimized augmentations used for supervised DeepLabv3+ learning (and finetuning experiments) is described below. See the Appendix for more details and ablation on these augmentations.

- *Random crop with distortion*: We take a random crop of predefined input size from the dataset image. Crop distortion is set to $s = 0.5$ which means that each side of the image is randomly stretched between $(1 + s)$ and $(1 - s)$ times the original size. This augmentation makes the model robust to minor perturbations in resolution.
- *Rotation/flips*: Each cropped image undergoes random horizontal flip, random vertical flip and random 90-degree rotations.
- *Appearance augmentation*: Color jitter is applied with 0.5 probability. Color jitter randomly changes brightness, contrast, saturation and hue of an rgb image. Saturation and hue augmentations are, however, not meaningful for non-rgb images. For such images, we replace saturation and hue with per channel brightness and contrast augmentations. As opposed to standard contrast and brightness which apply augmentation on each channel by the same strength factor, per channel brightness and contrast apply augmentation on each channel with a separate randomly chosen factor. This generalized color jitter - which includes brightness, contrast, per channel brightness and per channel contrast - makes our model robust to variations coming from the satellite sensor, post-processing and domain of training data, and forces it to consider spatial structures when making predictions.

B. SimCLR Self-supervised model

SimCLR [3] is a self-supervised learning method, which maximizes the similarity between representations of differently augmented views of the same image by applying a contrastive loss [55]. Models pre-trained using SimCLR provide a much better network initialization than random weights for downstream tasks, especially when labeled data is scarce. A ResNet-50 [56] based SimCLR model [3] trained only on the images of ImageNet (without labels) achieves 48.3% top-1 accuracy when finetuned over 1% of labeled ImageNet data, surpassing the supervised baseline (with random initialization) which achieves only 25.4% top-1 accuracy.

The contrastive loss is defined per mini-batch of images. Specifically, given a randomly sampled mini-batch containing N examples, each image is augmented twice, resulting in $2N$ augmented images $\{x_k, k \in [1, ..., 2N]\}$. Each augmented image $\{x_i\}$ has a corresponding positive example $\{x_j\}$ and the remaining $2(N-1)$ images are considered negative examples. These images are encoded with the encoder network $f(.)$ to generate feature representations $\{h_k\}$ (see Figure 2). Note that we use the same encoder used in DeepLabv3+ in order to allow transfer learning. These representations are fed to a non-linear projection network $g(.)$ to generate $\{z_k\}$. For each positive pair (i, j) of the mini-batch, SimCLR computes the contrastive loss as follows:

$$l_{i,j} = -\log \frac{exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} exp(sim(z_i, z_k)/\tau)}$$

where $sim(z_i, z_j)$ denotes the cosine similarity between vectors z_i and z_j. $1_{[k \neq i]} \in \{0, 1\}$ is an indicator function evaluating to 1 *iff* $k \neq i$ and τ denotes the temperature parameter. Intuitively, this loss maximizes the similarity between z_i and z_j (coming from the augmented views of the same image) and minimizes their similarities to other $\{z_k, k \notin \{i, j\}\}$ coming from different images of the same minibatch.

Augmentations: To generate augmented views of the input image, SimCLR [3] uses random distorted crop, random horizontal flip, random color distortion (color jitter and color dropping), and random Gaussian blur. We use the same augmentation policy for our work. However for non-rgb inputs, we use general color jitter as explained in Section II-A. Further details on the specific augmentation parameters are provided in the Appendix.

To leverage SimCLR self-supervised training on downstream segmentation tasks, we initialize the encoder of the DeepLabv3+ segmentation network with a trained SimCLR encoder $f(.)$ and finetune on the labeled dataset. The remaining parts of DeepLabv3+ architecture like ASPP layers and decoder layers, which constitute a small fraction of total parameters, are trained from scratch with random initialization. For example in the Resnet-50 based DeepLabv3+ model, 24M encoder parameters are initialized with SimCLR pretrained weights and remaining 3M parameters are trained from scratch.

C. FixMatch Semi-supervised model

FixMatch [2] is a label propagation based Semi-supervised learning algorithm for image classification. It has shown state-of-the-art performance in extremely low-data regimes across a variety of standard image classification benchmarks, including 94.93% accuracy on CIFAR-10 [57] (a 10-way classification task) with 250 labeled images and 88.61% accuracy with 40 labeled images (i.e 4 labels per class). The key idea in FixMatch is to use unlabeled data for consistency regularization ([17], [58]): Each unlabeled image undergoes two types of augmentations: weak and strong. Weak augmentations refer to less deforming augmentations like horizontal/vertical flipping, color jitter etc and strong augmentations refer to more deforming augmentations like shear, rotation, solarize and posterize. Pseudo-labels are generated using the model predictions (with current weights) on weak augmentations of unlabeled images. Only predictions that pass a threshold for the predicted probability (implying a high model confidence) are used as Pseudo-labels. These pseudo-labels are then used as training labels for stronger augmentations of the same input image via addition of an additional cross-entropy loss, the semi-supervised loss. The total loss for the model is a weighted average of supervised loss, using labeled samples, and semi-supervised loss, using unlabeled samples. By only considering pseudo-labels which are above a certain confidence threshold, FixMatch avoids the careful balancing of these losses needed by MixMatch [15], ReMixMatch [16], [59], and [60]. The max probability across all classes is usually low during the beginning of the training and increases as the training progresses, producing a curriculum effect in training. This allows Fixmatch to be trained on datasets with as low as a single labeled sample per class.

We extend the FixMatch algorithm from image level classification to pixel wise segmentation tasks. Unlabeled images are passed through the DeepLabv3+ model to generate per pixel pseudo-labels and valid masks are computed via thresholding the probabilities. The additional step needed is to spatially transform the prediction (and the validity mask) on the weakly augmented image, to generate pseudo labels that are correctly aligned with the strongly augmented image (see Figure 3).

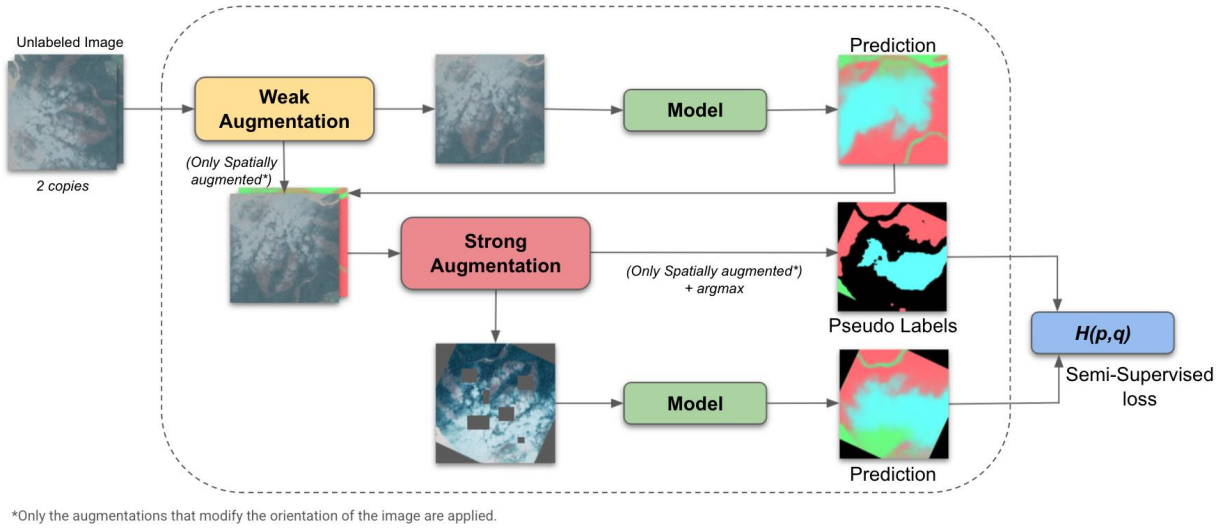


Fig. 3: FixMatch architecture with the semi-supervised loss adopted for semantic segmentation. The model generates predictions for weakly augmented unlabeled images. Weakly augmented images and their predictions undergo strong augmentations, with only the spatial augmentations being applied to predictions to generate pseudo labels. Semi-supervised cross-entropy loss is computed on this strongly augmented image and pseudo labels.

Augmentations: The same augmentations used for supervised DeepLabv3+ learning (rotation/flipping and color jitter, see Section II-A) are used as weak augmentations for FixMatch. For strong augmentations, we use the same configuration for the strong augmentations proposed by Cubuk *et al.* in Autoaugment [61]: out of a list of many augmentation functions (like Equalize, solarize, posterize, shear, rotation), two are chosen and applied in succession, followed by a cutout (removes a part of the image) – unlike FixMatch, which removes a single large rectangle for cutout, we remove multiple smaller rectangles. The complete details of the augmentations can be found in the Appendix.

III. Datasets

We selected datasets for model evaluation based on these criteria: (i) The task is a pixel wise segmentation task, (ii) We have the ability to sample unlabeled data outside of the dataset using the same satellite sources, and (iii) Location information is available per image so that the dataset can be split into geographically non-overlapping training/validation/test regions.

For each dataset, we create two types of train/validation/test splits in order to test for domain generalization: (i) IID (Independent and Identically Distributed) partition: Where the train/validation/test splits are created by IID sampling the entire dataset, and (ii) Domain-shifted partition: Where the train/validation/test splits are sampled from different geographical areas.

