CS230

Segmentation of Costa Rican Pineapple Plantations from Satellite Imagery Using U-Net

Sage McGinley-Smith
Department of Computer Science
Stanford University
sagems@stanford.edu

Abstract

Dengue is an mosquito-borne disease affecting the country of Costa Rica, as well as global health more broadly. Scientists hypothesize that pineapple plantation development in Costa Rice might be related to dengue transmission, but no dataset of modern pineapple plantations exists for the country of Costa Rica. This project develops a pipeline for a U-Net model for binary satellite image segmentation of pineapple plantations from Sentinel-2 data provided by the European Space Agency. Results show that additional positive examples are needed to improve performance, and that Focal Loss is a good choice for this problem. Code available at this Github: `https://github.com/sagems/230_final_project`.

1 Introduction

Background and Problem Statement: Dengue fever, a mosquito-borne viral disease caused by Aedes aegypti and Aedes albopictus mosquitoes, poses a significant global public health challenge, particularly in tropical and subtropical regions. In Costa Rica, dengue has become a pressing health concern due to the country's warm and humid climate, compounded by agricultural expansion. Pineapple plantations, a major agricultural industry in Costa Rica, have been linked to increased dengue risk as their irrigation systems and drainage ditches often create standing water, ideal breeding grounds for mosquitoes [1]. Moreover, the use of agrochemicals and habitat modifications in such plantations may influence vector ecology, potentially exacerbating disease transmission [2]. The Mordecai Lab at Stanford is working with researchers at Centro Nacional de Alta Technologie (CeNAT) in Costa Rica to study the impacts of pineapple cultivation on dengue transmission. To study this, they need the data on pineapple plantation distribution from 2019-present. No dataset currently exists for 2020-present. This project seeks to address this problem by creating a model which predicts pineapple plantation distribution for the years 2020-2024 based on satellite imagery.

Satellite data has long been used as a way to observe the Earth at a spatial and temporal scale. In recent years, the availability of high quality, temporally regular, and public satellite image data has coincided with the rise of Machine Learning and Computer Vision to produce advancements in satellite image segmentation technology. Additionally, the advancements in resolution of spectral bands in satellite imagery have allowed for multi spectral image segmentation as opposed to traditional RGB image segmentation [3]. Applications range from mapping landslides and waterways to analyzing poverty within cities, and more [4] [5] [6]. Much of this early research was focused on more traditional techniques, such as Random Forest prediction. Only more recently has Deep Learning, particularly Convolutional Neural Nets (CNNs) been used to segment satellite images [7] [8]. Thus, this project aims to build a CNN model that can identify pineapple plantations across the entirety of Costa Rica for the years 2020-2024 based on ground-truth data for plantation distribution in 2019.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

Input and Output: The input to my algorithm is a 128 x 128 satellite image tile containing 9 spectral features to predict pineapple plantation distribution within that patch. The project leverages a U-Net model infrastructure trained on roughly 10,000 image tiles for binary segmentation.

CS 191W Contribution: This project was completed in parallel with a CS 191W project advised by Jerry Cain and preapproved for collaboration. The components used for that class were the data cleaning and organization, as well as the linear baseline models. For CS 230, the components used are the model construction, training, and tuning, as well as the model evaluation. Further questions about this collaboration should be directed to Sage or CS 191W advisor Caroline Glidden (cglidden@stanford.edu).

2 Related work

Model Architecture: The state of the art for satellite image segmentation revolves around several models. Fully Convolutional Networks (FCNs) are frequently employed, as in [9]. Similar, and most aligned with this project is work similar to Masolele et. al, which uses an Attention U-Net architecture for agricultural segmentation in Africa. U-Net is an architecture that combines FCN architecture with skip connections to better preserve finer details. The Masolele paper also employs active learning, which was not employed in this projected but provides promising results. Other examples of U-Net for image segmentation more broadly can be found in [10][11][12].

