Report on Detecting and Counting Palm Trees Using Drone Images in Ecuador

1 Introduction to the Palm Dataset

1.1 Initial Labeled Data Preparation

We have been provided with two sets of data. The first set comprises an orthomosaic image, generated by GIS software. This software integrates individual drone images, considering their overlap, to form a unified, large-scale image, aided by longitude and latitude information. The second set consists of the locations of palm tree centers, which are utilized to crop tiles from the orthomosaic image of palm structures. Each cropped tile measures approximately 264 by 262 pixels in size, with the labeled palm center positioned at the center of the cropped tiles.

(a) Isolated Palm (b) Palm Obscured by Trees (c) Overlapping Palms

Figure 1: Representations of Various Palm Scenarios: An isolated palm (left), a palm obscured by surrounding trees (center), and overlapping palms (right).

Since the cropped tiles may not always fully encompass the palms (see Figure 1), we observe instances of well-isolated palms, palms obscured by surrounding trees, and palms overlapping with each other. These factors contribute to the low quality of the palm images, leading us to opt for a more efficient approach to capture palm features. The decision is to use smaller patches that can reveal the unique structure of palm leaves, making them more distinct from other tree crowns. Therefore, we divide the images into 40 by 40 pixel patches and select those in which palm coverage accounts for at least 90% of the area. We also choose

patches devoid of palm leaves, ensuring that no palm elements are included. Through this process, we end up with approximately 6,000 patches in total, with around 4,800 instances for palm and 1,200 instances for non-palm. The non-palm category encompasses a mixed assortment of other trees crowns, ground, road, trunks, etc. Meanwhile, there are blurred parts in the orthomosaic caused by the reconstruction of the images. Such blurriness can lead to misclassification, and since these regions are not worth predicting, we have chosen to delete them, see Figure 2.

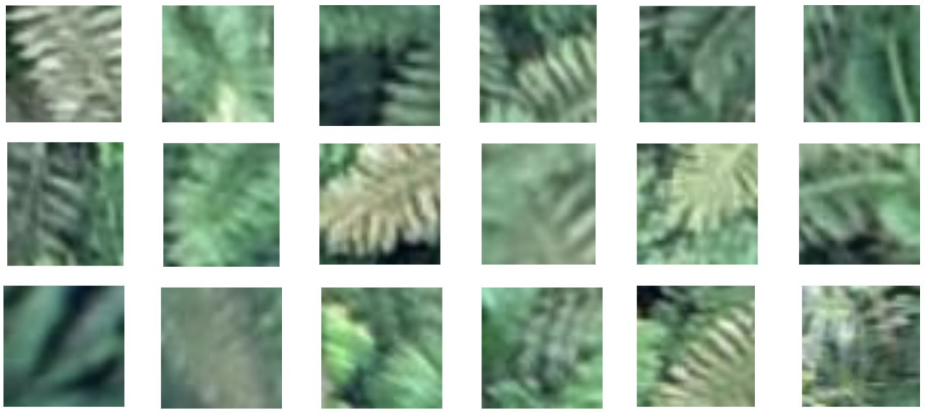
(a) The Original Orthomosaic Image

(b) The Cleaned Orthomosaic Image

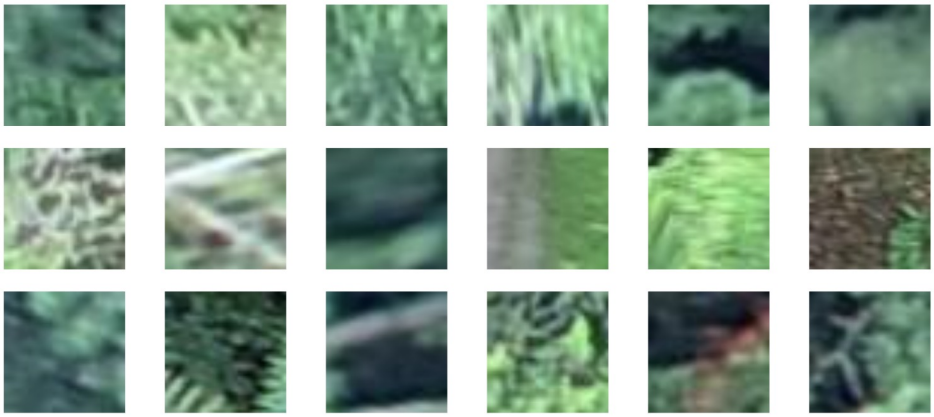
Figure 2: Visualizations of the orthomosaic image before and after cleaning.

