POLITECNICO
MILANO 1863

AN2DL - Second Homework Report
YNWA

Maria Aurora Bertasini, Marco Cioci, Francesco Rosnati, Luca Tramacere

mablearns, marcocioci, francescorosnati, freshtrama

260345, 252488, 252560, 247220

December 14, 2024

1. Introduction

This project addresses semantic segmentation of Mars terrain images, where each pixel is assigned a class label, dividing the image into meaningful regions. The dataset consists of 64x128 grayscale Mars images, each pixel belonging to one of five classes, paired with masks specifying pixel-wise class labels.

The objective is to develop a model for accurate pixel-wise classification, evaluated using the mean intersection over union (mIoU) metric, excluding the background class (label 0).

In this report, we detail every step of the journey that led to achieving reasonable results, though not as strong as initially hoped for. Several attempts were systematically tried, including exploring different approaches and extensive fine-tuning of hyperparameters. We also focused on finding a balance between efficiency and performance to ensure that the proposed solutions remained practical within the given constraints.

Although this paper outlines the main concepts and methodologies, it is designed to be read alongside a series of accompanying Jupyter notebooks. The `Template.ipynb` notebook serves to illustrate the key components of our networks architecture, from that each different `model.py` can be run and tested. Additionally, it acts as a guide, directing the reader to other notebooks that delve into specific aspects of the process, while the `FinalModel.ipynb` show our best result.

2. Dataset Preparation

The provided dataset consisted of 2,615 images, each paired with a corresponding mask, intended for training. Additionally, a set of 10,022 images was provided for testing, though their masks were not included.

2.1. Preprocessing

To prepare the data for training, the images and masks were normalized and reshaped to include a channel dimension, where necessary. The pixel values of the images were scaled to the range $[0, 1]$. The dataset was then split into training and validation subsets, with 80% of the data used for training and 20% reserved for validation.

2.2. Cleaning the Dataset

A cleaning process was applied to the training data to address the presence of outliers. Specifically, images with objectively incorrect masks were identified and removed, while for images with uncertain mask quality, a decision was made to retain most of them to preserve diversity and encourage generalization.

During the cleaning process, it was observed that several different images shared identical masks. In such cases, only those with clearly incorrect masks were removed. After an initial model was trained, this hypothesis was validated: overly aggressive cleaning, like removing all images with potentially incorrect masks, negatively impacted the model's

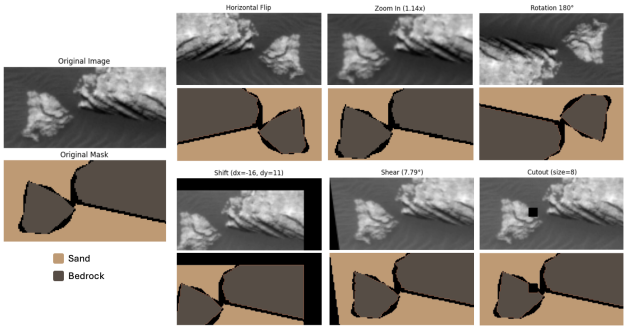
ability to generalize.

The final dataset used for training was saved as `clean_dataset.npz` and from this point forward, all models were trained on that.

3. Data augmentation

Before focusing on the architecture of the model, we began by testing data augmentation on a simple U-Net model. The goal was to evaluate various augmentations that could be beneficial for a segmentation task, taking a methodical, step-by-step approach.

We started by incorporating four basic augmentations: flip, zoom, rotation, and shift. Subsequently, we experimented with additional augmentations, such as cutout, shear, and elastic transformations. Based on the results obtained, we decided to retain all augmentations except for the elastic one, which was discarded due to limited improvement in performance. All the augmentations tried are described in the `Augmentations.ipynb` notebook and the one selected are shown in the picture below.



This phase characterized the initial days of the challenge. Even after finalizing the augmentation strategy, we continued to periodically review its effectiveness as the models evolved, ensuring that the augmentations continued to provide value. However, it became apparent that, in this case, augmentation was not the key factor driving performance improvements. As a result, we chose to maintain a light augmentation strategy while focusing most of our efforts on developing the best possible architecture to address the problem at hand.

4. Architecture

The architecture of our model builds upon a modified U-Net framework. Initially, we adopted a base U-Net architecture; this architecture consists of two main parts: a contracting (downsampling) path and an expansive (upsampling) path, both of which are connected by a bottleneck. On the base U-Net architecture several modification and additions were introduced to improve performance.