An alternative architecture newly popular for satellite image segmentation is the PSPNet, which uses a pyramid pooling module and slightly different upsampling structure than U-Net, and is demonstrated in Yuan et. al [13]. This architecture is less established in the satellite imagery field, but likely to become more prevalent in coming years. PSPNet was not chosen for this project because it is more computationally intensive and requires more training data than U-Net.

Loss Functions: Multiple loss functions have been popularized for image segmentation. Binary Cross Entropy (BCE) loss is often employed for situations where there is a balance of the number of pixels of each type in the training set. An example of BCE for satellite image segmentation is [14]. Weighted BCE is an adapted BCE that helps account for imbalances in the dataset. It is demonstrated in Li et. al [15]. Both BCE and Weighted BCE were employed in this project, but not used for the final model. Focal Loss, which is demonstrated in the Masolele paper, and is used specifically for projects where there are class imbalances. Given that only aroun 5% of the pixels were of the target class, Focal Loss was employed in this project, similar to its use in the Masolele paper, and further detailed in [16].

Feature Extraction and Spectral Bands: The use of bands outside of the visible spectrum from image segmentation is well substantiated in the literature. Additionally, using Normalized Difference Vegetation Index (NDVI), Soil Adjusted Vegetation Index (SAVI), Normalized Difference Moisture Index (NDMI), and Normalized Difference Water Index (NDWI) is well substantiated in [16][17][18][19][20].

Other models use additional bands and indices such as Weighted Difference Vegetation Index (WDVI), Global Environmental Monitoring Index (GEMI), Modified Soil-Adjusted Vegetation Index (MSAVI) are also popular in image segmentation, but are less prevalent and similar to the indices already chosen, so they were not considered in this project [18].

Transfer Learning: Transfer learning involves the retraining of a previously trained model on a given dataset. It is popular in image segmentation because it reduces training time and can allow a model to be used on multiple different datasets. Transfer learning is prevalent in satellite image segmentation such as in Wang et. al [21]. It was not employed in this project due to the expressed objective of implementing the learning algorithm from scratch.

3 Dataset and Features

Spectral Imagery: The Sentinel-2 Level 2A Harmonized dataset, provided by ESA's Copernicus program, offers atmospherically corrected optical imagery (2015–present) with consistent bottom-of-atmosphere reflectance values. Data was composited quarterly to select cloud-free images over four regions of interest (northwest, northeast, central, southwest). Features include RGB bands (B4, B3, B2), near-infrared (B8), and short-wave infrared (B11). Indices such as NDVI, SAVI, NDMI,

and NDWI were computed for land surface analysis. Bands were resampled to 10m resolution and normalized to [0, 1] for training. The dataset is georeferenced in the WGS 84 / UTM projection, distributed as part of the Copernicus Open Access Hub, and made available through Google Earth Engine (GEE).

Data Labels: Segmentation masks, sourced from Costa Rica's Centro Nacional de Alta Technologie (CeNAT), one of Costa Rica's premier research institutions via a partnership with Stanford Mordecai Lab, represent ground-truthed 2019 pineapple plantation distributions. Masks were rasterized, re-projected, and aligned with Sentinel imagery.

Image Tiling and Storage: Sentinel images and masks were tiled into 128x128 chips using `rasterio` in Google Colab and stored in a Google Cloud bucket. A train/dev/test split of 80/10/10 resulted in 4,863 training, 606 validation, and 612 test examples. Figure 3 illustrates a sample image tile and corresponding mask.

Data Sources:
Sentinel:`https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S2_SR_HARMONIZED#description`
Labels: Made available by CeNAT researchers. Available upon request.

4 Methods

Linear Baseline: To form a baseline for comparison to deeper learning models, a logistic regression model was trained on a subset of the data. The model selected 25,000 pixels that were labeled as 0 (not plantation) and 1,000 pixels that were labeled as 1 (plantation present) from the southwest quadrant image, which contains several plantations. These ratios reflect the ratio of each class in the quadrant as a whole. the southwest quadrant was chosen because the image is high quality (low cloud cover) and covers both plantation and non-plantation areas. A model was run on both the full spectral band pixels (9 features) as well as simple RGB 3 feature pixels.