1.2 Labeled Small Patches Refinement

Although we initially obtained 6,000 patches, each measuring 40 by 40 pixels, the data was imbalanced, leading the model to preferentially predict patches as palm. To address this issue, we introduced additional patches that are isolated in the landscape-level prediction map, specifically on the cleaned orthomosaic image. We added patches that were either predicted as palm but surrounded by non-palms (false positives) or vice versa (false negatives). In other words, we enhanced the training set by incorporating both false positive and negative samples to mitigate the imbalance and to train a more robust model. Additionally, some of the tree crowns are similar to palms in either leaf structure or color, potentially leading to misclassification by the classifier. To overcome this challenge, we manually picked the patches of such trees and added them as non-palm samples. By doing the above two jobs, we enlarged and balanced the dataset, resulting in around 5,900 instances each of palms and non-palms. The patches we labeled are visualized in Figure 3. The top image illustrates small patches containing palms, capturing the distinctive characteristics of the palm tree. The bottom image, on the other hand, showcases regions without palms, highlighting the variations that help the model to differentiate between palms and non-palms.



(a) Small Patches of Palms



(b) Small Patches of Non-Palms

Figure 3: Comparison of Small Patches: The top image displays regions containing palms, while the bottom image showcases regions without palms.

1.3 Labeled Large Patches

The construction of the dataset was aimed at addressing the challenges posed by the use of small patches, which could lead to false predictions or the misidentification of other types of tree crowns as palms. The adoption of large patches was intended to efficiently identify these discrepancies. Upon obtaining the probability heatmap from the trained network using small patches on the large landscape orthomosaic image (see Section 2.3), a 100×100 pixel sliding window was employed with a stride of 50 pixels to identify and label high and low probability regions. Thresholds for these probabilities were set at 5000 and 200, defined by the sum of probabilities inside each sliding window. Each window was evaluated based on the sum of its probabilities: if the sum exceeded the high threshold, the patch was categorized as a high probability region; if below the low threshold, it was considered a low probability region. Before assigning the label of non-palm to a patch, a filter was applied to ensure that

the patch did not contain more than 30% white or black pixels, effectively eliminating areas that might contain large empty white regions. The patches that met these criteria were then segregated into high and low probability directories for further manual inspection and labeling.

This process culminated in the manual labeling of high and low probability patches as palms and non-palms, resulting in approximately 1300 samples for each category (See Figure 4). Notably, some palms were identified within the low probability patches, and some non-palms within the high probability ones, highlighting the efficacy of using large patch datasets in the accurate identification of tree crowns.

