AN2DL - Second Homework Report
ANNalyzers

Edoardo Gennaretti, Francesco Lo Mastro, Riccardo Figini, Pierantonio Mauro

edoardogennaretti, francescolomastro01, riccardofigini, pierantonioma

258290, 243817, 251187, 258737

December 21, 2024

1 Introduction

The goal of this challenge was to classify different types of Martian surfaces from grayscale photographs, classifying each pixel into five main categories. In this paper we present the strategies and the trials we used to address this challenge, focusing on **image segmentation** techniques, and presenting the key **results** achieved.

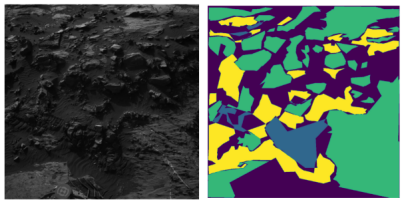

Figure 1: Segmentation example

2 Problem Analysis

The problem consisted in correctly classifying a dataset of **64x128px grayscale image** by generating 64x128px segmentation masks for each input image. To achieve this, we designed and implemented a **Convolutional Neural Network (CNN)** capable of performing pixel-wise classification.
The trained network will be evaluated on a test set of over 10,000 previously unseen images, with performance measured using the **Mean Intersection over Union (MIoU)** metric for the predicted segmentation masks.

3 Method

For enhancing the input data, we tried various data augmentation techniques provided by KerasCV and Tensorflow libraries, also performing some experiments with Albumentations library and size manipulation.
Since using pre-trained models or any sort of imported weights was not allowed this time, we chose to build each model entirely from scratch.
Indeed, **for model construction**, we made exclusive use of the basic building blocks of Tensforflow.keras library (such as Conv2D, BatchNormalization, Pooling layers etc).
For performance analysis, we relied on the most common metrics such as the Mean Intersection over Union (MIoU) and Accuracy both on Training and Validation.

Abbr.	Meaning
L_MIoU	Training MIoU (local)
L_VMIoU	Validation MIoU (local)
K_MIoU	Test MIoU on Kaggle

4 Experiments

OutliersRemover.ipynb (1):
During the initial inspection of the training dataset, we identified some **outlier** images and masks. The outlier images basically had aliens in the Foreground (Figure 2) but they were all slightly different, however their masks all looked the same. So we developed a script to remove all images that matched the identified mask, creating and exporting a clean dataset.

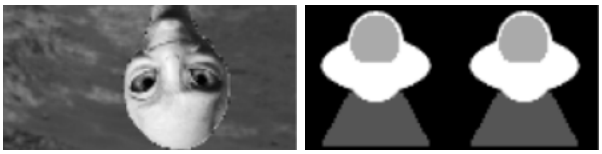

Figure 2: Outlier Image&Mask

PrimoTentativo.ipynb (2):
Our first approach was intentionally simplistic in design, serving as a preliminary model to establish a baseline for the problem and to familiarize the team with the dataset, testing the platform and overall workflow. We implemented a classic **UNet**, following the procedures seen during the lectures. Despite the simplicity of the network and the absence of any augmentation, the result was promising, K_MIoU of 0.4098, providing a solid basis for further improvements and optimizations.

UNet++.ipynb (3):
The second model developed aimed to increase the complexity of the basic UNet structure. **UNet++**, a network that has a hierarchical structure of skip connections arranged "in a triangle", was introduced with the idea of allowing a better flow of information between layers. The idea improved our score, reaching K_MIoU = 0.4779. In a second step, by adding minimal flip data augmentation, we were able to further improve the performance of the model, achieving a K_MIoU of 0.4955

Augmentations.ipynb (4):
The improvements achieved through data augmentation led us to explore how far these techniques could have enhanced our results. Consequently, we decided to dedicate a parallel branch of our team to this area of research, focusing on experimenting with and selecting the most effective augmentations for this specific problem. Once this work was completed, the selected augmentations were shared with the rest of the team.
We tested **generic** random (using Albumentations), and **custom** random **augmentations**. Among these, the most effective augmentations proved to be the custom ones like Translation, Flipping, Negative Filtering, and CutOut, while Rotations, Elastic Transformations and also CutMix showed limited effectiveness for this task.
To make this augmentations even more effective, we thought that we could **duplicate** (even triplicate) the **training dataset** and then let the stochastic augmentations produce (almost) new data and remove duplicates.

UNet++ Experiment.ipynb (5):
Given the good behavior of the UNet++ network, we decided to experiment with different loss functions. Noting a significant disparity in the frequency of the classes, particularly for class 4 which was underrepresented, we introduced a **weighted loss function**. This function assigns more weight to the underrepresented classes, promoting more accurate estimations for all categories. Achieving a solid K_MIoU = 0.50. We also experimented with several others loss functions, including Dice Loss, Boundary Loss and Focal Loss.
As a first step, we tested each function individually to evaluate their effectiveness at best. **Boundary Loss**, in particular, did not perform well in our case, never managing to exceed a L_VMIoU of 0.30. Our personal conclusion was that the low quality of the images negatively affected this loss. Indeed the edges and contours were not well defined (both in training masks as in the images) making it difficult for the model to accurately delineate them and take full advantage of this loss function.
In contrast **Focal Loss** and **Dice Loss** performed pretty well. The latter had been particularly effective, allowing us to achieve a score (K_MIoU) of about 0.49. This confirmed its ability to handle unbalanced classes.