All model training methods were implemented within the TensorFlow keras framework.

Model Architecture: The U-Net model employed in this project is a convolutional neural network (CNN) architecture specifically designed for semantic image segmentation tasks. It follows a symmetric encoder-decoder structure, comprising a contracting (downsampling) path, a bottleneck, and an expansive (upsampling) path. The contracting path consists of two blocks of convolutional layers. Each block applies two 3×3 convolutional layers with ReLU activations, followed by a 2×2 max-pooling layer, progressively reducing the spatial dimensions while doubling the number of feature channels at each stage (64 and 128 channels, respectively). At the bottleneck, two additional 3×3 convolutional layers with ReLU activations are applied, both with 256 feature channels.

The expansive path mirrors the contracting path and consists of two upsampling blocks. Each block begins with a 2×2 upsampling layer that restores the spatial dimensions, followed by concatenation with the corresponding feature maps from the contracting path via skip connections. Each upsampling block then applies two 3×3 convolutional layers with ReLU activations, halving the feature channels from the previous layer (128 and 64 channels, respectively). Finally, a 1×1 convolutional layer with a sigmoid activation function produces the output, which is a single-channel binary mask with pixel-wise probabilities. Parameter counts and a technical table description are in Figure ....

Loss Function: This project tested several loss functions, including Binary Cross Entropy Loss, Weighted Binary Cross Entropy Loss, and Focal Loss, all of which have substantiating literature supporting their use for satellite image segmentation. Ultimately, Focal Loss was used for the final model, but results were obtained for the performance of all three loss functions. Focal Loss is defined as follows:

$$\mathcal{L}_{Focal} = -\frac{1}{N}\sum_{i=1}^{N} \alpha(1-\hat{y}_i)^{\gamma} y_i \log(\hat{y}_i) + (1-\alpha)\hat{y}_i^{\gamma}(1-y_i)\log(1-\hat{y}_i)$$

- N: Number of samples in the batch ($N = 64$).
- $\hat{y}_i$: Predicted probability for the i-th sample.

- y_i: True label for the i-th sample (0 or 1).
- α: Balancing factor for positive and negative examples ($\alpha = 0.2$).
- γ: Focusing parameter to emphasize hard-to-classify examples ($\gamma = 2$).

This formula effectively reduces the loss contribution from well-classified examples, focusing more on hard-to-classify ones. The model uses the implementation of Focal Loss

Input Normalization: All input data was normalized to the range $[0, 1]$ and a 80% train, 10% dev, 10% test split was used. Further details on data preprocessing are presented in the above section titled Dataset and Features. For final model, all 9 bands were used. During tuning, a model was tests that used only the non-index bands (RGB, near infrared, short wave infrared).

Optimization: Adam (Adaptive Moment Estimation) Optimization is used in training to help smooth out convergence. Adam uses a combination of methods Momentum and RMSProp to adaptively tune the learning rate for each parameter based on its gradients. It also includes bias correction to account for the moving averages being initialized to zero.

Batching: Batching was used to increase the iterations of backpropogation per epoch. A batch size of 64 was used, as employed in Masolele et. al, 2024. Shuffling is used across epochs to reduce the likelihood of the model learning patterns based on the ordering of the data. The buffer size of the shuffling is 100 in this case, which reduces computational complexity but does not represent complete shuffling of the dataset.

Early Stopping: Early stopping, where the model stops training when performance plateaus to prevent overfitting, was explored. Given the limited computational resources of the project, the model never reached a meaningful plateau so and overfitting was not an issue and thus early stopping was ultimately not employed in the final model.

5 Experiments/Results/Discussion

Linear Baseline: Linear Baseline Results can be found in Figures...

