POLITECNICO
MILANO 1863

AN2DL - Second Homework Report
I ragazzi del Bav

Stefano Corti, Andrea Catelli, Alessandro Ciotti, Marco Giovanni Barbero

stecorti, Andrea Catelli, Alessandro Ciotti, MarcoBarbero

245554, 247446, 247754, 260314

December 14, 2024

1 Problem introduction

The project concerns **supervised image segmentation** using **Deep Learning** techniques. Our objective is to design a Neural Network model capable of accurately identifying the terrain type on Mars images, assigning the correct class to each pixel within the image. This task required us to focus on three different aspects: **data pipeline**, **loss** and **models**.

The starting point is a dataset of 2615 greyscale labeled images from Mars terrain of size (64x128), which we will address as **original dataset**. Overall, the number of labels is **five** that represents different type of Mars terrain (Background, Soil, Bedrock, Sand and Big rock).

As we have learned from the previous challenge, the correct analysis of the data is crucial to understand the task and build a powerful model.

2 Data pipeline

The first step in our analysis was to examine the **original dataset** to understand the characteristics of each type of Mars terrain. We observed several **outliers**, including images that depicted an alien and had **identical labels** pixel wisely. To ensure the integrity of our analysis, we removed these duplicates, resulting in a refined ***cleaned dataset*** for further study.

Upon closer inspection of the cleaned dataset, we identified additional issues, such as incorrectly labeled images. Hence, we first focused on images labeled with a single class, manually verifying and correcting them as necessary. This effort resulted in the creation of the ***fixed monocolor dataset***. Following the same procedure, we reviewed and corrected images labeled with two classes, leading to the development of the ***fixed bicolor dataset***.

Despite these corrections, testing multiple models on the revised datasets yielded unsatisfactory results. Further analysis suggests that the test set may have contained mislabeled images since it comes from real data, which could have contributed to the poor model performance.

We employed the **Augmentation** on the datasets in several ways: i) one by implementing by hand several geometrical transformation and ii) using the **library Albumentation**. We created a new larger dataset and it was used to test different architectures, seeing some improvement.

By analyzing the dataset and counting the occurrences of each class type, we observed that it was imbalanced, with pixels classified as "Big Rock" appearing $\mathbf{10^6}$ **times less frequent** than other classes. We experimented different methods to increase the number of "Big rock" pixels in our dataset. The most effective approach involved selecting images with a sufficient number of "Big Rock" pixels, manually cropping the regions of interest, and applying data augmentation to generate additional images.

These new images were added to the cleaned dataset, resulting in a new dataset named ***dataset balanced BigRock***. Additionally, we used the Albumentation library to perform standard data augmentation, creating the ***dataset augmented BigRock***.

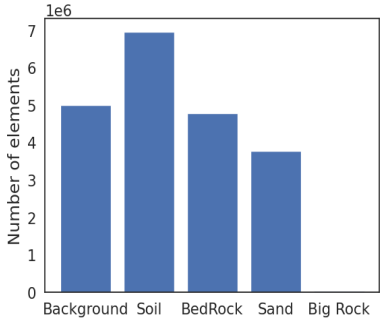

Figure 1: Pixel frequency distribution

3 Loss

Initially, we chose to use **Sparse Categorical Cross-Entropy** as the loss function to effectively train our models and we kept it as benchmark for others attempts. Subsequently, we implemented a **Dice Loss** in order to deal with the imbalance of our dataset and **Focal Loss** to focus more on challenging classification case, attempting to combine them with different weights to better handle our dataset. However, we observed that these strategies did not lead to significant improvements in performance.

Finally, ignoring the background class as in the competition metric, we decided to apply **weights** to make the model focus more on learning the other four classes. This approach proved to be highly effective in improving the model's ability to correctly classify each pixel in the images.

4 Models

4.1 Baseline U-Net architecture

As initial approach, we implemented a **simple U-Net architecture** consisting of an Encoder with sequence of convolutional, Batch normalization, Activation and Max Pooling layers and a Decoder with Up Sampling, Convolutional and Activation layers. We employed the standard Sparse Categorical Crossentropy as loss and we monitorate during training the mean intersection over union, which is the metric for our competition. However, as expected, this basic architecture was insufficient to perform well in a complex task as the one we were dealing with.

