AN2DL - Second Homework Report
CAMAL

Aleksa Mirkovic, Manuela Maroni, Carlotta Pecchiari
mirkotheking0405, manuelamaroni, carlottapecchiari
244893, 252121, 233381

December 14, 2024

1 Introduction

This project focused on the **semantic segmentation problem** of assigning the correct class label to each mask pixel of a given dataset. The dataset we worked on was composed of 64x128 grayscale real images from Mars terrain. Pixels in these images were categorized into five classes, each representing a particular type of terrain: *Background, Soil, Bedrock, Sand* and *Big Rock*. The objective of this project was to obtain the highest possible *pixel-wise classification accuracy* on the final set of data by using **deep learning** techniques in order to have an effective segmentation and classification of the different terrain types. Our approach consisted in trying out different architectures, losses and data augmentation techniques to see what worked best on the dataset, taking also inspiration from literature.

2 Problem Analysis

The dataset consisted of segmented images from Mars terrain, each paired with a mask representing the class of every pixel. It was already divided into *training set*, a Numpy array of shape (2615, 2, 64, 128), and *test set*, a Numpy array of shape (10022, 64, 128) containing the unlabeled test images. The initial assumptions were:

- Problem is *well-defined*: each pixel in the input images represents a meaningful part of the terrain and must be assigned a specific label from the defined classes.
- The dataset is assumed to be representative of real Mars terrain and to have sufficient diversity to generalize to unseen data.
- The training and test datasets are drawn from the same distribution, meaning the model can generalize well.

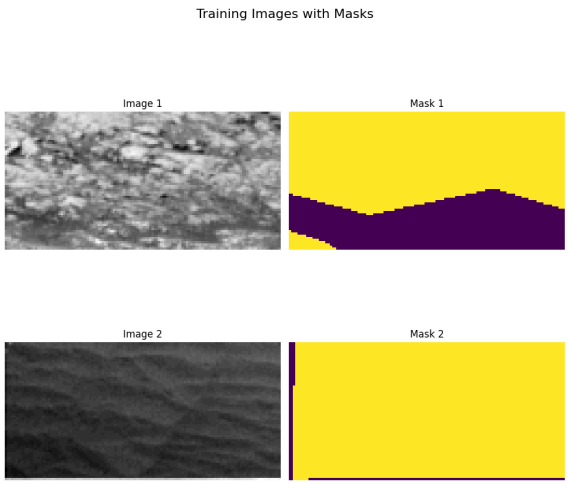

Figure 1: Two examples of dataset images and their masks.

When looking at the images we immediately found some of them that were outliers, as they contained pictures of aliens that were not supposed to be there. First we tried to remove them by using an **autoencoder** that could learn the features of the right data on the test set and eliminate the wrong images from the training set, but we actually got a better score and accuracy by simply directly deleting the pictures of aliens by using their mask, which appeared to be the same for all of them. Eventually we ended up removing a total of 110 images and corresponding masks from the training set:

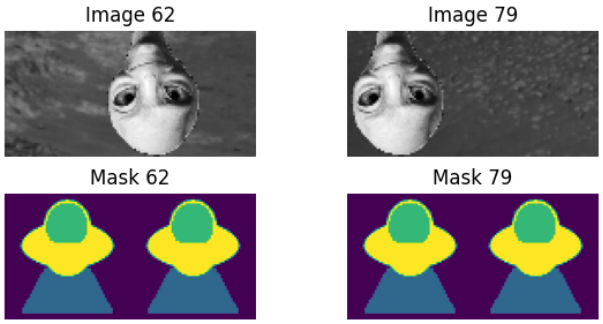

Figure 2: Two examples of alien images and their masks.

The main challenge for us was represented by the fact that with some kinds of augmentation methods, as well as losses and architectures that were more complicated and we thought would work better, we kept getting no improvement. One reason might be the fact that the five classes did not have a similar distribution, in particular the class "Big Rock" had only a few samples:

Figure 3: Distribution of classes in the training set.

In the end we managed to get our highest score by implementing a simpler network.