Hyperparameters: The learning rate was chosen to start at 0.001, using Adam optimization to adjust the learning rate dynamically for each parameter. 0.001 was chosen as the initial rate, as suggested in the original Adam paper by Kingma and Ba (2014). The mini batch size was chosen as 64. Originally, the batch size was 128, but that proved too computationally intensive for the system, and the batch size of 32 (mirroring that in Masolele paper) slowed down the training process. For the Focal loss, the alpha and beta parameters were chosen based on the original Focal Loss paper, Lin et. al (2017), which suggests that alpha should be set to 2.0 and gamma to 0.25 for most problems.

Metrics: Metrics used were precision, recall, accuracy, loss, and f1 score. The confusion matrix is structured as follows:

	Predicted Positive	**Predicted Negative**
Actual Positive	$TruePositives(TP)$	$FalseNegatives(FN)$
Actual Negative	$FalsePositives(FP)$	$TrueNegatives(TN)$

The key metrics used to evaluate classification performance are defined below:

- **Precision:** The proportion of predicted positive instances that are correct. Computed as TP / (TP + FP).
- **Recall:** The proportion of positive classes correctly identified by the model. Computed as TP / TP + FN.
- **Accuracy:** The proportion of all correctly classified instances. Computed as TP + TN / Total Instances.
- **F1-Score:** The harmonic mean of precision and recall, computed as:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Results: Before implementing normalization and dataset shuffling, the loss function was erratic and did not converge in a meaningful way, particularly in early iterations. An example loss curve for the

unnormalized model is shown in Figure 4. Once normalized, the initial loss curve smoothed out to the following, with a final loss of 0.0815 for training data and 0.0511 for validation data, shown in Figure 5.

Although initial epochs showed precision and accuracy hovering around 0.2-0.3, it quickly dropped to 0 in later epochs. The test accuracy was 95.30% and the dev accuracy was 95.63%. Although the accuracy seems high and the loss low, the model was found to be predicting all zeros by the end of the 50 epochs of training. Earlier models predicted some ones randomly, but the final model predicted all zeros on the test and dev sets. An example of this is shown In Figures 6 and 7. The confusion matrices for the dev and test sets are shown in Figure 8.

Discussion: The results indicate that the model performs more effectively with input normalization, achieving smoother convergence. Trials with alternative loss functions revealed that Focal Loss was the most suitable for this task, as other loss functions exhibited more erratic behavior during testing. However, the model ultimately failed to achieve its intended objective, as evidenced by its low precision and recall scores. Due to the dataset's significant class imbalance, with positive examples comprising only 5% of the data, the model learned to predominantly predict zeros to optimize metrics such as accuracy, which reached over 95%. Despite successful minimization of the loss function, the model's inability to capture minority-class features highlights several key challenges.

Firstly, the loss function did not adequately weight the minority class, which could have benefited from stronger weighting to counterbalance the class imbalance. Secondly, the training dataset likely lacked sufficient positive examples to enable effective learning. Lastly, the normalization of the red, green, and blue (RGB) pixel values resulted in very small feature magnitudes, potentially diverting the model's attention away from these critical features. These limitations underscore the need for further refinements in both the dataset and training approach to improve model performance.

6 Conclusion and Future Work

After training over ten models during this project, the final model was unable to perform effectively on the plantation segmentation task. The primary factors contributing to this outcome appear to be the lack of positive examples in the dataset and insufficient weighting of the minority class in the loss function. Improved normalization techniques, particularly for the RGB channels, should be explored in future work, as well as incorporating batch normalization to stabilize and enhance model training. Refining the loss function by experimenting with different weighting strategies for the minority class and fine-tuning parameters such as α and γ in Focal Loss could yield better results. Increasing the proportion of positive examples through data augmentation is another critical step to mitigate class imbalance. Additional data collection efforts to expand the dataset could also significantly enhance model performance.

With greater computational resources, larger and more complex models—similar to those used in studies such as Masolele et al.—could be trained to improve precision. Training multiple models simultaneously with varied hyperparameters (e.g., different α and γ values) and enhancing data shuffling between epochs may further improve generalization. This project successfully established a pipeline for integrating Google Earth Engine data with TensorFlow machine learning models and demonstrated the critical importance of data augmentation in handling imbalanced datasets. The findings suggest that Focal Loss with parameters ($\gamma = 2.0$, $\alpha = 0.25$) is insufficient for scenarios where positive examples represent only 5% of the dataset. With continued data collection and iterative refinement, this pipeline has the potential to produce a more robust and effective model for plantation segmentation in the future.

