AN2DL - First Homework Report
ReLUsive Minds

Nicola Chiarani, Nastaran Ghaffari Elkhechi, Andrea Parenti, Alessandro Staffaroni Biagetti

nicolachiarani, Nastarangfr4, anpare, alestaffa

11001700, 10900702, 10620763, 10659399

December 14, 2024

1 Introduction

This report outlines the work completed for the second homework project of the "Artificial Neural Networks and Deep Learning" course at Politecnico di Milano. The project focuses on the semantic segmentation of 64x128 pixel grayscale images of mars terrain into five predefined categories.

2 Problem Analysis

This project focused on semantic segmentation of Mars terrain using a dataset consisting of images and their corresponding pixel-wise masks. Each mask represents the terrain type for each pixel, with the following class labels: Background (Class 0), Soil (Class 1), Bedrock (Class 2), Sand (Class 3), and Big Rock (Class 4). The dataset was divided into a training set of 2,615 images and a test set of 10,022 images, both with dimensions of 64 $\times$ 128 pixels. Each image in the training set was paired with its corresponding mask.
The dataset demonstrated a severe imbalance in the distribution of pixels across the five classes, which required careful consideration during model development and training to ensure fair representation of all classes:

- Class 0: 4988826 instances;
- Class 1: 6957538 instances;
- Class 2: 4776810 instances;
- Class 3: 3770823 instances;
- Class 4: 26963 instances.

As in the first project, our initial step involved searching for anomalies in the images and masks. To achieve this, we utilized an image hashing technique, which allowed us to detect duplicated images efficiently. We hypothesized that outlier images or masks, if present, would likely be duplicated throughout the dataset. This assumption was validated as we identified 110 outlier images, including images with alien faces. Interestingly, all these outliers shared the same mask, which resulted in them being clustered into the same hash group. Removing these masks reduced the size of our cleaned training set to 2,505 images and masks. Before diving into the model development and training part, it is worth mentioning that we split the training dataset into training and validation sets, allocating 250 images for validation, normalized pixel values to the range [0, 1] and added a color channel to match the model's input requirements.

3 Model Developement

The development of the segmentation model involved extensive experimentation to optimize its performance. The initial implementation began with a straightforward U- Net architecture, designed without data augmentation. This version featured a 2- layer encoder (with 32 and 64 filters) and a 2-layer decoder employing UpSampling2D for upsampling. To further enhance the model's capability, the network depth was progressively increased by testing configurations with 3, 4, and 5 layers in the encoder-decoder structure. Correspondingly, the bottleneck filters were set to 256, 512, and 1024 for the 3-layer, 4-layer, and 5-layer architectures, respectively. After rigorous evaluation, the best performance was achieved using the 4-layer network. Larger models, such as the 5-layer configuration, did not demonstrate significant improvements. This limitation was likely attributed to the small size of the input images, which constrained the effective utilization of deeper architectures.

4 Data Augmentation

Before implementing augmentation techniques, we tried to balance the dataset by oversampling the minority class. Specifically, additional images featuring "big rocks" were introduced prior to any augmentations. This was achieved through two methods: duplicating entire images and inserting cropped regions containing big rocks while preserving their aspect ratios. Furthermore, all pixel values were normalized.

A comprehensive set of data augmentation techniques was tested, including horizontal and vertical flips, translations, rotations, and adjustments to contrast and brightness. Among these, horizontal and vertical flips yielded slight performance improvements. For augmentation methods involving pixel position changes, controlled randomness was incorporated to ensure consistent transformations were applied to both images and their corresponding masks, thereby maintaining the integrity of the segmentation task.

The CutMix augmentation technique was also tried, involving:

- Cropping a rectangular region from one image and replacing it with a padded patch from another image, ensuring precise overlap between the patch and the cropped region.
- Applying the same process to the corresponding masks to maintain perfect alignment between image and mask.

Additionally, the augmentation pipeline included an intermediate step to expand mask dimensions during augmentation operations and subsequently reduce them back to their original shape. This ensured seamless compatibility with the model's input requirements while optimizing computational efficiency.

Lastly, the augmented dataset was batched and prefetched to improve training efficiency. Augmentation techniques were applied exclusively to the training data, ensuring the validation set remained unchanged to provide a consistent basis for evaluation.

