POLITECNICO
MILANO 1863

AN2DL - Second Homework Report
Natural Stupidity

Camilla Bocciolone, Davide Fileccia, Luca Forte, Matilde Simonetti

camibocciolone, Davide Fileccia, calu02, matisimonetti

280315, 273538, 280428, 279810

December 14, 2024

1 Introduction

The goal of this project is to develop a Convolutional Neural Network (CNN) for **semantic segmentation** of 64x128 grayscale images representing **Mars terrain**. The network aims to classify each pixel into one of five categories: Background, Soil, Bedrock, Sand, Big Rock. This segmentation will facilitate detailed analysis of the Martian surface by distinguishing key terrain features.

2 Problem Analysis

1. **Dataset characteristics**
We began by examining the dataset, which contains 2,615 images. By displaying random batches of images alongside their corresponding masks for manual review, we identified an anomaly: 110 images featured an alien. Notably, all these alien images shared the same mask pattern. Using this insight, we efficiently isolated and removed these images, thereby cleaning the dataset.

2. **Main challenges**
One challenge was the limited size of the dataset, which was reduced to 2,505 images after the cleaning. After applying a 90/10 training-validation split, the training set consisted of only 2,254 images. Further analysis revealed significant class imbalance, particularly with class 4 (Big Rock), which was nearly nonexistent. Additionally, some masks were inaccurate, suggesting that the model would need to be robust enough to address these challenges.

3. **Initial assumptions**
We considered focusing on three key areas: developing a robust neural network, designing a loss function capable of addressing class imbalance, and implementing effective data augmentation strategies. However, we decided to prioritize the first two areas and defer data augmentation to a later stage of the project. All model evaluations were performed using the Mean Intersection over Union (Mean IoU) as primary metric 2.

3 Experiments

We began with a simple **U-Net** architecture, where the encoder section was constructed with two downsampling blocks, each one containing two 3x3 convolutional layers, batch normalization, ReLU activation, and the final 2x2 max pooling, The first block used 32 filters, the second 64, and the bottleneck had 128 filters. The decoder mirrored the encoder, with upsampling and concatenation of corresponding encoder features. The output used a 1x1 convolution with softmax activation for pixel-wise class predictions. Initial training and validation Mean IoU were low, prompting us to build a deeper model

(*Model 1*) with five downsampling blocks, increasing the number of filters from 32 to 512. Despite slight improvement, the performance and results remained underwhelming.

We then focused on class imbalance and explored three loss functions: **Focal Loss**, which prioritizes harder-to-classify examples 2; **Dice Loss**, which helps with imbalanced datasets; and **Boundary Loss**, which enhances segmentation accuracy by focusing on object edges. We combined these with **Sparse Categorical Crossentropy** and applied class weights to Focal Loss. After experimentation, we adjusted the loss weights to 0.4 for Crossentropy, 0.3 for Dice, 0.2 for Focal, and 0.1 for Boundary Loss, resulting in slight improvement (*Model 2*).

In order to achieve better results we then redefined each loss function to incorporate **class weights**. Unlike the built-in Keras losses, which apply a uniform penalty across all classes, the custom losses allow for different penalties for each class based on their relative importance. In Weighted Sparse Categorical Cross-Entropy (SCCE), instead of using the default behavior where each class is treated equally, we used class weights to assign different importance to each class, as shown in equation 1. The class weights adjust the loss for each pixel according to its class, making the loss function more sensitive to underrepresented classes. The same approach was applied to Dice Loss and Boundary Loss, where class weights ensure that the loss reflects the importance of each class, allowing the model to focus more on underrepresented classes during training. Since Focal Loss focuses on hard-to-classify examples, we excluded it, as we had already addressed class imbalance by incorporating class weights into each loss function. Seeing no significant improvements, we decided to experiment with a more complex network architecture by implementing a **dual U-Net**. The dual U-Net architecture consists of two separate U-Nets working in tandem: a coarse one and a finer one. The first consists of fewer and larger (7x7) filters, and shallower depth (3 downsampling and 3 upsampling blocks of 32, 64, 128 filters) designed to capture high-level features and context; the finer one has more and smaller (3x3) filters, and deeper architecture (4 upsampling and 4 downsampling blocks of 64, 128, 256 filters) aimed at capturing fine details and intricate structures in the segmentation task. We observed similar results compared to our previous experiments (*Dual U-Net*). To improve performance, we lowered the learning rate from 0.001 to 0.0001 and used a learning rate scheduler. We also increased filters in the fine U-Net and reduced them in the coarse U-Net, but these changes had no significant impact.

