Deep Learning for the Social Sciences (DLSS)

Final Project

Project Title: CNN for Detecting Deforestation (and Ice Melting)[1]

I. Introduction

Forest extension classification from satellite images helps monitor legal and illegal deforestation practices over time. Deforestation is a major environmental concern, affecting soil quality, species habitats, and biodiversity. It also contributes to climate change by increasing greenhouse gas levels in the atmosphere (Montenegro et al., 2005). In Argentina, the growth of agricultural activities, particularly the spread of soybean farming, has led to the widespread clearing of native forests across large Chaco Park regions (Cuadra et al., 2020; Montenegro et al., 2005).

This project aims to contribute to this topic by building satellite image classifiers based on Convolutional Neural Networks (CNN) and a change detection algorithm to assess forest loss between two different time points. The selected time frame is the six years before the COVID-19 pandemic, 2014-2019. This project will only focus on deforestation detection, due to follow-up feedback with the tutors and since additionally investigating in ice melting detection would go beyond the scope of this project.

II. Data and data preprocessing

II.1 Satellite image specificities

Using satellite images as data requires accounting for some particularities. This type of image usually has a higher number of pixels, which requires patching before inputting them into the CNN to avoid too computationally costly training (Preligens, 2019). In addition, satellite images can have noisy features due to clouds or variations in the spectral reflectance of a geographical event, such as water or forests (Poliyapram et al., 2019). To reduce these variations, we used composite satellite images, featuring the yearly median pixel values for the georeferenced area, instead of a unique satellite image. The limitations of this approach we will discuss in the final section. Using the quality assurance [QA] band of the Landsat 7 collection 2 data, allowed us to sort out all pixels in the area of interest and over the period of time containing either cloud shadows or clouds. Therefore, the composite satellite image uses only pixels without clouds or cloud shadows. Finally, the medium resolution of open satellite images can hinder the extraction of small features. Therefore, CNN architecture has to be chosen so that little information is lost with tools like skip connections between encoder and decoder substructures or avoiding max pooling, as explained below.

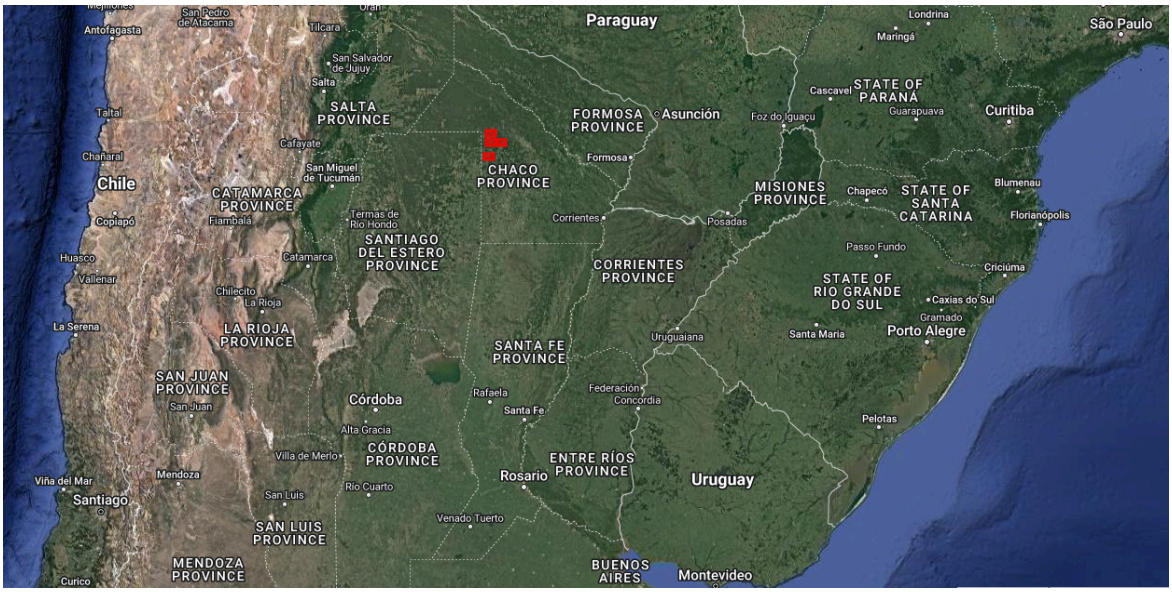
[1] Public GitHub repository: https://github.com/agustinapesce/deforestation_project_DLSS

