Enhancing Image Data Transmission Efficiency

RWANDA REGIONAL HACK4DEV Group 5 :

Manuella NAKAM
Tricart RWAYITARE
Delphine NISHIMWE

Table of Contents

Context

- Kyushu Institute of Technology and collaborators launched the VERTECS nanosatellite, equipped with a small-aperture telescope to study star formation by observing optical-wavelength extragalactic background light (EBL). However, its limited size and weight restrict onboard computational power and storage, significantly slowing data transmission [1].
- The goal is to develop a lightweight machine learning model to prioritize CubeSat image data for efficient transmission back to Earth.

Dataset

- The data is divided in three parts namely: **Training, validating and Testing**
- The training dataset consists of 9,711 samples of 512x512 RGB images, while the validation and testing sets each contain 3,237 samples.
- There are five classes in the data: **Blurry, Corrupt, Missing Data, Noisy, Priority**

Preprocessing

The preprocessing is applied in parrallel on the images of the whole dataset.

- **Normalization**: The image is converted to a tensor and normalized to a range between 0 and 1 by dividing by 255.0.
- **Resizing**: The image is resized to the target_size to reduce its dimensionality.
- **Contrast Adjustment**: The contrast of the image is enhanced using a factor of 1.5.
- **Data Augmentation**: If the augment flag is True,
- **Clipping**: Values are clipped to ensure they remain within the range [0.0, 1.0].
- **Flattening**: The processed image is converted back to a NumPy array and flattened into a 1D array.

Data after preprocessing

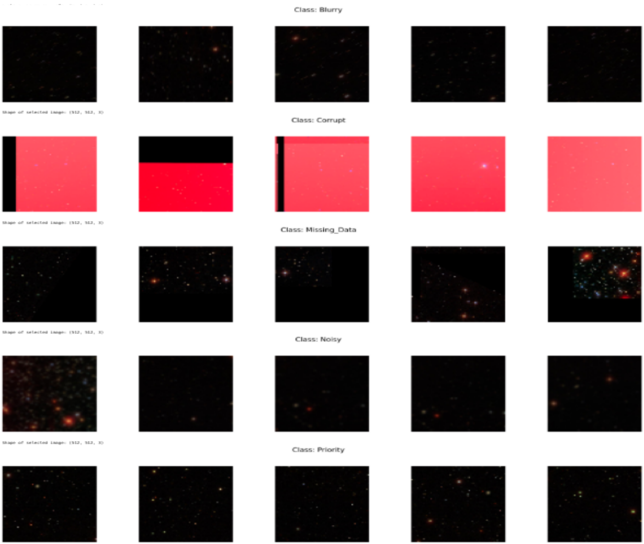

Figure 1: Data after preprocessing

Machine Learning Model: Random Forest

Configuration	Value
Model	Random Forest
Estimators	100
Execution	Parallel
Training time	7 s
Validation accuracy	0.8044
Model size	15 MB

Table 1: Structure of Random Forest

Results: Random Forest

```
### Evaluation Metrics ###

Evaluation Time:        7.11 seconds (The time it took for the pipeline to preprocess data and make predictions.)
Peak Memory Usage:      15593.02 MB (The maximum memory used during evaluation.)
Average CPU Usage:      84.65 % (The % shows how much of one CPU core was used during the evaluation.)
Algorithm code size:            15.06 MB (The size of the trained model and preprocessing function.)
Accuracy:               0.807 (The percentage of correctly classified samples.)
F1 Score:               0.782 (A balance of precision and recall, useful for imbalanced datasets.)

### Confusion Matrix ###
```

