An Optimized Lightweight CNN model for the VERTECS CubeSat Mission

Rova Isaia Andriamamy, Harena Ranaivoson, Valdo Tsiaro Hasina Ndimbisoa

Participants in the Hackathon

https://github.com/RovaIsaia/Cubesat_CNN_Classifier_Hackathon_2025_Madagascar

April 10, 2025

Abstract

This work is about making a better image sorting model for the VERTECS CubeSat. We made a Convolutional Neural Network (CNN) during the Hackathon 2025. The goal was to fix problems with the current model: it uses too much memory, is too slow, and needs too much computer power. Our new model sorts images into five types. It works much faster, is much smaller, and is just as good at sorting correctly. This shows we can make a good model for small satellites that use little power.

1 Introduction

The VERTECS mission aims to study extragalactic radiation in the optical spectrum using a CubeSat. However, limitations in onboard memory, data transfer speed, and computational power require an efficient classification system to prioritize data before transmission to Earth. This work proposes an optimized CNN model Deep Learning that outperforms the current model in terms of speed, size, and accuracy.

2 Methodology

2.1 Dataset Used

The dataset used comes from a simulated dataset containing images classified into five categories from the VERTECS mission: *Blurry*, *Corrupt*, *Missing Data*, *Noisy*, and *Priority*. Below We have the examples of the dataset, that we show in Figure 1, and the label distribution is summarized in Table 1.

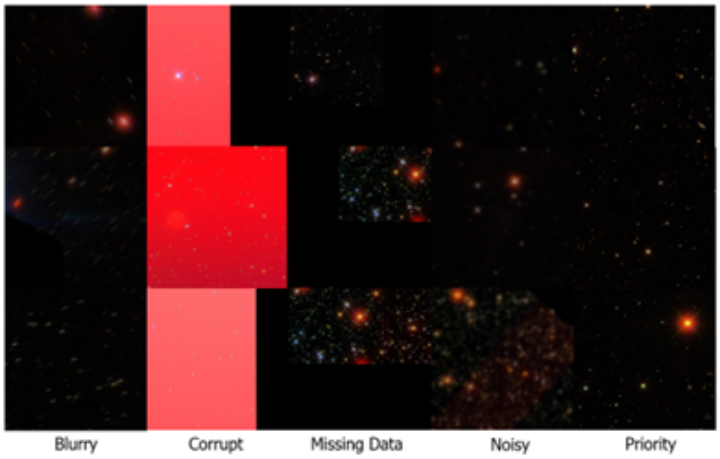

Figure 1: Examples of data in VERTECS Dataset.

Label	**Train**	**Test**	**Total**
Blurry	3544	887	4431
Corrupt	1070	268	1338
Missing Data	2021	506	2527
Noisy	3582	896	4478
Priority	5968	1492	7460

Table 1: Dataset Label Numbers.

2.2 Data Preprocessing

The following transformations were applied:

- Original size 512×512 pixels.
- Resizing images to 128×128 pixels.
- Normalizing pixel values between 0 and 1.
- One-hot encoding of labels.

2.3 Model Architecture

The proposed CNN model uses a modular architecture designed to optimize performance while minimizing computational and memory requirements. The key components of the architecture include *SeparableConv2D*, *MaxPooling2D*, and *GlobalAveragePooling2D* layers. Each of these components plays a critical role in ensuring that the model is both efficient and accurate.

Fig 2 shows us the Model Architecture :