II.2 Our satellite data (satellite specificities and extraction)

In this work, we used open Landsat 7 images with a 15-meter spatial resolution (Landsat Missions, 2024). This satellite has suffered a failure in its compensation of motion and therefore produces images with black traces (see “usgs_landsat” folder). Due to this drawback, the images were downloaded from the post-processed Google Earth engine data catalog[2] instead of the direct USGS site. The surface reflectance data for the years 2014 and 2019 were downloaded, including several spectral bands and quality assurance (QA) bands. The key spectral bands used in this study include: SR_B3 (Red band), SR_B2 (Green band) and SR_B1 (Blue band). These bands are combined to create true color images that visually represent the Earth's surface as seen by the human eye.

The investigated area was the north of Argentina in the province of Chaco (Figure 1) due to personal preferences and the abovementioned deforestation activities following a growth in agriculture in that region even when part of the native forest conservation program.

Figure 1. Area of analysis: North Argentina, Chaco province area.



We selected four regions for which we decided manually that deforestation activities could be observed between 2014 and 2019 (Figure 2).

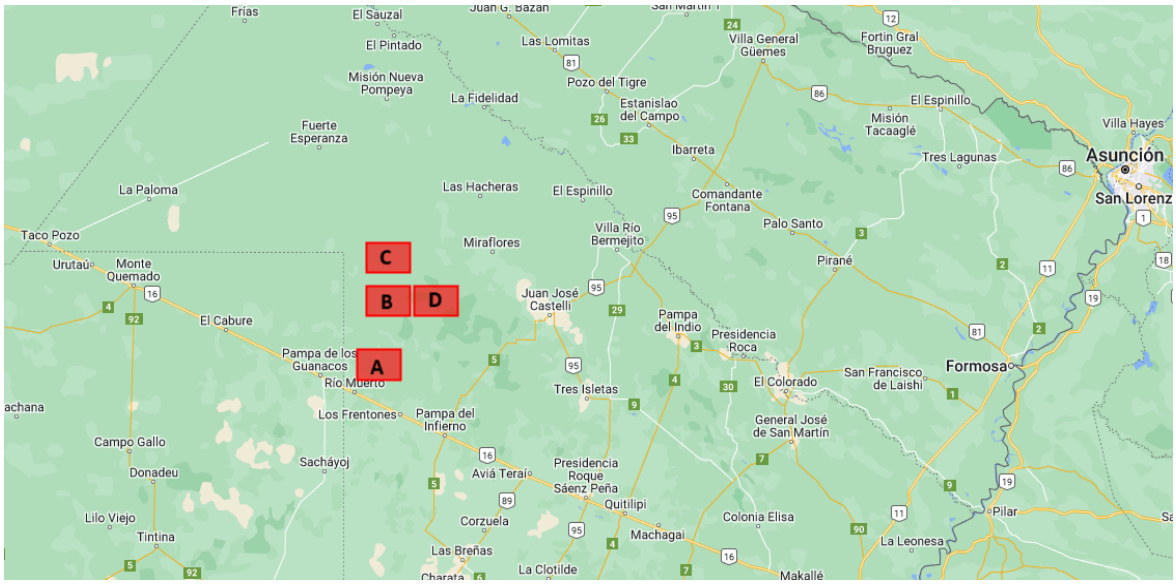
Georeferences of the four regions: *Region X = [(South-West), (North-East)]*

