POLITECNICO
MILANO 1863

AN2DL - Second Homework Report
ReLUctant Learners

Mattéo Bevilacqua, Vittorio Casalini, Gabriele Lorenzo Singe, Massimiliano Botta

cyhagha, VittoCasalini, Gabbo613, maxbot01

275840, 981802, 275589, 259740

December 14, 2024

1 Introduction

The objective of this challenge is to implement a Neural Network for **semantic segmentation** of images from Mars terrain. The workflow was divided in two phases: inspecting the data and the development of the code, with the aim of achieving the best performance.

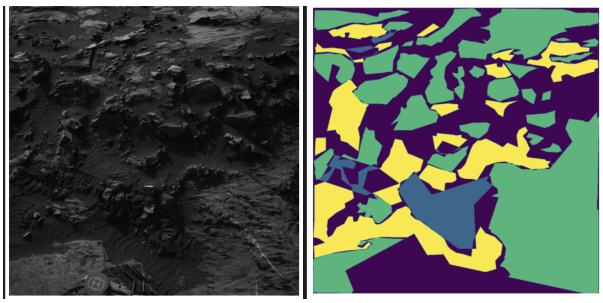

Figure 1: A sample segmented image

2 Problem Analysis

Given our limited experience with segmentation tasks, our initial focus was on analyzing the techniques available to address the problem effectively.

2.1 Challenges

Several challenges arose during the course of our analysis. First, the dataset consisted of grayscale images, which rendered many common augmentation techniques unsuitable. Second, the use of pretrained models was prohibited, adding complexity to the task. Consequently, significant effort was directed towards identifying strategies that would balance performance and portability while accommodating the unique constraints of the problem.

2.2 Data Cleaning

The first thing we did was inspecting the dataset. We soon realized that there were some outliers, in particular there were 110 images that were unrelated to the dataset.

3 Method

We began our experiments with a simple `UNet` model to establish a baseline for our approach. Our initial focus was on preprocessing and augmenting the dataset to improve model performance.

3.1 Workflow Traceability

Every modification to existing notebooks was consistently documented, along with its outcomes, in a shared Excel file. This practice was instrumental in minimizing redundant work and effectively recon-

structing our development workflow for the final report.

3.2 Data Augmentation

We experimented with various augmentation techniques, including geometric transformations such as Grid Mask, Grid Shuffle, Random Brightness, and flipping. However, these methods did not yield any significant improvement in performance. Consequently, we decided to discontinue their use work and to work on not-processed data.
Our focus switched on other aspects of the pipeline.

3.3 Network Architecture Exploration

Given the limited success with the initial network, we decided to explore stronger models without altering the dataset further. When selecting the network architecture to train, we considered models like `DeepLabV3+`, `ResNet`, `DenseNet`. However, most of these were too large and complex, often resulting in poor validation performance, likely due to overfitting.

Afterward, we attempted to implement a custom network inspired by a simplified version of `UNetV3+` [1], but this did not lead to any performance improvement. Ultimately, we decided to settle on a custom `UNet` model.

3.4 Customization

To enhance the performance of our simple network, we experimented with various modules, testing them individually to track their differential impact on results.
Squeeze-and-Excitation block: Improved the network's ability to focus on informative features by adaptively recalibrating channel importance.
Residual connections: Ensured effective feature fusion while preserving both low-level and high-level information. [3]
Skip gate: Regulated the flow of information between consecutive layers for better control and integration. [2]
These additions yielded encouraging results in our tests, indicating that we were on the right track. However, further attempts to implement deep supervision and pyramid pooling mechanisms did not lead to significant experimental success.

3.5 Class Imbalance Resolution

A significant improvement was made when we analysed the class distribution and discovered that the *background* class was too big compared its scientific value, while the *big rock* class was virtually absent. To address this, we adjusted the class weights to be inversely proportional to their frequencies and set the background weight to 0.0. This modification resulted in a substantial improvement in performance on both the validation and test datasets.

