POLITECNICO
MILANO 1863

AN2DL - Second Homework Report
SiumGPT

Leonardo Bianconi, Angelo Attivissimo, Armando Fiorini, Mattia Busso

leobia, angeloattivissimo, armafio, mattiabusso

232004, 233484, 259459, 252501

December 14, 2024

1 Introduction

This project focuses on *semantic segmentation*, where the objective is to assign one of five class labels to each pixel in 64x128 grayscale images of Mars terrain. The goal is to develop a neural network trained **from scratch** to accurately segment these images.

2 Problem Analysis

The dataset comprises a training set of 2615 labeled samples and a test set of 10022 unlabeled images. The training set included 110 outlier images. We removed them leveraging their predictable mask. Labels span five classes (0-4), though **Class 4 is heavily underrepresented**, posing a significant class imbalance problem. To address this, we utilized sample weights, and employed dice and focal loss functions to prioritize minority classes.

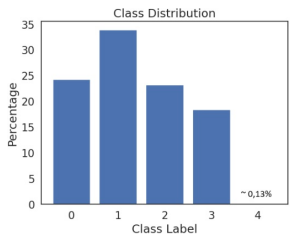


Figure 1: Histogram showing the class distribution

Additionally, as **the background (Class 0) is ignored in the competition evaluation metric**, we excluded it from the training process to streamline model performance on relevant classes.
Inspecting the dataset we found that many images were labeled inconsistently, so a couple of assumptions had to be made: **(i)** training data represent test data, **(ii)** the images contain sufficient information for effective segmentation.

3 Method

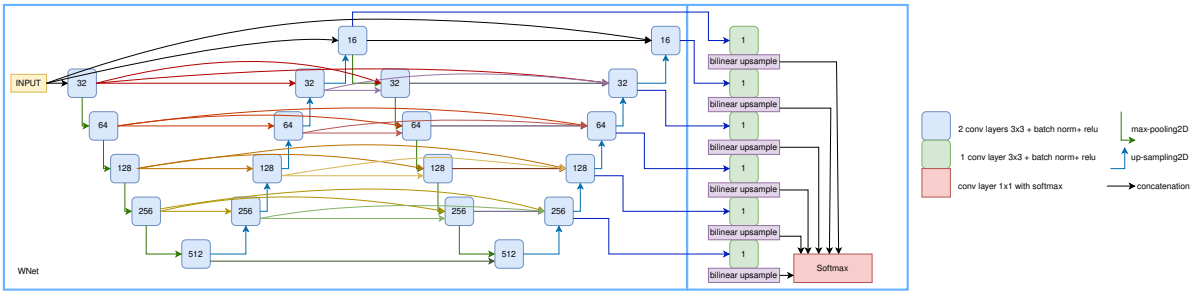
3.1 Data Augmentation

Data augmentation for segmentation tasks needs to be done carefully because using the incorrect filter would reduce performance. After experimenting with flipping, rotation, elastic transformations, coarse dropout, and random brightness, we ultimately discovered that utilizing the library Albumentations to rotate all of the photos 180° and flip them horizontally and vertically provides the best trade-off.

3.2 W-Net

We opted for developing a model that falls into the macro-category of the *W-Net* [4] models. Such models, which proved to perform considerably good on many other semantic segmentation settings [1], are

composed of **two cascaded *U-Net* networks** [2]; this approach gives the possibility to the first network to learn coarse-grained patterns in the image, while the second to refine such patterns. The two U-Net are equal in structure, with both encoder and decoder paths having 4 blocks, composed of 2 stacked convolutional layers with *dropout* and *batch normalization* followed by *max-pooling*. The number of filters (3x3 pixels) used in each block increase in the encoding path towards the *bottleneck* (32, 64, 128, 256, until arriving at 512 in the *bottleneck*, that has the same structure as the other blocks). We opted to increase the information flowing between layers by connecting through *skip-connections* not only layers from the same U-Net, but adding *skip-connections* between all possible forward connections through layers of the same activation size, across the two U-Nets, as in [3].



3.3 Feature fusion (feature pyramid)

As it can be seen in various state-of-the-art work[1], combining the output of the network with other intermediate outputs can enhance greatly generalization performances, as the patterns at the various scales and detail level can influence the output concurrently. This was achieved by rescaling the outputs of all the layers of the last decoder, and the output of the first U-Net into the shape of the image, by means of a convolutional layer and *bilinear upsampling*; then a final convolutional layer with 1x1 filters, and a *softmax* activation, combine these outputs into a single mask: the network's output.