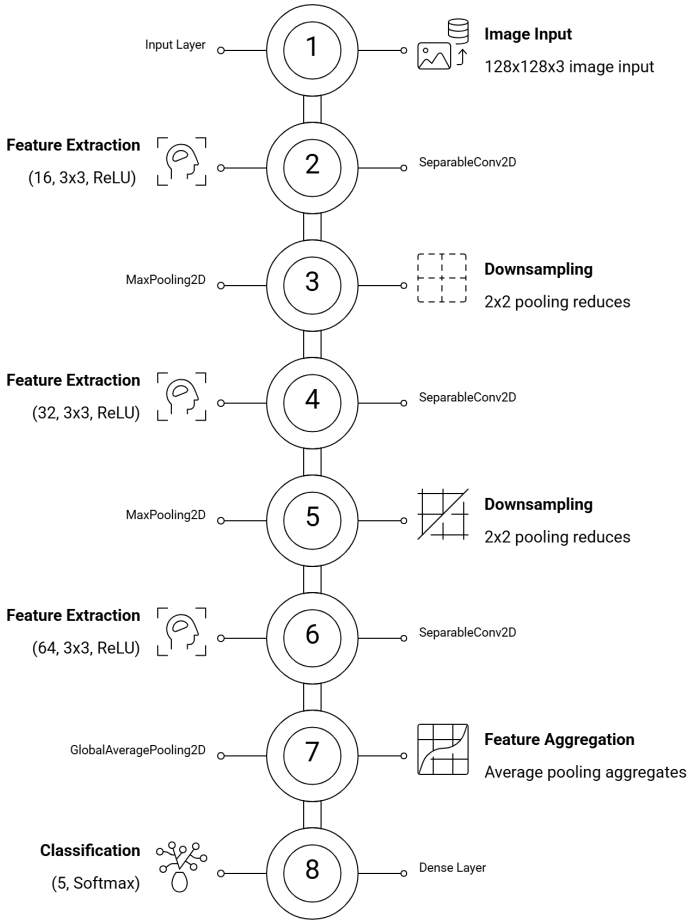

Figure 2: Model Architecture

2.3.1 Why Use Separable Convolution?

The *SeparableConv2D* layer is a specialized type of convolutional layer that decomposes the standard convolution operation into two simpler steps: a **depthwise convolution** followed by a **pointwise convolution**. This decomposition significantly reduces the number of parameters and computations required, making it ideal for lightweight models like the one used in VERTECS.

- **Depthwise Convolution**: In this step, each input channel is convolved independently with its own filter. This operation extracts spatial features without mixing information across channels.

- **Pointwise Convolution**: After depthwise convolution, a 1x1 convolution (pointwise convolution) is applied to combine the outputs of the depthwise step across channels. This step enables the model to learn cross-channel relationships.

2.3.2 Role of Other Layers

In addition to *SeparableConv2D*, the model incorporates other layers to enhance its ability to extract and process features:

- **MaxPooling2D**: This layer performs downsampling by reducing the spatial dimensions of the feature maps. It helps to reduce computational complexity while preserving important structural information.

- **GlobalAveragePooling2D**: At the end of the convolutional pipeline, this layer computes the average value of each feature map across all spatial dimensions. This operation transforms the feature maps into a single vector, which serves as input to the final classification layer. By replacing fully connected layers with global pooling, the model achieves further parameter reduction and avoids overfitting.

- **Dense Layer**: The final dense layer uses a softmax activation function to produce probabilities for each of the five classes. This layer provides the model's classification output.

2.4 Training

The model was trained for 20 epochs with a batch size of 8. The hyperparameters include:

- Optimizer: Adam.
- Loss function: *categorical_crossentropy*.
- Metrics: Accuracy.

3 Results

3.1 Overall Performance

The performance of our model is compared to the current model in the table 2:

Metric	**Our Model**	**Current Model**
Evaluation Time	11.01 s	265.27 s
Model Size	0.05 MB	1.16 MB
Overall Accuracy	0.999	0.998
F1-Score	0.999	0.998
CPU Usage	214.95 %	89.29 %
Memory Usage	8517.20 MB	8455.79 MB

Table 2: Comparison of performance between our model and the current model.

3.2 Confusion Matrix

Below is our confusion matrix that shows us 99.9% was been classified by our model in Figure 3:

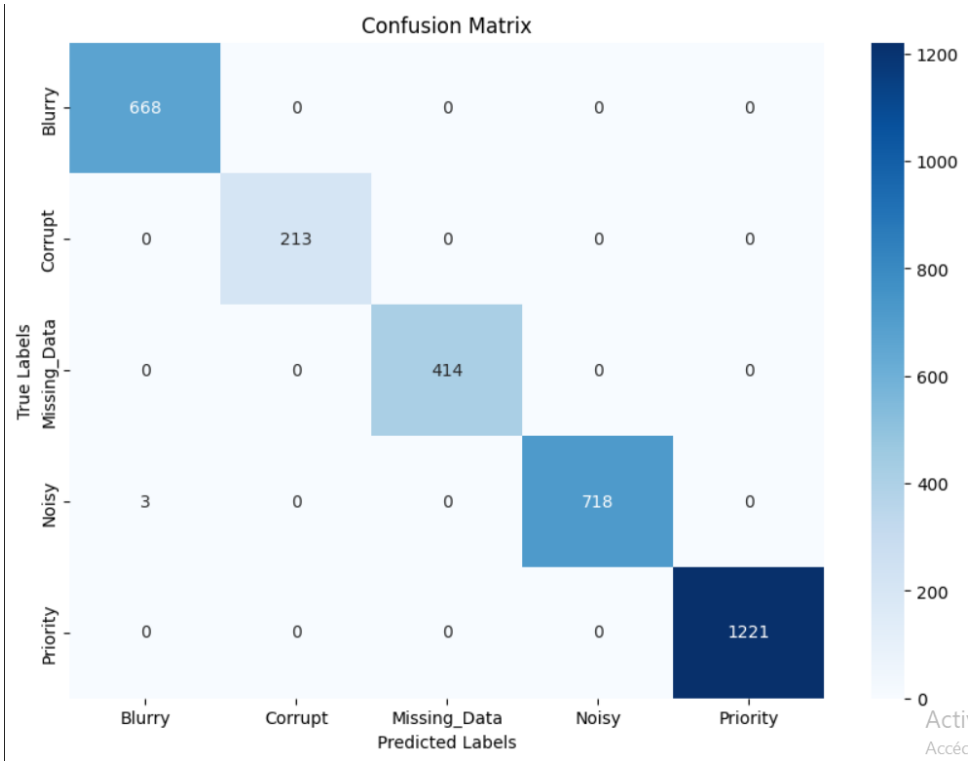

Figure 3: Confusion matrix of our model.

