Aerial Imagery - Challenge 1 AML

Alex Argese
Student
EURECOM
Biot, France
alex.argese@eurecom.fr

Cristian Degni
Student
EURECOM
Biot, France
cristian.degni@eurecom.fr

Miriam Lamari
Student
EURECOM
Biot, France
miriam.lamari@eurecom.fr

Enrico Sbuttoni
Student
EURECOM
Biot, France
enrico.sbuttoni@eurecom.fr

***Abstract*—This report presents a comparative analysis of different machine learning models developed to classify aerial images containing a specific type of cactus (Neobuxbaumia tetetzo). The goal of the project is to support biodiversity monitoring efforts such as the VIGIA project in Mexico, by identifying the presence of cacti in 32x32 aerial image patches. We describe the dataset, preprocessing methods, and the models trained so far. Final performance comparison and model selection for test set classification will follow upon completion of all experiments.**

***Index Terms*—Machine Learning, Cactus Detection, Aerial Imagery, CNN, ResNet**

I. Introduction

Monitoring biodiversity is a growing priority in the context of climate change and human-driven land transformation. In this challenge, we focus on detecting **Neobuxbaumia tetetzo**, a columnar cactus species, in 32×32 aerial images using machine learning. The dataset was derived from the VIGIA project in Mexico.

II. Dataset

The data set provided is divided into two main parts:

- **Train/ folder**: contains **17,500** 32x32 pixel RGB images, each associated with a label (class 0 or 1). This subset was used for training, validation and evaluation of the models developed.
- **Test/ folder**: includes **4,000** 32x32 RGB images of the same size, without labels. This set has been used exclusively to perform inference with the selected final model, in order to produce predictions to be submitted to the Kaggle platform.

The distribution of classes within the training set was unbalanced:

- **Class 1** (cactus presence): 13.136 images
- **Class 0** (no cactus): 4,364 images

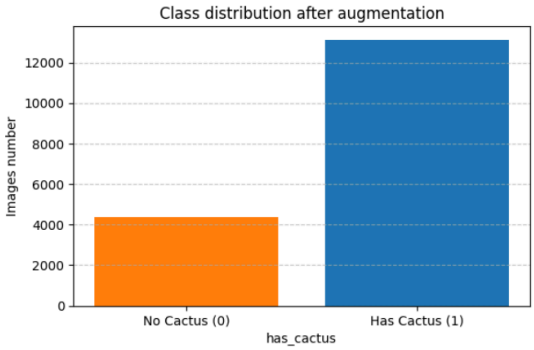

Fig. 1: Initial class distribution.

III. Preprocessing

A. Tensor transformation

The preprocessing involved first converting the images from PIL to PyTorch tensors and resizing them to a fixed resolution of 32x32 pixels, consistent with the format of the original images.

B. Data augmentation

To address the obvious class imbalance (about 13,000 images for class 1 and only 4,000 for class 0), a data augmentation strategy was applied targeting only minority class images. Small Random Rotation (10°) were applied, and then resized to 32x32 pixels, generating about 4,000 new synthetic images of class 0, bringing the total to 8,000 images for this class. This has significantly reduced the difference between the two classes, improving the ability of the models to generalize.

C. Data splitting

The original and augmented images were then linked into a single dataset, which was used for the training phase. Subsequently, the entire dataset was divided according to a stratified strategy (maintaining the proportion of classes in each subdivision) into three distinct subsets:

- **Training set**: 70%
- **Validation set**: 15%
- **Test set**: 15%

This balanced subdivision was crucial for reliable model tuning and an unbiased evaluation of the final performance.

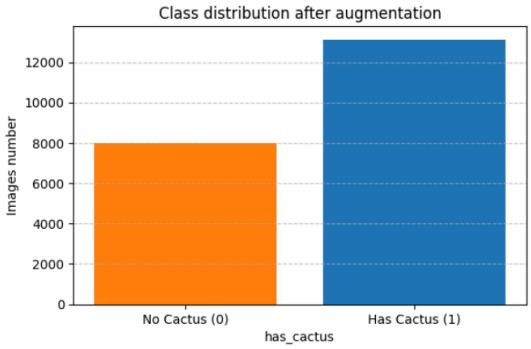

Fig. 2: Initial class distribution aftet data augmentation on Class 0.

IV. Models Evaluated

A. Logistic Regression

Logistic Regression was used as a baseline model due to its simplicity and interpretability. The model was implemented using the scikit-learn library. Since this algorithm does not exploit spatial structure in images, each 32×32 RGB image was flattened into a 3072-dimensional vector.

To improve performance and stability, we conducted a small grid search on key hyperparameters, tuning the regularization strength (C), the solver, and the maximum number of iterations. The tuning process was performed via 5-fold cross-validation on the training set using F1 score as the evaluation metric.

