Semantic Segmentation of Medium-Resolution Satellite Imagery using Conditional Generative Adversarial Networks

Aditya Kulkarni
Radiant Earth Foundation
San Francisco, CA 94105
adkulkar@eng.ucsd.edu

Tharun Mohandoss
Radiant Earth Foundation
San Francisco, CA 94105
tharun96@utexas.edu

Daniel Northrup
Benson Hill
Saint Louis, MI
dan@northrup.ag

Ernest Mwebaze
Sunbird AI
Kampala, Uganda
emwebaze@gmail.com

Hamed Alemohammad
Radiant Earth Foundation
San Francisco, CA 94105
hamed@radiant.earth

Abstract

Semantic segmentation of satellite imagery is a common approach to identify patterns and detect changes around the planet. Most of the state-of-the-art semantic segmentation models are trained in a fully supervised way using Convolutional Neural Network (CNN). The generalization property of CNN is poor for satellite imagery because the data can be very diverse in terms of landscape types, image resolutions, and scarcity of labels for different geographies and seasons. Hence, the performance of CNN doesn't translate well to images from unseen regions or seasons. Inspired by Conditional Generative Adversarial Networks (CGAN) based approach of image-to-image translation for high-resolution satellite imagery, we propose a CGAN framework for land cover classification using medium-resolution Sentinel-2 imagery. We find that the CGAN model outperforms the CNN model of similar complexity by a significant margin on an unseen imbalanced test dataset.

1 Introduction

Semantic segmentation techniques applied to Earth observation (EO) can be helpful in identifying patterns in imagery and detecting changes throughout time. Lately, semantic segmentation algorithms using Convolutional Neural Network (CNN) have produced state-of-the art results on various medium-resolution EO data like Landsat and Sentinel-2 [1, 2, 3, 4]. CNN architectures are designed to capture both the high and the low level details in the images. [5, 6] use encoder-decoder type of architecture and skip connections for this. Even though CNN based approaches outperform other hand-engineered feature based solutions, these approaches can lack the generalization required to handle the seasonal and regional diversity of EO imagery [7].

[8] provides an alternative perspective to computer vision problems, by looking at them as an image-to-image translation problem. This is done using Generative Adversarial Networks (GANs) [9] in the conditional setting [10]. GANs are trained to solve a two-player mini-max game, where generator G tries to fool the discriminator D by generating images that resemble the original dataset, while discriminator tries to differentiate the generated images from the original dataset. This results in two architectures trying to beat each other; improve together in the process, until eventually they reach Nash equilibrium or an impasse. The objective function is given by Eq. 1, and unlike CNN based approaches does not require problem specific loss functions. Here x is the original dataset sample, z

AI for Earth Sciences Workshop at NeurIPS 2020.

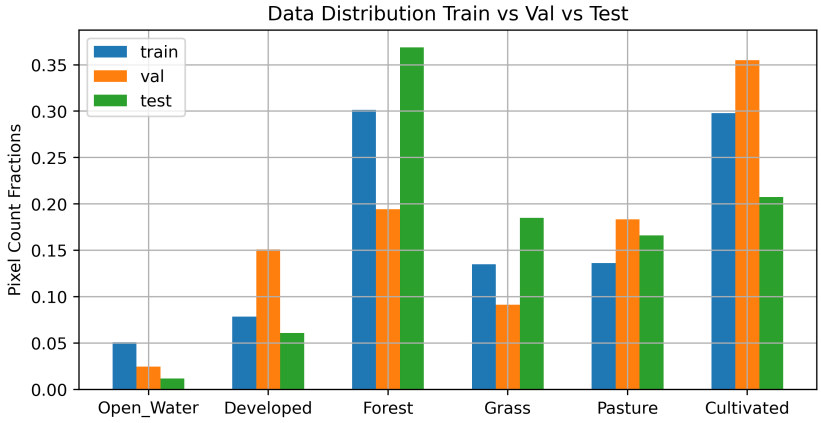

Figure 1: Class distribution in train, validation and test datasets

is the noise and $G(z)$ is generated data sample. The discriminator tries to maximize L_{GAN} and the generator tries to minimize L_{GAN}.