3 Method

After analyzing the class distribution and removing the alien images, we split the filtered data into training and validation sets. Then we normalized the pixel values of the training set to [0, 1] and added a channel dimension for compatibility with TensorFlow models. In order to deal with class unbalance we decided to implement a **weighted loss function**, so to assign weights inversely proportional to class frequencies to emphasize underrepresented classes (e.g., Big Rock). So we modified the **sparse categorical cross-entropy loss** to include class weights, like shown below:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} w_{y_i} \cdot \log(\hat{y}_{i,y_i}) \tag{1}$$

Then we performed **data augmentation** to increase data diversity and improve generalization. Subsequently we prepared the dataset by converting images and masks into a TensorFlow Dataset object for efficient processing, which applies augmentations during training for improved robustness and also to manage class unbalance. We implemented a **U-Net** architecture with a downsampling path, a bottleneck and an upsampling path. The *unet_block* function defines a modular block consisting of two stacked convolutional layers, batch normalization, and the **ReLu** activation function, hereby shown:

$$f(x) = x + \mathrm{ReLU}(x) = \max(0, x) \tag{2}$$

We used two of these modular blocks for the downsampling path (*Encoder*) and two for the upsampling path (*Decoder*). The model was then compiled with the Adam optimizer, the weighted loss function and using the accuracy and **MeanIou** as metrics. With this model we managed to obtain a validation accuracy of 43.69% and a score of 0.50798.

4 Experiments

We tried out many different models with various loss functions, architectures and data augmentation techniques, and some of the results are highlighted in table 1:

Table 1: Table of the main results achieved.

Model	MeanIou	Score
Baseline model	4.8242e-04	0.11758
WSCCE+R+DL	0.1147	0.12247
WBC+WDC+FC	0.3183	0.42602
Final model	**0.1175**	**0.50798**

5 Results

The first model is a minimal working neural network model that has a single convolutional layer. In the second presented model we used the weighted sparse categorical cross-entropy as loss function, the same one used in the final model, and added *regularization layers* and *dropout layers* to the architecture, but this did not seem to improve much the result. The third model shown in table 1 was obtained using the *weighted binary cross-entropy loss* combined with the *weighted dice loss function* and the *focal loss*, as suggested in [1]. This model also had dropout layers, and the normalization was changed from *batch normalization* to *group normalization*, differently from other experiments we had done before. The best model we got was unexpectedly simpler, as it did not include any dropout layers nor regularization layers, and also the loss function was less complicated.

6 Discussion

Our results were pretty satisfactory, considering that:

- Some classes like "Big Rock" were likely underrepresented.
- The 64x128 resolution of the data might have been limiting for capturing fine-grained terrain features.
- Pretrained models were not permitted, so the neural network was trained from scratch.

By taking a look at different papers and taking inspiration from some of them we tried to implement a model that could resemble the ones proposed[2], taking into consideration the memory constraint and high computational cost limitations. For example, we also tried to implement a **GAN (Generative Adversarial Network)** to generate images for the underrepresented class, but due to the high computational cost we were not able to include it in the model. Our model could for sure be improved by finding a way to generate more data for this specific class, or maybe also by implementing a different combination of techniques. Nevertheless, we tried out as many combinations of approaches as we could and we believe to have well managed the problems in our dataset.

7 Conclusions

Our work mainly focused on trying as many architectures as we could to see what worked best on our dataset. We worked in parallel and tried out different techniques, for example Mirkovic tried out different kinds of augmentations, parameters, losses. Maroni was able to get all the models illustrated in this report, and Pecchiari tried to implement the GAN, different losses and wrote the report. The model could for sure be improved with the use of **transfer learning**, so by fine-tuning a pre-trained encoder (e.g., ResNet, EfficientNet) to improve feature extraction and generalization. An improvement could also be achieved by creating more data for the underrepresented classes through **generative models**. Maybe there is also a better combination of techniques that we have not found that could improve the final score. Finally, a possible future development could be to extend the work to high-resolution Mars terrain imagery for finer-grained segmentation. It would also be interesting to develop a network capable of processing images at multiple resolutions to balance detail and computational efficiency.

References

[1] Mr-Talhallyas, *Loss Function Package Tensorflow Keras PyTorch.* Available at: https://github.com/Mr-TalhaIlyas/Loss-Functions-Package-Tensorflow-Keras-PyTorch?tab=readme-ov-file

[2] Junbo Li, Keyan Chen, Gengju Tian, Lu Li, Zhenwei ShiThe, *MarsSeg: Mars Surface Semantic Segmentation with Multi-level Extractor and Connector.* Available at: https://arxiv.org/html/2404.04155v1