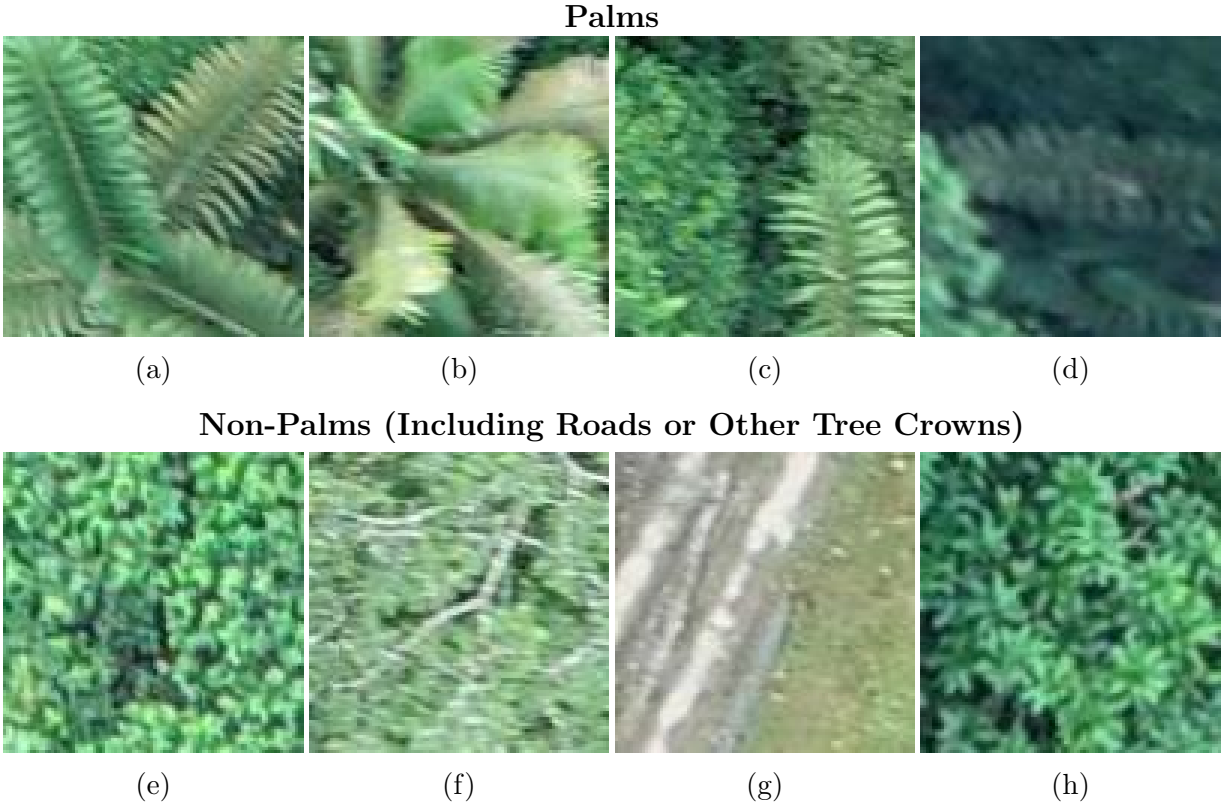


Figure 4: Large Patch Datasets in Identification: The first row displays patches containing palms, while the second row showcases regions with non-palms including roads or other tree crowns.

2 Pipeline

2.1 Feature Extraction

In the domain of deep learning, particularly for tasks involving image classification, transfer learning has emerged as a powerful strategy. Transfer learning leverages the knowledge gained while solving one problem and applies it to a different but related problem [1]. This

approach is highly beneficial when there is a lack of sufficient labeled data for the specific task at hand.

The Residual Network (ResNet) is a deep learning architecture that has made significant strides in addressing the vanishing gradient problem that hampers the training of very deep networks [2]. By introducing skip connections, or shortcuts, ResNet allows gradients to flow through the network more efficiently. The specific version utilized in this study, ResNet-18, consists of 18 layers, making it a relatively shallow variant within the ResNet family. The model has proven its efficacy in extracting intricate patterns and features from images.

For this palm detection and counting project, we do not have a large labeled dataset, making training a network from scratch impossible. Therefore, we use a pretrained ResNet to extract features from the 40 by 40 patches we cropped. By employing a pretrained model, one can capitalize on features that have already been learned from a substantially large dataset. In this study, we exploit the advantage of transfer learning by utilizing a pretrained ResNet-18 model. ResNet-18 was initially trained on the ImageNet dataset, a large-scale dataset encompassing over a million labeled images across a thousand categories [3]. Thus, it has learned a robust hierarchy of features that can be instrumental in various downstream classification tasks. Utilizing the pretrained ResNet-18, we extracted feature vectors from our dataset of palm and non-palm images, each feature vector with the shape 1000×1. This approach not only saved computational resources but also leveraged a well-proven model to obtain a rich set of features, thus laying a solid foundation for our subsequent analysis.

2.2 Classification

To classify the palm images based on the extracted features from ResNet, we employed a multilayer perceptron (MLP). An MLP is a type of feedforward artificial neural network that consists of at least three layers of nodes [4]. Unlike a single-layer perceptron, an MLP can capture complex patterns in data by introducing non-linearity. The architecture of our MLP consists of the following:

- **Input Layer:** Comprising 1000 nodes, corresponding to the features extracted from ResNet18.
- **Hidden Layer:**
  - *Fully Connected Layer (Linear):* This connects to the input layer with `nnodes` neurons.
  - *ReLU Activation:* Introduces non-linearity using a rectified linear unit.
  - *Batch Normalization:* Normalizes neuron activations across the mini-batch, improving training speed, and reducing overfitting by stabilizing input distributions.
  - *Fully Connected Layer (Linear):* A secondary fully connected layer with 2 output neurons.
- **Output Layer:** Utilizes a logarithmic softmax function `nn.LogSoftmax(dim=1)` for two-class classification.

The MLP's specific architecture, including the batch normalization layer, allows the model to learn from the feature vectors extracted from ResNet, mapping them to the final class labels. The parameter `nnodes` in the hidden layers can be user-defined, and in our experiment, it was set to 64, 128, and 256. Figure 5 shows the workflow of the study.

