AN2DL - Second Homework Report
Neural Network November

Michael Alibeaj, Matteo Bettiati, Lorenzo Bianchi, Francesco Ostidich

michaelgear01, betti38, lollyx21, francescoostidich

260044, 258730, 259946, 259863

December 14, 2024

1 Introduction

In this assignment[5][2] we were provided with grayscale images of Mars terrain, where each pixel is categorized into one of five terrain classes. This is a semantic segmentation problem, and our objective was to assign the correct class label to each pixel in the image. To address this challenge, we analyzed the data, built a UNet model, and enhanced its architecture to improve segmentation accuracy.

2 Problem Analysis

2.1 Dataset details

The dataset consists of 64x128 grayscale images (1 channel). There are five classes: 0 for *background*, 1 for *soil*, 2 for *bedrock*, 3 for *sand*, and 4 for *big rock*. The training set (comprehending images and labels) is an array of shape (2615, 2, 64, 128), the test set is instead an array of shape (10022, 64, 128) of unlabeled images.

2.2 Main challenges

Several challenges arose during this task. Firstly, the small image size limits the detail level, thus blurring class boundaries with varying terrain types. Class imbalance, with the *background* class dominating, hurted performance for underrepresented classes, i.e. *big rock*. Lastly, more generally, the Mars terrain complexity and irregular textures further complicated the model ability to generalize.

3 Methods

3.1 Exploratory data analysis

During the EDA phase, a grid of sample images along with their respective masks was plotted. This visualization helped identify a series of outliers, referred to as "aliens". These outliers were removed by leveraging the associated masks, which were identical across all identified outliers. The class distribution was then analyzed, revealing that class 4 was severely underrepresented

3.2 Data augmentation

To increase dataset diversity and improve generalization, we tried to apply random rotations, flips, scaling, cut mix, blur, elastic deformations, brightness and contrast to the images. Different combinations of functions made the model perform differently: the augmentation pipeline has thus been taylored ad hoc, based on the best metrics we observed when training the models. In the end, to address the class imbalance, images containing class 4 were duplicated multiple times (approximately 15 to 20 times) until a more balanced dataset was achieved.

After experimenting, the most impacting and used augmentation function was cut mix, which created new samples by pasting patches from 2 to 4 other samples onto the existing images. This technique increased the dataset size to around 10,000 samples. To ensure robust model performance evaluation, the validation split was also augmented. Finally, the entire dataset was fed into the model through a data loader, which dynamically applied simple augmentations such as flipping and rotations during training. Augmentations helped the model handle variations in terrain orientation and spatial configuration.

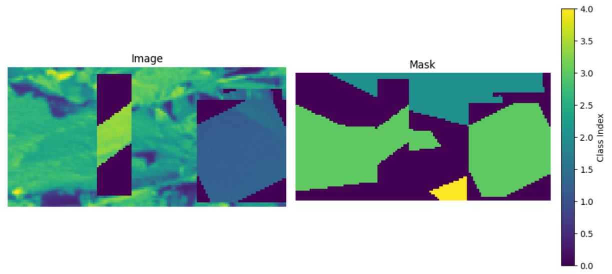

Figure 1: Mid-training prediction

3.3 UNet architecture

The main model architecture used was a custom UNet model, utilizing an encoder-decoder structure with skip connections to preserve spatial information. This architecture enabled the model to extract hierarchical features and make pixel-wise predictions. The training we made was made almost every time on the augmented dataset.

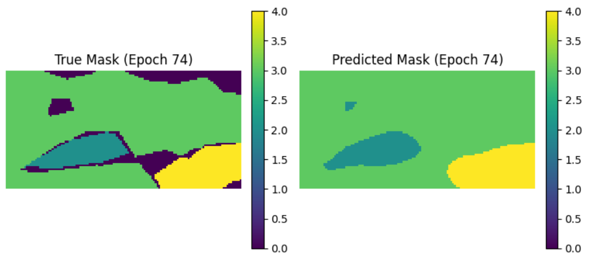

Figure 2: Mid-training prediction

3.4 Loss function exploration

We experimented with various loss functions during the training phase. We started with a *categorical cross-entropy*, for later exploring a *dice loss* and a mean intersection over union loss. Moreover, we also opted for combining losses together hoping to find better compromises and therefore better results. In the end, the *focal loss* was the best scoring alternative; we also exploited the *class weights* implementation of this loss function, which massively helped the model focus on smaller classes like big rock while ignoring the background class and improving both global and local segmentation accuracy and MIOU.

