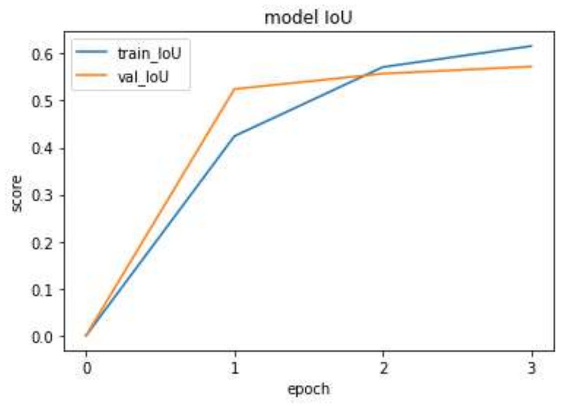
Image Segmentation

Homework 2 – Artificial Neural Networks and Deep Learning 2020

DATASET LOADING

The dataset is managed by the Python script *split_set*, which scans the provided folders and creates two text files containing a list of filenames for the training and validation sets, according to the settings passed as arguments. The validation examples are selected at random, depending on the *numpy* seed, which is fixed at the start of each notebook. The images are then loaded through the custom generator class *create_generator*, that also pre-processes and augments them.

SIMPLE U-NET WITH VGG-16 ENCODER



The model is the basic U-Net with VGG-16 backbone seen during the laboratory session. The weights for the encoder were initialised to the ones obtained on the ImageNet dataset, effectively implementing transfer learning. The decoder was trained from scratch, first on a single dataset (for example *bipbip – haricot*), then on multiple datasets combined (all *haricot*).

The following hyperparameters were tuned:

- Resolution of the input images
- Learning rate
- Batch size

The best results were obtained by using a combination of all the haricot datasets at almost full resolution (1536x2048) with a learning rate of 10^{-3} and a batch size of 4. This resulted in a mean *IoU* score of almost 51% on the test set in just three epochs. However, each training iteration took such a long time it was unfeasible to carry on with this method.

EXPERIMENTS WITH TILING AND LOSSES

The main issues with the previous approach were the strong class imbalance – as the number of pixels of the background class is significantly higher – and the high resolution of the input images, which required either extremely long training sessions or heavy resizing resulting in worse performance. Therefore, tiling was employed: every sample – and its mask – is divided into several, smaller overlapping patches on which the network is trained. Patches that only contain background pixels are discarded. The images of the test set are also split into patches for prediction but are later reconstructed using *average* interpolation.

Different patch dimensions were tested, but the best results on the test set were obtained using 800x800 patches, probably because smaller ones tend to incorporate only portions of the plants, consequently hindering the feature extracting capabilities of the network.

Tiling greatly improved training times but did not affect the score considerably.

The next step was switching from the *SparseCategoricalCrossentropy* loss to the *Dice* loss, which deals with class imbalance more efficiently. *Focal* loss was also tested. Neither of them had a positive impact on the performance of the network.

CUSTOM U-NET WITH SKIP CONNECTIONS

A different version of U-Net with skip connections was implemented and trained from scratch. The network uses *BatchNormalization* layers for better training performance. All previously described tests were replicated on this network. The best result was a mean *IoU* score of 48% on the test set, which looked promising considering transfer learning was not being employed here.

ALTERNATIVE BACKBONES

U-Net was tested with several other backbones, including ResNet, Inception, MobileNetV2, EfficientNet and DenseNet, taking advantage of the "Segmentation Models" library from GitHub. None of the models scored higher than the previous best result, despite careful tuning of the hyperparameters.

SEGNET-INSPIRED NETWORK

An experiment from the literature was partially replicated. The network is based on the SegNet architecture with the addition of residual blocks (Milioto 2018). The results were not particularly promising, therefore the idea was abandoned.

COMBINED APPROACH

Two separate neural networks were trained, one for the "crop" class and the other one for the "weed" class. This was achieved by setting to zero all pixels of the masks, except for the ones relevant to the current net (e.g., "crop" pixels for the crop net), before starting training. To obtain the final prediction on the test set, each image was fed to both networks, the results for the non-pertinent class were ignored while the interesting ones were merged by using the *argmax* function.

The two networks had a DenseNet backbone with additional *BatchNormalization* layers in the decoder.

A score of 30% *IoU* on the test set was achieved with this method, which deserves further experimentation efforts.

Marco Premi – 941388

Alessio Martignetti – 968430

Francesco Sammarco – 966229