4.1. Enhancements and modifications

AdamW Optimizer and Learning Rate Schedulers.

After experimenting with several optimizers, we chose the AdamW optimizer for its superior generalization and overfitting prevention. To further enhance training efficiency, we employed early stopping and a learning rate scheduler that reduces the learning rate when the model's performance plateaus.

Skip Connections Strategy

We tested to find the best possible skip connections mechanism. Standalone and mixed use of concatenation, add, multiply were experimented with. Concatenation was found to be the most effective for improving feature propagation, merging low- and high-level features.

Network Depth and Efficiency.

An important consideration was finding the optimal depth for the network to balance model performance with computational efficiency. We experimented with various depths of the network, ranging from shallower to deeper architectures. Deeper networks, although capable of capturing more complex patterns, suffered from increased computational costs and training instability. After extensive experimentation, we end up choosing a lightweight version.

Bottleneck Design.

We evaluated several bottleneck designs to optimize feature representation at the transition between the downsampling and upsampling paths. In the end, we found that combining Squeeze-and-Excitation (SE) blocks with Dilated Atrous (DA) convolution blocks provided the best feature representation, achieving superior performance with-

out incurring excessive computational cost.

Regularization.

We explored several regularization techniques, including dropout, L2 regularization, and elastic regularization, to determine which one provided the best performance. After testing, L2 regularization emerged as the most effective technique for our network.

This design allowed us to establish a starting point, achieving a meanIoU score of 0.45. Building upon this foundation, we tried to enhance the model's performance by experimenting with different loss functions, which are discussed in the next section.

5. Loss Function and Class Imbalance

The primary goals were to address class imbalance and reduce the influence of the background class on the model's learning. To achieve this, we tested various loss functions, including Dice Loss, custom Focal Loss, and custom soft Mean IoU Loss, along with weighting strategies for underrepresented classes. Both custom losses exclude class zero, with Focal Loss inherently addressing class imbalance.

Despite these efforts, addressing class imbalance using static or dynamic methods proved challenging. The extreme imbalance in certain classes, particularly the infrequent "big rock" class, caused issues with weighting strategies, leading the model to overemphasize this rare class. We also tried a "Copy and Paste" augmentation technique (detailed in `Augmentations.ipynb`) to reproduced unfrequent labels, but with limited success.

The custom Mean IoU Loss mitigated the background class issue by excluding it from the loss computation, helping the model focus on foreground features. While this didn't resolve the imbalance, it improved performance, achieving a mean IoU score of 0.60.

Given that further improvements with the current approach were difficult to achieve, this led us to explore a different strategy: a Coarse-Fine approach.

6. Corse-Fine Approach

Starting from the best-performing model, we introduced a dual-branch design: one branch was the original U-Net, while the other was a simplified U-Net block aimed at capturing global features through increased downsampling. However, this underperformed.

We refined the global branch by increasing its complexity and experimented with fusion mechanisms like concatenation, addition, and cross-scale fusion, with cross-scale fusion yielding the best results. Adding an Atrous Spatial Pyramid Pooling (ASPP) module and a stochastic depth layer to the global branch slightly improved performance.

Encouraged by this, we integrated a basic transformer module into the global branch. Although GPU memory constraints initially were difficult to handle, a simpler transformer design improved performance. Advanced transformers with gradient checkpointing managed memory constraints but slowed training to impractical levels.

We also explored alternatives, including pyramid pooling, dynamic convolutional layers, gated attention, positional embeddings, and hybrid fusion strategies. Unfortunately, none led to meaningful improvements.

Ultimately, the U-Net with a basic transformer module in the global branch remains our best-performing model, achieving a slight but significant improvement over the baseline U-Net (0.62)

7. Discussion and Conclusion

The final model (`FinalModel.ipynb`) achieved a validation meanIoU of **0.62**. However, despite all our efforts, we undoubtedly missed some key aspects and were unable to reach the results we had hoped for.

Contributions The entire homework was carried out collaboratively, with team members contributing to the overall development and implementation. Special mentions: Maria Aurora Bertasini, for dataset preparation and the baseline model; Francesco Rosnati, for the loss function and regularization process; Marco Cioci, for augmentations and hyperparameter tuning; Luca Tramacere, for the coarse-fine approach.

References

[1] Slides from course lectures and exercise sessions

[2] TensorFlow Documentation. `https://www.tensorflow.org/`.

[3] O. Ronneberger, P. Fischer, and T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*. arXiv preprint arXiv:1505.04597, 2015.

[4] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv preprint arXiv:1409.1556, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, 2016.