- Region **A** = [(-61.64, -26.25), (-61.41, -26.11)]
- Region **B** = [(-61.59, -25.95), (-61.36, -25.81)]
- Region **C** = [(-61.59, -25.75), (-61.36, -25.61)]
- Region **D** = [(-61.34, -25.95), (-61.11, -25.81)]

[2] https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LE07_C02_T1_L2

For each region, two satellite images were downloaded, one for each year (see Google Earth Engine Code, Appendix). The dimensions for each of the eight images are 1040 pixels high, 1708 pixels wide with three bands for RGB.

Figure 2. The four georeferenced areas used for model training, validation and testing.



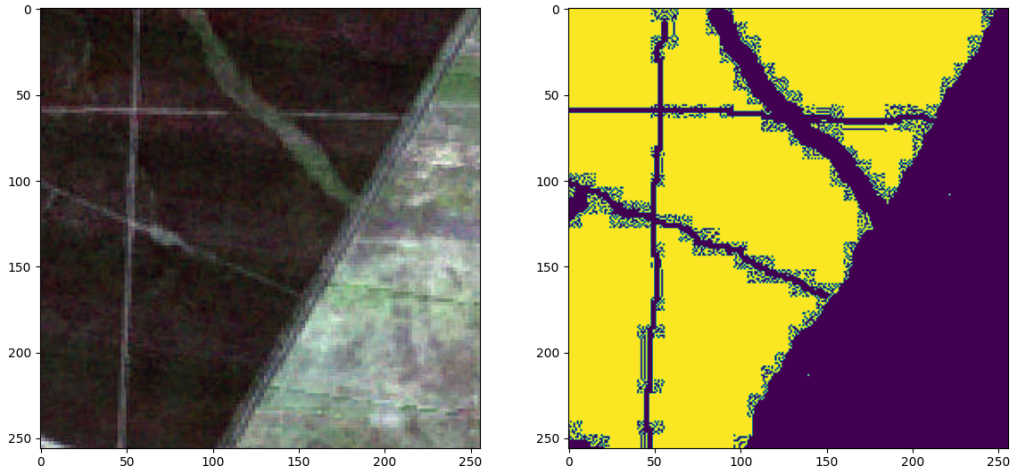
II.3 Masking, image preprocessing, datasets creation

To learn more about the hurdles of image data processing and ensure the compatibility of our training data with the region of interest, we created masks for the downloaded data. The Python package “label-studio” (1.13.1 version) was implemented to create binary labels signaling where in the images forest was present. This requires an account at label studio (free of charge). In this step, we gain many insights into the difficulties for non-specialists to be sure about the presence of forests in all areas of the image. Furthermore, the interface from the labeling open package lacked precision for small features.

To preprocess the images and masks, we divided them into patches in the standard measure of 256x256 pixels. From the masks, label arrays were created such that black pixels, the color that indicated the presence of forest for our masks, were set to a value of 1 and all the rest of the image to 0 (Figure 3, right). In Figure 3 it can be seen that this workflow takes to some noisy areas in the border between the two labels (see white noise points), however, these did not represent a big proportion of the image pixels, and therefore, the risk of inserting bias in the models was low. As a last step, the “to_categorical” function from “tensorflow.keras.utils” was implemented to create a one-hot vector for the categories. Although this is not strictly necessary for binary data with just 0 or 1 outputs, we decided to keep the step to allow for future more flexible implementations.

Each patch of the RGB satellite images was normalized with a min-max scaling to emphasize contrasts between forest and non-forest representations (Figure 3, left).

Figure 3. Example of a 256x256 patch normalized satellite image (left) and labeled mask (right).



Different datasets were created for training, validation and testing. Following De Bem et al. (2020) approach, different geographical areas were included in each of the datasets. Areas A and B account for the training data resulting in 96 patches. Area C accounts for validation data and finally area D for testing, each of them consisting of 48 patches.