5 Loss Functions, Metrics and Callbacks

Regarding the loss functions [1], a range of them was tested, including Focal Loss, Weighted Sparse Categorical Cross-Entropy, Dice loss, FTversky loss and combinations of these. As for the combination, we adopted a two-stage warmup approach. Initially, the model was compiled using Sparse Categorical Cross-Entropy as the loss function and the Adam optimizer with a learning rate of 0.001. This setup was chosen to stabilize the training process in the early epochs.

In the second stage, in order to address the severe class imbalance in the dataset, we replaced Sparse Categorical Crossentropy with FTversky loss. This loss function adjusts the weights of false positives and false negatives using alpha and beta parameters. These parameters were set to 0.7 and 0.3, respectively, to prioritize reducing false negatives over false positives. This weighting was particularly important for our segmentation task, as missing smaller or underrepresented classes (false negatives) could significantly impact the model's performance and utility. For this second stage, the model was recompiled with a reduced learning rate of 0.0005 and continued training of the first stage.

Among all these attempts, Dice loss produced the best results, due to its robustness in handling class imbalances and variations in class sizes. Next, class weights were computed based on pixel frequencies, with further experiments using logarithmic weights to refine results.

As for callbacks, both ReduceLROnPlateau and EarlyStopping monitored the MeanIoU metric to adjust the learning rate and stop training if no improvement was observed. The MeanIoU metric was particularly relevant for our semantic segmentation task as it measures the overlap between predicted and ground truth masks across all classes, providing a robust evaluation of segmentation performance.

6 Additional Features

We experimented with additional features to assess their impact on the results:

- Replaced "UpSampling2D" with "Conv2D Transpose" during upsampling, which demonstrated better results.
- Weight decay was explored using the AdamW optimizer with various values, including 0.001, 0.005, 0.0001, and 0.0005, where the optimal configuration was achieved with a decay of 0.0001.
- Nadam with an exponential learning rate scheduler: the latter is able to progressively lower the learning rate (which starts at 1e-3) after a fixed amount of epochs, fixing some fluctuations.
- Batch sizes of 8, 16, 32, and 64 were tested. Among these, a batch size of 16 provided the optimal results.
- Skip connections were employed to retain high-resolution features.
- Attention mechanisms such as squeeze-and-excitation and attention modules, between layers or after the bottleneck, though these did not yield significant benefits.
- Dilated convolutions were incorporated into both the bottleneck and skip connections to capture larger receptive fields.
- Squeeze-and-excitation blocks to the skip connections.
- Implemented a NAS (Neural Architecture Search) using Keras Tuner to find the best hyperparameters. Initialized NAS with experiments on 3, 4, 5, and 6 encoder-decoder blocks, with and without features like squeeze and excitation, CBAM, STN. However, NAS proved to be too computationally expensive, and achieving an adequate number of epochs for each tested model was not sustainable.
- Explored generation of image-mask pairs with a custom GAN. This approach proved to be unstable and highly resource-intensive, and raised concerns about the reliability of the results.

7 Dual U-Net

A dual U-Net architecture was developed to integrate global and fine-grained features for segmentation. The global branch, a 2-layer encoder-decoder, processed downscaled images to extract contextual information, while the fine branch, a 4-layer encoder-decoder, handled full-resolution images for detailed features. Outputs from both networks were combined and refined through additional convolutional layers to produce the final segmentation. Training employed weighted loss functions, using Sparse Categorical Crossentropy for the global output and Dice loss for the fine and final outputs. This architecture ended up being very resource-intensive and did not outperform the single optimized U-Net, making it less practical for deployment.

8 Final Model

To sum up, the final optimized model consisted of a U-Net with 4-layer encoder and decoder, Conv2DTranspose for upsampling, Dice loss, horizontal and vertical flips as augmentations, AdamW with a weight decay of 0.0001, and a batch size of 16, ReduceLROnPlateau with a starting learning rate of 0.001 and EarlyStopping. This configuration achieved a score of 0.53063 on the first test set.

9 Personal Contribution

All the members tried different solutions, augmentations, architectures and loss functions looking for the best possible solution.

References

[1] H.-T. T. Quang Du Nguyen. Crack segmentation of unbalanced data: The role of loss functions. *Available online at* `https://www.sciencedirect.com/science/article/pii/S0141029623014037`.