$$L_{GAN} = E_x[log(D(x))] + E_z[log(1 - D(G(z))] \tag{1}$$

Thanks to GAN's promising performance, their applications has emerged in EO too [7, 11, 12]. Further [7] has proposed that this technique can be applied to semantic segmentation problem on satellite imagery, and demonstrated that GANs can achieve near CNN performance. But CNNs still emerge victorious in high resolution imagery like that of [13]. We find that Conditional Generative Adversarial Networks (CGAN) has the potential to outperform CNN model of similar architectural capacity by a significant margin; when trained on medium-resolution satellite imagery, and tested on unseen imagery from a different location. Further, objects of the same land cover class in satellite imagery (e.g. cropland) are very distinct across different parts of the world. This also requires a segmentation model that can generalize beyond the initial training dataset, as these labeled data are scarce at global scale.

2 Dataset

In this work, we use Sentinel-2 satellite imagery, which has a resolution of 10 meters. Even though the dataset has a total of 13 spectral bands we choose Red, Green, Blue and Near Infrared (NIR) bands as a first step towards the problem. We use land cover classes from the National Land Cover Database (NLCD) that provides a 16-class semantic labels for each 30m pixel in the Sentinel-2 imagery. We selected multiple regions within the continental US to generate skewed training and test dataset. To simplify the complicated nuances in the similar classes, we apply some post processing on the labels. First, we merge together classes belonging to similar categories: deciduous forest, evergreen forest, mixed forest, and shrubs as forest class. Also a single developed class instead of developed open space, developed low intensity, developed medium intensity, and developed high intensity. Second, we dropped images belonging to extremely rare classes like barren land, woody wetlands and emergent wetlands. Thereby, reducing the total number of classes to six: Open Water, Developed, Forest, Grass, Pasture and Cultivated. Train and validation were selected from same regions, while test datasets was selected from a different region. Details about the locations can be found in the section C of the appendix. In total we have around 11943 train, 3989 validation and 6795 test images. The distribution of train, validation and test data is presented Figure 1. A sample image and its corresponding label is shown in Figure 2.

3 Methods

Inspired by [7] and [8], we use a special variant of GANs, called CGAN, to train the model for segmentation task. The dense image labelling task is looked at as that of an image-to-image translation:

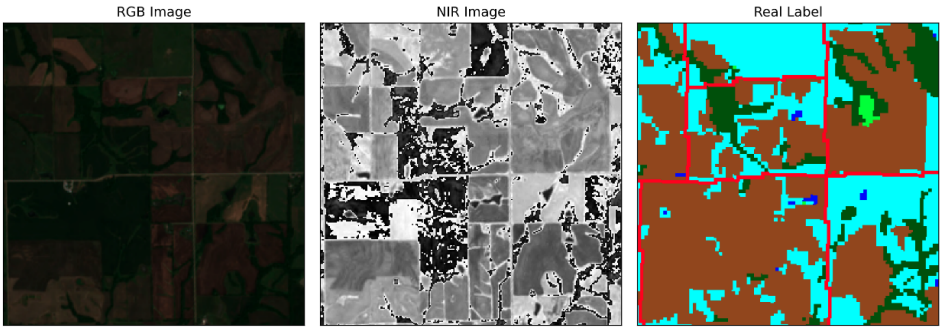

Figure 2: Sample RGB image with corresponding NIR band and labels. (label colors: red: developed, dark blue: open water, cyan: pasture, dark green: forest, light green: grass, brown: cultivated)

from a source image domain X i.e. from RGB+NIR to a target image domain Y i.e. dense labels. Here the generated output $\hat{Y}$ is conditioned on the input X. CGANs are popular for their superior performance over CNN in image-to-image translation tasks like image coloration, style transfer and dense labelling [7]. Like GAN, CGAN optimizes a two-person mini-max objective function given by Eq. 2. The generator G tries to minimize L_{CGAN} while discriminator D tries to maximize L_{CGAN}.