II.4 Data augmentation

After running our baseline training with the abovementioned training set, we decided to multiply our data using a data augmentation strategy. We decided to rotate the satellite images three times (90°, 180°, 270°). Since the data quadruplication took very long training times for the models, which complicated the exploration of their best-performing hyperparameters, we used the 90° and 180° augmentations for the first model and none for the second, which required even longer training time. Nevertheless, it should be noted that the inclusion of the augmented data in the first model did not lead to a significant increase in its performance, and we therefore inferenced it would not take too big insufficiencies to the second model.

III. CNN architectures

Two different CNN architectures were implemented to complete the segmentation task to assess current approaches performance for our topic of interest. We first created a U-Net neural network, this architecture was created to overcome the complications of convolutional neural networks on tasks requiring localisation (Ronneberger et al., 2015). To do this, upsampling operators are used to increase the resolution of the max-pooled output from the decoder and skip connections are inserted to propagate information from the decoder layers to the encoder layers. Upsampling is executed by transposed convolutions of 2x2 that counteract the previous 2x2 max pooling layers. Moreover, “long” skip connections or bridges are concretely made by concatenating the previous decoder output and the corresponding output within the encoder.

More specifically, the U-Net model encoder consists of blocks with two convolutions of 3x3 filters each, a dropout mechanism after the first convolution to avoid parameters overfitting and a max pooling of 2x2 at the end of the block to reduce outputs' dimensionality. The decoder mirrors the encoder, with the difference that its convolutional blocks first have the upsampling transposed convolutions, also with 2x2 stride, followed by the bridge mechanism. All layers have ReLU as activation function and padding was set to "same" to keep the images' proportions compatible along the blocks. The details of the U-Net architecture such as the number of blocks and filters per layer, is available in "report_figures/model1.png".

It has been argued that the downsampling implemented by U-Net can also be insufficiently sensitive for lower-resolution images (Poliyapram et al., 2019). These authors, therefore, proposed creating a CNN without max pooling and also added dilated convolutions, also known as "convolution with a dilated filter" (Yu & Koltun, 2016; see paper's Figure 1), where filters are not implemented to adjacent pixels but with spacing between them. This action increases the surrounding or contextual information without loss of resolution or coverage.

As in the reference paper, the final architecture for model 2 consisted of a first block of four convolutional layers with no dilation (also known as 1-dilated filters), three pyramid dilation layers with dilations of 2, 4, and 6, and four additional layers with no dilated filters and two skip connections with the first block. Filters were all of 3x3, activation functions were set to ReLU and padding to "same" as in model 1. The details on the whole model 2 network architecture can be seen in "report_figures/model2.png".

III.1 CNNs training hyperparameters

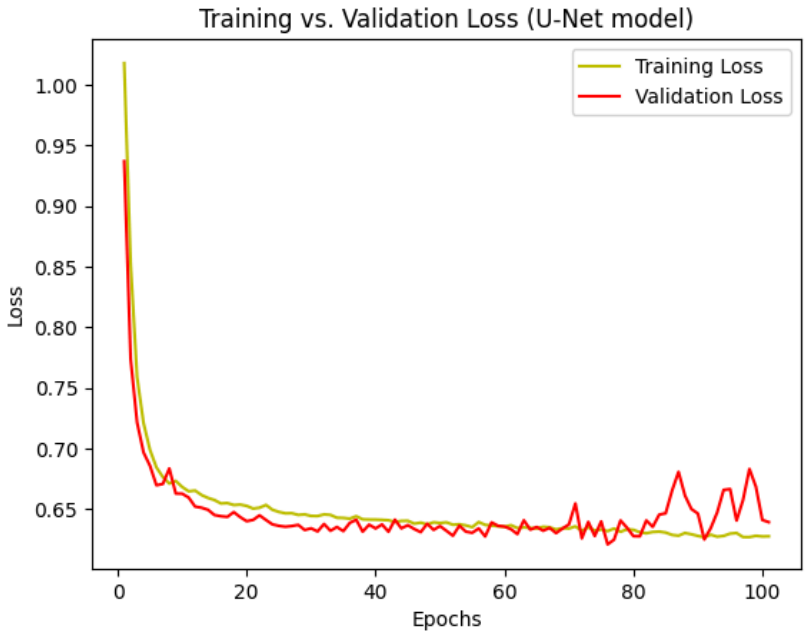
Regarding the general specifications in the networks training, in both models, kernels were initialized with the method proposed by He et al. (2015), the final learning rate was set to 0.001 and batch size was 16 patches for model 1 and 6 for model 2. In model 1, training optimization had in all layers a L2 regularization of 0.0001. In model 2, batch normalization (BN) was added before the activation functions; this method is meant to decrease training time by normalizing not only the input layer but also the input of each layer in the network (Garbin et al., 2020). However, BN are one of the most computationally intensive non-convolutional layers (Jung et al., 2019), which should be taken into account in the future when implementing a method with no down-sampling.

