Artificial Neural Network and Deep Learning
Politecnico di Milano

Project 2 : Image segmentation

Luca Massini 10504929

Arsène Marzorati 10785084

25-12-2020

The **ACRE competition** proposes an interesting subject of image segmentation. The idea is to use Deep Learning to improve the robot tasks in agriculture. Here, the dataset is provided for two crops (with different tools and different times of the day for the images). The task is to recognize where the crop is (in order to protect it) but also the weed in order to remove it.
We choose to keep all the images in order to have the largest possible dataset and also augmented with data augmentation. We decided to use an aggressive data augmentation process since this kind of task is really rare to obtain a transformed picture that is no more on. We had to focus on the meanIoU which is the main criterion for the competition. For that we worked with small batch size (around 10 images) and a learning rate between $1e-4$ and $1e-6$.

We split the work in four parts :

First we wanted to implement a basic network with encoder, decoder and prediction layer and then improve it with some techniques such as **Early Stopping** or **Drop Out**, in order to avoid **Overfitting**. The results were very low (around 19.1%), and we only detected one class (the crop) among the both interesting. The weed was too small to be captured by the model. We tried to increase the number of filters, but the Overfitting was more dangerous in this way. The Upsampling process was performed only by the concatenation of blocks composed of Upsampling and convolution (not the ideal block of Keras layers for that task). We did some trials with all the datasets and also with only one of them.

After this basic try, the idea was to use **Transfer Learning** for the encoder part because that part of the network acts like an image classifier and this method was already successfully used in the project for image classification. We used one pre-trained model (VGG16) and used the preprocess for the images. We also implemented fine-tuning on a part of the layers. The results were better but there is still the problem to capture the weed. The Upsampling was performed in the way described previously and also here we did some trials with all the datasets and also with only one of them. We got 19.6%

Then we returned to a basic model, trained on all the datasets, but with skip connections between the encoder and the decoder at each step (implemented as sums). This method allowed us to avoid a too important loss of information with the Max-pooling and Upsampling. Here, one important achievement, was to perform Upsampling through transpose convolution with stride= (2,2) to make this process more flexible and even learner in a better way. After many trials, another idea was to make two convolutions with different kernel sizes in order to select more information about the small class of different dimensions such as the weed. We performed the average of these 2 different convolutions in each network block. We trained this network and we decided to concatenate a simple network to it in order to capture some other details of the masks. We did this by freezing all the layers of

the previous trained network and we concatenated a simple network without skip connections to it.At this step we also changed the size of the images fed to the networks in order to select more details and to stay similar to the initial size (isometric transformation) in order to avoid a too important loss information due to the resizing process necessary to do the submission.
The results were very improved (34%).

Then we wanted to mix the improvements with skip connections. For this part we decided to build four networks with the same architecture of the previous one (expect for the number of filters) to be trained on the four different datasets of each project to have four specialized and finally to merge the predictions. One difficulty was that, since in this case we had fewer data for each network (even considering data augmentation), it was more difficult to tune the parameters in order to avoid the overfitting and the risk of the latter was higher. In order to avoid this, we built the networks with less filters in order to have a simpler model with respect to before. We got a little improvement compared to before (37.5%).

We tried a different approach in which we built 2 specialized networks, one on Maize and one on Haricot. The networks were simpler with just one convolutional layer for each block and therefor a unique filter size of size (3,3) and a number of filters equal to 55 which doubled at every layer in the encoder and it reduces of a factor of 2 in the decoder. A depth equal to 4. We decided to do so since we observed that one single network for all the datasets was good in some datasets but really bad in others. We merged the predictions of those networks in a unique file and the results were much better (around 42%).

Then we decided to train a single network on all the datasets again but with a single kernel size and with 2 convolutional layers concatenated for each model block. For the rest the network architecture was really similar to the previous one. The network was able to improve the score even better with respect to all the other cases. We achieved a meanIoU value around 0.4862 overall which was our best result in this competition.