TABLE I: Hyperparameter tuning for Logistic Regression

Hyperparameter	Ranges	Best Parameter
C	0.1, 1, 10	10
solver	lbfgs	lbfgs
penalty	l2	l2
max_iter	1500, 3000	1500

While the model is limited by its inability to learn spatial features, it remains a valuable baseline that performs surprisingly well in this context, particularly when combined with proper preprocessing and class balancing.

B. Support Vector Machine

Support Vector Machines (SVM) are well-suited for binary classification tasks with high-dimensional input spaces. In our setup, the images were first flattened into 3072-dimensional vectors and then standardized using z-score normalization.

We performed a grid search on key hyperparameters using 5-fold cross-validation to optimize model performance. The tested hyperparameters included different kernel types (rbf, sigmoid, poly), values for regularization parameter C, and kernel coefficient gamma.

TABLE II: Hyperparameter tuning for SVM

Hyperparameter	Ranges	Best Parameter
C	0.1, 1, 10	10
kernels	'rbf', 'sigmoid', 'poly'	rbf
gammas	'scale', 'auto'	scale

The optimal configuration consisted of an RBF kernel with C=10 and gamma='scale'. Although computationally more expensive than logistic regression, SVM provided improved generalization capabilities by modeling non-linear boundaries in the data space.

C. Convolutional Neural Network (CNN)

To better capture spatial structures in the images, we implemented a custom Convolutional Neural Network (CNN) using PyTorch. The architecture was designed with simplicity in mind to ensure interpretability and fast training, yet flexible enough to benefit from data augmentation.

The model consists of two convolutional layers followed by max-pooling, batch normalization, ReLU activation, and dropout. The final layers are fully connected with a sigmoid output unit for binary classification.

We performed a grid search on four key hyperparameters to fine-tune the model:

TABLE III: Hyperparameter tuning for general CNN

Hyperparameter	Ranges	Best Parameter
learning rate	0.001, 0.005, 0.01	0.001
drop_out	0.25, 0.4, 0.5	0.25
initial_filters	8, 16, 32	32
fc layers	64, 100, 128	64

The model was trained for 10 epochs using the Adam optimizer and binary cross-entropy loss. Data augmentation included random horizontal/vertical flips and slight rotations, applied on the fly during training. Thanks to this setup, the CNN was able to learn meaningful spatial patterns, achieving substantial improvements over non-convolutional models.

D. ResNet18

To leverage deep feature representations learned from large-scale image datasets, we adopted a transfer learning approach using PyTorch's pretrained ResNet18 model. The model, originally trained on ImageNet, was adapted to our binary classification task by replacing the final fully connected layer with a custom head: a single neuron followed by a sigmoid activation function.

We froze the pretrained layers and fine-tuned only the final block and classification head. Images were resized to 224×224 to match the input size expected by ResNet18.

A small grid search was conducted to tune the learning rate and optimizer:

TABLE IV: Hyperparameter tuning for general ResNet18

Hyperparameter	Ranges	Best Parameter
learning rate	0.001, 0.0001	0.001
optimizer	'adam', 'sgd'	'adam'

The model was trained using binary cross-entropy loss and the Adam optimizer. We applied moderate data augmentation (horizontal/vertical flips and rotations) and early stopping to avoid overfitting. The results demonstrated superior performance compared to all other models evaluated.

V. Metric Justification

Since the dataset is slightly imbalanced, we adopted **F1 score** as the primary evaluation metric. It balances precision and recall, which is crucial in ecological monitoring where false negatives (missed cacti) may have high cost.

VI. Results Summary

A. *Detailed Results – Logistic Regression*

The best logistic regression model, trained with the selected hyperparameters, was evaluated on the held-out test set. The table below summarizes the classification performance per class:

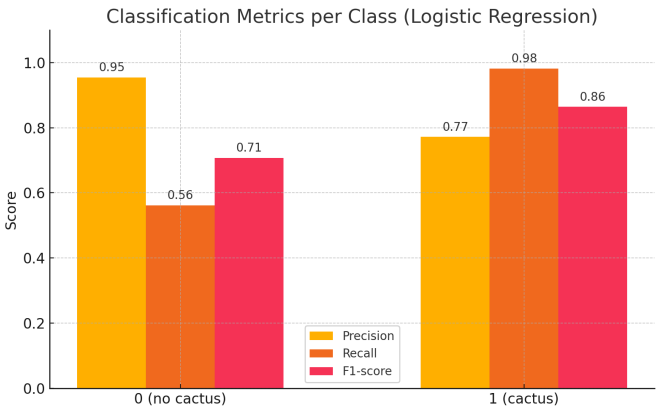

Fig. 3: Metrics for Logistic Regression

The model achieved a final test accuracy of 81.01%. It showed very high precision for class 0 (no cactus), but the relatively low recall (0.55) indicates many false negatives. For class 1 (cactus), the model achieved strong performance across all metrics, reflecting its bias toward the majority class. This suggests that while the logistic regression model benefits from the augmentation strategy, it still struggles with class imbalance and lacks the capacity to model complex spatial patterns.