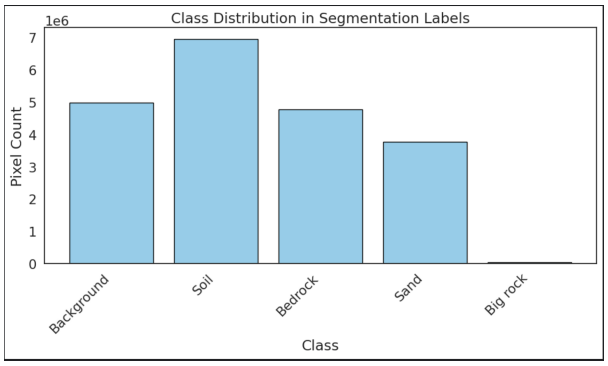

Figure 2: Distribution of the classes in the cleaned dataset

3.6 Loss Function

We also made notable progress by experimenting with different loss functions. Each of these attempts involved fine-tuning the parameters to optimize the performance of the respective loss. Good results were obtained with *Focal Loss*, *Dice Loss*, and a linear combination of both. We also tested *Categorical Cross-Entropy Loss* and *Tversky Loss*, but these did not yield significant improvements. Ultimately, we decided to focus on a variation of **Focal Loss** with `gamma = 0.25`, `beta = 2.0`, and an additional `class_weights` parameter to account for class distribution.

4 Experiments

We dedicated most of our training instances to optimizing the hyperparameters, attempting to anticipate the outcomes of each change to achieve the desired results. Below is a summary of our observation:
Dataset split: 80% of the dataset was used for training, with the remaining 20% split evenly between validation and testing.

Backbone	Loss	Notes	Test Score
Unet	Categorical Cross-Entropy	AutoContrast+RandomBrightness	0.456
UNetV3+	Focal	background = 0, batch size = 16	0.507
UNetV3+	Focal + class weights	Add layer	0.623
Custom UNet	Focal + class weights	Add layer + Skip layer	0.64
Custom UNet	Focal + class weights	Leaky ReLU + Residual Connections	0.678
Custom UNet	**Focal + class weights**	**Add Squeeze Layer**	**0.701**

Table 1: A condensated summary of our model exploration, any subsequent identical labels are assumed to include the same notes as previously indicated. Best result is highlighted in **bold**

.

Optimizers: We used `AdamW`, which achieves a good balance between convergence speed and generalization by decoupling weight decay from adaptive learning.
Weight Decay: To promote generalization, a relatively high weight decay value of 1×10^{-3} was applied in conjunction with the AdamW optimizer.
Learning Rate: For the Adam optimizer with weight decay, the initial learning rate was set to $\eta = 1 \times 10^{-4}$. A learning rate plateau strategy was adopted, halving the rate after ten consecutive epochs without improvement
Batch size: Great effort was dedicated to tuning the batch size; we finally settled for a relatively small value of 16 to maximise adaptability and generalization.
Patience: A patience threshold of 30 epochs was implemented in conjunction with the learning rate plateau strategy. This approach enabled further refinement of the final performance across three different learning rate stages.
Activation Function: We observed that *Leaky ReLU* in the UNet blocks provided better results then other activation functions.

5 Results

Our best results with each model are listed in Table 1 above. As shown, the best results came from the Custom UNet with Leaky ReLU as activation function for the UNet blocks, residual connections and squeeze layers, with a test score of 0.701.

6 Discussion

Due to the restriction on using pre-trained models, we focused on maintaining a lightweight architecture to avoid complex feature extraction. In this context, various generalization techniques proved beneficial. Notably, we observed performance improvements with smaller batch sizes, as well as the integration of attention layers and the implementation of a learning rate plateau strategy. However, the emphasis on maintaining a simple network may have limited our ability to achieve higher performance.

7 Conclusions

We explored a wide range of materials, including basic UNet and state-of-the-art models such as *DeepLab*. The dataset was thoroughly analyzed, and custom layers were incorporated to enhance model performance. Fine-tuning hyperparameters played a pivotal role, as even small adjustments often led to significant improvements in segmentation quality. Despite not having identified a suitable augmentation, we acknowledge that it could have potentially improved our results. Given more time, we would have explored a two-stage semantic segmentation approach, dividing the task in recognizing the background from everything else and then segment the rest into the other classes.
Another technique that captured our attention was deep supervision in UNetV3+, however we faced several issues managing multiple outputs.

7.1 Group contribution

All team members contributed equally to every stage of the project, collaboratively refining the main notebooks through shared insights and efforts, ensuring a cohesive and comprehensive final outcome.

References

[1] Huimin et al. Huang. UNet 3+: A full-scale connected UNet for medical image segmentation. April 2020.

[2] Yun Jiang, Huixia Yao, Shengxin Tao, and Jing Liang. Gated skip-connection network with adaptive upsampling for retinal vessel segmentation. *Sensors (Basel)*, 21(18):6177, September 2021.

[3] Gui Yu, Juming Dong, Yihang Wang, and Xinglin Zhou. RUC-Net: A residual-unet-based convolutional neural network for pixel-level pavement crack segmentation. *Sensors (Basel)*, 23(1):53, December 2022.