We shifted our focus to optimizing a single U-Net architecture, exploring enhancements like attention mechanisms 4. Implementing an **Attention U-Net** slightly improved our results. This architecture uses attention gates in the decoder to enhance relevant features in skip connections and suppress irrelevant ones, improving segmentation accuracy. Its downsampling blocks extract features hierarchically, while the upsampling ones reconstruct outputs by combining context and detail through attention gates, and skip connections. The introduction of the attention gates made the use of addional loss and class weights superfluous. Indeed the attention gates single-handedly achieved better results in highlighting the less represented class compared to the combination of them with custom loss and class weights. We decided then to only use the Sparse Categorical Cross-Entropy (SCCE) loss without incorporating the Dice and Boundary losses and set class weights to 1 except for the background class, whose weight was still set to 0.

We then turned our attention to **data augmentation**, using the ImgAug library for its suitability in segmentation tasks, aiming to enhance the model's generalization without overly altering the data. This was crucial given our limited training set

Finally our last improvement consisted of swtiching our optimizer from Adam to **AdamW**: after testing different optimizers we found that the best results were achieved using the AdamW (Adam with weight decay). Additionally, we employed a learning rate scheduler with a patience of 15 and an initial learning rate of 0.001, along side **Early Stopping** to prevent overfitting.

4 Method

Weighted Sparse Categorical Crossentropy loss:

$$\mathcal{L}_{\text{Weighted SCCE}} = -\frac{1}{N}\sum_{i=1}^{N} w_{y_i} \cdot y_i \log(\hat{y}_i) \quad (1)$$

Table 1: The scores of some of our models

Model	Model 1	Model 2	Dual U-Net	Attention U-Net
Validation MeanIoU	0.2301	0.4573	0.4608	0.6738
Kaggle MeanIoU	0.23975	0.45232	0.44973	0.68415

Mean Intersection over Union excluding the Background class:

$$\text{Mean IoU}_{\text{exclude}} = \frac{1}{C-1} \sum_{i=1,C,i\neq 0} \text{IoU}_i \qquad (2)$$

5 Results

Our main results are illustrated in the table 1. *Model 2* is an extension of *Model 1*, incorporating the custom loss function described in the Experiment section. The *Attention U-Net* described in the previous section represents our final model.

6 Discussion

Upon analyzing our results, it appears that the final accuracies are not particularly satisfactory. The primary reason for this is the data scarcity in our dataset, particularly for class 4, which is significantly underrepresented. Below are the key findings regarding the performance of our network:

- **Training time:** The network requires a substantial amount of time for training due to the low learning rate employed. While necessary to achieve optimal results, this choice increases computational complexity.
- **Classification of class 4:** Despite improvements and extensive testing, the network still struggles to classify class 4 accurately, mainly due to limited data. We attempted to address this through image augmentation (augment4 in makedataset), but resource limitations prevented full implementation.
- **Data augmentation:** The data augmentation strategy used is designed to generalize the problem and enhance the overall robustness of the network. However, it is a generic approach and not class-specific, which could limit its effectiveness, particularly for underrepresented classes.
- **Class weights:** The implementation of class weights should be revisited and refined to ensure better integration. This adjustment has the potential to further improve the network's performance.

7 Conclusions

Following a thorough analysis of the problem, it was evident that the U-Net architecture was the most suitable choice. We began by implementing a basic version of the model and iteratively refined it through successive developments. After multiple iterations, as previously detailed, we converged on the Attention U-Net, a model enhanced with attention gates in the decoder section to effectively identify and prioritize the most significant features within the skip connections. Subsequently, after finalizing the network architecture, our efforts shifted towards data augmentation and hyperparameter tuning to further optimize the results.

1. **Future work** : Future improvements could involve the development of a ResU-Net, incorporating residual connections within the U-Net architecture to enhance its performance. Additionally, a ResU-Net++ could be explored, integrating advanced features such as Squeeze-and-Excitation blocks, Atrous Spatial Pyramid Pooling (ASPP), and attention mechanisms. These enhancements aim to further optimize the model's performance and improve the overall effectiveness of the augmentation process.

2. **Member's Contribution**: Concerning individual contributions, Matilde implemented both the initial U-Net model and the final Attention U-Net. Davide developed Model 2, while Luca created the dual U-Net. Camilla handled the data augmentation. All other tasks, including hyperparameter tuning and report writing, were carried out collaboratively.

References

[1] *Batch normalization layer*: Link

[2] *Focal loss*: Link

[3] *ImgAug*: Link

[4] *Attention U-Net*: Link

[5] *ResUnet*: Link

[6] *ResUnet++*: Link