CrossValidation.ipynb (6):
After the last singular trial, we implemented a cross-validation procedure to find an optimal combination of the three losses (**Dice Loss**, **Boundary Loss**, and **Focal Loss**). To manage the complexity, we added an "if" condition to halt training if the first k-fold did not meet a predefined threshold. De-

spite these efforts, the highest score achieved was K_MIoU = 0.50, suggesting that even combining losses offered no significant advantage over the Dice Loss alone.

PAN.ipynb (7):
Failing to improve scoring with previous strategies, we developed a Pyramid Attention Network (**PAN**), designed to extract features at different scales and improve segmentation accuracy. As stated on *paper 6*, this net leverages pyramidal attention for better spatial representation, initially achieving K_MIoU = 0.51. Improvements such as **cross-validation**, **new augmentations**, **squeeze-and-excitation** levels, and **dropouts** slightly enhanced performance, reaching K_MIoU = 0.53.

FeaturesFusion.ipynb (8):
We also experimented with feature fusion, using **two cascade connected UNets**. For the development of this network, we relied on what was described in *Paper 5*. The network was trained both holistically and individually for each component, this was done by the Transfer Learning technique (on our custom model). However, we could not find any particularly optimal combination, achieving at most K_MIoU = 0.52.

Unet-Vgg16.ipynb (9):
In the final days of the challenge, we decided to revert to the classic UNet architecture, but with one significant modification: we used a **transformer as the bottleneck** and used a **gating block** to manage the skip connections. This approach leveraged the transformer's ability to capture global relationships between features, enhancing the model's performance. Remarkably, on the very first attempt, this particular model achieved a score of K_MIoU = 0.54.

FiveNets.ipynb (10):
Another interesting experiment was carried out by constructing five separate networks, each specialized in a specific class. Their outputs were then combined for a final segmentation task. Our first and only attempt achieved K_MIoU = 0.48. Unfortunately, during further trials, we encountered some strange problems. For example, due to computational limits and session interruption, we lost some models. In addition, we couldn't load ".keras" files on Kaggle (the buffering icon looped indefinitely). It remains unclear if the failure was due to Kaggle or due to the way we saved our models.

UNet_post_processing.ipynb (11):
In our analysis of the results produced by the UNet model, we observed a significant occurrence of false positive classifications for pixels labeled as class 0 (background). To address this issue, we implemented a **post-processing strategy**: replacing the predicted label 0 with the second most probable class for each pixel. This naive yet effective approach yielded notable improvements, achieving a K_MIoU score of 0.61 with a basic UNet architecture trained for a limited number of epochs. Extending the training period by increasing the number of epochs, we further enhanced the model's performance, reaching a K_MIoU of 0.70. We also experimented by applying this post-processing technique to more complex network architectures. However, these experiments did not replicate the performance gains observed with the simpler UNet model.

5 Discussion

Throughout this challenge, we explored a wide range of techniques, attempting to test as many strategies as possible. However, no matter how **modern** or **unconventional** was the architecture we tried, we couldn't achieve results that exceeded a score of 0.55. The most significant improvement came from the post-processing method discussed in the final experiment. With more focus on techniques such as ensemble, we could have likely achieved even better results.

6 Conclusions

We are fairly satisfied with the results we've achieved, especially considering that the use of pretrained models could have led to even better performance. Even though such models were not allowed, this challenge was still a valuable experience that pushed us to explore new techniques and improve our problem-solving skills.
Contributions: **Francesco** and **Edoardo** worked on notebooks (1, 2, 4, 5, 8), **Riccardo** on (7, 8, 9, 10,) while **Pierantonio** worked on notebooks (3, 11).

7 Bibliography

[1] Keras.io. Keras. [Online] https://keras.io.

[2] Image augmentation. Keras. [Online] https://keras.io/api/layers/preprocessing_layers/image_augmentation/

[3] *U-Net: Convolutional Networks for Biomedical Image Segmentation* Olaf Ronneberger, Philipp Fischer, Thomas Brox https://arxiv.org/abs/1505.04597

[4] *DoubleU-Net: A Deep Convolutional Neural Network for Medical Image Segmentation* Debesh Jha, Michael A. Riegler, Dag Johansen, Pål Halvorsen, Håvard D. Johansen https://arxiv.org/abs/2006.04868

[5] *UNet++: A Nested U-Net Architecture for Medical Image Segmentation* Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, Jianming Liang https://arxiv.org/abs/1807.10165

[6] *Feature Pyramid Network for Multi-Class Land Segmentation* Selim S. Seferbekov, Vladimir I. Iglovikov, Alexander V. Buslaev, Alexey A. Shvets https://arxiv.org/abs/1806.03510

[7] Albumentations. [Online] https://albumentations.ai/