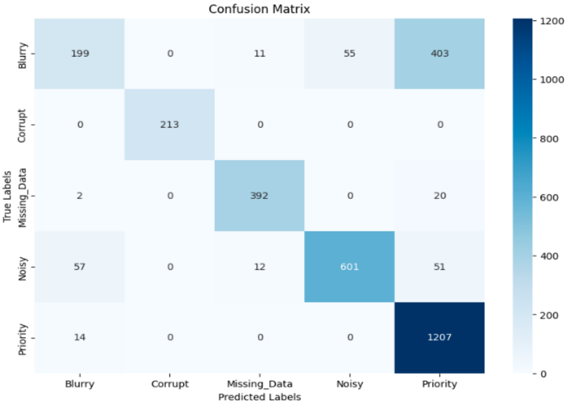

Figure 2: Evaluation Random Forest

Results: Random Forest

The model we proposed :

- Is significantly faster (approx. 5 times) during the evaluation phase (preprocessing + prediction)
- Uses less RAM during evaluation (3 GB less)
- Uses slightly less CPU power during evaluation
- Classifies correctly a much larger percentage of the total samples compared to the existing model.
- Is larger in terms of storage.

Deep Learning Model: CNN

Configuration	Value
Model	Convolutional Neural Network
Number of layers	10 (Convolution, Dense, Pooling, Flatten, DropOut)
Input shape	(128, 128, 3)
Loss function	Categorical Crossentropy
Optimizer	Adam
Execution	Parallel
Training time	7 min
Validation accuracy	0.9988
Model size	38 MB

Table 2: Structure of CNN model

Results: CNN

```
### Evaluation Metrics ###

Evaluation Time:        47.68 seconds (The time it took for the pipeline to preprocess data and make predictions.)
Peak Memory Usage:      25400.27 MB (The maximum memory used during evaluation.)
Average CPU Usage:      54.25 % (The % shows how much of one CPU core was used during the evaluation.)
Algorithm code size:          37.87 MB (The size of the trained model and preprocessing function.)
Accuracy:               0.999 (The percentage of correctly classified samples.)
F1 Score:               0.999 (A balance of precision and recall, useful for imbalanced datasets.)

### Confusion Matrix ###
```

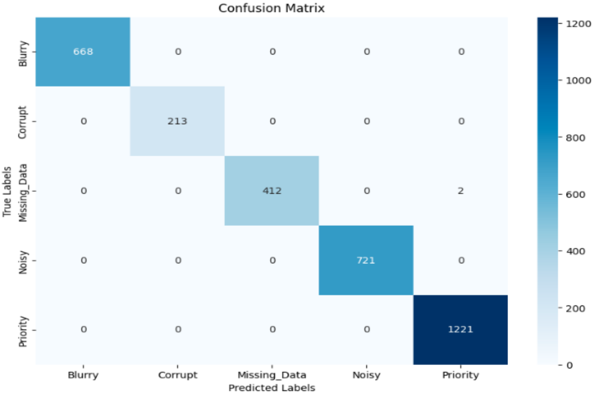

Figure 3: Evaluation CNN

Results: CNN

The model we proposed :

- Makes slightly fewer errors than the existing model.
- Delivers predictions much more quickly
- Uses less CPU on average during its shorter runtime.
- Is larger in terms of storage size.
- Uses significantly more RAM

Conclusion and perspectives

Conclusion

- The preprocessing: contributes significantly to improve the performance and efficiency of the model, ensuring that it receives high-quality, well-structured data during training and inference.
- The parallelisation: enhances performance, resource utilization, and scalability.
- Our proposed models provide high accuracy and efficiency during inference but come with a large model size.

Perspectives

- **Model Pruning**: Identify and remove redundant weights or connections within the network that contribute little to the final prediction.
- **Cross-Validation**: To identify the best parameters such that we increase confidence in the reported metrics.

References

[1] Keenan Chatar et al. "Data downlink prioritization using image classification on-board a 6U CubeSat". In: *Sensors, Systems, and Next-Generation Satellites XXVII*. Ed. by Toshiyoshi Kimura, Sachidananda R. Babu, and Arnaud Hélière. SPIE, Oct. 2023, p. 19. DOI: 10.1117/12.2684047. URL: http://dx.doi.org/10.1117/12.2684047.