# ANN Homework2

For this second homework, the preliminary part of dataset organization and file analysis was crucial, since the dataset was divided in different teams and crops, with images of various sizes and kind. At first we began experimenting with a really small portion of the dataset, selecting a single team from a single crop (Haricot/Bipbip). Due to the limited dimension of the dataset, we started approaching the problem with a transfer learning solution right away, because we assumed that training an entire model from scratch would have led us to really poor results. After focusing on a single crop/team and reaching fairly satisfying results, we started using a wider dataset (all the images from Haricot) to test our model on all the other teams. The homework was approached using encoder-decoder architecture.

## Dataset Management

In the first three notebooks, we managed the organization of the datasets; briefly:

- *"0-dataset-organisaton.ipynb":* we splitted the dataset in training set (90%) and validation set (10%)
- "*1_files_splitting.ipynb":* we emulated the data organization needed in the class *CustomDataset of* Multiclass Segmentation notebook
- *"2_test_dataset_unification.ipynb":* we took all the images from the *TestSet* and put them in a single directory, in order to make the testing process easier.

Further details are addressed in the notebooks.

## Resnet-50 model

We first approached the problem exploiting the notebook scheme that we observed on lab sessions, in particular *"Multiclass Segmentation"*. At first we had to make some adjustments to the *CustomDataset* class to make it fit to our problem. After testing the model with VGG16 as encoder, we decided to use Resnet-50 instead, in order to improve the performance in this

downsampling part of the model. After performing some tests and hyperparameter tuning, we decided to use 1024x1024 as target size for our predictions (tradeoff between accuracy and training time). For the decoder, we added some Convolutional layers in order to reduce the depth of the output, and then we iteratively applied blocks composed by the following layers:

- UpSampling2D (doubling the dimension of the map)
- Convolution2D (dividing by 2 the initial number of filters each time)
- BatchNormalization
- Relu Activation

This structure of the model allowed us to reach an overall meanIoU value of 0.38 on the test set.

### U-Net model

Simultaneously, we implemented an U-Net architecture following the scheme seen on the lectures. One of the advantages of this neural network is the fact that is trainable with small datasets and, given our slight one, we believed that it would have fitted nicely. In the opposite, one of the disadvantages is that U-Net wasn't trained on *imagenet* dataset, but on biomedical images, so the features extracted from those types of datasets may not have been ideal for our task. At first we reached 0.28 in val_meanIoU, but then we were not able to get close to that value anymore. Thus we decided to try to implement an *U-shape* architecture similar to the U-Net one, by creating a decoder from scratch and connecting it to the ResNet encoder with some skip connections.

### Resnet-50 model with Skip Connections

After experimenting with the previous ResNet model, we started thinking about ways to improve our results. We decided to implement *Skip Connections* in order to be more accurate in reconstructing the image in the Decoder part. Therefore we retrieved some output maps from the ResNet encoder (those matching the dimensions of the maps in the decoder), and we introduced some *Add* layers that simply sums the output map of the encoder and the map obtained after the Activation layer of the decoder; then the result of the sum

is upsampled. This tweak led us to a 10% improvement in the validation meanIoU with respect to the previous Resnet model, while on the test set we obtained an overall meanIoU value of 0.44.

### SegNet model

We were substantially satisfied with the results obtained with our models, but we continued looking up on the web to collect further suggestions, and we ended up discovering a very interesting new Deep Convolutional Encoder-Decoder Architecture called Segnet. The innovation of this model is all about the so-called *Unpoooling.* Basically, while performing the usual 2×2 MaxPooling in the Encoder, the corresponding MaxPooling indices (locations) are stored; then during UpSampling in Decoder, the maps are reconstructed in the original locations using those stored indices. This model allowed us to reach an overall meanIoU value of 0.22 on the test set. Despite being a pretty powerful architecture, the results were not that impressive, mainly due to the fact that our dataset was limited and not suitable to train a model from scratch.