Finally, as in the de Bem et al. (2020) paper, both models used focal loss as loss function to avoid possible problems with uneven categories. While model 1 used adaptive moment estimation (ADAM) algorithm for gradient descent optimization, model 2 used Stochastic Gradient Descent (SGD).

For both trainings an early stopper was implemented with a patience of 25 epochs which monitored the validation data loss output for the epochs. This way we avoided model overfitting to the training data along the maximum number of epochs. The best model for the validation data was saved and kept as the final model.

Regarding training time, training epoch duration for model 2 almost tripled that of model 1 although a third of the training data was used. Despite avoiding max pooling seems a right logic to more detailed segmentation outcomes and BN is deemed helpful for model convergence, this type of architecture should be implemented when great computer power is at hand.

Figure 4. Training curves for the best performing U-Net model for the validation data



It should be noted, however, that when observing model 2 validation loss curve ("report_figures/tr_val_loss_m2_300epochs.png"), it seems there was space for improvement for the model's parameters to get to maximize performance for this architecture. Since this model implements BN, higher learning rates could be applied in the future to optimise the model in fewer training steps (Garbin et al., 2020).

III.2 Metrics to assess classification results

As metrics to assess the model's performance, confusion matrices for the classification performance were visualized together with accuracy, precision, recall, F1, and the Jaccard index, also known as Intersection over Union (IoU) (de Bem, 2020). This last metric is calculated by dividing the overlap of the predicted and true area (true positives) with the union of these two regions (true positives, false positives, and false negatives).

IV. Change detection

To detect the change over time we are comparing the two outputs our model generated with each other by comparing the shares of pixels labeled by the CNN as forest/non-forest pixels over the total pixels of the output mask. Since the CNN outputs a binary black and white mask for a given new satellite image in a given year, the change in shares of two satellite images from

two years accounts for the change detected by our model. With a white pixel meaning the algorithm predicts forest and a black pixel meaning no forest predicted.

$$CHANGE = N_{px:\, class = forest,\ time = 1} - N_{px:\, class=forest,\ time = 2}$$

V. Results

Our results will be presented in a twofold fashion. First, we will introduce the performance of the different models we trained, afterwards we will present the output masks our CNN model provided, finally moving on to the pixel change analysis.