$$L_{CGAN} = E_{X,Y}(log(D(X,Y))) + E_{X,\hat{Y}}(log(1 - D(X, G(X)))) \tag{2}$$

Along with L_{CGAN}, we use L_2 norm loss to train the generator. This helps to capture all the low frequency features from the source images to target images. We found the performance of L_2 to be much better than L_1 in terms of F1-score, visual perception and also training stability. And since our dataset is imbalanced we use weighted L_2 norm loss Eq. 3 where the weight used is inverse of the class fraction in the training set. Hence the overall generator loss is given by Eq. 4 where w_c is the fraction of pixels of class c, $\lambda = 100$, Y and $\hat{Y}$ are the target labels and generated labels respectively.

$$L_{L^2} = \sum_{c=1}^{6} \frac{\frac{1}{w_c}||Y - \hat{Y}||_2}{\sum_{c=1}^{6} \frac{1}{w_c}} \tag{3}$$

$$L_G = L_{CGAN} + \lambda L_{L^2} \tag{4}$$

3.1 CGAN Model

Based on the success of benchmarks in pic2pic [8] and GeoGAN [7], we use a U-Net type architecture for our generator. We choose a 14 layer deep U-Net for our generator, and use Patch GAN [14] with 5 layers deep for our discriminator. Architectural details can be found in the table A.1, A.2 of the appendix. In contrast with GeoGAN, we found that a shallower network was better in learning the difference between classes. This could be because of the lower resolution of Sentinel-2 images compared to [13] used in GeoGAN. We optimize Eq. 4 by using Adam optimizer for G with $\beta = (0.5, 0.99)$ and plain stochastic gradient descent (SGD) for D. Both optimizers use a learning rate of $2e - 4$.

3.2 CNN Model

We also train a 14-layer deep U-Net in a fully supervised setting using cross entropy loss. This architecture is exactly same as the generator used in CGAN based approach. To have a fair comparison with CGAN architecture, we use a pixel-wise weighted cross entropy loss given by Eq. B.1 (in the appendix) to account for class imbalance.

Table 1: F1 Scores for train, validation and test datasets

Set	Architecture	Open Water	Developed	Forest	Grass	Pasture	Cultivated
Train	CGAN	**97.686**	76.371	78.939	**68.553**	**65.518**	84.569
	CNN	91.578	**84.899**	**81.443**	62.734	57.784	**90.693**
Validation	CGAN	**94.872**	83.537	70.247	**39.265**	**49.682**	81.899
	CNN	84.291	**84.029**	**71.858**	36.511	32.258	**85.956**
Test	CGAN	**82.985**	**54.402**	**62.093**	34.382	**56.655**	**59.197**
	CNN	49.081	50.343	61.635	**40.816**	40.203	57.606