7 Contributions

This project was completed as an individual project. All code was written by Sage as was this final report and the final poster. The only notable external contribution was code from a Senior Project Sage was completing this quarter, which was used for data cleaning.

8 Figures

:

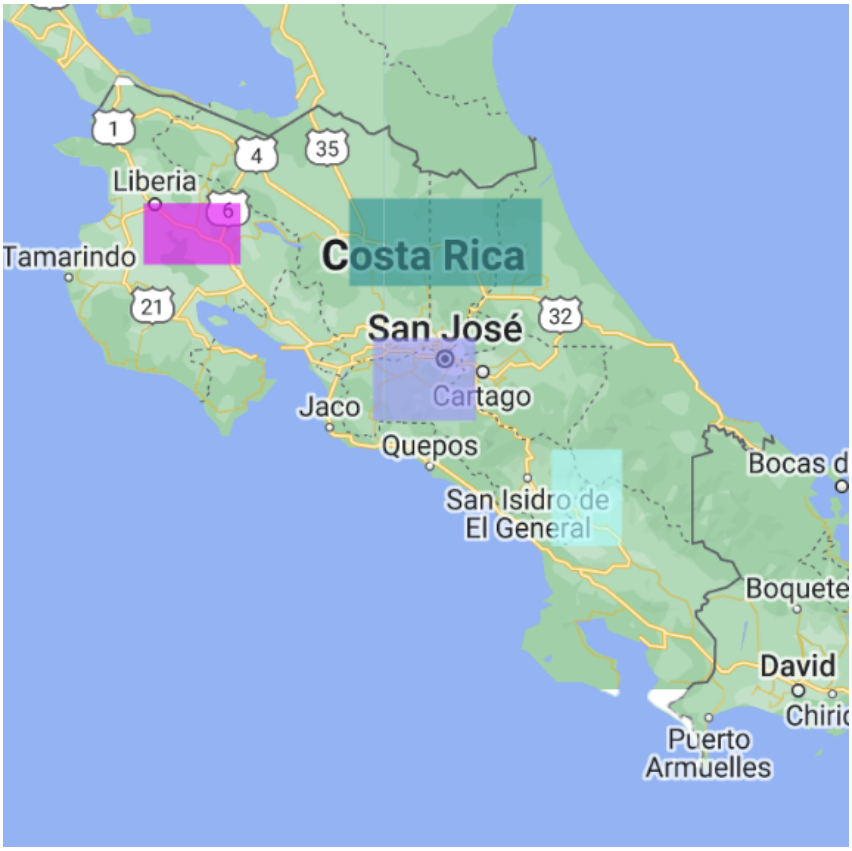

Figure 1: Quadrants for Training

```
Accuracy: 0.98
Classification Report:
              precision    recall  f1-score

           0       0.98      0.99      0.99
           1       0.83      0.60      0.70
```

Figure 2: Linear Baseline, All Features

```
Accuracy: 0.9625
Classification Report:
              precision    recall  f1-score

           0       0.96      1.00      0.98
           1       0.58      0.10      0.16
```

Figure 3: Linear Baseline, RGB Only

]