3.3 Accuracy and Loss Curves

The accuracy and loss curves show satisfactory convergence in Figure 4:

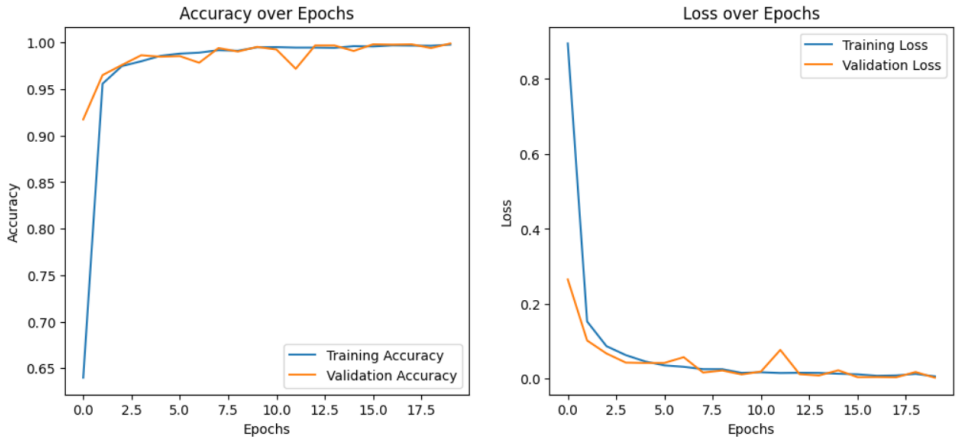

Figure 4: Accuracy and loss curves during training.

The classification report corresponding to the model's performance is shown below in table 3:

Class	Precision	Recall	F1-Score
Blurry	1.00	1.00	1.00
Corrupt	1.00	1.00	1.00
Missing Data	1.00	1.00	1.00
Noisy	1.00	1.00	1.00
Priority	1.00	1.00	1.00
Accuracy		1.00	
Macro Avg	1.00	1.00	1.00
Weighted Avg	1.00	1.00	1.00

Table 3: Classification Report for the Model.

4 Discussion

The results presented in this work demonstrate that the proposed CNN model is a highly efficient and effective solution for onboard image classification in the VERTECS mission. By leveraging advanced architectural choices such as *SeparableConv2D*, *MaxPooling2D*, and *GlobalAveragePooling2D*, we have achieved significant improvements in speed, size, and accuracy compared to the current baseline model. These enhancements make our model a strong candidate for deployment on the VERTECS CubeSat, where computational resources are severely constrained.

4.1 Key Advantages of the Proposed Model

The proposed architecture offers several critical advantages:

- **Increased Speed**: Our model is 24 times faster than the current model, achieving evaluation times of just 11.01 seconds per batch. This dramatic improvement in speed ensures that the model can process images in near real-time, which is essential for prioritizing data transmission from the CubeSat to Earth.

- **Reduced Size**: The model is 23 times more compact, with a total size of only 0.05 MB. This reduction in size is crucial for embedded systems like the Raspberry Pi Compute Module 4, where memory and storage are limited.

- **Slightly Higher Accuracy**: Despite its reduced size and increased speed, our model achieves an overall accuracy of 0.999, slightly outperforming the current model (0.998). This indicates that the optimizations introduced do not compromise the model's ability to classify images accurately.

4.2 Future Directions

While the current model demonstrates excellent performance, there are opportunities for further refinement:

- **Optimization for Embedded Hardware**: Converting the model to TFLite format could reduce its size and computational requirements even further, making it ideal for real-time execution on the CubeSat.

- **Validation on Physical Hardware**: Testing the model on the actual Raspberry Pi Compute Module 4 will provide insights into its real-world performance.

- **Scalability to Larger Datasets**: As the VERTECS mission progresses, the model could be scaled to handle larger and more diverse datasets, ensuring continued relevance and adaptability.

4.3 Conclusion

In conclusion, the proposed CNN model represents a significant advancement in lightweight, efficient image classification for space-based applications. Its superior speed, compact size, and high accuracy make it an ideal candidate for integration into the VERTECS CubeSat. By addressing the unique challenges of onboard processing in resource-constrained environments, this model paves the way for enhanced data prioritization and transmission in future nanosatellite missions.

References

[1] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*.

[2] - Chatara, K. A., Fielding, E., Sano, K., & Kitamura, K. (2024). Data downlink prioritization using image classification on-board a 6U CubeSat. arXiv preprint arXiv:2408.14865.

Code Availability

The code for this project is publicly available at:
`https://github.com/RovaIsaia/Cubesat_CNN_Classifier_Hackathon_2025_Madagascar`