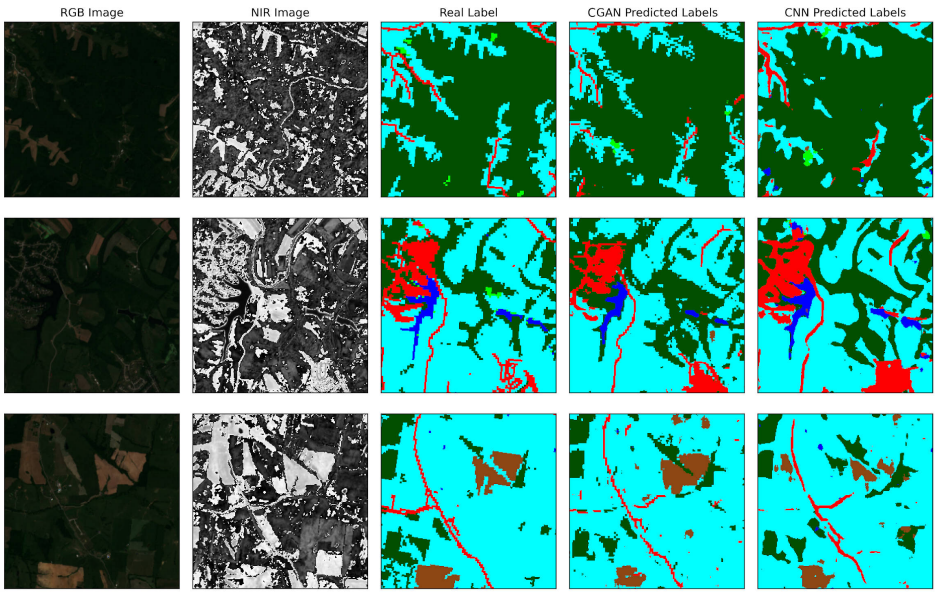

Figure 3: Comparison of CGAN and CNN predictions in the test dataset. (label colors: red: developed, dark blue: open water, cyan: pasture, dark green: forest, light green: grass, brown: cultivated)

4 Results and Discussions

We split the training dataset into train-fold and validation-fold, and tune for hyper-parameters for CGAN and CNN based architectures. Then the models are tested on an unseen dataset from a totally different geographical location as described in Appendix C. Table 1 presents the F1-score for train, validation and test datasets. We observe that on train and validation there was a tie between the two architectures with each exhibiting dominating performance over 3 classes. However, on test data CGAN outperforms CNN in five out of six classes. Figure 3 shows 3 examples of the input imagery, true labels, and predicted labels from CGAN and CNN. More examples is provided in Figure D.1

The performance enhancement observed in CGAN based approach demonstrated better generalization capability of the adversarial training over fully supervised training - for similar architectures. The CGAN based approach uses a total of 44.6M parameters, while CNN based approach uses 41.83M parameters. Hence, we can see that by using an additional 6.62% parameters we can achieve significant performance improvement on unseen test data. Additionally, we can observe from the predicted labels that the CGAN generated labels have learned all the nuances present in the target labels. The contours of objects in the CGAN labels are much similar to the target labels than the CNN based approach even on test data (Figure 3). Here, we demonstrated the performance gain of CGAN over CNN on unseen data and, as a future work we can explore the performance benefits of CGAN over CNN in a wider transfer learning setting: from a label rich region like the US to label scarce regions like Africa and Asia. We believe this approach can also be a good step towards building semantic models to produce labels across different seasons as well.

References

[1] Lisa Knopp, Marc Wieland, Michaela Rättich, and Sandro Martinis. A deep learning approach for burned area segmentation with sentinel-2 data. *Remote Sensing*, 12(15):2422, Jul 2020.

[2] A. Pomente, M. Picchiani, and F. Del Frate. Sentinel-2 change detection based on deep features. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 6859–6862, 2018.

[3] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov. Deep learning classification of land cover and crop types using remote sensing data. *IEEE Geoscience and Remote Sensing Letters*, 14(5):778–782, 2017.

[4] Xin-Yi Tong, Gui-Song Xia, Qikai Lu, Huanfeng Shen, Shengyang Li, Shucheng You, and Liangpei Zhang. Land-cover classification with high-resolution remote sensing images using transferable deep models. *Remote Sensing of Environment*, 237(July 2019):111322, feb 2020.

[5] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

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

[7] Vladimir V. Kniaz. Conditional GANs for semantic segmentation of multispectral satellite images. In Lorenzo Bruzzone and Francesca Bovolo, editors, *Image and Signal Processing for Remote Sensing XXIV*, volume 10789, pages 259 – 267. International Society for Optics and Photonics, SPIE, 2018.