The next three subsections give specific details on each dataset that is evaluated in this work and Table I provides a summary view of these datasets.

A. Riverbed Segmentation Dataset

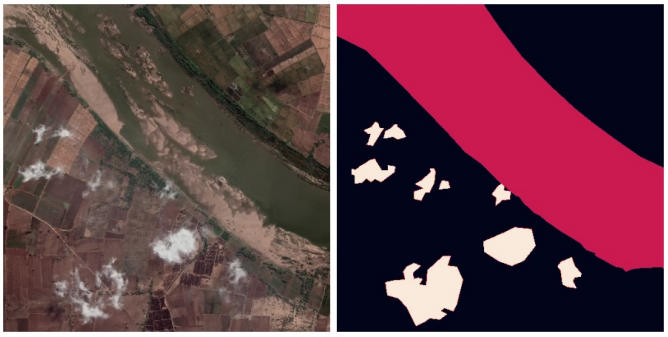


Fig. 4: Riverbed segmentation task example with the RGB image on the left and the segmentation groundtruth label (with riverbed and cloud) on the right. The riverbed includes both the water and the sandy portion of the river.

The segmentation task in this dataset is to demarcate riverbeds from RGB satellite images sampled from Google's satellite imagery (see Figure 4). It consists of 26,112 satellite RGB images of size 513x513 at 4m resolution sampled from riverbed regions of five Indian rivers - Ganga, Brahmaputra, Narmada, Krishna, and Kaveri (see Figure 5). Each image was labeled by human operators, who classified each pixel into one of the 3 classes – riverbed, clouds and background using a polygon drawing tool.

Fig. 5: (Left) The riverbed regions where satellite images were sampled for the labeled dataset. As shown in the samples, each riverbed has distinct characteristics. (Right) The regions where the unlabeled dataset was sampled.

The IID partition of this dataset was done in 60:20:20 ratio resulting in 15,708 train, 5,243 validation and 5,161 test images. The domain-shifted partition (Figure 11) sampled 15,504 training images from the Ganga and Brahmaputra riverbed region (black polygons in Figure 5(Left)), 7,568 validation images from the Narmada riverbed (red polygon in Figure 5(Left)), and 3,040 test images from the Krishna and Kaveri riverbed (green polygon in Figure 5(Left)). Each riverbed has distinct visual characteristics (both the water and sandy portion), thus making it harder for models trained on one region to generalize on the other.

The unlabeled data around riverbed regions of several rivers of India (polygons in Figure 5(Right)) was sampled from the same satellite imagery collection as the labeled data. In total, 183,668 satellite RGB images of size 1024x1024 were sampled at the same 4m resolution. The same unlabeled data was used for both semi-supervised and self-supervised experiments.

B. Chesapeake Land Cover

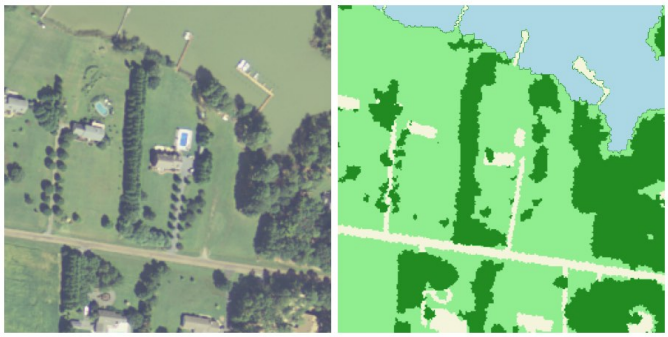


Fig. 6: Chesapeake Land Cover task example with the RGB image on the left and the segmentation groundtruth label on the right. [Blue: Water, Light green: Low vegetation, Dark green: Forest, Light yellow: Impervious land]

The publicly available Chesapeake Land Cover dataset [62] is sourced from the Chesapeake conservancy region in the United States. It covers an area of about 100,000 sq miles

	Riverbed Segmentation	Chesapeake Land Cover	Sen1Floods11
Source	Google Earth	NAIP	Sentinel-1
Bands	RGB	RGB+NIR	VV and VH
Resolution	4m	1m	10m
Image size	513x513	256x256	512x512
Label classes	3 (Riverbed, Clouds, Background)	4 (Water, Forest, Field, Impervious)	2 (Water, No-water)
Domain-shifted Partition			
# Train images	15,504	50,000	275
# Validation images	7568	2500	83
# Test images	3040	42,762	88
Train regions	Ganga, Brahmaputra	West Virginia	Ghana, India, Nigeria, Paraguay, USA
Validation regions	Narmada	Maryland	Somalia, Sri-Lanka, Bolivia
Test regions	Krishna, Kaveri	New York, Pennsylvania, Delaware	Mekong, Pakistan, Spain
IID Partition			
# Train images	15,708	50,000	252
# Validation images	5243	2500	89
# Test images	5161	14,750	90
Unlabeled Dataset for FixMatch			
Source	Google Earth	NAIP	Sentinel-1
Image size	1024x1024	256x256	512x512
# Images	183,668	1,999,840	4384
Sampling regions	All major rivers of India	East half of mainland USA	11 flood events from Sen1Floods11
Unlabeled Dataset for SimCLR			
Source	Same as Above	NAIP	Sentinel-1
Image size		256x256	512x512
# Images		1,999,675	4384(above) + 63k
Sampling regions		Mainland USA	Global

TABLE I: Summary of various parameters for the three datasets used for evaluation. The first section outlines information about the labeled dataset and image sources, followed by details about the domain-shifted partition, IID partition and unlabeled dataset sampling for FixMatch and SimCLR.

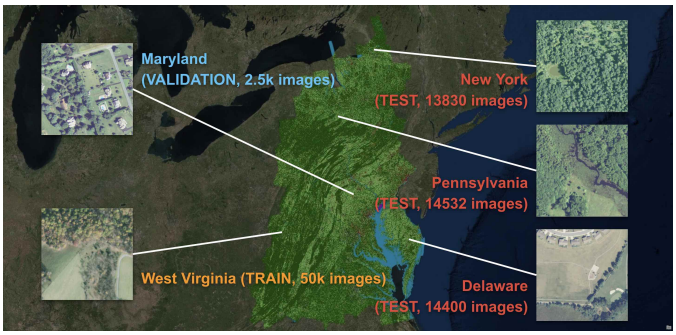


Fig. 7: The region where Chesapeake Land Cover data was sampled, along with the domain-shifted partition and sample images from each state.

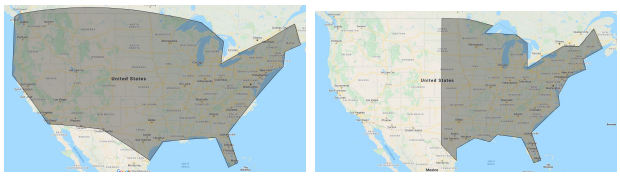


Fig. 8: (Left) Unlabeled data from the mainland USA was used to train the SimCLR model for Chesapeake dataset. (Right) Unlabeled data from the east half of the mainland USA was used to train FixMatch semi-supervised models.

that spans over 6 states: New York, Pennsylvania, Maryland, Delaware, Virginia and West Virginia (Figure 7). The dataset consists of multi-spectral imagery from the NAIP program [63] at 1m resolution. Labels were created using semi-automated feature extraction, rule-based clustering, followed by corrections from experts. The complete details can be found on their website [64]. Pixels are classified into 4 categories: (i) water, (ii) forest, (iii) field, (iv) impervious surfaces (Figure 6). For our experiments, we use the visual spectrum (RGB) and the near-infrared band (NIR) images of size 256x256 pre-sampled by the dataset creators. Each state has a pre-populated IID split of 50,000 train samples and 2,500 validation samples per state. The test split is provided as large tiff tiles ($\sim$ 6000x7500), which we slice into images of size 256x256.

For the IID partition, we use the data from the state of Maryland, consisting of 50K training examples, 2,500 validation examples and 14,750 testing examples(from 20 test tiles) from the same state. For the domain-shifted partition,

the training split consists of the training data from West Virginia (50k examples), validation split uses the validation data from Maryland (2.5k examples) and test split uses the test data from New York, Pennsylvania and Delaware (13,830+14,532+14,400=42,762 examples from a total of 60 tiles) (see Figure 12 for example imagery).

Unlabeled data for this task was sampled from publicly available NAIP imagery (RGB+NIR) on Google Earth Engine [65]. For SimCLR training, approximately 2 million 256x256 sized images were uniformly drawn from the entire region of the mainland United States from 2011 to 2014 (see Figure 8(left)). This dataset contains a rich variety of geological features that is useful in representation learning for the SimCLR model. A second sample of unlabeled data containing approximately 2 million images was drawn from the eastern half of the United States (see Figure 8(right)). We observed that using the unlabeled data from the east half of the US resulted in better performance for the Fixmatch semi-supervised model. Our hypothesis for this behavior is that semi-Supervised learning benefits more from unlabeled data that is closer in distribution to the eventual test data.