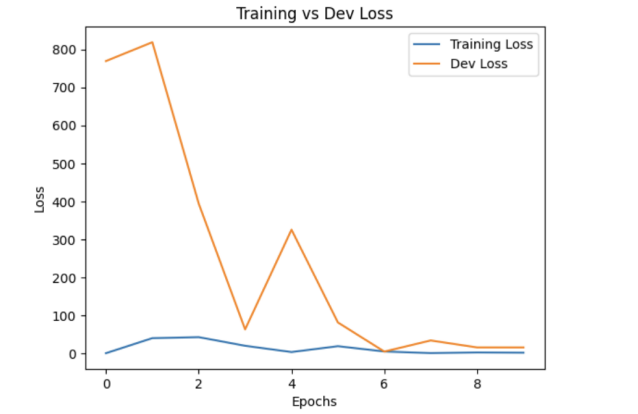

Figure 4: Loss Curve, Pre Normalization

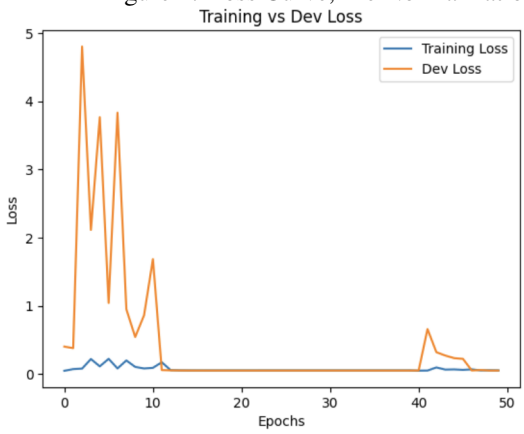

Figure 5: Loss Curve, Post Normalization

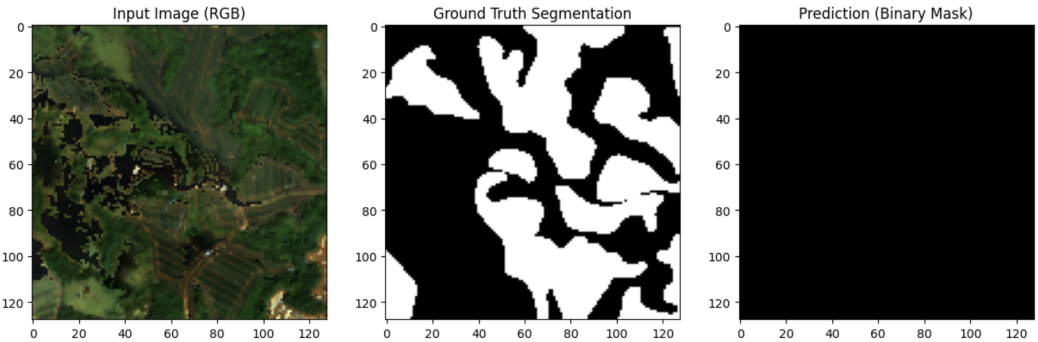

Figure 6: Example Prediction All Zeros, Plantation Tile

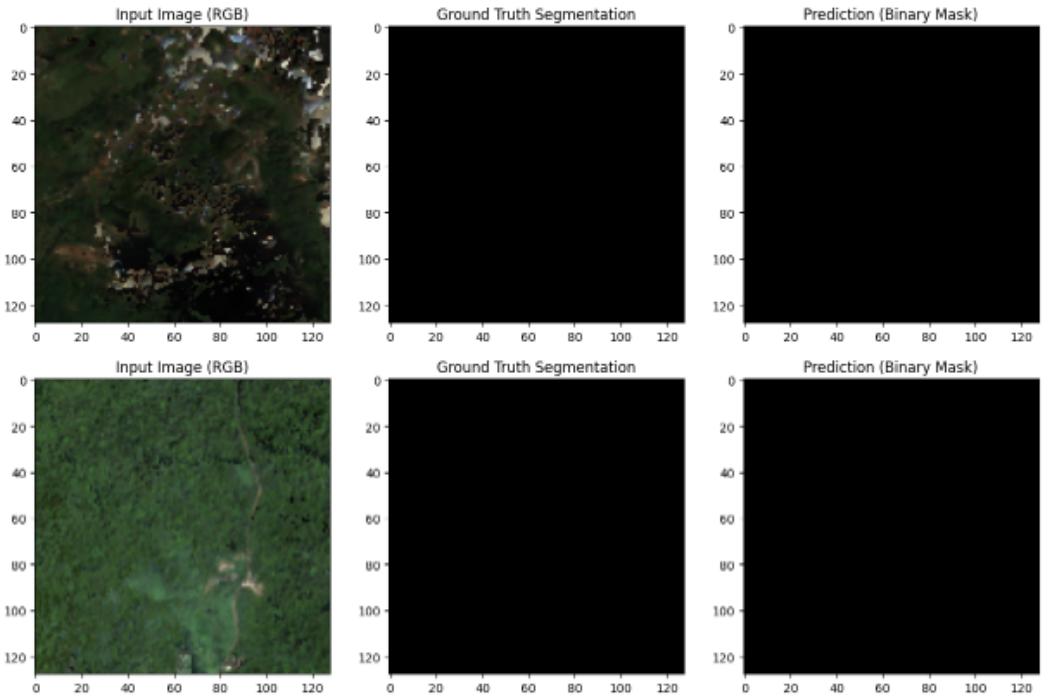

Figure 7: Example Prediction All Zeros, Non-Plantation Tile

Dev Set Confusion Matrix

	Predicted Negative	**Predicted Positive**
Actual Negative	9479017	0
Actual Positive	433303	0

Test Set Confusion Matrix

	Predicted Negative	**Predicted Positive**
Actual Negative	9556150	0
Actual Positive	470858	0

Figure 8: Confusion Matrices

9 References

[1] Troyo, A., Fuller, D. O., Calderón-Arguedas, O., Solano, M. E., Beier, J. C. (2009). Urban structure and dengue incidence in Puntarenas, Costa Rica. Singapore journal of tropical geography, 30(2), 265-282.

[2] Ferraguti, M., Magallanes, S., Suarez-Rubio, M., Bates, P. J., Marzal, A., Renner, S. C. (2023). Does land-use and land cover affect vector-borne diseases? A systematic review and meta-analysis. Landscape Ecology, 38(10), 2433-2451.

[3] Vali A, Comai S, Matteucci M. Deep Learning for Land Use and Land Cover Classification Based on Hyperspectral and Multispectral Earth Observation Data: A Review (2020). *Remote Sensing* **12(15)**:2495. https://doi.org/10.3390/rs12152495