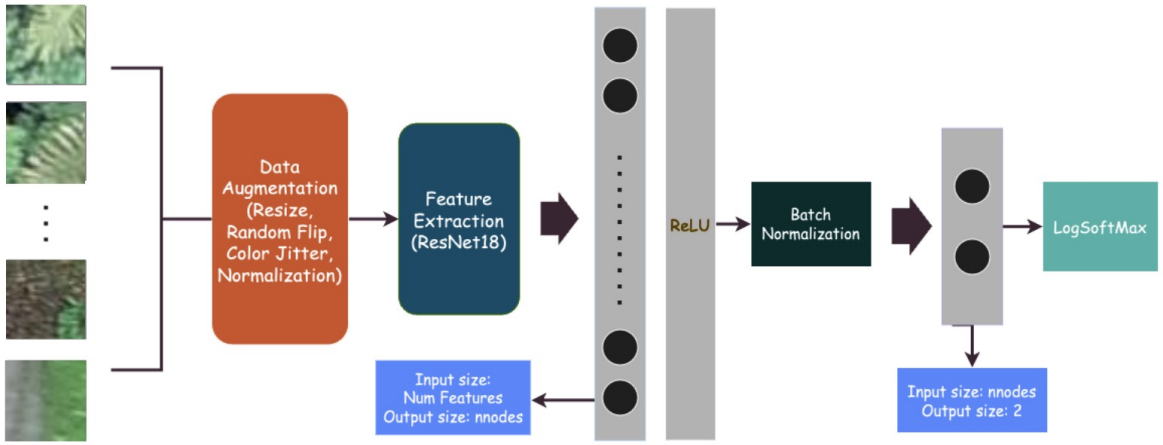


Figure 5: Workflow Pipeline of Palm Classification: The process begins with palm patch inputs, followed by data augmentation. Features are then extracted using a pretrained ResNet18 model, and passed through a sequence of layers, including a dense layer with `nnodes` nodes, batch normalization, and a final dense layer with 2 nodes for binary classification.

2.3 Application to the Landscape Orthomosaic Image

2.3.1 Application of the ResNet18 Trained by Small Patches

The application of a trained ResNet18 model to large landscape orthomosaic images involves several stages for the identification of palms, as detailed below. In the initial stage, the large image is loaded and preprocessed. Hyperparameters such as the patch size (40×40) and stride (10) are defined. The image is subsequently divided into small patches of 40×40 pixels, with patches containing 40% or more black or white pixels labeled as 'non-palm' for immediate classification. The full image is further segmented into 30 individual images, each of size 5000×5000, to be analyzed and later recomposed. The trained ResNet18 model is then loaded for feature extraction, followed by a multilayer perceptron (MLP) for binary classification.

During the next phase, the extracted patches are sequentially processed by the ResNet18 model. Features from each patch are used for predictions using MLP, and the output probabilities are collated into an accumulator array. This technique ensures an average probability prediction for each region, taking overlapping patches into account.

Upon completion of the predictions, the accumulated probabilities are transformed into a heatmap by undergoing a process of resizing and normalization. The heatmap is overlaid onto the original small images, highlighting the areas containing palms. This operation is performed iteratively for all 30 small images, each of size 5000×5000, generating a visual representation of palm locations. The individual highlighted images and corresponding heatmaps are assembled to form a new large image and a comprehensive heatmap, respectively. These are then cropped to match the original large image size, preserving the size of the initial landscape orthomosaic. The final results, including the large highlighted image indicating the palm regions and the large heatmap, are saved for in-depth analysis, capturing all computed probabilities.

2.3.2 Application of the ResNet18 Trained by Large Patches

In addition to the application using 40×40 pixel patches, the process can also be applied using larger patches of 100×100 pixels. When the image size is changed to 100×100, the rest of the pipeline remains consistent with the methodology described in the previous section.

3 Experiment

3.1 Data Division and Augmentation

To ensure a fair evaluation of the model, we divided our dataset with 40 by 40 patches into two parts: 80% for training and 20% for testing. The training set was further used to perform 5-fold cross-validation, allowing us to assess the model's performance more reliably.

In the field of machine learning, particularly in tasks dealing with image data, data augmentation is essential to enhance the performance and robustness of a model. Data augmentation helps the model to generalize better, thus reducing the risk of overfitting, especially when the available labeled dataset is limited. For our palm detection and counting project, the following preprocessing methods with augmentation were implemented:

- **Resize:** The images were resized to 224×224 pixels to make them compatible with the pretrained ResNet-18 model.
- **Random Horizontal and Vertical Flip:** These two transformation mirror the image either horizontally or vertically, each of these effectively doubling the data and allowing the model to learn from different orientations.
- **Color Jitter:** By randomly changing the brightness, contrast, and saturation of the image, we enabled the model to be more invariant to slight changes in coloration, simulating various lighting conditions.
- **Normalization:** This step is essential for scaling pixel values to a standardized range, aiding in faster convergence during training.