V.1 Performance comparison of the two models

Table 1. Metric Comparison of the two models

Metric	Model 1	Model 2
Loss	0.6189	0.8144
Accuracy	0.8778	0.821
Jaccard Index (IoU)	0.7184	0.6334
Precision (No_Forest)	0.81	0.7
Precision (Forest)	0.92	0.9
Recall (No_forest)	0.84	0.83
Recall (Forest)	0.9	0.81
F1-Score (No_forest)	0.83	0.76
F1-Score (Forest)	0.91	0.86
True Negative Rate (No_forest)	0.84	0.83
False Positive Rate (No_forest)	0.16	0.17
False Negative Rate (Forest)	0.1	0.19
True Positive Rate (Forest)	0.9	0.81

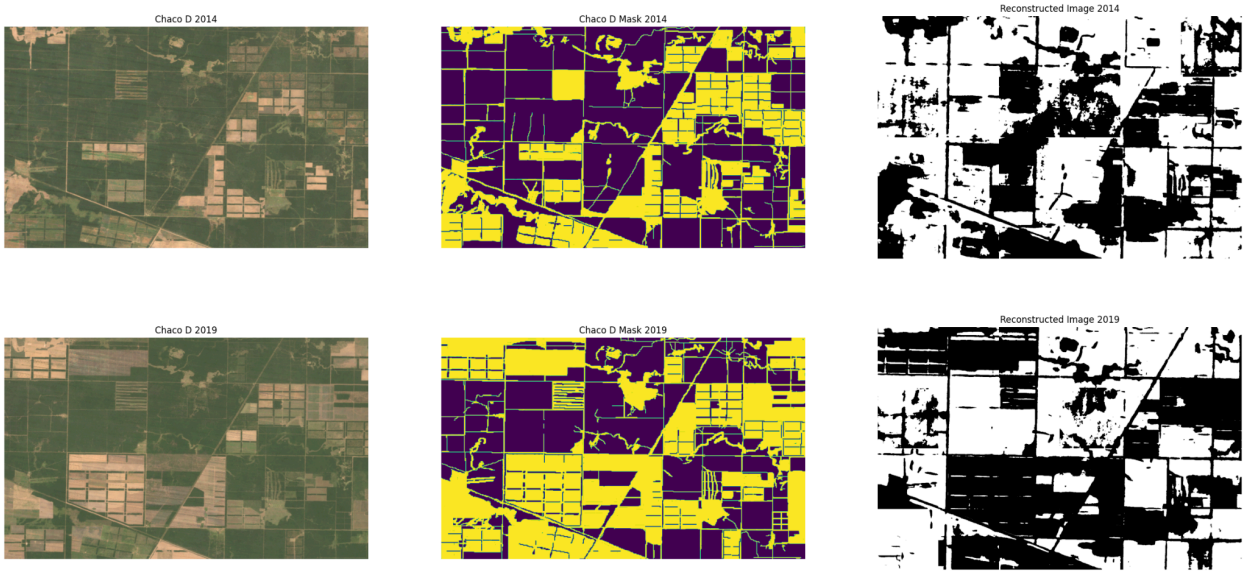
Based on the evaluation metrics and confusion matrices, it is clear that the first model outperforms the second model across multiple dimensions. The first model achieves a higher accuracy (87.78% vs. 82.10%), a better Jaccard Index (71.84% vs. 63.34%), and a lower loss (0.6189 vs. 0.8144), indicating superior overall performance. The confusion matrix for the first model shows a higher true positive rate for the "Forest" class (90% vs. 81%) and a lower false negative rate (10% vs. 19%), suggesting that it is more effective in correctly identifying forested areas. Additionally, the first model also has a slightly better true negative rate (84% vs. 83%) and a lower false positive rate (16% vs. 17%), indicating it is more accurate in distinguishing

non-forested areas. Therefore, considering both the evaluation metrics and the confusion matrices, the first model is clearly the better performer, making it more reliable for the classification task at hand.