C. Sen1Floods11

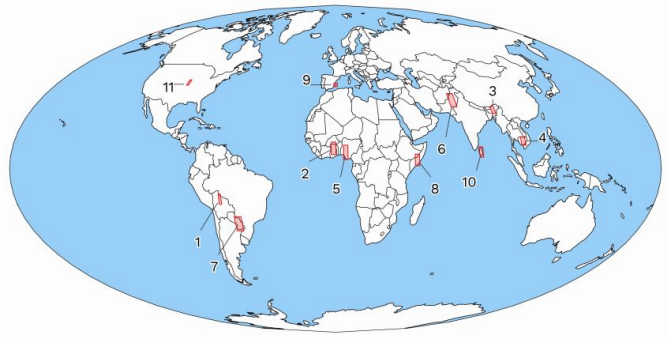


Fig. 9: The locations of 11 flood events where the flood data of Sen1Floods11 was sampled (Courtesy of [66]).

Fig. 10: Sentinel-1 image (left), groundtruth floodmap (middle) and Sentinel-2 image (right) from Sen1Floods11 dataset. Sentinel-2 image is only for visualization and not used during training. Labels include water (yellow), no-water (cyan), and masked out region (purple).

Sen1Floods11 [66] is a surface water dataset including raw Sentinel-1 imagery with classified permanent and flood water. A part of the dataset includes 4,831 Sentinel-1 images and aligned Sentinel-2 images sampled during 11 flood events from different regions of the world (see Figure 9). The dataset authors create a flood map using the Sentinel-2 image by thresholding over NDVI (Normalized Difference Vegetation Index) and MNDWI (Modified Normalized Difference Water Index). Out of these 4831 images, 446 were selected to be hand corrected by experts yielding high quality water labels. The final task is to perform flood segmentation using Sentinel-1 images only (see Figure 10).

We carry out all our experiments on expert-labeled 446 images because they have high quality labels (curated by experts) and to test self and semi-supervised methods on the real world constraint of an extremely small dataset. The authors provide an IID partition of this data containing 252 train samples, 89 validation samples and 90 test samples (excluding the Bolivia flood event). The same partition was used as the IID partition for our experiments. Our domain-shifted partition includes 275 train images from 5 events (Ghana, India, Nigeria, Paraguay, USA), 83 validation images from 3 events (Somalia, Sri-Lanka, Bolivia) and 88 test images from the remaining 3 events (Mekong, Pakistan, Spain) (see Figure 13).

The 4385 images that were not labeled by experts, are used as unlabeled data for FixMatch semi-supervised learning (weak labels are not used). However, this unlabeled dataset is quite small for SimCLR to learn rich features. So we sample additional 63K sentinel-1 images randomly distributed across the globe from the year 2019. This large dataset combined with 4385 unlabeled images from Sen1Floods11 is used to train the SimCLR model.

IV. Experiments and Results

The next section describes the overall experiment design, followed by a separate section each for dataset specific setup and its results.

A. Experiment Design

For each dataset, we compare the performance of following baseline models that only use labeled data:

- *Supervised(Random)* : Supervised training with randomly initialized DeepLabv3+ network.
- *Supervised(ImageNet)* : Supervised training where the DeepLabv3+ encoder is initialized with an Imagenet pretrained checkpoint (as done in [46]).

and the following models that use unlabeled data:

- *Supervised(SimCLR)* : Supervised training with the DeepLabv3+ encoder initialized from a SimCLR-pretrained checkpoint.
- *FixMatch(Random)* : FixMatch training with a randomly initialized DeepLabv3+ network.
- *FixMatch(ImageNet)* : FixMatch training with the DeepLabv3+ encoder initialized with an Imagenet pretrained checkpoint.
- *FixMatch(SimCLR)* : FixMatch training with the DeepLabv3+ encoder initialized from a SimCLR pretrained checkpoint, combining the benefits of both self and semi-supervised learning.

To understand how performance varies with label scarcity, we subsample the labeled part of the dataset at 1%, 10%

Fig. 11: Randomly sampled images from train (top), validation (middle) and test (bottom) split of domain-shifted partition of Riverbed segmentation dataset. It can be noticed that the riverbed differs in structure and color among the splits.

Fig. 12: Randomly selected images from train (top), validation (middle) and test (bottom) split of the domain-shifted partition of the Chesapeake Land Cover dataset.

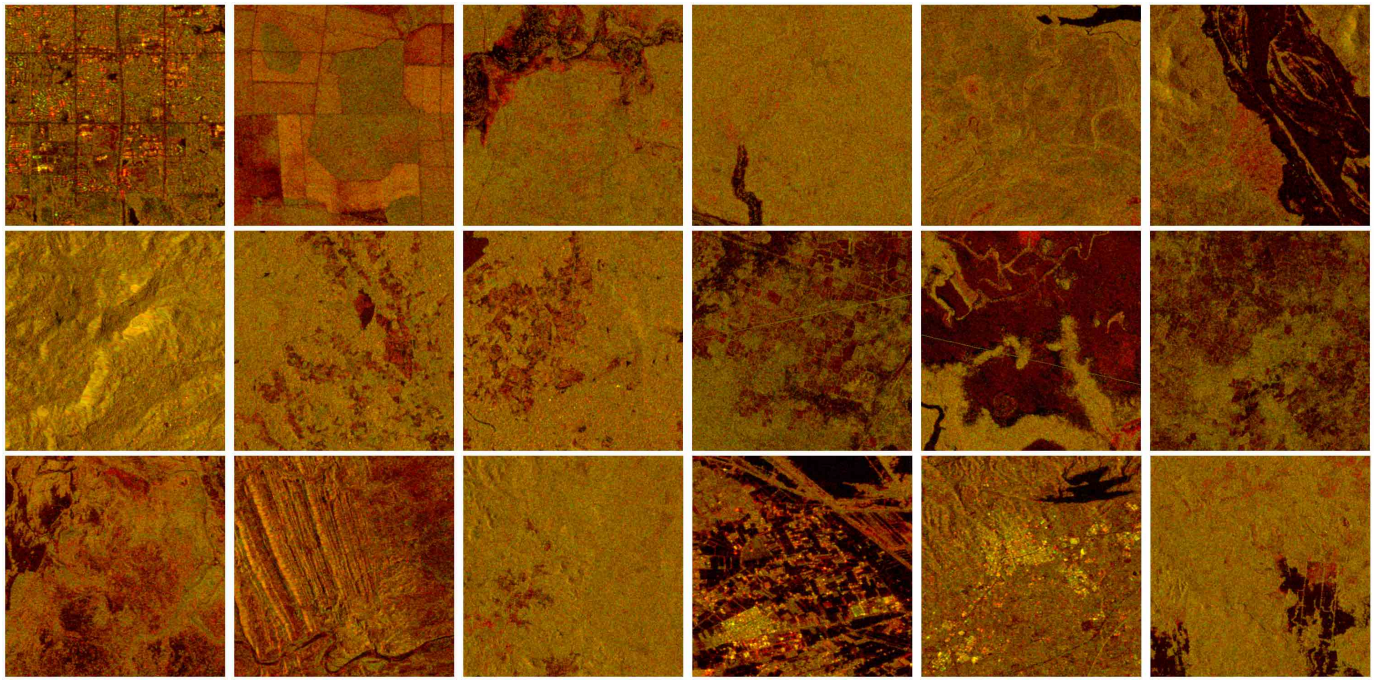


Fig. 13: Randomly selected Sentinel-1 images from train (top), validation (middle) and test (bottom) split of domain-shifted partition of Sen1Floods11 dataset.

and 100%. We sample five draws of 1% of the dataset and three draws for the 10% dataset – these samples are chosen only once and fixed for all experiments. However, for Sen1Floods11, 1% of the labeled data amounts to only 3 labeled images which is too small for supervised learning/finetuning. Hence we conduct our experiments only on 100% and five samples of 10% dataset. The results on these partial datasets are reported by mean and standard deviation of the chosen metric. We run the 100% experiment three times and report aggregated metrics similarly.

All models are trained and evaluated on both the IID and domain-shifted partition of each dataset. The domain-shifted partition often reflects real world deployment scenarios more accurately, and provides a good test bed to quantify improvements that come from using unlabeled data (as this data provides an opportunity for the network to learn to generalize better). For each dataset partition, the neural network weights are optimized on the training split, all hyperparameters selection, checkpoint selection and experimental analysis is done on the validation split. And once we have frozen our model and all hyper-parameters, we finally report results by running inference on the test split.

B. Training Details

Supervised: We use the Resnet-50 model as our backbone encoder with the first 7x7 convolution layer replaced with two 3x3 convolution layers (as done in DeepLabv3+ [46]). We use batch normalization [67], batch size of 64, and momentum optimizer with momentum set to 0.9. We also use the exponential moving average of model parameters which helps to stabilize the evaluation of the model throughout the training. Atrous rates are set to $(3, 6, 9)$ for all experiments. The skip connection between encoder and decoder is applied at the layer with downsampling factor of 4. Learning rate is decayed with a polynomial schedule (with power 0.9) from its initial value to zero. For each dataset, we set output stride (the ratio of input image size and the spatial resolution of intermediate feature maps) and input image size according to the dataset image size (see Table II for these details). Refer to Chen *et al.* [46] for more details on these hyper-parameter values.

