POLITECNICO
MILANO 1863

AN2DL - Second Homework Report
stiratissimi

Nicolò Vacis, Giovanni Vaccarino, Vittorio Palladino, Maria Fernanda Molina Ron

nicolovacis, giovaaaaa, vittoriopalladino, fermolina12

274316, 274402, 278541, 250258

June 1, 2025

1 Introduction

The AN2DL challenge involves developing a **supervised segmentation model** to categorize five classes of Mars terrain. The primary objective is to maximize mean intersection over union metric.

2 Problem Analysis

Dataset: The training set consists of 2,615 grayscale Mars images across five classes: background, soil, bedrock, sand, and big rock. An unlabeled test set includes 10,022 grayscale images.

Architecture: Our aim was to determine the most effective model architecture through systematic testing and analysis on the test set.

Main challenges: The primary challenges involved balancing the dataset, optimizing augmentation strategies, and building a cohesive network with well-integrated blocks. While the model performed well in defining class boundaries, accurately classifying individual classes remained difficult.

Initial assumptions: We assumed that data balancing was crucial to prevent biases in the model. Additionally, we recognized the importance of augmentation while ensuring that image transformations did not alter their structure.

3 Methods applied

Dataset preprocessing: We converted NPZ files to JPG for manual inspection, identifying and removing outliers with alien patterns. Next, we analyzed the test set and found it visually similar to the training set, indicating consistency in data characteristics.

We analyzed dataset distribution by image count per class and total pixel count per class. The pixel count was crucial for revealing a significant imbalance in Class 4. To address this imbalance, we applied a two-step solution. First, images containing Big Rock were duplicated 2–3 times with additional augmentations to prevent overfitting. Then, pixel-based balancing was applied by extracting Big Rock pixels from labeled data, pasting them onto random images, and duplicating the originals to retain information [4].

Finally, we applied general augmentations like rotations and solarization to the dataset to enhance generalization[2].

For the final experiments, we utilized the entire dataset for training to fully leverage its potential and maximize model performance. However, it did not improved the result as expected.

Model Architectures: We started with U-Net as our baseline architecture. While experimenting, U-Net remained central to our process. We

also tested models like SegNet[5], DeepLab[11], and Mask R-CNN[3], but they performed worse, except for U-Net++[1], which delivered positive results.

After finalizing U-Net++ as our base, we experimented with integrating features from other architectures, like adding SegNet blocks to the U-Net framework.

The trials focused on testing different blocks within the architecture, dividing the problem into encoder, bottleneck, and decoder components.

4 Experiments

Going deeper in the previous points, those are some experiments made.

Network architectures: In the initial experiments, we tested various architectures, starting with a custom U-Net[16][9]. Comparisons were made with SegNet [5][15] (without a pretrained ResNet), DeepLab [13][11] (without a ResNet-50 backbone), a customized Mask R-CNN [14][3]using segmented images as labels and others listed here 1. Attempts to freeze ResNet backbone layers during training to stabilize convergence proved ineffective. U-Net++ outperformed other models, leveraging its advanced skip connections for more refined feature extraction.

With U-Net++ established as the baseline, we explored combining features from other architectures, though these efforts yielded no significant improvements. Focus shifted to fine-tuning U-Net++ components:

For the **encoder**, advanced modules like inception blocks, dilated convolutions, transformer-inspired multi-attention mechanisms[7][17], muscle attention blocks, and MLP blocks were explored to improve the network's ability to capture multi-scale and contextual features. Among these, only dilated convolutions showed noticeable improvements, likely due to their capacity to expand the receptive field without sacrificing resolution.

In the **bottleneck**, we tested multiscale residual path bridges, transfer bridges, dense attention modules, and atrous spatial pyramid pooling, among other techniques, to enhance information flow between the encoder and decoder while preserving spatial details. The residual dense attention module proved highly effective.

For the **decoder**, we experimented with fusion blocks and custom modifications to convolutional layers.

While some changes brought marginal improvements, the most significant gains came from general blocks applied across the network. A custom U-Net block, featuring a range convolution layer paired with a squeeze and excitation module[6], enhanced the model's focus on relevant features. The **global context block** [8], nested throughout the network, significantly improved segmentation by integrating contextual information. Additionally, the **cross-branch attention block**strengthened feature interaction across the network.

Fine Tuning: To optimize fine-tuning, we successfully used an LR reducer added with Warmup, while the cyclic LR proved ineffective. Class weights were calibrated to address dataset imbalances yielding modest accuracy gains. Early stopping was used to halt training at peak performance, reducing overfitting risk.

Categorical cross-entropy was retained as the core loss function, complemented by experiments with dice[10], focal [12], and boundary loss[10]. Combining these into a custom loss function provided a modest but measurable improvement in fine-tuning performance.

5 Results

Dataset Preprocessing: Balancing the dataset at the pixel level notably improved the model's performance, with the copy-and-paste method for Class 4 being pivotal. This approach included translation and rotation of cropped regions to enhance variability, effectively addressing class imbalance and making augmentations more realistic. Additional augmentations further reduced overfitting and improved the model's generalization to unseen data.

Architectures: U-Net++ outperformed as a baseline with refined skip connections, adding nested pathways and extra convolutions for better encoer-decoder integration, boosting segmentation results.

In the encoder, dilated convolutions demonstrated their strength by effectively capturing larger receptive fields, enabling the model to understand spatial relationships over broader areas.