4 Experiments

4.1 Model architecture adjustments

The layers of out UNet model architecture were fine tuned during the training process. By adjusting the number of convolutional layers, filter sizes, and kernel depths, feature extraction improved, and pixel-wise predictions got more distincts. These adjustments helped the model capture more complex terrain features.

Training was conducted on Colab[3] and Kaggle[4] GPUs The TensorFlow[8] Keras[1] and PyTorch[6] frameworks were also utilized during model development and training.

To get an idea of the training metrics evolution, here after we present a table with the values corresponding to some models throughout the experimentation phase.

Mean IoU		
Model Name	**Local**	**Submission**
Early Stage Models		
UNet	0.43	0.44
UNet ResNet34	0.60	0.43
Middle Stage Models		
UNet	0.47	0.52
UNet squeeze & excitement	0.66	0.57
Late Stage Models		
DeepLabV3Plus	0.61	0.69
SegFormer	0.63	**0.74**

Table 1: Metrics throghout training

4.2 Advanced architectures

We also explored more advanced architectures. One such architecture was a dual UNet model, which combined the outputs of two parallel UNet networks. Another enhancement involved the incorporation of squeeze-and-excite blocks, which improve the model's ability to recalibrate channel-wise feature responses. We also experimented with transformer-based architectures, which excel at capturing long-range dependencies. Furthermore, residual blocks were integrated to improve gradient flow and prevent overfitting, ultimately boosting the model's learning capacity.

5 Conclusions

5.1 Final Model

For the final submission, we employed a well-established and reliable architecture known as "Segformer." This model leverages the power of self-attention mechanisms to efficiently capture both local and global dependencies within the input data, which is particularly beneficial for segmentation tasks. Given the relatively small size of our dataset, we initially began with a lightweight configuration, progressively increasing the model complexity while carefully monitoring the risk of overfitting. This approach allowed us to balance model performance with generalization.

The model was trained using a 10,000-sample augmented training set and evaluated on a separate 1,000-sample augmented validation set, providing a more robust assessment of its performance. The optimizer used was AdamW with a weight decay of 1e-3 and gradient clipping set to a norm of 1.0, ensuring stable training. Due to the diverse nature of the augmented data, a batch size of 32 was selected to maintain a good trade-off between memory efficiency and model convergence. Callbacks like model checkpoints, early stopping and LRSchedulers were employed, leading to a training session of 100 epochs on average.

For the final iteration of Segformer, we utilized the 'timmregnetx_160' encoder, with a depth of 5 and 256 channels in the decoder block. This choice of encoder was driven by its ability to balance model efficiency and expressive power, making it well-suited for the task at hand. The implementation of this model was facilitated by the Segmentation Models library[7], which provides a framework for building segmentation networks.

5.2 Contribution

Experimentation and training tasks were carried out equally and coordinately by each team member. Everyone mainly focused on a specific objective, as follows. Alibeaj worked on the development and implementation of the UNet architecture. Bettiati focused on refining and optimizing the UNet model for better performance. Bianchi investigated and experimented with various loss functions to improve segmentation accuracy. Ostidich led the data augmentation efforts, applying various transformations to enhance model generalization.

References

[1] F. Chollet. Keras: The python deep learning library, 2015. Accessed: 2024-11-24.

[2] P. di Milano. Artificial neural networks and deep learning (polimi, ay 2024/2025): course slides, 2024. Available from the course platform or upon request.

[3] G. Inc. Google colaboratory, 2024. Accessed: 2024-11-24.

[4] K. Inc. Kaggle: Your home for data science, 2024. Accessed: 2024-11-24.

[5] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[6] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. Advances in Neural Information Processing Systems (NeurIPS).

[7] Qubvel. segmentation_models: Python package for image segmentation models, 2024. Accessed: 2024-12-14.

[8] TensorFlow. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. Accessed: 2024-11-24.