SimCLR: The same Resnet-50 backbone that was used for the DeepLabv3+ based model is also used for SimCLR training to allow transferring weights. For each dataset, SimCLR is trained on its corresponding unlabeled dataset with a setup similar to the SimCLR paper [3]. We use the LARS optimizer with momentum 0.9, weight decay of 0.0001, batchsize of 4096, initial learning rate of 2.4 and cosine learning rate decay schedule. These parameters work well and we didn't try to optimize them per task, though it is possible that careful tuning for each downstream task may further increase its performance. For Riverbed segmentation, each large unlabeled image of size 1024x1024 is split into tiles of size 256x256 and then crop size of 224x224 is used to train for 160k steps. For Chesapeake, crop size of 224x224 is used to train for 200k steps. For Sen1Floods11, crop size 256x256 is used to train for 100k steps. After SimCLR pretraining is done on the unlabeled data, the last checkpoint is used to initialize the model encoder for *Supervised(SimCLR)* and *FixMatch(SimCLR)* .

FixMatch: For Semi-supervised experiments, a hyperparameter sweep is run over the confidence threshold with values $[0.75, 0.8, 0.85, 0.9, 0.95]$. A threshold of 0.9 worked best for pseudo label generation across all datasets. The train batch consists of 32 labeled and 32 unlabeled images in

	Riverbed Segmentation	Chesapeake Land Cover	Sen1Floods11
Encoder	ResNet-50	ResNet-50	ResNet-50
Batch size	64	64	64
Atrous rates	(3,6,9)	(3,6,9)	(3,6,9)
Optimizer	Momentum(m=0.9)	Momentum(m=0.9)	Momentum(m=0.9)
Decoder output stride	4	4	4
Train crop size	321x321	241x241	321x321
Output stride	16	8	16
Evaluation Metric	Mean IoU of Riverbed class	Average of mean IoU of all classes	Mean IoU of Water class
Number of train steps	60k for Supervised 120k for FixMatch	100k for Supervised 120k for FixMatch	20k for Supervised 120k for FixMatch
Partial datasets	1 sample of 100% 3 samples of 10% 5 samples of 1%	1 sample of 100% 3 samples of 10% 5 samples of 1%	1 sample of 100% 5 samples of 10%
Learning rate weight decay	Decided by a sweep over learning rate values [0.3, 0.1, 0.03, 0.01, 0.003, 0.001] and weight decay values [0.001, 0.0001, 0.0003, 0.00001, 0.000001] for each model on each dataset.		

TABLE II: Training details of Supervised and FixMatch learning methods for all three datasets.

all cases. The total loss is the sum of supervised loss and semi-supervised loss, optimized using Momentum optimizer with momentum 0.9 and polynomial schedule (with power 0.9). Batch normalization parameters are updated only on the labeled and the strongly augmented images. Table II summarizes the other dataset specific hyperparameter choices.

Other details: For each dataset and for each model (except SimCLR pretraining), we do a grid search to find out the optimal values of learning rate from [3e-1, 1e-1, 3e-2, 1e-2, 3e-3, 1e-3] and weight decay value from [1e-3, 1e-4, 3e-4, 1e-5, 1e-6] because their optimal values vary for each dataset and model. The grid search is done on the 100% sample of the domain-shifted partition and the same parameters are used for the 1% and 10% data experiments as well as all IID partition experiments. The exact hyper-parameters chosen per dataset are documented in the Appendix. For each model, the checkpoint with the best evaluation metric on the validation set is chosen for testing. The Resnet-50 backbone used in all models is fully convolutional [68], which allows us to transfer weights from models trained on ImageNet and SimCLR using different image sizes, as well as evaluate on image sizes that are different from the training image sizes. For datasets with non RGB inputs, whenever we initialize the model from an ImageNet checkpoint, the first conv layer weights for each non-rgb channel are initialized with the mean of RGB weights of the first conv layer of the ImageNet checkpoint (averaged across channels).

C. Results: Riverbed Segmentation

We use the pixelwise Intersection over Union(IoU) of riverbed class to validate our model performance on this dataset. Even though this dataset has another labeled class (i.e. clouds), these annotations are ambiguous even for human annotators as some images contain hazy clouds. In addition, some of the validation and test splits had extremely small number of cloud pixels (due to their geographical location), which made metrics on cloud class unreliable. Hence, we run our evaluation only on the riverbed class metric.

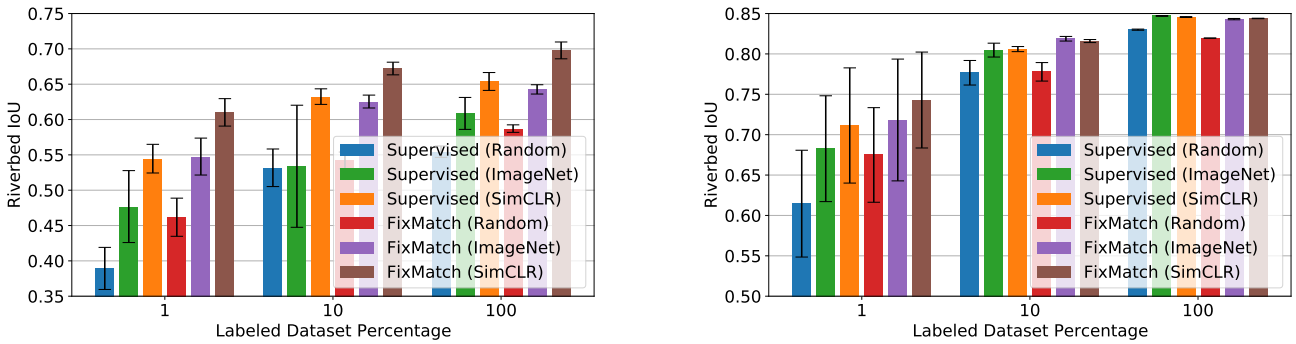
1) Domain-shifted partition: Figure 14(a) shows riverbed IoU on the domain-shifted partition for *Supervised(SimCLR)* , *FixMatch(Random)* and *FixMatch(ImageNet)* against the baselines *Supervised(Random)* and *Supervised(ImageNet)* . Because of in-domain pretraining, *Supervised(SimCLR)* provides better model initialization and consistently outperforms *Supervised(ImageNet)* (7%, 10% and 4.5% absolute riverbed IoU improvement in 1%, 10% and 100% labeled dataset respectively). For semi-supervised learning, *FixMatch(Random)* provides an absolute riverbed IoU improvement over *Supervised(Random)* of 7% in 1% dataset, going down to 3% in 100% dataset. *FixMatch(ImageNet)* shows a consistent increase in the absolute riverbed IoU of 7%, 9.1% and 3.4% over *Supervised(ImageNet)* in 1%, 10% and 100% of the labeled dataset respectively.

Our best model *FixMatch(SimCLR)* which combines the benefits of SimCLR pretraining and FixMatch semi-supervised training, significantly outperforms all other models: *FixMatch(SimCLR)* trained only on 1% labeled dataset has a riverbed IoU of 61.0%, that matches the commonly used *Supervised(ImageNet)* baseline trained on full 100% dataset (60.9% riverbed IoU). This shows that combining self and semi-supervised models can make deep learning extremely data efficient for the riverbed segmentation task.

2) IID partition: Figure 14(b) compares the models on the IID partition of the riverbed dataset in a similar way as the domain-shifted partition. Initializing the model with ImageNet or SimCLR pretrained weights always helps against random initialization for both Supervised and Semi-supervised models. When labeled data is available in enough quantity (i.e 10% and 100%), in this easier problem setup, model performance saturates and there isn't much gain to be seen in using self and semi-supervised techniques. But for the 1% labeled data sample, *Supervised(SimCLR)* , *FixMatch(ImageNet)*

