POLITECNICO
MILANO 1863

AN2DL - Second Homework Report
Team: Alan_tuning

Team members: Cerino Marco, Gallego Toscano Olivia, Pompigna Leonardo, Sichili Giulio.

Kaggle usernames: cerino23, oliviagallegotoscano, leonardopompigna, giulio8.

Matricola codes: 244780, 276398, 248985, 244842.

December 14, 2024

1 Introduction

This paper is the final report of the second homework assigned to students enrolled to the AN2DL course held in Politecnico di Milano in 2024/2025.
The project aimed to develop a deep learning model to segment martial terrain images. Techniques learned in the course and additional methods were used to achieve high prediction performances on unseen data.

2 Problem Analysis

Initially, we inspected the data and identified several artificial outliers.
After removing the outliers, a significant class imbalance became evident, requiring corrective measures. Specifically, the percentage of class 4 (large rocks) was 0.13%, which was over 100 times lower than the percentage of the other classes.
The key challenge in this segmentation task is that the classes correspond to pixel labels, while the model processes entire images. This complicates tasks such as maintaining consistent class distributions during dataset splitting or augmenting specific pixel classes.

3 Method

Our approach was divided into several key stages, each employing specific techniques to optimize the model's performance.

3.1 Preprocessing

To ensure better representation of class 4, the dataset was split by separately handling images with and without class 4, using a 0.1 validation fraction. This approach increased the percentage of class 4 pixels in the validation set from 0.06% (non-stratified split) to 0.19%, improving both training and the alignment of the validation set performance with test set performance. To mitigate the under-representation of class 4, we augmented the training set with a targeted approach. Images containing class 4 were extracted, and the centroids of class 4 labels were calculated. These images were cropped around the centroids, reducing their dimensions to 25% of the original size. This process produced a dataset primarily featuring large rocks, which was subsequently used for various augmentation methods detailed in Section 4.2.

3.2 Model

The model used was a U-Net [5] designed from scratch, with a GridMask layer[1] added at the input stage to improve generalization.

Residual connections[2] were incorporated to mitigate the vanishing gradient problem. The bottleneck layer was enhanced with the Squeeze-and-Excite[3] technique, which dynamically adjusts the weights of feature map channels, improving the model's ability to learn meaningful representations. To prevent overfitting, an early stopping mechanism was implemented by monitoring the validation metric, specifically the Mean Intersection over Union (IoU) on the validation set.
Since the problem involved multi-class segmentation, a softmax activation function was applied in the model's final layer. The softmax formula is as follows:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}, \tag{1}$$

where z_i represents the logit of class i, and C is the total number of classes.
The chosen loss function was a weighted cross-entropy loss, where the class weights were computed statically at the start, based on the initial class distribution. This loss function revealed to be the most performing with respect to others like the dice loss function, the focal loss function or a linear combination of the three.
The formula for the weighted categorical cross-entropy loss is:

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{C} w_j \cdot y_{ij} \cdot \log(\hat{y}_{ij}), \tag{2}$$

where N is the number of samples, C the number of classes, w_j the weight associated with class j, y_{ij} the ground-truth value (one-hot encoded) for sample i and class j, and $\hat{y}_{ij}$ the predicted probability for sample i and class j.
The formula for calculating class weights is:

$$\text{weight}[i] = \frac{\text{total samples}}{\text{number of classes} \cdot \text{class count}[i]}. \tag{3}$$

However, the weights were further modified as described in Section 4.1
To optimize the model, we used the AdamW optimizer[4], which decouples weight decay from the gradient update, enabling better control over regularization. The learning rate scheduler with the Reduce on Plateau technique dynamically adjusted the learning rate when the validation metric ceased to improve.

3.3 Performance Evaluation

The metric used to evaluate the model's performance was the Mean IoU of the single classes excluding the background class (class 0). The formula for computing the Mean IoU is given by:

$$Mean\ IoU = \frac{1}{N}\sum_{i=1}^{N}\frac{|A_i \cap B_i|}{|A_i \cup B_i|}, \tag{4}$$

where A_i represents the predicted pixels for class i, and B_i the ground truth pixels for class i.
Apart from this metric, we analyzed the IoU of individual classes to identify which classes were performing the worst.