3.4 Loss function

We opted for a linear combination of different loss functions, in order to capture various aspects of interest in the segmentation process:

$$\mathcal{L}_{\text{dice}} = 1 - \frac{2\sum_{i=1}^{N} p_i g_i + \epsilon}{\sum_{i=1}^{N} p_i^2 + \sum_{i=1}^{N} g_i^2 + \epsilon} \tag{1}$$

$$\mathcal{L}_{\text{focal}} = -\sum_{i=1}^{n} \alpha(i - pi)^{\gamma} CCE(y, \hat{y}) \tag{2}$$

$$\mathcal{L}_{\text{combined}} = k_{\text{dice}}\mathcal{L}_{\text{dice}} + k_{\text{focal}}\mathcal{L}_{\text{focal}} \tag{3}$$

Loss (3) combines a *Dice loss* component, that approximates better the *Mean IOU* metric to optimize, rather than *categorical cross-entropy* (CCE), and a *Focal CCE* loss component, that allows for (a) smoother gradients and (b) focusing on examples on which the model is not confident about. We also experimented with a boundary loss function based on the *Structural similarity Index Measure* (SSIM) between predicted and true masks, which did not improve performances.

3.5 Models Ensemble

Our solution employed an ensemble of two models with identical architectures (3.2) but trained using different loss functions: one trained with (3), and the other trained with *categorical cross-entropy* for general-purpose segmentation.
The ensemble prediction $\hat{y}$ for input x is computed as follows:

$$\hat{y} = \arg\max_{c} \left(\sum_{i=1}^{n} w_i \cdot p_i(x, c) \right) \tag{4}$$

where n is the number of models in the ensemble, $p_i(x, c)$ represents the probability prediction of class c by the i-th model for input x, and w_i is the weight assigned to the i-th model (the final weights were determined by an optimization procedure on the validation set).
This approach leverages the **complementary strengths** of different loss functions—enhancing performance on underrepresented classes while maintaining robust overall predictions. By combining models, the ensemble reduces individual biases

and improves generalization, yielding more reliable and accurate segmentation results.

4 Experiments

During the development of this project we experimented with a variety of techniques, related to different aspects of the model, to try to improve its performance.
Basing mainly on the model architectures showed in precedence we then proceeded trying to work on the structure, sample weights and ensemble learning.

4.1 Network Structure

We tried different techniques to improve our architecture:

- **Normalization:** We experimented with several normalizing layer types (group and layer normalization), but we discovered that batch normalization worked best for our structure.
- **Skip connections:** We experimented with several skip connection types (addition, concatenation, weighted additions) between the encoder and decoder or residual in the same decoder/encoder and discovered that concatenation is the best option.
- **Bottleneck:** Using squeeze-and-excitation, dilatation, and global context modules, we experimented with several bottleneck types with parallel branches; nonetheless, a single bottleneck provided us with better performance.
- **Adaptive fusion:** Before the final softmax, we attempted to apply an Adaptive fusion module; nevertheless, concatenation performed better on the test set.

4.2 Sample weights

To fight the problem of data scarcity regarding in particular class four, we tried to adopt class weights, initially using them to weight the value of the loss function on the frequency with which each real mispredicted class appeared in the data (this did not lead though to a significant improvement in the performance), and then to compute sample weights, to give more importance in the learning to the images whose labels contains more samples of the less frequent class: this strategy was the one which gave us the best results, and which we kept using in our final model.

5 Results

Table 1: Class-wise MeanIoU (validation)

	0	1	2	3	4
model1	0.00	0.85	0.80	0.86	0.32
model2	0.00	0.88	0.82	0.89	0.00
ensemble	0.00	**0.88**	**0.84**	**0.89**	**0.32**

Table 2: MeanIoU for all the models

Model	MeanIoU [val]	MeanIoU [test]
model1	0.71318	0.69710
model2	0.65132	0.49594
ensemble	**0.73554**	**0.72216**

Final per-class *Mean IoU* for the various models can be seen in 1. We can observe the effectiveness of the ensemble in equalizing performances among classes.

6 Discussion

The ensemble of two specialized models achieved good overall results, but class 4 remains the main bottleneck for the model's performance. The limited data for this class and the risk of overfitting on the test set, which was consistently the same during evaluations, are the most likely causes of the performance limitations.

7 Conclusions

7.1 Main Contributions

Main architecture: A. Attivissimo, L. Bianconi.
Ensemble development: M. Busso, A. Fiorini.
Report: all team members.

7.2 Possible improvements

Even though we've conducted a lot of research and trials, performance might be enhanced by using a different U-Net block and bottleneck. Preprocessing data to enhance additional images with class 4 could be an additional option.

References

[1] S. Khouloud, M. Ahlem, T. Fadel, and S. Amel. W-net and inception residual network for skin lesion segmentation and classification. *Applied Intelligence*, 52:3976 – 3994, 2021.

[2] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[3] L. Shuai, W. Zou, N. Hu, X. Gao, and J. Wang. An advanced w-shaped network with adaptive multi-scale supervision for osteosarcoma segmentation. *Biomedical Signal Processing and Control*, 80:104243, 2023.

[4] X. Xia and B. Kulis. W-net: A deep model for fully unsupervised image segmentation, 2017.