AN2DL - Second Homework Report
LastHoursGang

Andrea Ferrarini, Chiara Fossá, Ermelinda Giulivo, Marina Mastroleo

aafnii, keirafox, ermelinda15, marinamastroleo

259198, 259481, 246877, 260102

December 14, 2024

1 Introduction

This project focused on the development of a deep learning model for semantic segmentation of Mars terrain images. The primary goals were:

1. Perform analysis and pre-processing of the provided dataset.
2. Build and train a deep learning model for semantic segmentation.
3. Evaluate the performance of the model using appropriate metrics.

This report will delve into the analysis performed and methodologies used during the development phase, as well as the results obtained by the team.

2 Problem Analysis

During the initial analysis of the problem we focused our attention on the dataset characteristics, trying to detect potential challenges and ways to tackle them.

2.1 Dataset Characteristics

The initial dataset was comprised of 2.615 gray-scale images of size 64×128 where each pixel was segmented into one of five possible classes *(background, soil, bedrock, sand and big rock)*. The dataset was stored as a numpy array of shape $(2615, 2, 64, 128)$, where the two channels on axis 1 contained the gray-scale image and its segmentation map *(with integers in the* $[0, 5)$ *range)*.

2.2 Main Challenges

After a preliminary inspection of the dataset the following issues and challenges were uncovered:

- **Outliers:** 110 samples had the picture of an alien superimposed on them, rendering them unusable.
- **Class Imbalance:** Pixels representing big rocks were severely underrepresented.

2.3 Dataset cleaning and pre-processing

The 110 outlier samples were removed very simply after noticing that their class map was always identical *(depicting the shape of a flying saucer)*. At this point, we normalized all samples into the $[0, 1]$ range and divided them into train and validation sets *(with rates 70%:30%. Given the small size of the dataset, no separate test set was created, as we opted to perform testing directly on Kaggle)*.
In order to tackle class imbalances and improve generalization we proceeded to augment our dataset; this was done by means of a custom `albumentation` pipeline as it handles geometric transformations

on both image and segmentation mask automatically *(crucial in our problem)*. The designed pipeline mainly performs geometrical transformations *(affine, flips and elastic)* and light non-geometrical transformations *(brightness, contrast and CLAHE)*.
In order to define the number of augmentations of each sample, we developed the following algorithm:

1. Compute total number of **pixel occurrences** of each class.
2. **Normalize** each count with respect to total number of pixels ($N \times 64 \times 128$).
3. Compute frequencies by **inverting** the obtained values.
4. **Limit** all values to a maximum value equal to the sum of all other values.
5. Augment each sample a number of times equal to the **maximum factor** among those of the classes in its class map.

This procedure can be described by means of the following equation:

$$\hat{f}_i = \frac{N \times 64 \times 128}{C_i} \implies f_i = \min\left\{\hat{f}_i, \sum_{j \neq i} \hat{f}_j\right\}$$

where C_i indicates the total number of pixels mapped as class i.
As a result of this pre-processing operation which ensured a greater number of augmentations for those images containing rare classes, the final training set was comprised of a total of 11.345 samples.

3 Model development

Our development was comprised of two separate phases: an initial experimental one followed by a more stable second phase where only hyper-parameter and augmentation adjustments were made.

3.1 Initial development

Our initial model, which was named `mark1.1`, was based on a U-Net architecture [5] with simple custom designed encoder and decoder, both built according to general design patterns [3]. This approach resulted in the development of the following architecture:

1. **Encoder** *(batch-augmentation after each convolution and ReLU activation)*.
   (a) 2x $64 \times 128 \times 32$ layers with max-pooling.
   (b) 3x $32 \times 64 \times 64$ layers with max-pooling.
   (c) 3x $16 \times 32 \times 128$ layers with max-pooling.
2. **Bottleneck** with 3x $8 \times 16 \times 256$ layers and nearest-neighbor up-sampling.
3. **Decoder** *(batch-augmentation after each convolution and ReLU activation)*.
   (a) 3x $16 \times 32 \times 128$ layers with up-sampling.
   (b) 3x $32 \times 64 \times 64$ layers with up-sampling.
   (c) 2x $64 \times 128 \times 32$ layers.
4. **Output map** of shape $64 \times 128 \times 5$ obtained by means of a 1×1 convolution with softmax activation.

This model reached a final mean-IOU on the validation set of 46.555%, which was later also confirmed upon upload on Kaggle *(45.942%)*.

