POLITECNICO
MILANO 1863

# AN2DL - Second Homework Report
# Autobot

Gabriele Carminati, Gabriele Carrino, Matteo Briscini

carmigab, gabricarr, matteobriscini

245680, 246970, 259098

December 14, 2024

## 1 Introduction

This project focuses on *semantic segmentaiton* of 64x128 grayscale images of **mars**. The task consists in correctly **predict the class for each pixel** among **5** possible **classes** (includind the background one) using deep learning techniques, in particular using **Convolutional Neural Networks**.

## 2 Problem Analysis

### 2.1 Dataset Analysis and Preprocessing

**PCA** and **T-SNE** were applied to inspect the dataset and identify outliers. While these methods highlighted some anomalies, they were **insufficient** for comprehensive detection. **Clustering with DBSCAN** on T-SNE-preprocessed data identified **81 outliers** across 8 clusters, plus **29 additional outliers** in the noise cluster (-1). These outliers shared a common visual feature with variations in orientation or background, explaining the limited effectiveness of PCA and T-SNE alone.

### 2.2 Dataset Splitting

The dataset was divided into **90% training and 10% validation** sets for model training.

### 2.3 Class Imbalance

**Class imbalance** was evident, with class 4 being significantly underrepresented. This issue was mitigated by applying **class weighting in the loss function**.

## 3 Method

After various experiment described in the following paragraph the final model was structured as follow. It is an **ensemble model** made by **5 models** applying the **bagging** technique with **majority voting** to make predictions. The five base models were all model performing **at least 61% intersection over union** on the test set. The base architecture was Unet for each model. Two of the group were a single Unet while other three were made of two parallel Unets (see experiments for more detail). Then they also differed a bit in the details, two of them had a transformer block in the bottleneck, four had a squeeze excite block. All of them had spatial attention layers and three of them also learnable upsampling. The final score obtained by this model was **66.3% mIoU**.

## 4 Experiments

A series of tests and experiments were conducted to reach the final method described in the previous

section.

## 4.1 Model Testing

To determine the best **model** for the task, several were tested, initially without any augmentation (see next section for augmentation testing). All the tested models used the **Unet architecture** as a base, and different additions and changes were performed.

### 4.1.1 Unet architecture

The core of this model is a **Unet block**, consisting of a stack of a **Conv2D**, a **BatchNormalization**, and a **ReLU** activation function. The stack dimension used was **2**. The model combined two of these blocks as a **downsampling path** (each followed by a MaxPooling layer), one as a **bottleneck**, and two as an **upsampling path** (each preceded by an UpSampling block and a concatenation with the corresponding downsampling block).

### 4.1.2 Tested variations

Different variations of the base architecture were tested, both alone and in combination with other variations:

- **Increase depth**: Different numbers of Unet blocks in the down and upsampling paths were tested. The best results were recorded with **3 blocks per path**.
- **Added residual connections**: A **summation** between the **input** and **output** of the Unet block, followed by a **ReLU** layer, was added, leading to slight improvements.
- **Attention gates**: Both **spatial attention** layers and **channel attention** layers were added. While the channel attention layer did not provide significant improvements, the spatial attention layer proved to be more useful.
- **Learnable upsampling**: The **Upsampling** layer in the **upscaling path** was replaced by a **Conv2DTranspose** layer, which allowed the learning of the upsampling operation, providing a slight performance increase.
- **Dual Unet - coarse and fine stream**: Two parallel Unets were used to capture both **larger** and **finer contexts**, with one having a **doubled number of filters** for each layer. The outputs of these layers were then combined.
- **Transformer layer**: A transformer layer consisting of **LayerNormalization**, a **MultiHeadAttention** layer, and a small **feed forward network** was added in the bottleneck, resulting in a small increase in segmentation performance.
- **Squeeze excite layer**: A layer consisting of a **GlobalAveragePooling**, followed by **two Dense** layers, was added in the Unet bottleneck. This layer improves feature representation by enhancing **channel sensitivity**, leading to notable improvements in results.