Dataset	Model	Domain-shifted Partition			IID Partition		
		1%	10%	100%	1%	10%	100%
Riverbed Segmentation (Riverbed mIoU)	*Supervised(Random)*	38.93 ± 3.0	53.17 ± 2.7	55.45 ± 0.8	61.46 ± 6.6	77.67 ± 1.5	83.00 ± 0.1
	Supervised(ImageNet)	47.68 ± 5.1	53.39 ± 8.6	60.87 ± 2.3	68.26 ± 6.6	80.47 ± 0.9	$\mathbf{84.70 \pm 0.0}$
	Supervised(SimCLR)	54.46 ± 2.0	63.24 ± 1.1	65.39 ± 1.3	71.14 ± 7.1	80.60 ± 0.3	84.57 ± 0.1
	FixMatch(Random)	46.17 ± 2.7	54.21 ± 1.2	58.72 ± 0.5	67.49 ± 5.9	77.78 ± 1.1	81.95 ± 0.0
	FixMatch(ImageNet)	54.76 ± 2.6	62.54 ± 0.9	64.27 ± 0.7	71.82 ± 7.5	$\mathbf{81.88 \pm 0.3}$	84.31 ± 0.1
	FixMatch(SimCLR)	$\mathbf{61.01 \pm 1.9}$	$\mathbf{67.22 \pm 0.9}$	$\mathbf{69.78 \pm 1.2}$	$\mathbf{74.29 \pm 5.9}$	81.59 ± 0.2	84.39 ± 0.0
Chesapeake Land Cover (Avg. mIoU)	*Supervised(Random)*	63.63 ± 1.9	72.06 ± 1.0	76.89 ± 0.2	79.06 ± 2.3	82.18 ± 0.8	83.41 ± 0.2
	Supervised(ImageNet)	62.48 ± 5.0	73.65 ± 1.0	78.31 ± 0.3	80.60 ± 1.1	82.60 ± 0.6	83.65 ± 0.1
	Supervised(SimCLR)	65.89 ± 2.0	74.05 ± 0.8	78.51 ± 0.4	80.03 ± 1.5	82.43 ± 0.7	$\mathbf{84.25 \pm 0.3}$
	FixMatch(Random)	67.72 ± 3.3	72.23 ± 1.6	76.34 ± 0.3	80.71 ± 2.0	81.81 ± 1.4	82.54 ± 0.2
	FixMatch(ImageNet)	63.73 ± 5.7	75.38 ± 1.7	78.64 ± 0.3	80.86 ± 1.3	82.55 ± 0.4	83.80 ± 0.1
	FixMatch(SimCLR)	$\mathbf{69.15 \pm 2.5}$	$\mathbf{76.26 \pm 2.1}$	$\mathbf{79.01 \pm 0.0}$	$\mathbf{81.76 \pm 0.7}$	$\mathbf{82.86 \pm 0.4}$	83.64 ± 0.1
Sen1Floods11 (Water mIoU)	*Supervised(Random)*		60.11 ± 10.3	67.77 ± 0.8		53.69 ± 9.7	63.51 ± 1.2
	Supervised(ImageNet)		61.91 ± 7.6	67.43 ± 0.5		56.68 ± 1.9	64.79 ± 0.5
	Supervised(SimCLR)		64.29 ± 3.9	70.95 ± 0.1		$\mathbf{59.56 \pm 3.0}$	$\mathbf{66.92 \pm 0.5}$
	FixMatch(Random)		66.05 ± 3.5	$\mathbf{71.56 \pm 0.6}$		56.24 ± 5.2	62.61 ± 0.6
	FixMatch(ImageNet)		$\mathbf{66.46 \pm 3.6}$	71.07 ± 0.3		59.17 ± 3.4	64.66 ± 0.4
	FixMatch(SimCLR)		64.08 ± 6.3	70.63 ± 0.5		59.47 ± 1.6	61.35 ± 2.4

TABLE III: Results on test set of the domain-shifted and IID partitions of all three datasets. The numbers show the aggregated mean and standard deviation of metrics.



(a) Domain-shifted partition

(b) IID partition

Fig. 14: Results on the test split of River segmentation dataset. Note that the y-axis range is different for the two dataset partitions as the performance range is significantly different between the two.

and *FixMatch(SimCLR)* provide a 2.9%, 3.6% and 6% absolute riverbed IoU improvement respectively over the *Supervised(ImageNet)* baseline. It is interesting to note that on 100% dataset, *FixMatch(Random)* performance degrades by 1.1% absolute rivedbed IoU from the *Supervised(Random)* baseline. Our hypothesis is that the FixMatch model might be losing some accuracy on the labeled training data (which is sampled from the same region as the test data) while learning to generalize better on other regions of the unlabeled dataset.

D. Results: Chesapeake Land Cover

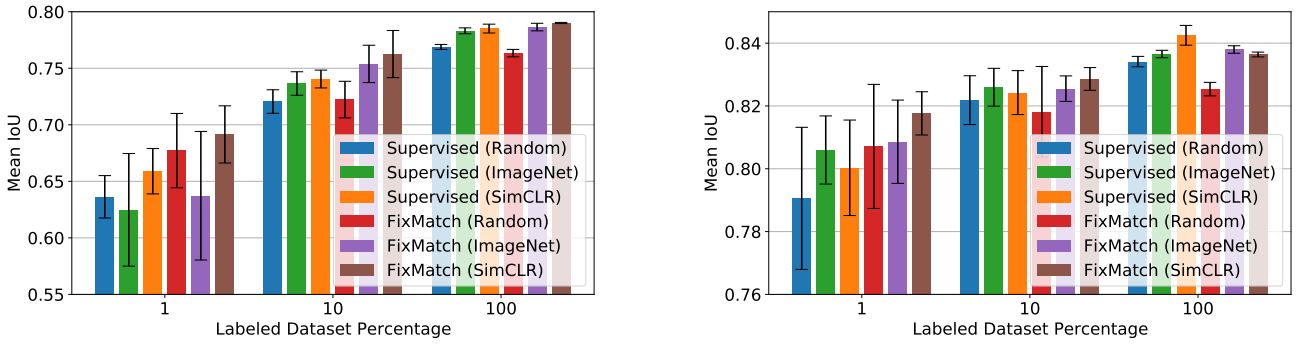
We use mean IoU (average of the pixelwise IoU of all 4 classes) to validate our model performance on this dataset as done by [62].

1) Domain-Shifted partition: Figure 15(a) shows the mean IoU on the domain-shifted partition of this dataset for all models. *Supervised(SimCLR)* performs slightly better than *Supervised(ImageNet)* baseline for 10% and 100% dataset. However, the benefit of SimCLR pretraining is more pronounced in the 1% labeled dataset where it outperforms *Supervised(ImageNet)* by an absolute margin of 3.4% in mean IoU and *Supervised(Random)* by 2.3% in mean IoU (9.1% and 6.2% relative reduction in error rate).

FixMatch(ImageNet) provides an absolute improvement of by 1.25% and 1.73% in 1% and 10% dataset respectively over *Supervised(ImageNet)* . On the 100% dataset however, there are no significant gains in using *FixMatch(ImageNet)* over the baseline *Supervised(ImageNet)* . *FixMatch(Random)* performs better than *Supervised(Random)* by 4.1% for the 1% dataset but is similar in performances for 10% and 100% datasets.

The combined *FixMatch(SimCLR)* model outperforms all other models, and provides an absolute mean IoU improvement of 6.7%, 2.6%, 0.7% over *Supervised(ImageNet)* for 1%, 10% and 100% datasets respectively.

2) IID partition: Figure 15(b) compares all models on the IID partition of this dataset. *Supervised(SimCLR)* outper-



(a) Domain-shifted partition

(b) IID partition

Fig. 15: Results on the test split of Chesapeake dataset. Note that the y-axis range is different for the two dataset partitions as the performance range is significantly different between the two.

forms *Supervised(ImageNet)* in 100% by a small margin of 0.6% absolute mean IoU, but lags behind the *Supervised(ImageNet)* baseline by 0.6% in the 1% dataset. *FixMatch(ImageNet)* and *FixMatch(SimCLR)* models, on the other hand, match the *Supervised(ImageNet)* baseline in 100% dataset and outperform them in the 1% dataset by a small margin (0.3% and 1.2% absolute mean IoU improvement respectively).

Overall from Figure 15(b), we do not see much gains in using unlabeled data techniques over the supervised baselines in this partition. The change in absolute performance of *Supervised(ImageNet)* between 1% and 100% of labeled data is also fairly small (from 80.6% to 83.65%). Our hypothesis is that in the IID setting, this task is easier than the domain-shifted partition, and we have enough labels needed to get high accuracy even at the 1% sample size (which corresponds to 500 labeled images). In addition, in our qualitative review, we found that the ground truth labels on Chesapeake have some errors, and it's possible that at 100% sample size, we might be at the limit of ground truth accuracy and cannot see further model improvements.

E. Results: Sen1Floods11

The authors of this dataset used the mean of imagewise IoU of the water class as their evaluation metric [66]. To keep our metric consistent across other datasets above, we used the mean IoU (computed pixelwise) of the water class to validate our model performance on this dataset. Unlike the imagewise IoU, this metric is more robust and is not sensitive to the choice of the tile size and proportion of water pixels in a particular image.

1) Domain-Shifted partition: Figure 16(a) compares the performance of all models (using IoU of water class) on the domain-shifted partition of the Sen1Floods11 dataset. *Supervised(SimCLR)* provided modest gains over the *Supervised(ImageNet)* baseline in 10% and 100% dataset of 2.4% and 3.5% absolute IoU respectively.

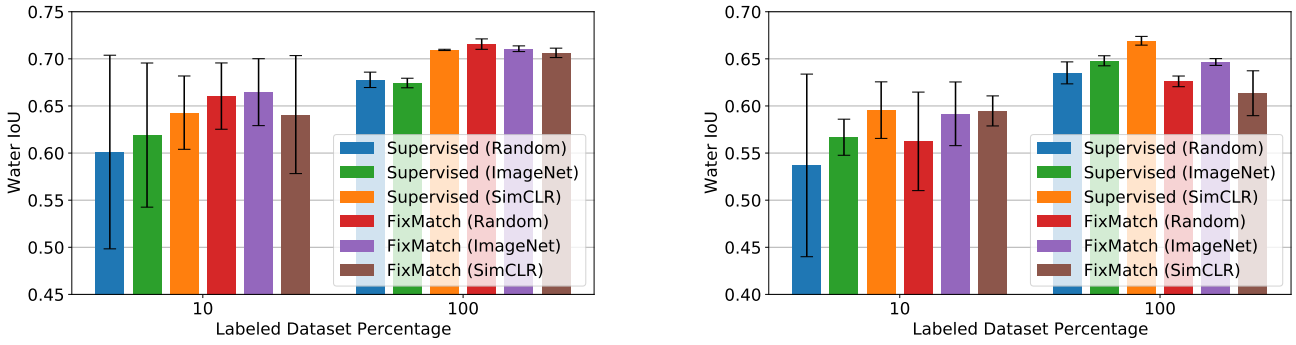
FixMatch(Random) outperforms *Supervised(Random)* by 6.0% and 3.8 % absolute IoU on the 10% and 100% dataset respectively. With *FixMatch(ImageNet)* , we see a gain of 4.55% and 3.6% absolute IoU over *Supervised(ImageNet)* on the 10% and 100% dataset respectively. *FixMatch(SimCLR)* shows an improvement of 2.2%, 3.2% over *Supervised(ImageNet)* for 10% and 100% datasets respectively (matching *Supervised(SimCLR)* in each case, but not providing any more gains than that).