[8] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2016.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[10] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[11] Vladimir V. Kniaz. Deep learning for dense labeling of hydrographic regions in very high resolution imagery. In Lorenzo Bruzzone and Francesca Bovolo, editors, *Image and Signal Processing for Remote Sensing XXV*, volume 11155, pages 283 – 292. International Society for Optics and Photonics, SPIE, 2019.

[12] Chuan Yang and Zhenghong Wang. An ensemble wasserstein generative adversarial network method for road extraction from high resolution remote sensing images in rural areas. *IEEE Access*, 09 2020.

[13] 2D Semantic Labeling Contest Potsdam ISPRS. http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html.

[14] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks, 2016.

[15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

A Network Architectures

Our experiments are performed using PyTorch libraries and relies on the code released by pix2pix [15]. The architectural details that we considered for the CGAN and CNN are listed in table A.1, A.2. The generator architecture is also the same one used for CNN. Here, we use 4×4 kernel for performing both 2D convolutions (Conv2D) and transpose 2D convolutions. LReLU stands for leaky relu with negative slope = 0.2, BN stands for batch normalization and DO stands for drop out with probability 0.5.

Table A.1: Generator Architecture

Block	Input Shape	Operations	Output Shape
1.	4 x 256 x 256	Conv2D, LReLU	64 x 128 x 128
2.	64 x 128 x 128	Conv2D, BN, LReLU	128 x 64 x 64
3.	128 x 64 x 64	Conv2D, BN, LReLU	256 x 32 x 32
4.	256 x 32 x 32	Conv2D, BN, LReLU	512 x 16 x 16
5.	512 x 16 x 16	Conv2D, BN, LReLU	512 x 8 x 8
6.	512 x 8 x 8	Conv2D, BN, LReLU	512 x 4 x 4
7.	512 x 4 x 4	Conv2D, ReLU	512 x 2 x 2
8.	512 x 2 x 2	ConvTrans2D, BN, ReLU	512 x 4 x 4
9.	1024 x 4 x 4	ConvTrans2D, BN, DO, ReLU	512 x 8 x 8
10.	1024 x 8 x 8	ConvTrans2D, BN,DO, ReLU	512 x 16 x 16
11.	1024 x 16 x 16	ConvTrans2D, BN, ReLU	256 x 32 x 32
12.	512 x 32 x 32	ConvTrans2D, BN, ReLU	128 x 64 x 64
13.	256 x 64 x 64	ConvTrans2D, BN, ReLU	64 x 128 x 128
14.	128 x 128 x 128	ConvTrans2D, Softmax	6 x 256 x 256

Table A.2: Discriminator Architecture

Block	Input Shape	Operations	Output Shape
1.	10 x 256 x 256	Conv2D, LReLU	64 x 128 x 128
2.	64 x 128 x 128	Conv2D, BN, LReLU	128 x 64 x 64
3.	128 x 64 x 64	Conv2D, BN, LReLU	256 x 32 x 32
4.	256 x 32 x 32	Conv2D, BN, LReLU	512 x 16 x 16
5.	512 x 16 x 16	Conv2D, Sigmoid	1 x 8 x 8

B CNN Optimization

We optimize a weighted cross entropy loss as in Eq. B.1 where w_c is the fraction of pixels of class c in the train dataset, and $\hat{y}_{pred}$ and y_{true} are the predicted class probability and one-hot encoded target respectively. We optimize the Eq. B.1 by using Adam optimizer with $\beta = (0.5, 0.99)$ and a learning rate of $2e-4$.

$$L_{CNN} = -E_{pixels}[\sum_{c=1}^{6} \frac{y_{true}}{w_c} log(\hat{y}_{pred})] \tag{B.1}$$

C Geographical Details of the Training and Test Datasets

In order to build the training dataset, we ensured that there were non-zero number of pixel samples for each of the six classes. But, as the dataset collected from a single city or region always tends to have highly skewed class distribution, we collected both training and test data from a number of regions around the US. To build the training dataset we sampled images from Iowa, Sacramento, Dallas, Chicago, Los Angeles, San Diego, Houston, New York, Boston, Baltimore, Philadelphia, Seattle and Detroit areas. Similarly, to build the test dataset we considered Montana, Kentucky, Green Bay Wisconsin and area around Lake Michigan.