4.2 DeepLab V3

In order to improve our performances, we implemented a version of the DeepLab architecture which employ a key feature in the **bottleneck**. Thanks to the presence of **Atrous Spatial Pyramid Pooling** (ASPP) module, we were able to capture features at multiple scales. This module leverages on dilated convolutions with different **dilatation rate**s and a **global average layer** which are then combined in a single convolutional block. This architecture was able to improve our performances, adding global context on the predictions.

4.3 W-Net

It was then implemented a **double U-Net** (or W-Net). The aim of the architecture is to produce a segmentation mask with the first U-Net and, subsequently, refine it through the second one. Different optimizers and loss functions were used but we did not get any expected improvements.

4.4 Pyramid Scene Parsing Network - PSP-Net

After the implementation of the first models, by looking to the output segmentation map we have noticed the presence a pattern with a patchy appearance.

The PSP-Net tries to catch information at at different scales, allowing us to paying attention to different scales, and so also to the global context. This seems coherent to the fact that in the dataset a lot of segmentation maps show wide portions of pixels with a specific label.

The **bottleneck** of this model is the key point: after an encoder very similar to the one described in the U-Net architecture, the bottleneck split the feature in **4 different paths**. In each of these, the features go through an average pooling with a size reduction specific for each part, and after applying two convolutional layers, along with activation, we go back to the original size before the pooling.

This feature manipulation allowed us to obtain satisfying outcomes, as it can be seen in Table 1. Combining this model with the attention gate allowed us to achieve the best performance.

Table 1: Mean Intersection over Union on Validation and Kaggle test set

Model	**Dataset**	**Loss**	Validation **Mean IoU**	Kaggle **Mean IoU**
Baseline U-Net	Cleaned	SCCE	45.23%	43.01%
DeepLabV3	Cleaned	SCCE	52.00%	48.77%
W-Net	Cleaned	Weighted SCCE	57.49%	51.34%
PSP	Cleaned	Weighted SCCE	72.71%	64.13%
DeepLabV3	Aug. Big Rocks	Weighted SCCE	72.89%	69.12%
Attention + PSP	Aug. Big Rocks	W. Focal+SCCE	72.88%	71.78%

4.5 Attention Gate

The attention gate mechanism aims at improving the model capabilities to **focus only on relevant information**. This objective is achieved by dynamically computing the weights that will ponder input data, increasing the input values that we expect to obtain in output. In our case the **Key/Value** are represented by the information coming from the downsampling and the **Query** from the first stages of upsampling. The resulting weights multiply the inputs that then will propagates in final upsampling stages. We saw that adding this Attention gate has improved our model predictions.

5 Results

In Table 1 we reported the performances on both **Validation** and **Kaggle test set** obtained by model employed, the dataset and loss used. The main improvement in performances were due to the right choice of **loss** and of the **dataset**, rather than the model. Excellent results have been achieved **combining different losses**, trying to establish the right weight for each loss.

Most of the results in the table are reported on the cleaned dataset since its dimension allowed us to obtain useful insights in a fast way. For the **final model** we run the training on the whole dataset without taking out a local test set.

6 Conclusions

The problem of segmenting Mars terrain images proved to be challenging due to the **similarity** between the different **type of terrains** and due to the **mislabeled images**, which is common in **real dataset**. The first attempt of fixing the labels of dataset was not big source of improvement in performance, however it helped us to better understand the **characteristic of the images**. The idea of **balancing and augmenting** our dataset to let the model learn key features, along with the **loss functions combination**, it leads to a great improvement in performance. By analyzing the model architecture, adding **PSP bottleneck and Attention gate mechanism** in the decoder, allowed to obtain best results.

By using the weighted loss, excluding the background, we induced the model to focus more on the other classes which help for our performances. Doing in this way the model was prone to predict much less background, introducing a new source of error. To tackle this, a possible solution could be adding a **refinement net** for creating background/non-background masks. These should be considered in our model.

7 Members Contribution

At the beginning we analyzed the problem together, removing the outliers from the data and building the baseline model. **Alessandro** worked on augmentation with **Andrea**, who also helped **Stefano** in the attempt of improving the images maps. **Marco** worked on attention gates while Andrea implemented DeepLabV3, Stefano tried PSPNet and Alessandro W-Net. In the last part all together we studied loss functions and figured out the best model.

Just for reporting an example of what said above, the final and best result is obtained using the dataset created by Alessandro, the architecture of the model has been implemented by Stefano and improved by the attention mechanism of Marco while the losses function that enters in the combined loss have been developed by Andrea.