2) IID partition: Figure 16(b) compares the performance of all models on the IID partition of Sen1Floods11 dataset. Similar to the domain shifted partition, *Supervised(SimCLR)* provided modest gains over the *Supervised(ImageNet)* baseline in 10% and 100% dataset of 2.9% and 2.1% absolute IoU respectively.

Semi-supervised learning shows mixed results on this partition. *FixMatch(Random)* outperforms *Supervised(Random)* by 2.5% on 10% dataset but shows a small degradation of 0.9% for the 100% dataset. The other variants of FixMatch (i.e. *FixMatch(ImageNet)* and *FixMatch(SimCLR)*) also show mixed performance. *FixMatch(ImageNet)* improves performance over *Supervised(ImageNet)* by 2.5% in 10% dataset and matches it for the 100% dataset. *FixMatch(SimCLR)* matches *Supervised(SimCLR)* in performance for 10% dataset but decreases by 5.4% for the 100% dataset.

V. Conclusion

We evaluated the efficacy of using self and semi supervised methods on three different remote sensing tasks, especially in two real world settings: geographical domain shifts and small amount of labeled data. One consistent trend observed across all datasets was that in settings with small number of labels, using both SimCLR and FixMatch techniques (and combining the two as well) improved model performance significantly. The gains were even more pronounced when there are geographical domain shifts at test time, signifying that these techniques can leverage unlabeled data to improve model generalization. Such gains are especially useful in real world deployments, where the test distribution is uncertain and often contains such geographical domain shifts. In easier scenarios where the performance of the purely supervised



(a) Domain-shifted partition

(b) IID partition

Fig. 16: Results on the test split of Sen1Floods11 dataset. Note that the y-axis range is different for the two dataset partitions as the performance range is slightly different between the two.

model saturates, signifying that there is enough labeled data for good generalization (e.g. the IID partition with 100% labels on Chesapeake), such techniques do not provide significant benefits (and FixMatch can sometimes lower the performance over supervised baselines by small amounts). However, in real world deployments where some amount of geographical domain shifts is expected, using the self and semi supervised techniques evaluated in this paper improve model generalization significantly.

ACKNOWLEDGMENTS

We would like to thank Vishal Batchu for insightful discussions, Aparna Taneja for the help in generating the Sentinel-1 unlabeled data, Valerie Pasquarella for her very thorough review and feedback on this paper, and Oliver Guinan, Noel Gorelick and John Platt for reading drafts of this paper and providing feedback.

REFERENCES

[1] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018. [Online]. Available: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

[2] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 596–608.

[3] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[4] L. Zhong, L. Hu, and H. Zhou, "Deep learning based multi-temporal crop classification," *Remote Sensing of Environment*, vol. 221, pp. 430–443, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425718305418

[5] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros, "Context encoders: Feature learning by inpainting," in *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *ECCV*, 2016.

[7] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," *arXiv preprint arXiv:1803.07728*, 2018.

[8] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *European conference on computer vision*. Springer, 2016, pp. 69–84.

[9] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[10] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent - a new approach to self-supervised learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, 2020, pp. 21 271–21 284.

[11] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," 2020.

[12] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve model robustness and uncertainty," *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[13] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17. MIT Press, 2005.

[14] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Mutual exclusivity loss for semi-supervised deep learning," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 1908–1912.

[15] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[16] D. Berthelot, N. Carlini, E. D. Cubuk, A. Kurakin, K. Sohn, H. Zhang, and C. Raffel, "Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring," *arXiv preprint arXiv:1911.09785*, 2019.

[17] J. Jeong, S. Lee, J. Kim, and N. Kwak, "Consistency-based semi-supervised learning for object detection," *Advances in neural information processing systems*, vol. 32, pp. 10 759–10 768, 2019.

[18] J. Li, R. Socher, and S. C. H. Hoi, "Dividemix: Learning with noisy labels as semi-supervised learning," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[19] H. Dong, W. Ma, Y. Wu, J. Zhang, and L. Jiao, "Self-supervised representation learning for remote sensing image change detection based on temporal prediction," *Remote Sensing*, vol. 12, no. 11, p. 1868, 2020.

[20] W. Chen, X. Zheng, and X. Lu, "Hyperspectral image super-resolution with self-supervised spectral-spatial residual network," *Remote Sensing*, vol. 13, no. 7, p. 1260, 2021.

[21] Z. Zhao, Z. Luo, J. Li, C. Chen, and Y. Piao, "When self-supervised learning meets scene classification: Remote sensing scene classification based on a multitask learning framework," *Remote Sensing*, vol. 12, no. 20, p. 3276, 2020.

[22] S. Vincenzi, A. Porrello, P. Buzzega, M. Cipriano, F. Pietro, C. Roberto, I. Carla, C. Annamaria, and S. Calderara, "The color out of space: learning self-supervised representations for earth observation imagery," in *25th International Conference on Pattern Recognition*, 2020.

[23] D. Lin, K. Fu, Y. Wang, G. Xu, and X. Sun, "Marta gans: Unsupervised representation learning for remote sensing image classification," *IEEE*

Geoscience and Remote Sensing Letters, vol. 14, no. 11, pp. 2092–2096, 2017.

[24] X. Lu, X. Zheng, and Y. Yuan, "Remote sensing scene classification by unsupervised representation learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 9, pp. 5148–5157, 2017.

[25] N. Jean, S. Wang, A. Samar, G. Azzari, D. Lobell, and S. Ermon, "Tile2vec: Unsupervised representation learning for spatially distributed data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3967–3974.

[26] H. Jung, Y. Oh, S. Jeong, C. Lee, and T. Jeon, "Contrastive self-supervised learning with smoothed representation for remote sensing," *IEEE Geoscience and Remote Sensing Letters*, 2021.

[27] V. Stojnic and V. Risojevic, "Self-supervised learning of remote sensing scene representations using contrastive multiview coding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1182–1191.

[28] K. Ayush, B. Uzkent, C. Meng, K. Tanmay, M. Burke, D. Lobell, and S. Ermon, "Geography-aware self-supervised learning," *arXiv preprint arXiv:2011.09980*, 2020.

[29] J. Kang, R. Fernandez-Beltran, P. Duan, S. Liu, and A. J. Plaza, "Deep unsupervised embedding for remotely sensed images based on spatially augmented momentum contrast," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 3, pp. 2598–2610, 2020.

[30] M. Leenstra, D. Marcos, F. Bovolo, and D. Tuia, "Self-supervised pre-training enhances change detection in sentinel-2 imagery," in *Pattern Recognition. ICPR International Workshops and Challenges, 2021, Proceedings*, ser. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 12667, 2021, pp. 578–590.

[31] W. Li, H. Chen, and Z. Shi, "Semantic segmentation of remote sensing images with self-supervised multitask representation learning," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 6438–6450, 2021.

[32] G. Camps-Valls, T. V. B. Marsheva, and D. Zhou, "Semi-supervised graph-based hyperspectral image classification," *IEEE transactions on Geoscience and Remote Sensing*, vol. 45, no. 10, pp. 3044–3054, 2007.

[33] D. Tuia and G. Camps-Valls, "Semisupervised remote sensing image classification with cluster kernels," *IEEE Geoscience and Remote Sensing Letters*, vol. 6, no. 2, pp. 224–228, 2009.

[34] L. Bruzzone, M. Chi, and M. Marconcini, "A novel transductive svm for semisupervised classification of remote-sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 11, pp. 3363–3373, 2006.

[35] U. Maulik and D. Chakraborty, "A self-trained ensemble with semisupervised svm: An application to pixel classification of remote sensing imagery," *Pattern Recognition*, vol. 44, no. 3, pp. 615–623, 2011.

[36] ——, "Learning with transductive svm for semisupervised pixel classification of remote sensing imagery," *ISPRS journal of photogrammetry and remote sensing*, vol. 77, pp. 66–78, 2013.

[37] B. Chaudhuri, B. Demir, S. Chaudhuri, and L. Bruzzone, "Multilabel remote sensing image retrieval using a semisupervised graph-theoretic method," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 2, pp. 1144–1158, 2017.

[38] J. Hu, D. Hong, and X. X. Zhu, "Mima: Mapper-induced manifold alignment for semi-supervised fusion of optical image and polarimetric sar data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 11, pp. 9025–9040, 2019.

[39] D. Hong, N. Yokoya, J. Chanussot, J. Xu, and X. X. Zhu, "Learning to propagate labels on graphs: An iterative multitask regression framework for semi-supervised hyperspectral dimensionality reduction," *ISPRS journal of photogrammetry and remote sensing*, vol. 158, pp. 35–49, 2019.

[40] H. Wu and S. Prasad, "Semi-supervised dimensionality reduction of hyperspectral imagery using pseudo-labels," *Pattern Recognition*, vol. 74, pp. 212–224, 2018.