[4] Zhang Y., Wu W., Qin Y., Lin Z., Zhang G., Chen R., Song Y., Lang T., Zhou X., Huangfu W., et al. (2020) Mapping Landslide Hazard Risk Using Random Forest Algorithm in Guixi, Jiangxi, China. *ISPRS International Journal of Geo-Information* **9**(11):695. https://doi.org/10.3390/ijgi9110695.

[5] Felton B.R., O'Neil G.L., Robertson M.-M., Fitch G.M., Goodall J.L. (2019) Using Random Forest Classification and Nationally Available Geospatial Data to Screen for Wetlands over Large Geographic Regions. *Water* **11**(6):1158. https://doi.org/10.3390/w11061158.

[6] Zhao X., Yu B., Liu Y., Chen Z., Li Q., Wang C., Wu J. (2019) Estimation of Poverty Using Random Forest Regression with Multi-Source Data: A Case Study in Bangladesh. *Remote Sensing* **11**(4):375. https://doi.org/10.3390/rs11040375.

[7] Ayushi, Preetpal Kaur Buttar (2024) Satellite Imagery Analysis for Crop Type Segmentation Using U-Net Architecture. *Procedia Computer Science* **235**:3418-3427. https://doi.org/10.1016/j.procs.2024.04.322.

[8] Masolele, R.N., Marcos, D., De Sy, V. et al. (2024) Mapping the diversity of land uses following deforestation across Africa. *Scientific Reports* **14**:1681. https://doi.org/10.1038/s41598-024-52138-9.

[9] Buttar, P. K., Sachan, M. K. (2024). Land cover segmentation using 3D FCN-based architecture with coordinate attention. IEEE Geoscience and Remote Sensing Letters.

[10] O. Ronneberger, P. Fischer, T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany, 2015.