These specific augmentations were chosen based on the nature of the data and the task. For the test set, we avoided augmentations that might alter the intrinsic characteristics of the images, preserving consistency with the model's expectations.

3.2 Model Training

For our model training, we primarily focused on the multilayer perceptron (MLP), tuning its parameters while freezing those in ResNet to ensure effective feature extraction. The training process was initiated with 5-fold cross-validation. During each fold, the training data was augmented using previously described techniques, leaving the validation data unaltered. To accelerate the computation, we submitted slurm jobs on a Tesla V100 GPU node of the DEAC Cluster. The main training phase utilized the Negative Log-Likelihood loss function, a commonly used criterion that measures the model's ability to predict the correct classes.

The core training phase consisted of 500 epochs for small patches and 100 epoches for large patches, employing a batch size of 64. Throughout this stage, we carefully monitored validation loss, using it as a criterion to select the best model, marked by the lowest loss across epochs and folds. After identifying the best model, we proceeded to retrain it on the full 80% training data, adhering to the earlier applied augmentation strategies. Subsequently, the model was tested on the remaining 20% that had been preprocessed without augmentation, facilitating evaluation on unseen instances. This methodical training approach, combining cross-validation, diligent data division, and validation loss emphasis, established a robust evaluation framework. It enabled precise model selection and maximal utilization of the labeled data, thus optimizing overall performance and reliability.

3.3 Evaluation Metrics

To evaluate the performance of our model, we utilized several metrics:

- **Accuracy:** A straightforward metric that quantifies the proportion of correctly classified instances out of the total number of instances.
- **Kappa:** Cohen's Kappa score evaluates the agreement between the predicted and actual classes, taking into account the possibility of agreement occurring by chance.
- **Precision:** This metric quantifies the proportion of true positive predictions among the total positive predictions.
- **Recall:** Recall assesses the proportion of actual positives that were correctly classified by the model.
- **ROC AUC:** The area under the Receiver Operating Characteristic (ROC) curve, representing the model's ability to discriminate between the classes.

3.4 Numerical Results

In the process of evaluating the numerical results, both pretrained models, corresponding to large and small patches, were applied to the landscape orthomosaic image. Utilizing a stride of 10 during this application ensured that the models were efficiently able to scan and analyze the entire image.

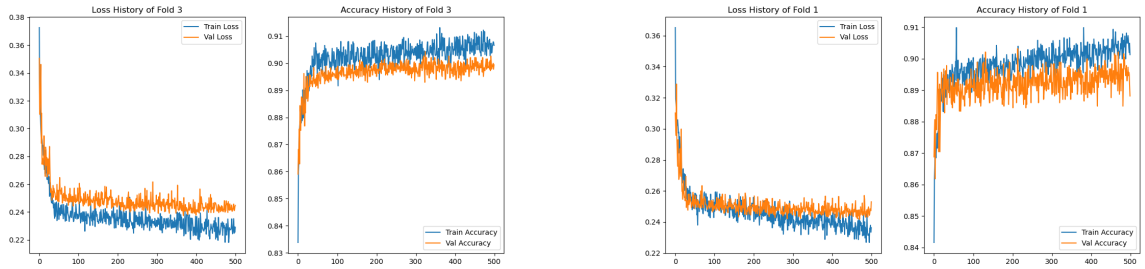
3.4.1 Small Patches Classification

In the classification of small patches, three different configurations of hidden layer nodes (`nnodes`) were tested: 64, 128, and 256. The Table 1 below encapsulates the performance across various metrics including training, validation, and test accuracies, ROC AUC, average accuracy, precision, and recall.

`nnodes`	Train Acc	Val Acc	Test Acc	ROC AUC	Avg Acc	Precision	Recall
64	90.21%	88.97%	91.01%	0.9642	91.05%	91.26%	91.01%
128	90.59%	**89.78%**	90.41%	0.9653	90.47%	90.99%	90.41%
256	**91.11%**	89.13%	**91.10%**	**0.9674**	**91.14%**	**91.32%**	**91.10%**

Table 1: Results of Small Patches Classification for Different Hidden Layer Configurations

Analysis of the results in Table 1 reveals a consistent performance across different configurations. All three settings demonstrated substantial success in classifying small patches, with metrics such as accuracy, precision, and recall remaining relatively stable. There is a slight trend towards improvement with an increase in hidden layer nodes, with the 256-node configuration yielding the highest values in most metrics. However, the improvements are incremental, suggesting that further tuning might be required to discern significant differences. The optimal threshold based on the ROC curve varies between the configurations, indicating a need for a careful selection depending on the specific application or desired trade-off between precision and recall. Overall, these experiments demonstrate robustness in the classification approach, with different hidden node configurations providing comparable performance.