[41] X. Tang, C. Liu, J. Ma, X. Zhang, F. Liu, and L. Jiao, "Large-scale remote sensing image retrieval based on semi-supervised adversarial hashing," *Remote Sensing*, vol. 11, no. 17, p. 2055, 2019.

[42] X. Sun, A. Shi, H. Huang, and H. Mayer, "Bas4net: Boundary-aware semi-supervised semantic segmentation network for very high resolution remote sensing images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 5398–5413, 2020.

[43] B. G. Weinstein, S. Marconi, S. Bohlman, A. Zare, and E. White, "Individual tree-crown detection in rgb imagery using semi-supervised deep learning neural networks," *Remote Sensing*, vol. 11, no. 11, p. 1309, 2019.

[44] W. Han, R. Feng, L. Wang, and Y. Cheng, "A semi-supervised generative framework with deep learning features for high-resolution remote sensing image scene classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 145, pp. 23–43, 2018.

[45] D. Hong, N. Yokoya, G.-S. Xia, J. Chanussot, and X. X. Zhu, "X-modalnet: A semi-supervised deep cross-modal network for classification of remote sensing data," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 167, pp. 12–23, 2020.

[46] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[47] M. Huh, P. Agrawal, and A. A. Efros, "What makes imagenet good for transfer learning?" *CoRR*, vol. abs/1608.08614, 2016. [Online]. Available: http://arxiv.org/abs/1608.08614

[48] S. Kornblith, J. Shlens, and Q. V. Le, "Do better imagenet models transfer better?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2661–2671.

[49] R. Pires de Lima and K. Marfurt, "Convolutional neural network for remote-sensing scene classification: Transfer learning analysis," *Remote Sensing*, vol. 12, no. 1, p. 86, 2020.

[50] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[51] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[52] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[53] J. Wang and L. Perez, "The effectiveness of data augmentation in image classification using deep learning," *Convolutional Neural Networks Vis. Recognit*, vol. 11, pp. 1–8, 2017.

[54] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[55] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742.

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

[57] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, Ontario, Tech. Rep. 0, 2009.

[58] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29, 2016.

[59] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[60] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. Goodfellow, "Realistic evaluation of deep semi-supervised learning algorithms," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[61] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 113–123.

[62] C. Robinson, L. Hou, K. Malkin, R. Soobitsky, J. Czawlytko, B. Dilkina, and N. Jojic, "Large scale high-resolution land cover mapping with multi-resolution data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 726–12 735.

[63] "Naip imagery," https://www.fsa.usda.gov/programs-and-services/aerial-photography/imagery-programs/naip-imagery/, accessed: 2021-08-13.

[64] "Land cover data project 2013/2014," Aug 2020, accessed: 2021-02-11. [Online]. Available: https:

//www.chesapeakeconservancy.org/conservation-innovation-center/high-resolution-data/land-cover-data-project/
[65] "Earth engine naip imagery," https://developers.google.com/earth-engine/datasets/catalog/USDA_NAIP_DOQQ, accessed: 2021-08-13.
[66] D. Bonafilia, B. Tellman, T. Anderson, and E. Issenberg, "Sen1floods11: A georeferenced dataset to train and test deep learning flood algorithms for sentinel-1," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.
[67] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.
[68] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

APPENDIX A
HYPERPARAMETERS

Learning rate and weight decay significantly affect the model performance, especially for domain-shifted partition. Hence, we perform a hyperparameter grid search and arrive at the following parameters for each model on each dataset. All these hyper-parameters are selected using validation metric score of the 100% dataset experiment of domain-shifted partition.

A. Hyperparameters for Riverbed Segmentation Dataset

Model	Learning rate	Weight decay
Supervised(Random)	0.01	0.001
Supervised(ImageNet)	0.03	0.0001
Supervised(SimCLR)	0.003	0.0001
FixMatch(Random)	0.03	0.0001
FixMatch(ImageNet)	0.01	0.00001
FixMatch(SimCLR)	0.003	0.0001

B. Hyperparameters for Chesapeake Land Cover Dataset

Model	Learning rate	Weight decay
Supervised(Random)	0.03	0.0001
Supervised(ImageNet)	0.01	0.0003
Supervised(SimCLR)	0.03	0.0003
FixMatch(Random)	0.03	0.0003
FixMatch(ImageNet)	0.03	0.0001
FixMatch(SimCLR)	0.01	0.0001

C. Hyperparameters for Sen1Floods11 Dataset

Model	Learning rate	Weight decay
Supervised(Random)	0.01	0.0001
Supervised(ImageNet)	0.001	0.000001
Supervised(SimCLR)	0.01	0.0001
FixMatch(Random)	0.03	0.0001
FixMatch(ImageNet)	0.03	0.0001
FixMatch(SimCLR)	0.1	0.0001

APPENDIX B
AUGMENTATIONS

A. Supervised Training Augmentations

We use the following augmentation policy for supervised training of the DeepLabv3+ models: *Supervised(Random)* , *Supervised(ImageNet)* and *Supervised(SimCLR)* . We also use this same policy for weak augmentation in all FixMatch models.

- Random distorted crop with 0.5 distortion.
- Random horizontal flip
- Random vertical flip
- Random rotation by a multiple of 90 degrees
- Color jitter with 0.5 probability:
  - `ColorJitterRGB` with strength 0.4 for RGB images
  - `ColorJitterGeneral` with strength 0.4 for non-RGB images

Refer to Listing-1 below for the definition of `ColorJitterRGB` and `ColorJitterGeneral`.

B. Ablation on Supervised Learning Augmentation

Our augmentation policy for *Supervised(ImageNet)* has 2 hyperparameters - distortion of crop and strength of color jitter. On an initial ablation done on Sen1Floods11 dataset, the crop distortion of 0.5 and color jitter strength of 0.4 worked the best. With this defined augmentation policy, we carried out a formal ablation on the effect of augmentation on *Supervised(ImageNet)* baseline model performance. For domain-shifted partition of Riverbed segmentation dataset and Sen1Floods11 dataset, we trained two more models: (1) *Supervised(ImageNet)* with color jitter removed (i.e. no appearance augmentation) and (2) *Supervised(ImageNet)* with all augmentation removed (no geometric and appearance augmentation). Hyperparameters were chosen again for these new models with the same hyperparameter sweeps as discussed in Section IV-B. The results on test splits are shown in the Figure 17 below. The figures clearly demonstrate that the chosen set of augmentations significantly improve the overall performance of *Supervised(ImageNet)* across datasets and percentage splits, making it a strong baseline.

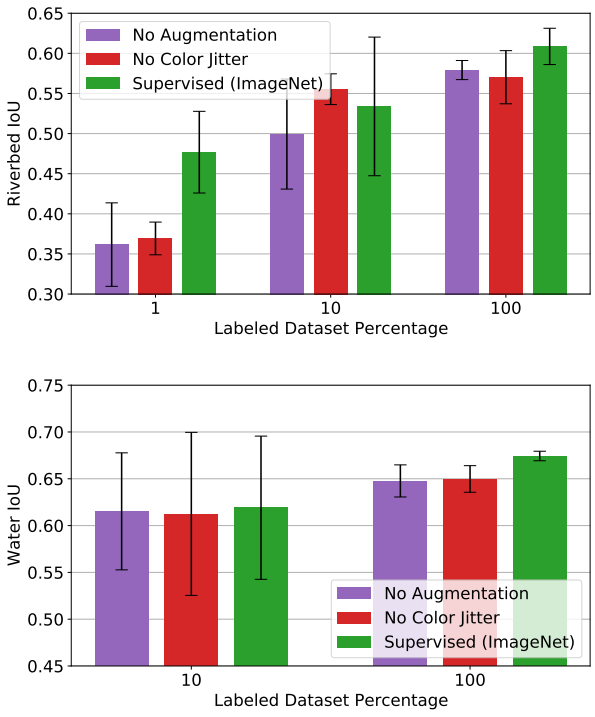


Fig. 17: Augmentations ablation on the domain-shifted partition of Riverbed segmentation(top) and Sen1Floods11(bottom) dataset.

C. SimCLR Augmentations

For RGB images, we use the same augmentations proposed by SimCLR [3]. For non-RGB images, we replace RGB color jitter with general color jitter. For color dropping, we use the mean of all channels for Chesapeake Land Cover dataset (RGB+IR images). But the notion of color dropping is not defined for Sentinel-1 images. Hence for Sen1Floods11, we don't use color dropping and instead adjust color jitter probability to 0.85. Below is the full list of SimCLR augmentations used:

- Distorted bounding box crop
- Random horizontal flip
- Random vertical flip
- Color jitter with 0.8 probability (probability 0.85 for Sen1Floods11)
  - `ColorJitterRGB` with strength 0.8 for RGB images
  - `ColorJitterGeneral` with strength 0.8 for non-RGB images
- Color dropping with 0.2 probability
  - Riverbed segmentation: tf.image.rgb_to_grayscale for RGB images
  - Chesapeake Land Cover: Taking mean of all channels
  - Sen1Floods11: No-operation
- Random Gaussian blur

Refer to Listing-1 below for the definition of `ColorJitterRGB` and `ColorJitterGeneral`.