Further developments took the above described architecture as template and tried to further improve it; the first update consisted in adding attention mechanisms at the end of convolutional stacks, the idea being that deeper layers with a high number of channels could benefit from learning which channels carry more relevant information.
This was initially done by means of simple **Squeeze and Excitation** blocks [2], and later also using **Convolutional Block Attention Modules** [7], or CBAM, which not only learn which channels are most relevant, but also apply spacial attention mechanisms, that is, apply weights to individual pixels as well *(which is of particular interest for wider layers such as the entry of the encoder and ending of the decoder)*.
These intuitions *(SE blocks at the end of lower convolutional stacks and CBAM blocks at the end of higher convolutional stacks)* led to the improved model `mark4.3.7` which obtained a mean-IOU score of 49.066% on Kaggle.
Still related to the concept of attention, further inspiration was drawn from the **Attention-UNet** [6] architecture, where skip connections are concatenated with the decoder's up-sampled tensors by

means of **attention gates** which also apply dynamically computed weights to individual pixels based on their relevance.
Because this update only increased slightly the performance of the model *(52.172% on Kaggle)*, the attention turned toward the bottleneck of the network. In order to combine local and global information from the network's input, adding convolutions both with low and high dilation rates was considered. These would occur in parallel and then be concatenated, essentially replicating an **Atrious Spacial Pyramid Pooling** [4] block as in the **DeepLab-v3** architecture. This, coupled with the previous updates, produced a model which obtained a mean-IOU of 57.425% on Kaggle (`mark6.5.4`).
Finally, in order to be able to produce richer features, the encoder was deepened: an additional convolutional stack followed by max-pooling was added, leading to a bottleneck of size $4 \times 8 \times 512$.
Because the depth and size of the network could now prove to be challenging to train, an approach similar to that implemented in the **GoogLeNet** [1] network was developed: intermediate outputs are extracted from the bottleneck and middle point of the decoder *(by means of* 1×1 *kernel convolutions with softmax activation, followed by appropriately sized up-sampling)*.
This **Multi Scale Supervised** architecture *(akin to that of **DS6** [8])* was then trained weighting the three losses respectively 0.2, 0.3 and 1.0, the idea being that lower levels will always produce less accurate predictions hence should be weighted less, yet their use should help the overall model converge, guiding the deeper layers toward already approximately correct segmentations.
This last model (`mark6.5.7`) yielded a slight improvement in mean-IOU reaching a score of 58.656% on Kaggle.

3.2 Our final solution

Our final model is based on the `mark6.5.7` network, which starts from the base U-Net architecture *(with 4 down-samplings, leading to a* $4 \times 8 \times 512$ *bottleneck)*, with added **attention gates** in skip connections, **SE blocks** at the end of first two decoder stacks, **CBAM blocks** at the end of last two decoder stacks, **ASPP block** in the bottleneck *(with rates of 1 and 2)* followed by two **transformer blocks** in series which then enter the decoder.
The resulting network, named `mark6.5.8`, obtained a mean-IOU score of 60.450% on Kaggle, which was further increased to 61.820% by `mark6.5.10` after simply substituting all CBAM blacks with SE blocks instead.
Mean-IOU scores for training, validation, and testing of all mentioned models are reported in table 1.

3.3 Training

The final model was trained without MSS and using the `AdamW` optimizer along with an early stopping with 20 epochs patience. In order to help learning convergence, a learning rate reduction on plateau callback was also added with a patience of 3 epochs.
Several loss functions were implied throughout the development, starting from categorical cross-entropy and class-weighted dice, however in the end categorical **focal cross-entropy** proved to be the best choice.

Model	**Training**	**Validation**	**Kaggle**
`mark1.1`	48.010%	46.555%	45.942%
`mark4.3.7`	51.731%	50.121%	49.066%
`mark6.5.3`	58.209%	53.020%	52.172%
`mark6.5.4`	66.778%	58.650%	57.425%
`mark6.5.7`	70.120%	61.169%	58.656%
`mark6.5.8`	78.882%	62.250%	60.450%
mark6.5.10	**81.241%**	**63.335%**	**61.820%**

Table 1: Mean-IOU scores of all mentioned models

4 Contributions

- **Andrea:** main notebooks code structure, development of `mark6.5.10`, `mark1.1` and CBAM blocks.
- **Chiara:** development of `mark6.5.7` and MSS learning. Code documentation.
- **Ermelinda:** development of `mark6.5.3`, augmentation pipeline and evaluation code.
- **Marina:** initial dataset pre-processing, development of `mark6.5.4` and SE blocks.

References

[1] Y. J. P. S. S. R. D. A. D. E. V. V. A. R. Christian Szegedy, Wei Liu. Going deeper with convolutions. 2014.

[2] S. A. G. S. E. W. Jie Hu, Li Shen. Squeeze-and-excitation networks. 2019.

[3] N. T. Leslie N. Smith. Deep convolutional neural network design patterns. 2017.

[4] F. S. H. A. Liang-Chieh Chen, George Papandreou. Rethinking atrous convolution for semantic image segmentation. 2017.

[5] P. F. Olaf Ronneberger and T. Brox. U-net: Convolutional networks for biomedical image segmentation. 2015.

[6] J. S. L. L. F. M. L. M. H. K. M. K. M. S. M. N. Y. H. B. K. B. G. Ozan Oktay1, 5 and D. Ruecker. Attention u-net: Learning where to look for the pancreas. 2018.

[7] J.-Y. L. Sanghyun Woo, Jongchan Park and I. S. Kweon. Cbam: Convolutional block attention module. 2018.

[8] M. P. G. B. C. S. F. D. H. M. M. d. B. O. S. A. N. Soumick Chatterjee, Kartik Prabhu. Ds6, deformation-aware semi-supervised learning: Application to small vessel segmentation with noisy training data. 2022.