Basic Network

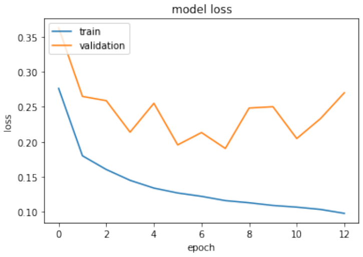

Figure 1: *Loss during the training of basic network*

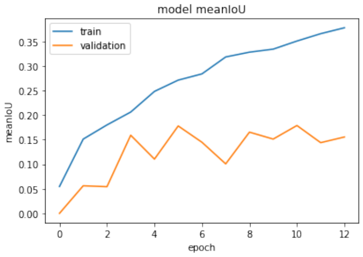

Figure 2: *MeanIoU during the training of basic network*

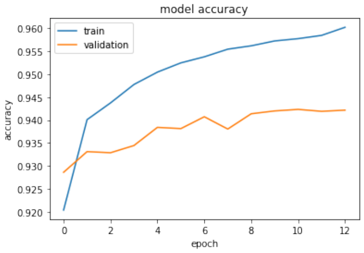

Figure 3: *Accuracy during the training of basic network*

Transfer Learning

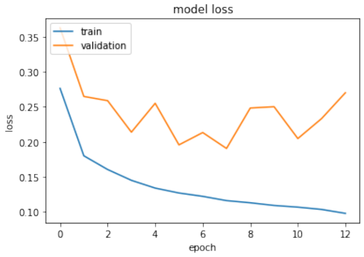

Figure 4: *Loss during the training with Transfer Learning*

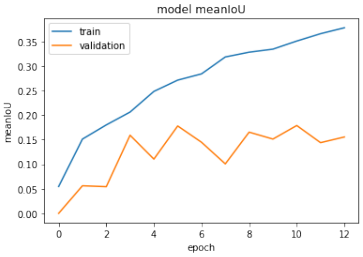

Figure 5: *MeanIoU during the training with Transfer Learning*

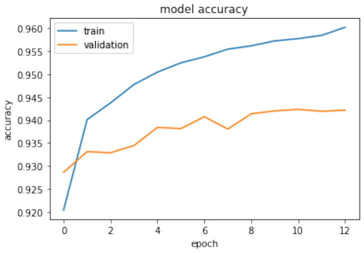

Figure 6: *Accuracy during the training with Transfer Learning*

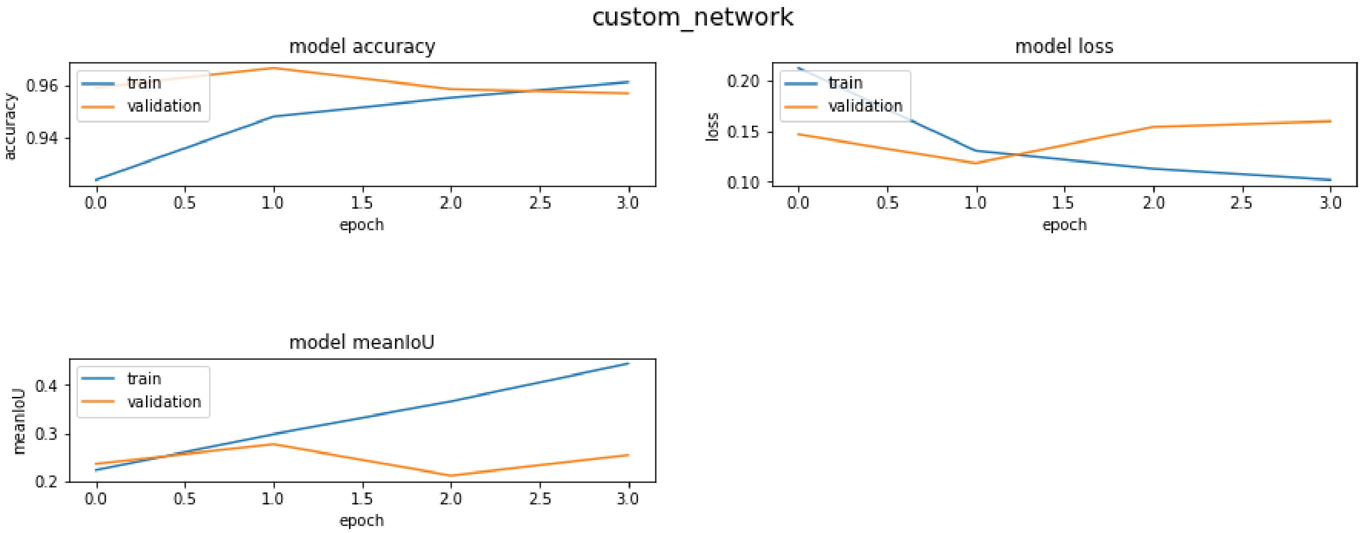

Figure 7: *Accuracy, MeanIoU and loss of the first network trained on all the dataset without concatenation*

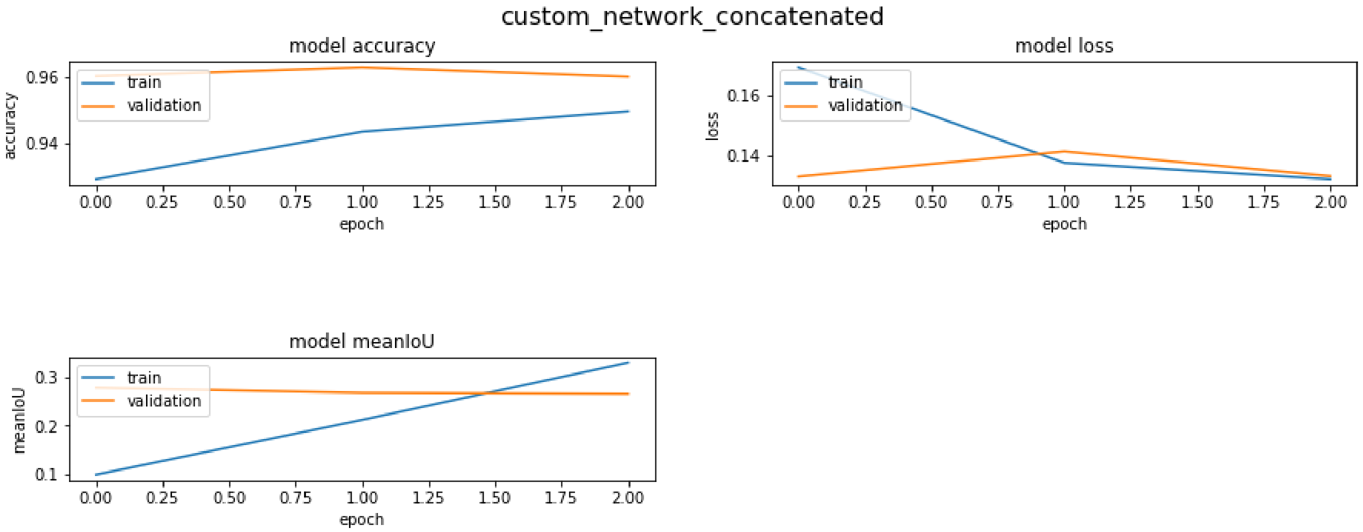

Figure 8: *Accuracy, MeanIoU and loss of the first network trained on all the dataset after the concatenation*