POLITECNICO
MILANO 1863

AN2DL - Second Homework Report
Team: fmap

Filippo Galli, Alessandro Howe, Matteo Callini, Paolo Bellezza

filippogalli, alehowe, matteocallini, paolobellezza

251720, 251125, 244635, 251364

December 14, 2024

1 Introduction

This report outlines the methodologies and strategies adopted for the **Mars Terrain Segmentation** challenge proposed during the Artificial Neural Networks and Deep Learning course at Politecnico di Milano. The objective of the challenge is to accurately segment each image of the Martian surface into distinct terrain classes. To achieve this goal, a series of image **pre-processing techniques** are implemented, and a **neural network** is designed and trained from the ground up.

NB: Code.zip includes the final code fmap.ipynb, which contains the finalized model, and all other note books which support the results discussed in the report.

2 Data Preprocessing

The dataset contains 2615 training and 10022 testing grayscale images, each with a size of 64x128 pixels. The task requires the classification of each pixel into one of five terrain classes: background, soil, bedrock, sand and large rocks. For each training image, an associated mask is provided to label the corresponding terrain classes.

An initial exploration of the training set identifies **110 outliers**, which contain irrelevant content showing aliens. These are removed from the dataset.

Following this analysis, **data augmentation** is performed.

Specifically, in the **first augmentation version**, the entire dataset is **augmented to 4000 images** by randomly flipping, cropping, rotating or translating the image and its respective mask.

Then, noticing that the **large rocks class** (class number 4) is **very rare** - only 50 images contain a large rock - two approaches are explored to address the imbalance. The first uses the same augmentation as before but applies it **only to images containing class 4** and then to the entire dataset. This produces 900 images containing class 4 and the **second augmentation version** is obtained. The second approach involves **pasting large rocks** onto images that do not contain them, followed by augmentation on the entire dataset, leading to the **third augmentation version**.

Additional experiments apply augmentations involving **colors**, but this method results in a worse final score and therefore is abandoned.

3 Model Development

The model development begins with the implementation of a basic **U-Net** model, a standard architecture widely used for segmentation tasks. The structure of the model is illustrated in Figure 1.

Initially, the model employs **convolutional**

blocks, followed by the ReLU activation function and batch normalization. To reduce the number of parameters, it replaces classical convolutional blocks with **SeparableConvolutions**. Additionally, **Group Normalization** is experimented instead of Batch Normalization, as this slightly improves performance. The chosen evaluation metric for the task is the **mean intersection over union**, with the background class excluded from the calculation.

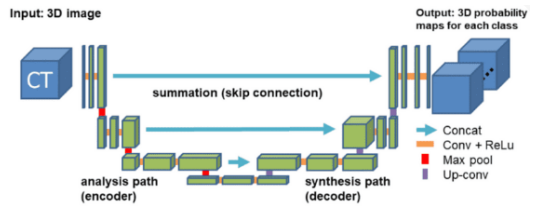

Figure 1: U-Net model architecture

At such point, the development aims to improve this baseline model by incrementally introducing new features.

Specifically, the standard convolutional blocks are replaced with alternative blocks, such as **residual blocks**, which incorporate more convolutional layers and utilize skip connections to facilitate gradient flow.

The **bottleneck** (i.e. the deepest block of the model), becomes the next focus. Several approaches are experimented:

Dilated Convolutions: These introduce parallel convolutional paths that capture features at different resolutions.

Global Context Modules: These modules capture the global information of the input tensor using global average pooling and recalibrate feature importance through a squeeze-and-excitation mechanism.

Squeeze-and-Excitation Blocks: These blocks have a purpose similar to global context modules, aiming to improve feature recalibration.

These modifications demonstrate improvements over the basic CNN block.

Also the loss function for the task is modified. Instead of relying solely on categorical cross-entropy we tried to implement an **ensemble of losses**. Specifically:

Dice Loss: Measures overlap between predicted and true.

Focal Loss: Addresses class imbalance and focuses on hard-to-classify examples.

Boundary Loss: Penalizes boundary mismatches between predicted and true segmentation labels.

However, we obtained better results using the cross-entropy, even if we tried all the possibilities.

To improve the detection of important features, the **background class is excluded from the loss calculation**.

3.1 Further architectures

Driven by the success of this approach in many segmentation tasks (see, e.g., [2]), also a **U-Net++** architecture is experimented. This is summarized in Figure 2.