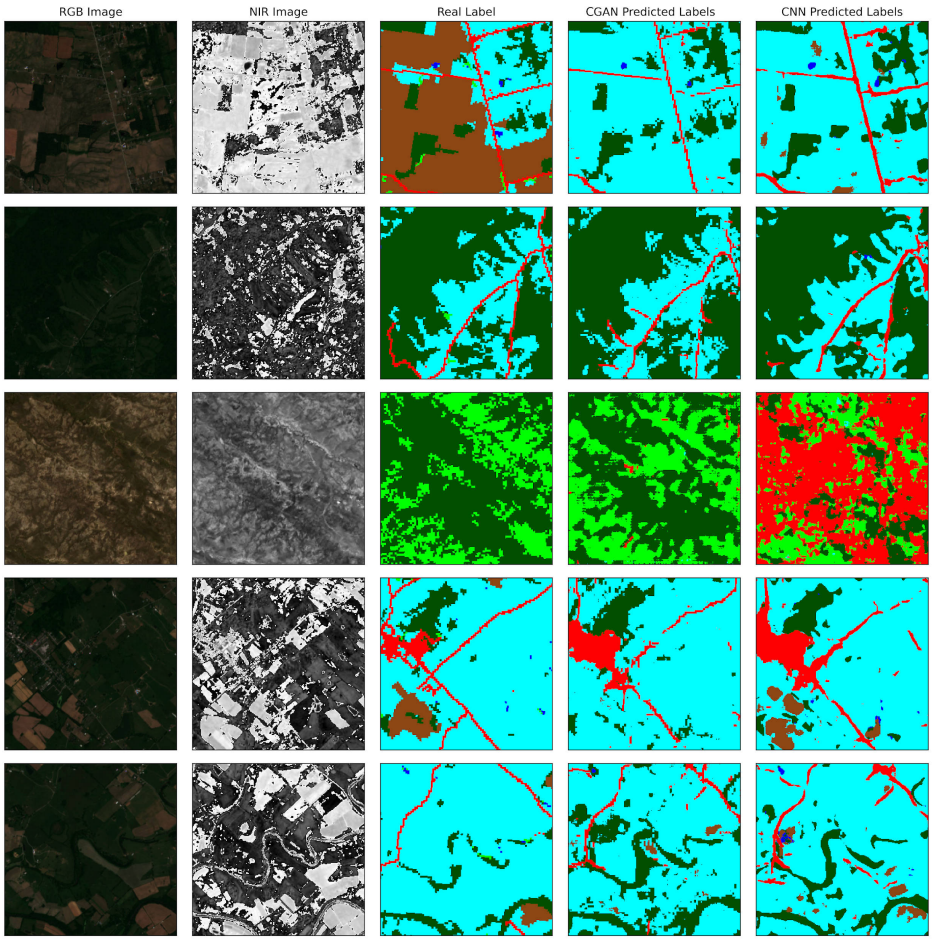

Figure D.1: More examples similar to Figure 3. (label colors: red: developed, dark blue: open water, cyan: pasture, dark green: forest, light green: grass, brown: cultivated)

D Additional Test Image

More predicted classes from the test dataset are shown in Figure D.1. We can see that the models struggle with cultivated class in the test set. It is worth noting that the CGAN model performs better on grass class in the test data (different region), while CNN confuses such patterns with development class. Even visually CGAN generated labels are more similar to the real labels than the CNN generated labels.