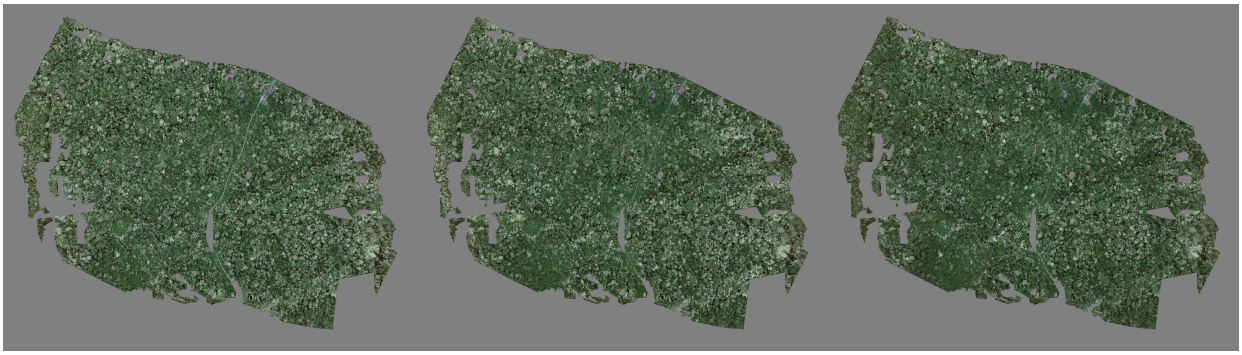
(a) Third Fold with `nnodes` = 64

(b) First Fold with `nnodes` = 128

Figure 6: Training Process Visualization: Figure 6a represents the third fold using 64 hidden layer nodes, and Figure 6b represents the first fold using 128 hidden layer nodes. Both plots include the training and validation loss and accuracy.

Figure 6 illustrates the variations in loss and accuracy during the training process for different folds and configurations, specifically the third fold with 64 hidden layer nodes and the first fold with 128 hidden layer nodes. Both training and validation loss and accuracy are displayed. Notably, there is no evident sign of overfitting in either of the configurations. This favorable outcome can be attributed to two crucial factors. Firstly, the parameters within the ResNet18 were frozen during the training process because we have limited amount of

labeled data, meaning that only the MLP part of the network was subjected to training. This approach ensures a stable and robust feature extraction, preventing the model from fitting too closely to the training data. Secondly, the implementation of data augmentation techniques has further assisted in enhancing the model's generalization capability. By artificially expanding the dataset through various transformations, data augmentation increases the diversity of training examples, thereby reducing the risk of overfitting. These strategies collectively contribute to the model's consistent performance across different configurations and validate the chosen methodology.



(a) With `nnodes` = 64 (b) With `nnodes` = 128 (c) With `nnodes` = 256

Figure 7: Large Highlighted Images Indicating Palm Regions for Different Hidden Layer Configurations produced by the network trained on small patches.

Figure 7 showcases heatmaps from large highlighted images, representing the likelihood of palm presence across different regions. In a heatmap, the intensity of a 10×10 area conveys the probability of it being part of a palm, as predicted by the model. Specifically, brighter regions indicate higher probabilities, signifying a stronger belief that the area contains a part of a palm.

Considering three different hidden layer configurations (`nnodes` = 64, 128, 256), the results, while broadly similar, exhibit nuanced differences upon closer inspection. Each configuration efficiently detects a majority of the palm trees, though there are occasions where non-palm tree crowns are erroneously classified as palms. Such instances emphasize the necessity for more refined model adjustments to reduce false positives. A visual evaluation suggests that configurations with `nnodes` = 64 and `nnodes` = 256 marginally surpass the performance of `nnodes` = 128. The consistent detection patterns across diverse node configurations underline the reliability of the feature extraction and classification phases, simultaneously indicating areas where further tuning could elevate model accuracy.