D. FixMatch Augmentations

Augmentations are used for all 3 version of images during the training pass: labeled, weakly-augmented and strongly-augmented. For the labeled images and weakly augmented images, the augmentation policy for supervised DeepLabv3+ models (defined in Appendix B-A) is used. For strong augmentations, two simple augmentation functions are applied successively, followed by a random cutout to generate a strong augmentation. For the cutout augmentation, while FixMatch [2] only removes a single large rectangle, we remove multiple smaller rectangles. Table V lists the compositions of these augmentations. For each image one of the composition is randomly selected and the individual functions are applied successively.

E. Pseudo Code

We follow AutoAugment [61] and SimCLR [3] for the implementation of most augmentation functions. Please refer to the papers and their public implementations for specific details. Listing-1 contains pseudo code for remaining augmentation functions that we use.

Listing 1: Pseudo-code for augmentations using Tensorflow

```
def BrightnessPerChannel(image, strength):
  channels_list = split_channels(image)
  aug_channels_list = [
    Brightness(channel, strength) for channel in
    channels_list]
  return stack_channels(aug_channels_list)

def ContrastPerChannel(image, strength):
  channels_list = split_channels(image)
  aug_channels_list = [
    Contrast(channel, strength) for channel in
    channels_list]
  return stack_channels(aug_channels_list)

def ColorJitterRGB(image, strength):
  # We also randomize the order of these
```

Function	Parameter	Description
AutoContrast		Maximizes the image contrast by setting the darkest (lightest) pixel to black (white).
Brightness	S	Change the brightness of all channels by a random factor in $[1-S, 1+S]$.
Color	C	Controls the contrast of the image. A $C=0$ returns a gray image, $C=1$ returns the original image.
Contrast	S	Change the contrast of all channels by a random factor in $[1-S, 1+S]$.
Equalize		Equalizes the image histogram.
Hue	S	Change the hue of RGB image by a random factor in $[1-S, 1+S]$.
Invert		Adjusts each pixel value p to $(255-p)$
Posterize	B	Reduces each pixel to B bits.
Rotate	θ	Rotates the image by θ degrees.
Saturation	S	Change the saturation of RGB image by a random factor in $[1-S, 1+S]$.
ShearX	R	Shears the image along the horizontal axis with rate R.
ShearY	R	Shears the image along the vertical axis with rate R.
Solarize	T	Inverts all pixels above a threshold value of T.
SolarizeAdd	A, T	Add A to each pixel then invert each pixel with value above the threshold T.
TranslateX	λ	Translates the image horizontally by $(\lambda \times image\_width)$ pixels.
TranslateY	λ	Translates the image vertically by $(\lambda \times image\_height)$ pixels.

TABLE IV: Functions used in data augmentation policies for training the models.

Augmentation 1	Augmentation 2
Equalize(0.8, .1)	ShearY(0.8, 0.4)
Color(0.4, .9)	Equalize(0.6, 0.3)
Color(0.4, .1)	Rotate(0.6, 0.8)
Solarize(0.8, .3)	Equalize(0.4, 0.7)
Solarize(0.4, .2)	Solarize(0.6, 0.2)
Color(0.2, .0)	Equalize(0.8, 0.8)
Equalize(0.4, .8)	SolarizeAdd(0.8, 0.3)
ShearX(0.2, .9)	Rotate(0.6, 0.8)
Color(0.6, .1)	Equalize(1.0, 0.2)
Invert(0.4, .9)	Rotate(0.6, 0.0)
Equalize(1.0, .9)	ShearY(0.6, 0.3)
Color(0.4, .7)	Equalize(0.6, 0.0)
Posterize(0.4, .6)	Autocontrast(0.4, 0.7)
Solarize(0.6, .8)	Color(0.6, 0.9)
Solarize(0.2, .4)	Rotate(0.8, 0.9)
Rotate(1.0, .7)	TranslateY(0.8, 0.9)
ShearX(0.0, .0)	Solarize(0.8, 0.4)
ShearY(0.8, .0)	Color(0.6, 0.4)
Color(1.0, .0)	Rotate(0.6, 0.2)
Equalize(0.8, .4)	Equalize(0.0, 0.8)
Equalize(1.0, .4)	Autocontrast(0.6, 0.2)
ShearY(0.4, .7)	SolarizeAdd(0.6, 0.7)
Posterize(0.8, .2)	Solarize(0.6, 1.0)
Solarize(0.6, .8)	Equalize(0.6, 0.1)
Color(0.8, .6)	Rotate(0.4, 0.5)

TABLE V: Augmentation strategy proposed in AutoAugment [61] used as strong augmentation for FixMatch training. Each augmentation is specified as $function(probability, strength)$. For non-RGB inputs like Sen1Floods11 we replace the `Color` augmentation with `ColorJitterGeneral`.

```
  # augmentations each time.
  image = Brightness(image, strength)
  image = Contrast(image, strength)
  image = Hue(image, strength/4)
  image = Saturation(image, strength)
  return image

def ColorJitterGeneral(image, strength):
  # We also randomize the order of these
  # augmentations each time.
  image = Brightness(image, strength)
  image = Contrast(image, strength)
  image = BrightnessPerChannel(image, strength/2)
  image = ContrastPerChannel(image, strength/2)
  return image
```

APPENDIX C
VALIDATION SPLIT PLOTS

Figures 18, 19, 20 show the results of experiments on the selected hyper-parameters on the validation split of the datasets.

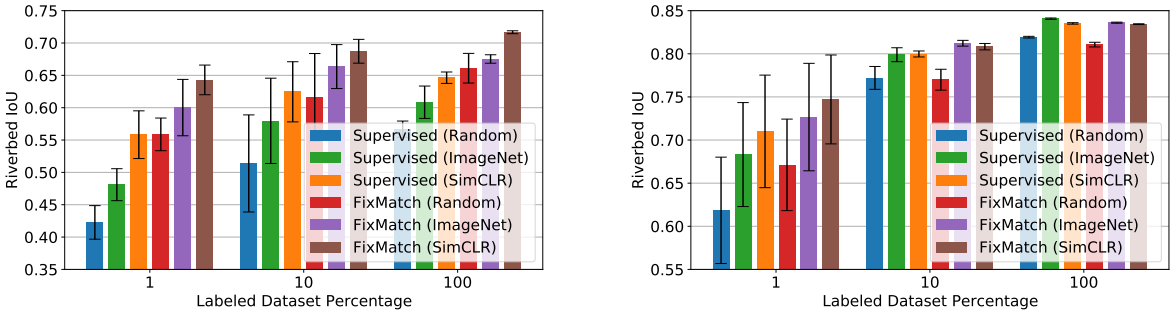


Fig. 18: Results on the validation split of River segmentation domain-shifted partition (left) and IID partition (right).

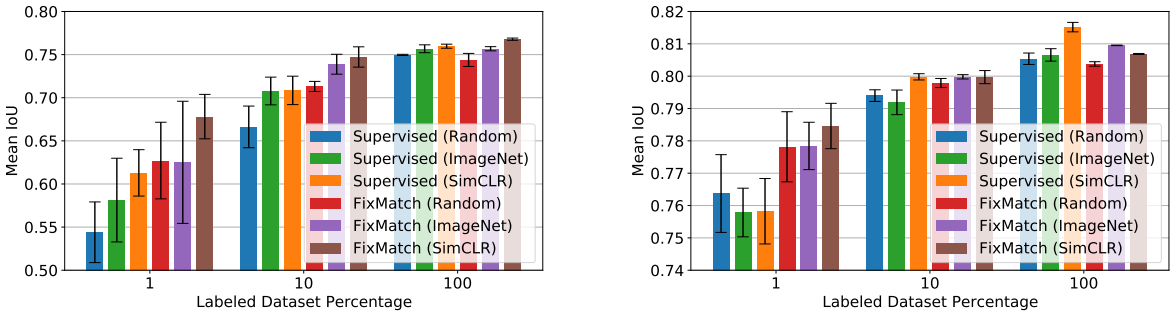


Fig. 19: Results on the validation split of Chesapeake domain-shifted partition (left) and IID partition (right).

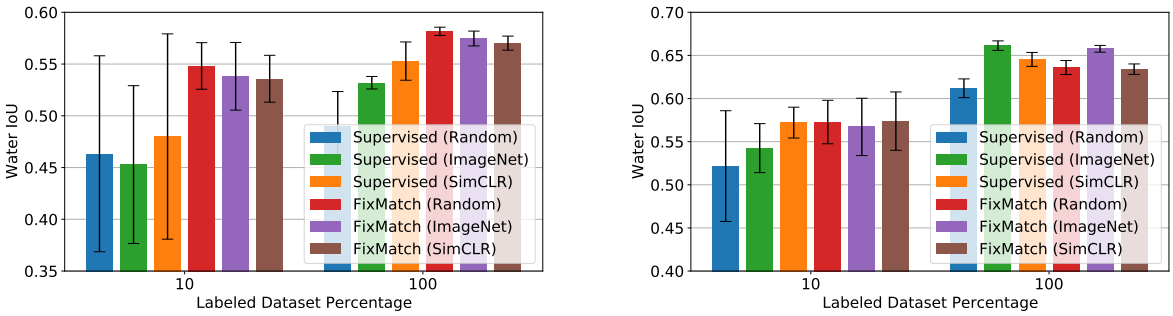


Fig. 20: Results on the validation split of Sen1Floods11 domain-shifted partition (left) and IID partition (right).