B. *Detailed Results – Support Vector Machine*

The best-performing SVM model was evaluated on the test set. The performance breakdown per class is shown below:

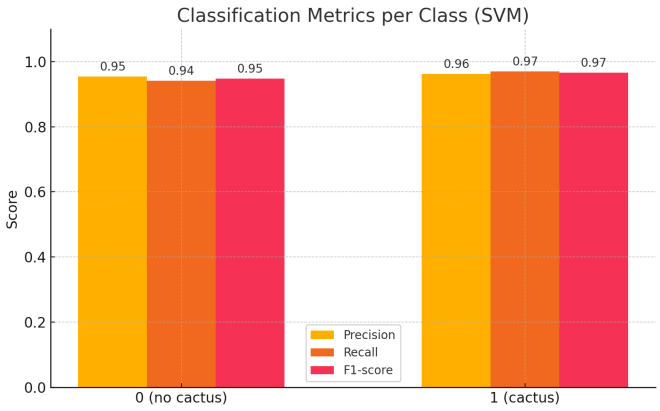

Fig. 4: Metrics for SVM

The overall test accuracy was 96.5%, with a weighted F1 score of 0.9654. The SVM showed higher recall for the cactus class (class 1), and moderately improved recall on the minority class (class 0) compared to logistic regression. This confirms the model's ability to capture non-linear decision boundaries through the RBF kernel, although it still struggled with some overlap in feature space.

C. *Detailed Results – Convolutional Neural Network*

The best CNN model was evaluated on the held-out test set. The performance results per class are as follows:

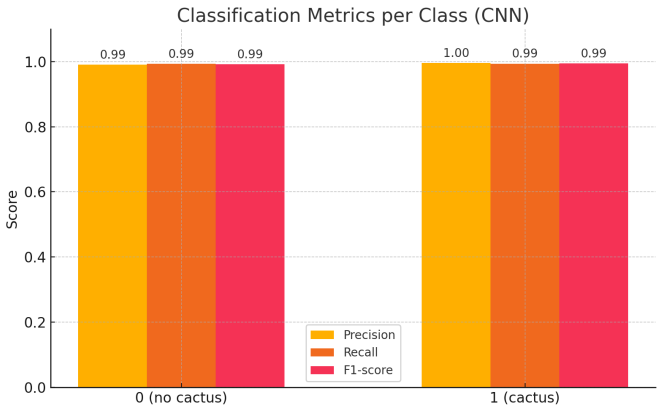

Fig. 5: Metrics for SVM

The final test accuracy was 99.3%, with a weighted average F1 score of 0.9944. The CNN outperformed both logistic regression and SVM by a wide margin, particularly in identifying minority class (class 0) samples, thanks to its ability to learn local spatial features and generalize well from augmented examples.

D. Detailed Results – ResNet18

The fine-tuned ResNet18 model achieved the best performance on the test set. Below is the detailed classification report:

TABLE V: Classification Report on the Test Set (ResNet18)

Class	Precision	Recall	F1-score	Support
0 (no cactus)	0.97	0.96	0.96	1309
1 (cactus)	0.98	0.99	0.98	1971

ResNet18 reached a final test accuracy of 98.0%, with a weighted F1 score of 0.9776. It demonstrated excellent generalization and balance between precision and recall across both classes, validating the power of transfer learning even in low-resolution, small-format image classification tasks.

TABLE VI: Performance Comparison of Models on the Validation Set

Model	F1 Score	Accuracy
Logistic Regression	0.8098	83.4%
SVM	0.8571	86.2%
CNN	0.9554	96.0%
ResNet18	0.9776	98.0%

VII. Conclusion and Next Steps

Despite its simplicity, Logistic Regression achieved a strong baseline performance, demonstrating that even linear models can be effective when supported by appropriate preprocessing and balancing techniques. The SVM outperformed logistic regression, especially in terms of class 1 recall, but was still constrained by the lack of spatial awareness in flattened input representations. The custom CNN significantly improved classification accuracy and balance across classes, confirming the advantage of convolutional architectures in image-based ecological tasks. Among all tested models, ResNet18 stood out with outstanding precision, recall, and F1 scores, confirming the effectiveness of transfer learning even when applied to small aerial images of ecological relevance.

VIII. References

This report is inspired by the VIGIA project as described in:

Efren López-Jiménez et al., **Columnar Cactus Recognition in Aerial Images using a Deep Learning Approach**, Ecological Informatics, 2019.