4 Experiments and discussion

Table 1 shows the results of the most relevant experiments.

Table 1: Experiments.

Class weights	**Test Accuracy (%)**
Initial class weights	46
Class_weights[0]=0	58-68
Normalized class weights	**70-73**
Augmentation	**Test Accuracy (%)**
Baseline	62
Augmentation A	**68**
Augmentation B	67
Augmentation C	68
Model	**Test Accuracy (%)**
Baseline model	46-58
Sq&Ex + Residual connections	62-70
More filters	**73**

4.1 Impact of manipulating the class weights

The class weights in the loss function played a crucial role in addressing the dataset's inherent class imbalance. Since the Mean IoU metric excluded class 0 (background), we hypothesized that nullifying the weight for this class would simplify loss function optimization. However, this simplification risked misclassifying background pixels as other

classes, potentially reducing their IoU scores.
To test this, we set the weight of class 0 to zero. Surprisingly, the metric improved by approximately 10 percentage points. This result suggested that simplifying the loss function significantly benefited the model's learning process, outweighing the negative impact on the IoU scores of other classes. Ignoring class 0 in the loss computation enabled the model to focus on predicting the classes of interest, ultimately enhancing overall performance.
Additionally, we experimented with normalizing the remaining class weights between 0 and 1 to mitigate the extreme influence of underrepresented classes. This approach led to a moderate improvement in performance, highlighting the advantage of balancing the influence of different classes during training.

4.2 Impact of the different augmentations

Several oversampling techniques for class 4 were tested on the model utilizing the Squeeze-and-Excitation architecture, with the first class weight set to zero. The number of synthetic images generated was optimized to 1000 for each technique.

- **Baseline**: replicating images containing large rocks improved the performance on that class, despite producing overfitting. The performance on the internal training set, indeed, was more promising than the actual final results. This however made us increase our effort to augment the class 4 samples' variety.

- **Augmentation A**: placing the cropped rocks in random position of other images revealed the best solution among the ones experimented. This produced more varied samples, and mitigated the unbalance of the dataset (Class 4: 5%, approximately 5 times lower the other classes).

- **Augmentation B**: placing the cropped rocks only on the prevalently background images revealed less effective, maybe because the produced images were less *natural*, more different to the ones possibly found in the dataset.

- **Augmentation C**: Applying morphological transformations to the rocks before integrating them into other images did not enhance performance. While this approach was intended to introduce greater variance into the dataset, the results indicated that the model was already generalizing effectively, making further augmentation efforts on the dataset unnecessary.

4.3 Impact of different models

- **Baseline Unet**: the initial simple model was able to achieve decent results, but was too simple to effectively embed information at very different complexity levels.

- **Squeeze & Excite + Residual connections**: these advancements lead the model to considerably improve the performance. The former, by using a mechanism to compress and then decompress information using a dense layer, effectively extracting the most relevant high level features from the channels incoming at the bottleneck, achieving higher generalization capacity; the latter, by obtaining a better gradient flow and a more fluent propagation of the information through the layers.

- **Adding more filters**: by doubling the number of convolutional filters per each block in the network, from 32 to 64, the model was more flexible to learn more patterns, i.e. it was more powerful, obtaining a moderate increase in performance.

5 Conclusions

This project was particularly valuable in learning how to approach a challenging segmentation task. It provided an opportunity to experiment with different techniques to address highly imbalanced classes and to build more effective models.
Although the final results are encouraging, the IOU for class 4 remained very low (9%), therefore there are still many steps that could be taken in the future to more effectively address class imbalance, including, for instance, incorporating additional data sources, such as unlabeled image datasets

References

[1] P. Chen and C.-C. Hsieh. Gridmask data augmentation. *arXiv preprint arXiv:2001.04086*, 2020.

[2] CodeLikeAGirl. U-net vs residual u-net vs attention u-net vs attention residual u-net. *Code Like A Girl Blog*, 2021.

[3] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1904.08254*, 2019.

[4] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2019.

[5] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *arXiv preprint arXiv:1505.04597*, 2015.