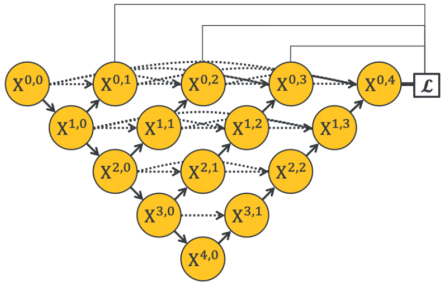

Figure 2: U-Net++ model architecture

The U-Net++ architecture introduces intermediate blocks that perform upsampling starting from a given depth. Skip connections are used to concatenate different blocks at the same depth. This design aims to extract features at different semantic levels and compute a weighted loss for various resolutions. To avoid a significant increase in the number of parameters, basic Separable Convolutional blocks are used.

Initially, the U-Net++ is implemented as shown in Figure 2. However, the higher computational time does not increase the performance of the model. To address this issue, a modified version of U-Net++ is created, which uses fewer blocks in the skip connections.

Additionally, another approach explores training **concatenated models** simultaneously. Here two models are trained simultaneously: the first is more complex, and its output serves as the input for the second, smaller. Both outputs are considered to compute the loss. Multiple configurations of double-model settings, such as two U-Nets or two

U-Net++ models, are tested. This method achieves good results using the U-Net approach; however, the **computational time significantly increases**.

3.2 Hyperparameter tuning

Observations reveal that a high learning rate is ineffective. As a consequence, a dynamic learning rate procedure is adopted, using the *ReduceLROnPlateau* function in Keras, with an initial learning rate of 10^{-4}. Based on empirical results, the models are trained for 200 epochs with early stopping monitoring the validation metric. The batch size is set to 32. AdamW is used as optimizer. It is observed that good performance are achieved using the following configuration: **Dice:** 0.7 - **Focal:** 0.2 - **Boundary:** 0.05 - **Sparse Crossentropy:** 0.05, but the best model is obtained considering all the weights equal to zero, except the **sparse crossentropy** one.

4 Results and Discussion

The simple U-Net model is used as a baseline. Table 1 displays the obtained results.

The results indicate that the best performance is achieved by the Concatenated U-Net with the sparse categorical cross-entropy loss. This outcome may be attributed to its simpler training process and reduced number of parameters compared to the Concatenated U-Net++.

We noticed that in a double configuration the U-Net++ does not perform as well as the U-Net. This could be caused by the training phase, since the model complexity of the first UNet++ affects the input of the second one. Moreover, in the double Unet case, the training MeanIoU reached **0.8159** in a smooth and regular way.

5 Challenges and further development

The development of the models prioritizes performance improvement while maintaining small and fast architectures, even if, as previously explained, model complexity increases.

A challenge which was encountered is that the performance metric often reaches its **peak performance in the middle of the training phase**. After this peak, the metric stabilizes approximately 20% below its maximum value. High patience results in unnecessary computational time, while lower patience risks to prematurely stop the training before the optimal value is reached.

For further improvements there are several steps which could be taken. The primary strategies include optimizing the hyperparameters of the ensemble loss, improving the monotonicity of the metric progression during training and testing alternative losses and callbacks for each network output, particularly for intermediate outputs.

6 Author contributions

The main focuses of each team member are outlined below. The work was characterized by strong collaboration and significant overlap in tasks.

Filippo Galli: Loss function implementation, Unet++ implementation and led experiments regarding the neural networks implementations.

Alessandro Howe: Unet++ implementation and Concatenated Unet++ implementation, led experiments regarding the neural networks implementations.

Matteo Callini: Concatenated Unet++ and Unet implementation, led experiments regarding ensemble loss parameters.

Paolo Bellezza: Data exploration and augmentation implementation, Unet implementation and led experiments regarding ensemble loss parameters.

Table 1: Validation and test performance for different models

Model	Validation Mean IoU	Test Mean IoU
U-Net	**0.3810**	**0.4442**
U-Net++	**0.4765**	**0.4932**
Concatenated U-Net	**0.6405**	**0.6324**

References

[1] H. Liu, M. Yao, X. Xiao, H. Cui, *A hybrid attention semantic segmentation network for unstructured terrain on Mars*, 2023, School of Artificial Intelligence, Jilin University, China.

[2] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, J. Liang, *UNet++: A Nested U-Net Architecture for Medical Image Segmentation*, 2018, Arizona State University, Tempe, USA.