3.4.2 Large Patches Classification

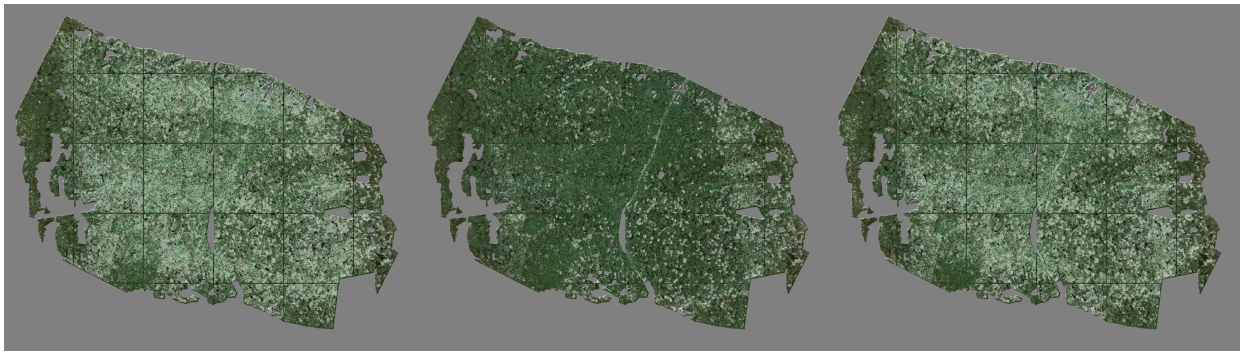
For the classification of large patches, three different hidden layer node numbers (`nnodes`) were also examined: 64, 128, and 256. The Table 2 below summarizes the performance across various metrics as we did for small image patches.

The results from Table 2 show a substantial success in classifying large patches with each configuration, achieving excellent metrics across all parameters. There is a clear variation in

nnodes	Train Acc	Val Acc	Test Acc	ROC AUC	Avg Acc	Precision	Recall
64	**0.9899**	0.9691	0.9696	0.9968	0.9697	0.9698	0.9696
128	0.9798	**0.9810**	**0.9829**	**0.9977**	**0.9829**	**0.9829**	**0.9829**
256	0.9846	0.9691	0.9753	0.9975	0.9754	0.9755	0.9753

Table 2: Results of Large Patches Classification for Different Hidden Layer Configurations

the performance as the number of hidden nodes changes, reflecting a sensitivity in the model to this parameter when dealing with larger patches. Notably, the 128-node configuration seems to outperform the other two slightly in terms of validation and test accuracy, precision, recall, and ROC AUC. However, the differences are not significant enough to decisively favor one configuration over the others.



(a) With `nnodes` = 64 (b) With `nnodes` = 128 (c) With `nnodes` = 256

Figure 8: Highlighted Images of Patches Indicating Palm Regions for Different Hidden Layer Configurations produced by the network trained on large patches.

Figure 8 presents visualizations of the classification results for large patches, with each image corresponding to a different hidden layer configuration. These images highlight the palm regions, as identified by the model when trained on large patches. Upon visual inspection, the configuration with `nnodes` = 128 is clearly superior, outperforming the other two by a notable margin. While the configurations with `nnodes` = 64 and 256 misclassify many non-palm areas and other tree crowns as palm, the 128-node configuration more accurately delineates the target regions.

However, a common issue across all three images is the handling of edges, particularly where the large orthomosaic was divided into 30 separate 5000×5000 images. The visible artifacts at these division seams indicate challenges in managing the transitions between adjacent image sections. This observation emphasizes the importance of using overlap in large-scale image analysis and underscores the need for specialized techniques to handle edge regions more effectively.

4 Future Work

4.1 Result Improvement

In an effort to enhance the model's performance in future iterations, several strategies could be explored:

Expanding the Training Set: Bolstering the training set by labeling more data using the existing workflow could prove beneficial. An expanded dataset can enhance the model's ability to generalize from training to unseen data, thereby improving the accuracy and robustness of the predictions. This strategy is particularly pertinent for the large patch dataset, where each class — palm and non-palm — comprises approximately 1300 samples.

Utilizing Overlapping in Segmentation: During the segmentation of large images into smaller 5000×5000 patches, an overlapping method could be employed to avoid the creation of visible edges in the combined large image. Such an approach could create more seamless transitions between patches, resulting in more coherent overall predictions. In particular, this strategy would help in accurately detecting palms that are located on or near edges.

Experimenting with More Epochs: Increasing the number of epochs during the training process may help in producing a better-converged model, thereby enhancing the predictive performance. This suggestion is especially applicable for the small patch dataset, where no obvious overfitting has been observed. As the large patch dataset is enlarged, the number of epochs can be increased correspondingly. However, caution should be exercised to monitor for potential overfitting to ensure that the model continues to perform well on validation data.