These variations were tested both individually and in combination, leading to the final model architecture, which achieved the best results in terms of mean intersection over union on the test set.

## 4.2 Loss Function Testing

In parallel with model architecture testing, different loss functions were also evaluated:

- **Sparse Categorical Cross Entropy**: The first attempt used the default Sparse Categorical Cross Entropy loss from Keras, establishing a solid baseline.
- **Focal Categorical Cross Entropy**: This loss function was chosen to address class imbalance by assigning different weights to each class (details in the next section).
- **Multiclass Dice loss**: This loss is effective for handling class imbalance and optimizing the intersection over union metric.
- **Structural similarity**: Aimed at maintaining the image structure, improving boundary identification.

The best performance was achieved by combining **Focal Categorical Cross Entropy** and **Sparse Categorical Cross Entropy** with specific class weights.

#### 4.2.1 Class Weights

Due to the class imbalance in the dataset, class weights were computed using inverse frequency, with higher weights assigned to underrepresented classes. These weights were calculated using the **compute_class_weight** function from the **sklearn.utils** library, without normalization to improve performance. A logarithmic version of the weights was also tested but did not yield better results. The background class weight was set to 0 to avoid affecting the intersection over union computation, which significantly improved the test score.

### 4.3 Augmentation

To properly implement augmentations on images and labels, the **Albumentations library** [1] was utilized within an **augmentation pipeline** designed to expand the dataset. The tested augmentations included geometric transformations (scaling, translation, rotation), horizontal flipping, image downscaling, coarse dropout, and Gaussian blur. The results of the tests on a **Dual UNet network (coarse + fine stream)** using **Categorical Focal Cross-Entropy** as the loss function are shown below:

| **Augmentation** | **val Mean IOU (%)** |
|---|---|
| Empty Model | 44.43 |
| Image Downscaling | 56.51 |
| Gaussian Blur | 35.11 |
| Coarse Dropout | 55.81 |
| Horizontal Flipping | 56.68 |
| Geometric | 41.82 |
| Geometric (no rotation) | 53.87 |

The results indicate that Gaussian Blur and Rotation had detrimental effects and were removed. It was also noted that applying multiple augmentations to a single image worsened model performance. The final approach involved increasing the training set size by fourfold, applying a single, distinct augmentation per image.

### 4.4 Majority Voting

To exploit potential differences in the prediction accuracy of each model, a **bagging** technique was used. The five most performant models on the test set were combined in a **majority voting system**, aiming to leverage the diverse features learned by each model.

## 5 Results

The ensemble model achieved a **66.3%** mean intersection over union (mIoU) on the test set, slightly outperforming baseline models. Majority voting among five Unet-based models improved robustness, while spatial attention layers and squeeze-excite blocks enhanced feature representation. Effective augmentations like horizontal flipping and coarse dropout contributed to performance gains, while **weighted loss functions** addressed class imbalance, improving segmentation for minority classes.

## 6 Discussion

The ensemble approach proved effective by combining complementary architectures and exploiting diverse features. Tailored augmentations boosted performance, though some, like Gaussian blur, degraded results, highlighting the need for dataset-specific tuning. Class weighting improved minority class segmentation but introduced trade-offs for dominant classes. Excluding the background class from loss calculations enhanced test performance but may reduce applicability in tasks requiring background segmentation.

## 7 Conclusions

This project achieved significant improvements in semantic segmentation through an ensemble model leveraging attention mechanisms, augmentation, and loss weighting. Future efforts could refine augmentation pipelines, explore advanced architectural enhancements, and balance class weighting strategies. Overall, the model demonstrates strong segmentation capability.

## 8 Individual Contributions

Each team member took part in almost every aspect of the project, however this was the main work distribution:

- **Gabriele Carrino**. Different model architecture testing.
- **Matteo Briscini**. Data exploration and augmentation testing.
- **Gabriele Carminati**. Help with model testing and loss function testing.

## References

[1] Albumentations Documentation. *albumentations*. URL: `https://albumentations.ai/`.