Homework 2 - Semantic Segmentation

January 21, 2021

Dataset The dataset is divided in four sub-dataset where each contains 2 folders of Haricot and Mais plants. Those plants have to be segmented correctly from the bad weeds. We have a total of 1440 elements (720 images and 720 masks used to train the model). The dataset is small, but we didn't suffer from the size of the dataset as we did on the classification homework.

Models Tried We first tried to create a small version of a Unet with some dropout layers to randomly drop pixels of the activation masks. The model was training pretty well, but we faced a lot of problems in the first submissions: our results were always near 1% IoU accuracy. We discovered that this problem was caused by the size of the predicted mask, which had to be at the original size of the image.

While we discovered this, we were already working on a much more powerful model: we used a VGG16 to build a custom Unet. Basically before all MaxPool layers (except for the last one) we added skip connections that are concatenated with Upsampling layers composing the decoder part of the model.

As this idea (using transfer learning on the decoder and training a whole decoder model with skip connection) was working really well we tried using networks which were really performing on classification tasks: Xception and ResNetV2. Surprisingly results were very bad compared to our VGG16 model (approximately 0.15 less in IoU on our validation split). We've suddenly returned to the VGG16 idea.

Optimization We used both Kaggle and Google Colab as cloud infrastructures to train our model. Images at full size were way too big to use as the input of our model because the VRAM wasn't enough, so we used reshaped images at 512x512 pixels and the RAM used was approximately 9.5GB. As the training time was astonishingly long we avoided using generators, which are useful for very large datasets but overkill for such small datasets which can be easily loaded in the RAM. Therefore we used numpy arrays to load the whole train dataset (images and masks reshaped at 512x512). We were really surprised how much the train time was improved. We went from 400 seconds for each epoch to 110 seconds.

Overfitting We used checkpoints to save the best weights (monitoring the validation loss), training the model for 30 epochs. We also used different percentages of validation split. We had to decrease the validation set from 80 images to 8 to reach our best score (0.7508). Data augmentation wasn't used as it wasn't a game changer as it could be in classification tasks where images are in different three dimensional scales. We've tried dropout and weight decay but this didn't help performing better.

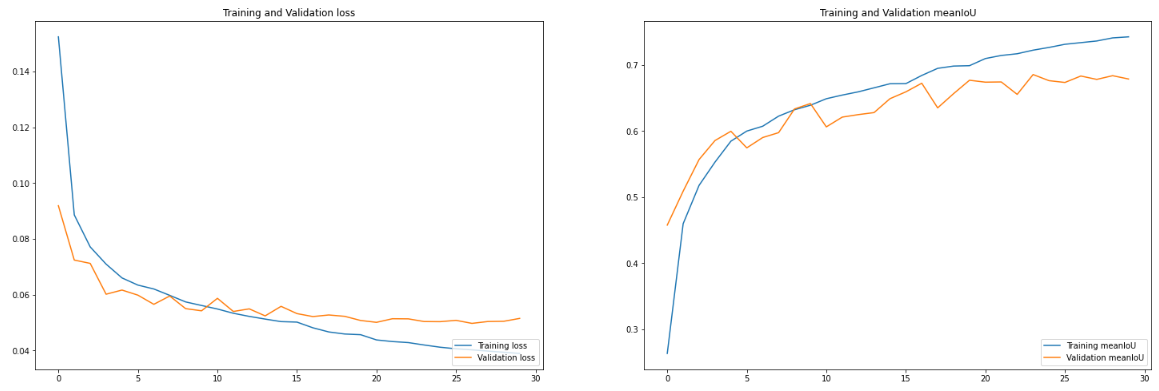

Figure 1: Training curves, the validation split is composed of 8 images so it's normal it's so noisy

Hyperparameters Tuning and Loss We didn't go crazy on hyperparameter tuning like the first challenge as no weight decay/data augmentation/dropout is used. The only hyperparameters we had to tune was the learning rate, the number of convolutional filters and the percentage of VGG16 to train. We had our best performance (0.75 meanIoU on the test set) training the whole VGG16 with a learning rate of $1e-5$. We also used a callback to schedule the decay of our learning rate after 15 epochs, observed to be the start of a plateau for validation set. For the loss function we implemented a simple form of weighted sparse categorical cross entropy, giving more weight to the class of weeds, which is less present than the background and other crops.

$$\eta^{t+1} = \eta^{t} \times e^{-0.1}$$

learning rate decay, triggered after 15 epochs

Final test update To make the final submit we merged manually the new data into the old dataset and we retrained the model from scratch changing some parameters: the validation split to 10% and the VGG fine tuned layers are now the final 80%.

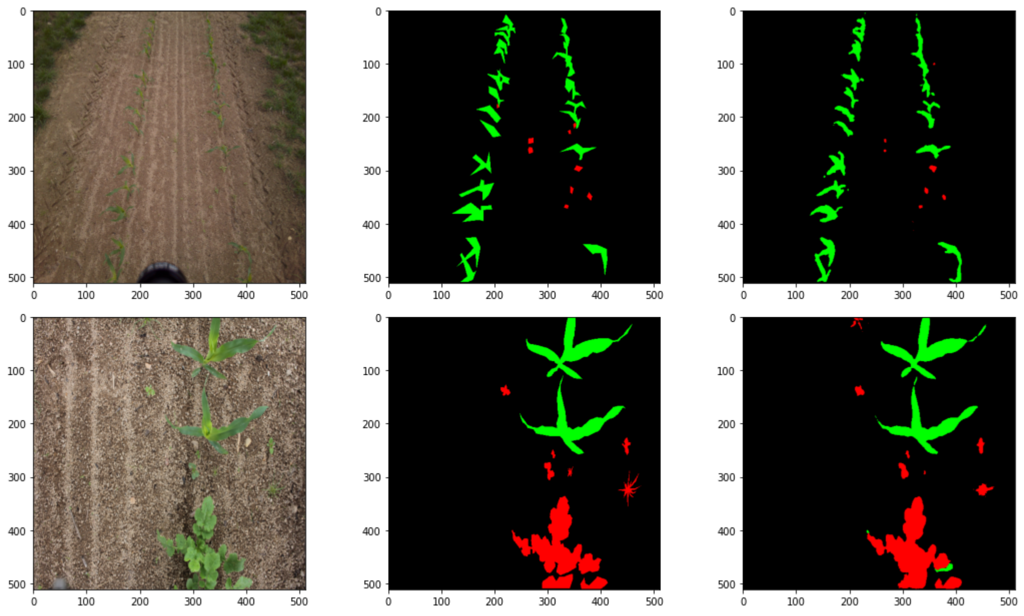

Figure 2: Examples of predictions: center images are the target masks, on the right the predicted masks