V.2 Results for output masks

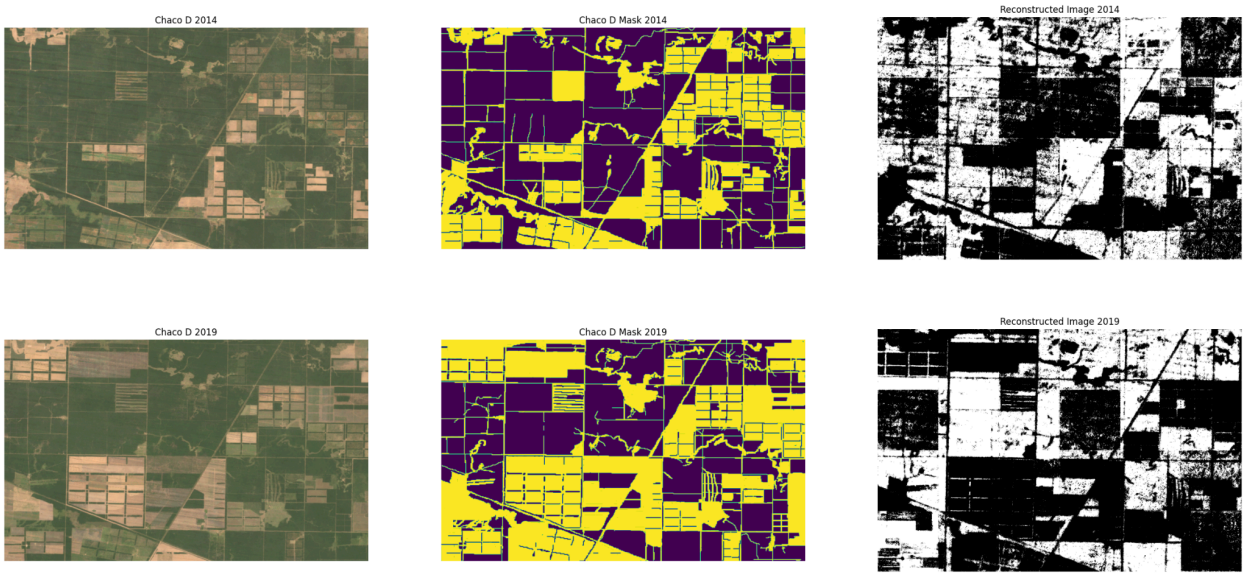
In this section we will present the masks produced by the CNNs. As can be seen in Figure 5. the output mask provided by the first CNN does a fairly good job in detecting the forest/non-forest areas. Not very surprisingly deforested areas characterized by lightgreen appearance account for flaws in detection, whereas bright brown areas are easily detectable. The first model also does have some issues with slim paths through the forest, as can be seen in the 2014 classification mask, especially for the diagonal path.

Figure 5. Model 1: Output masks compared to the original satellite images and manually labeled masks.



In Figure 6. we show the results for the second model. What is striking is that the second model provides a more fine grained mask. It seems to understand finer shapes better. On the other hand it also includes more black pixel clouds (no-forest) in areas that have been manually annotated as forest. It appears as if the second model rather assesses every pixel individually, while the first model tries to find underlying shapes in the satellite image. It therefore more often and due to the ambiguous nature of forest detection, as we as annotators experienced as well, fails to correctly classify a pixel. Even though this approach leads to, in our opinion, better output masks, that is, more precise representations of the satellite image, it incorporates more mistakes, which in our opinion also leads to the worse performance metrics.

Figure 6. Model 2: Output masks compared to the original satellite images and manually labeled masks.



V.3 Results for change detection

As described above the change detection will be the difference over the years 2014 and 2019 (time = 1, 2) in pixels classified by the CNNs as forest.

$$CHANGE = N_{px:\, class = forest,\ time = 1} - N_{px:\, class=forest,\ time = 2}$$

In Table 2 we compare the different changes in forest coverage detected or classified on our test data (area D specified above) by our two CNN-models with the ground truth. Therefore the ground truth also was transformed from its RGB form into a binary form changing all pixels above 128 to 1 and everything else to 0.

Table 2. Change in forest in area D over ground truth and our two models.