[11] Singh, N. J., Nongmeikapam, K. (2023). Semantic segmentation of satellite images using deep-unet. Arabian Journal for Science and Engineering, 48(2), 1193-1205.

[12] Wimmers, Anthony J., Sarah Griffin, and Christopher Velden. "A UNet retrieval of tropical cyclone inner-core wind fields from microwave and infrared satellite imagery." Artificial Intelligence for the Earth Systems (2024).

[13] Yuan, Y., Yang, L., Chang, K., Huang, Y., Yang, H., Wang, J. (2024). DSCA-PSPNet: Dynamic spatial-channel attention pyramid scene parsing network for sugarcane field segmentation in satellite imagery. Frontiers in Plant Science, 14, 1324491.

[14] Badr, M. A., Elrewainy, A. F., Elshafey, M. A. (2024, May). Efficient Lossy Satellite Image Compression Using Hybrid Autoencoder Model. In 2024 14th International Conference on Electrical Engineering (ICEENG) (pp. 78-83). IEEE.

[15] Li, X., He, M., Li, H., Shen, H. (2021). A combined loss-based multiscale fully convolutional network for high-resolution remote sensing image change detection. IEEE Geoscience and Remote Sensing Letters, 19, 1-5.

[16] Ross, T. Y., Dollár, G. K. H. P. (2017, July). Focal loss for dense object detection. In proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2980-2988).

[17] Tempa, K., Ilunga, M., Agarwal, A., Tashi. (2024). Utilizing Sentinel-2 Satellite Imagery for LULC and NDVI Change Dynamics for Gelephu, Bhutan. Applied Sciences, 14(4), 1578.

[18] Huete, A. R. (1988). A soil-adjusted vegetation index (SAVI). *Remote Sensing of Environment*, *25*(3), 295–309. `https://doi.org/10.1016/0034-4257(88)90106-X`

[19] Vasantrao, C. P., Gupta, N. (2023). Wader hunt optimization based UNET model for change detection in satellite images. International Journal of Information Technology, 15(3), 1611-1623.

[20] Suyeda, F. D., Setiawan, B. (2022, April). Utilization of NDMI Method in Landsat 8 Satellite Imagery for Analysis of Multi-Hazard Susceptibility. In GMPI Conference Series (Vol. 1, pp. 63-70).

[21] McFeeters, S. K. (1996). The use of the normalized difference water index (NDWI) in the delineation of open water features. *International Journal of Remote Sensing*, *17*(7), 1425–1432. https://doi.org/10.1080/01431169608948714

[22] Wang, S., Waldner, F., Lobell, D. B. (2022). Unlocking large-scale crop field delineation in smallholder farming systems with transfer learning and weak supervision. Remote Sensing, 14(22), 5738.

[23] Kingma, D. P. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Data:

European Space Agency. (2023). *Sentinel-2 Harmonized Dataset*. Copernicus Open Access Hub. https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S2_SR_HARMONIZED#description

Centro Nacional de Alta Tecnología (CeNAT). (2019). *Land use and land cover mapping for Costa Rica. National Dataset*. Retrieved from CeNAT database in partnership with Stanford Mordecai Lab.

Packages:

Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., & Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*, *202*, 18–27. https://doi.org/10.1016/j.rse.2017.06.031

TensorFlow Developers. (2023). *TensorFlow: Large-scale machine learning on heterogeneous systems*. TensorFlow. https://www.tensorflow.org/

TensorFlow Addons Developers. (2023). *TensorFlow Addons: Extending TensorFlow functionality*. GitHub. https://github.com/tensorflow/addons

Google Cloud. (2023). *Google Cloud Storage Client Library for Python*. Google Cloud. https://cloud.google.com/python/docs/reference/storage/latest

Gillies, S., & Contributors. (2023). *Rasterio: Geospatial raster data access for Python programmers*. Mapbox. https://rasterio.readthedocs.io/

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., ... & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Chollet, F., & others. (2015). *Keras: Deep learning library for Python*. GitHub. https://keras.io/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. https://jmlr.org/papers/v12/pedregosa11a.html