Fine-tuning ResNet18's Last Layer: One of the approaches in transfer learning involves unfreezing the last layer of a pre-trained network and fine-tuning it to better adapt to the current dataset. In the context of our model, unfreezing and subsequently training the last layer of the ResNet18 architecture might prove beneficial. The underlying rationale is that while the initial layers of the network capture generic features useful for a wide range of tasks, the final layers are more specialized. By fine-tuning the last layer, we can tailor ResNet18's capabilities to better recognize patterns specific to palm and non-palm classifications in our dataset. This approach requires a careful balance in learning rate to ensure that the pre-learned weights aren't drastically altered but are adjusted enough to optimize performance for our specific task.

Exploring Complex Architectures: Further exploration and fine-tuning of more complex architectures for the multilayer perceptron (MLP) section of the model could yield improved classification accuracy. The introduction of additional hidden layers, new loss function, or new activation functions could potentially increase the model's capacity to discern intricate patterns in the data.

4.2 Palm Counting

Palm counting is a necessary procedure for studying the distribution and density of palm trees in a specified area. Two models, each trained on either large or small labeled patches, were used to predict palm locations on the large orthomosaic image, resulting in two distinct heatmaps. The subsequent steps outline a method to combine these heatmaps and provide an estimation of the palm count:

Merging Heatmaps: The first step combines the results from both large and small patches by either averaging or using a weighted combination. Averaging ensures equal contribution from both, while a weighted scheme might favor the large patch predictions due to the labeling method, which relies on the small patch predictions. To increase the reliability of the findings, the large patch dataset should be expanded.

Clustering or Blob Detection: After merging the heatmaps, clustering algorithms or blob detection techniques can be applied to identify groupings of high-probability regions representing individual palms. Techniques such as K-Means clustering or simple binary blob detection can be utilized here.

Count and Validate: The final step involves counting the identified clusters or blobs to estimate the number of palms. To confirm the accuracy of this estimate, a validation step might include manual checking of a sample or comparison with known ground-truth data.

A Appendix

All files are included in this drive, and all the code can be found in this GitHub repository.

A.1 Data

The datasets detailed below are integral to the training, validation, and testing phases outlined in this report.

Large Orthomosaic: This is the primary high-resolution drone image dataset. Generated through GIS software, the orthomosaic provides a thorough overview of the targeted region. Beyond the original image, a refined version has been created where artifacts—like blurred or stretched segments—are removed, as depicted in Figure 2. Both of these images are then segmented into 30 smaller images, each of dimensions 5000×5000 pixels, and housed in the "small_images" and "small_images_new" folders.

Small Patches: These patches, each sized 40×40, are derived from the larger images, referencing the palm centers and the overarching orthomosaic (as detailed in Section 1). Serving a key function, these patches were used to train the model tailored for smaller data segments. The patches are distributed across various subfolders. To distinguish between the patches that showcase palms from those that don't, consult the provided code.

Large Patches: This dataset consists of patches that measure 100×100. Each patch is annotated to indicate the presence or absence of a palm. Originating from the heatmap predictions of the model trained on the smaller patches, a comprehensive explanation of this data is available in Section 1 and 2.3.

A.2 Results

This research has produced a range of results, each shedding light on the model's performance, its effectiveness, and areas needing further development:

Heatmap Predictions: Models trained on both small and large patches have yielded heatmap predictions essential for measuring the model's capabilities and identifying areas ripe for enhancement. The folder contains heatmaps spanning the entirety of the large orthomosaic region, assembled from predictions on the 30 smaller images. There are areas, especially along the edges, where predictions fall short, evident in regions that are evidently darker than their surroundings. Such discrepancies highlight areas for future improvements, potentially by utilizing overlaps when segmenting into smaller images.

Other Outputs: This folder encompasses a rich collection of outputs. Among these are heatmaps of small patches, generated by models trained on both the large and small patches. These heatmaps are integral in assembling the comprehensive heatmaps for the orthomosaic. The folder also includes plots detailing the model's accuracy and loss across epochs for each of the five folds in every setting. The Receiver Operating Characteristic (ROC) curves are available for detailed assessment. Additionally, the folder houses the ".pth" file of the trained model, enabling users to load and utilize the model for subsequent applications.

References

[1] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[2] 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.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[4] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, *Learning internal representations by error propagation*, 1985.