	Pixelshare, class = forest, time = 1	**Pixelshare, class = forest, time = 2**	**Total Change**
Ground Truth	*62.32%*	*48.87%*	*-13.45%*
Model 1	*58.70%*	*50.68%*	*-8.02%*
Model 2	*61.02%*	*56.42%*	*-4.60%*

Again model 1 is closer to the ‘actual’ change annotated by us manually. The argument prevails that due to the individual pixel classification in contrast to the underlying shape detection a lot of

wrong single pixels distort the results. But since for all numerical metrics model 1 exceeds model 2, we are going to select model 1 as our best performing model.

VI. Limitations

The study has several limitations that could impact the results. First, regarding the data, the use of medium-resolution satellite images (15 meters) may hinder the detection of smaller features and fine-grained details, potentially leading to inaccuracies in forest classification. In addition, many works that deal with satellite images use a greater number of bands, which could also enhance input information for model detection.

Secondly, in what entails data post-processing, the reliance on manually labeled masks done by non-experts introduces the possibility of human error and subjectivity, which could affect the training and evaluation of the models. Even in cases where experts participate in the process, a workflow could be implemented where all masks are done by at least two of them and a final mask that only incorporates areas where both annotators indicated deforestation was present. The data augmentation strategy, while helpful for increasing the training dataset, may also introduce biases that do not necessarily represent real-world variations. As seen in Figure 5, most deforestation lines are done in what is seen as a horizontal line in the satellite frame.

Lastly, concerning models' training, the computational intensity of training CNN models, especially the more time-intensive architecture without max pooling and with BN, limited the exploration of hyperparameters and model configurations, which could have improved performance further. Also, as mentioned in III.1, greater learning rates could be implemented for model 2 to arrive at optimum parameter values in fewer training time steps. Finally, ample space for exploration of different model architectures is still in place, such as implementing downsampling techniques that involve increased stride for the convolutional filters or additional skip connections.

VII. References

Cuadra, D. E., Insaurralde, J. A., & Montes Galbán, E. J. (2020). Evaluación espacio-temporal de la deforestación en el noroeste de la provincia del Chaco (1986-2018): mediante el uso combinado de Sistemas de Información Geográfica y Procesamiento Digital de Imágenes.

De Bem, P. P., de Carvalho Junior, O. A., Fontes Guimarães, R., & Trancoso Gomes, R. A. (2020). Change detection of deforestation in the Brazilian Amazon using landsat data and convolutional neural networks. *Remote Sensing*, *12*(6), 901.

Garbin, C., Zhu, X., & Marques, O. (2020). Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia tools and applications*, *79*(19), 12777-12815.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026-1034).

Jung, W., Jung, D., Kim, B., Lee, S., Rhee, W., & Ahn, J. H. (2019). Restructuring batch normalization to accelerate CNN training. *Proceedings of Machine Learning and Systems*, *1*, 14-26.

Landsat missions (2024). Landsat 7 https://www.usgs.gov/landsat-missions/landsat-7

Montenegro, C., Strada, M., Bono, J., Gasparri, I., Manghi, E., Parmuchi, E., & Brouver, M. (2005). Estimación de la pérdida de superficie de bosque nativo y tasa de deforestación en el norte de argentina. *Buenos Aires, UMSEF Unidad de Manejo del Sistema de Evaluación Forestal, Dirección Bosques, Secretaría de Ambiente y Desarrollo Sustentable*.

Poliyapram, V., Imamoglu, N., & Nakamura, R. (2019, July). Deep learning model for water/ice/land classification using large-scale medium resolution satellite images. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium* (pp. 3884-3887). IEEE.

Preligens (2019). The real life of an Earthcube data scientist. https://www.youtube.com/watch?v=62mHkK_qT0I&t=209s

Yu, F., & Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

VIII. Appendix

GitHub repository:
Access: public
https://github.com/agustinapesce/deforestation_project_DLSS

GoogleEarthEngine [GEE] Code:
Access: GEE Sign-in necessary.
https://code.earthengine.google.com/6ef1f7468379c85d474c01d0e499abce
And in GitHub Repository