For the bottleneck, the residual dense attention and global context modules performed best. The

former enhanced focus on key features while preserving spatial details, while the global context module excelled at capturing high-level semantic features for segmentation.

Notably, the U-Net block with a squeeze-and-excitation module significantly improved accuracy by emphasizing important feature maps, aiding in more accurate class classification. The cross-branch attention mechanism enhanced interactions between feature maps from different branches, effectively combining complementary information and improving segmentation of complex and ambiguous regions.

Interestingly, the two best networks were: one with all advanced blocks and another using a U-Net with a Squeeze-and-Excite block and the global context block as the bottleneck. Mixing them or adding advanced blocks to the second worsened performance. Both achieved similar IoU scores: 0.563 for the bottleneck-only model and 0.561 for the full-block model. 1

Fine Tuning: Class weights were crucial in addressing the imbalance of Class 4, the primary challenge. Optimizing these weights improved prediction reliability, resulting in a significant boost in the online evaluation score. Using a large batch size of 64 ensured smoother gradient updates, as it averaged out noise in the gradients. The Adam optimizer proved to be the most effective, including its variant with weight decay. Dynamic learning rate scheduling further enhanced this process by automatically adapting the step size based on the model's progress, ensuring faster convergence. Categorical cross-entropy was the most effective loss function for minimizing classification errors. However, fine-tuning with a custom loss combining Dice, Focal, and Boundary losses improved performance by addressing class imbalance (Focal), boundary accuracy (Boundary), and overlap quality (Dice), optimizing the older networks but curiously not the final one.

Table 1: Model Used (Kaggle score) (* ResNet pretrained models were retrained from scratch)

Parameter	Value	Parameter	Value
Unet++	**0.56**	DeepLab*	0.53
MaskRCNN	0.47	RUnet	0.44
SegNet	0.45		

6 Discussion

Careful dataset balancing and advanced augmentations were key to improving performance while avoiding biases. Iterative testing and refinement proved effective in evaluating different blocks. Ultimately, the enhanced U-Net architecture consistently outperformed more complex models less suited to the dataset. The main challenge lay in correctly classifying pixels, which was significantly mitigated by the squeeze-and-excitation block. Strong correlations between training, validation, and test results indicated good generalization. Additionally, the architectures maintained reasonable computational efficiency, ensuring practicality.

A key limitation was the scarcity of real images for Class 4, which impeded the model's ability to generalize. Additionally, the lack of color information in the dataset may have constrained the model's ability to utilize chromatic cues for better class differentiation.

7 Conclusions

We developed a robust U-Net++ based system, utilizing dynamic learning rates, fine-tuning strategies, and focused preprocessing with data balancing to optimize performance. Future directions include exploring GANs to simulate realistic colors and generate images for underrepresented classes, addressing imbalances more effectively. Additionally, incorporating pretrained models in the encoder could be investigated to assess their impact on performance and generalization.

Contributors

vacisnicolo: Data cleaning, architectures, training improvements.

giovanniv: Preprocessing, fine-tuning optimization.

vittpall: Preprocessing, architectures.

fermolina12: Architectures, residual connection experiments.

References

[1] T. Abdualimov. Unet++: Implementation of the unet++ architecture on tensorflow for segmentation of cell nuclei. https://medium.com/@abdualimov/unet-implementation-of-the-unet-architecture-on-tensorflow-for-segmentation-of-cell-nuclei-528b5b6e6ffd.

[2] albumentations (nick github). Using albumentations for a semantic segmentation task. https://albumentations.ai/docs/examples/example_kaggle_salt/.

[3] A. F. Gad. Object detection using mask r-cnn with tensorflow 1.14 and keras. https://www.digitalocean.com/community/tutorials/mask-r-cnn-in-tensorflow-2-0.

[4] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. https://keras.io/guides/keras_cv/cut_mix_mix_up_and_rand_augment.

[5] D. Gupta. Segnet. https://arxiv.org/abs/2307.13215, 2019.

[6] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu. Squeeze-and-excitation networks. https://arxiv.org/abs/1709.01507.

[7] Keras.io. How to correctly use test-time data augmentation to improve predictions. https://keras.io/api/layers/attention_layers/multi_head_attention/.

[8] Y. Li, Z. Shen, Y. Shan, T. P. A. R. Center, and T. U. of Hong Kong). Fast video object segmentation using the global context module. https://arxiv.org/abs/2001.11243.

[9] M. Maynard-Reid. U-net image segmentation in keras. https://pyimagesearch.com/2022/02/21/u-net-image-segmentation-in-keras/, 2021. Accessed: 2024-11-16.

[10] S. Paul. Highly accurate boundaries segmentation using basnet. https://keras.io/examples/vision/basnet_segmentation/.

[11] S. Rakshit. Multiclass semantic segmentation using deeplabv3+. https://keras.io/examples/vision/deeplabv3_plus/, 2021.

[12] Tensorflow. tf.keras.losses.categoricalfocalcrossentropy. https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalFocalCrossentropy.

[13] Vacis-Vaccarino-Palladino-Molina. Deeplab-script. Final/Models/DeepLab.

[14] Vacis-Vaccarino-Palladino-Molina. Maskrcnn. Final/Models/MaskRCNN.

[15] Vacis-Vaccarino-Palladino-Molina. Segnet. Final/Models/SegNet.

[16] Vacis-Vaccarino-Palladino-Molina. Unet script. Final/Models/UNet.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. https://arxiv.org/abs/1706.03762.