28462000 - Bourez Nicolas
68721900 - Sonnet Thibaut
55431800 - Coppée Thomas

LELEC2885
12th of December 2023

LELEC2885 - Project report - Group 2

1 Network architecture

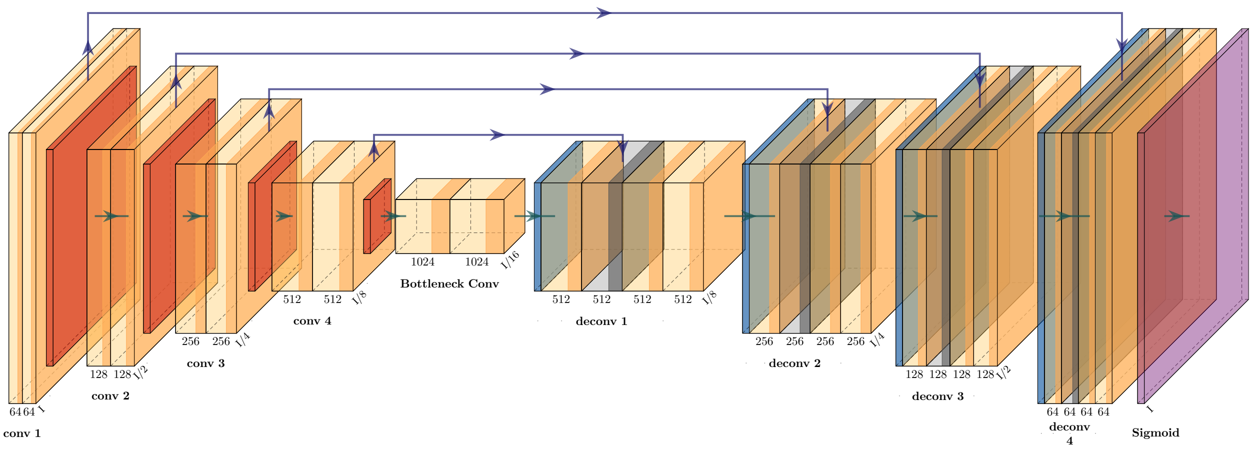


Figure 1: Initial network architecture (UNet)

There are many different types of network architecture[1] used to perform image segmentation. On one hand, some are really easy to implement but produce approximate results, while on the other hand, some produce very accurate segmentation but are build on a hard and complex implementable architecture. We choose a compromise between these constraints using the UNet architecture. Our first UNet on Figure5 has 4 blocks of convolution layers given ... parameters. But, because of the limited amount of available data ($\sim 7.5k$), having lots of parameters is contra-productive, so we decided to reduce it, keeping only 3 blocks :

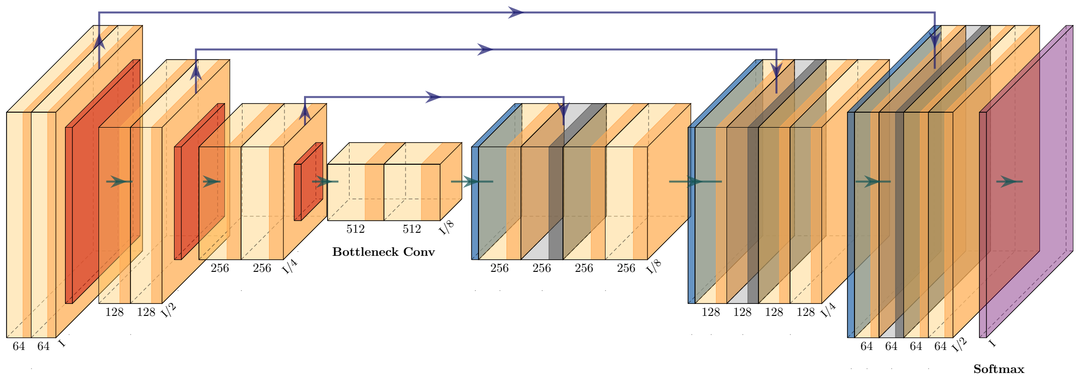


Figure 2: Smaller network architecture (UNet small)

As the follwing graphs 5 show it, with 64×64 images, the losses have almost the same curve. The smallest UNet is capable of computing before reaching overfitting.

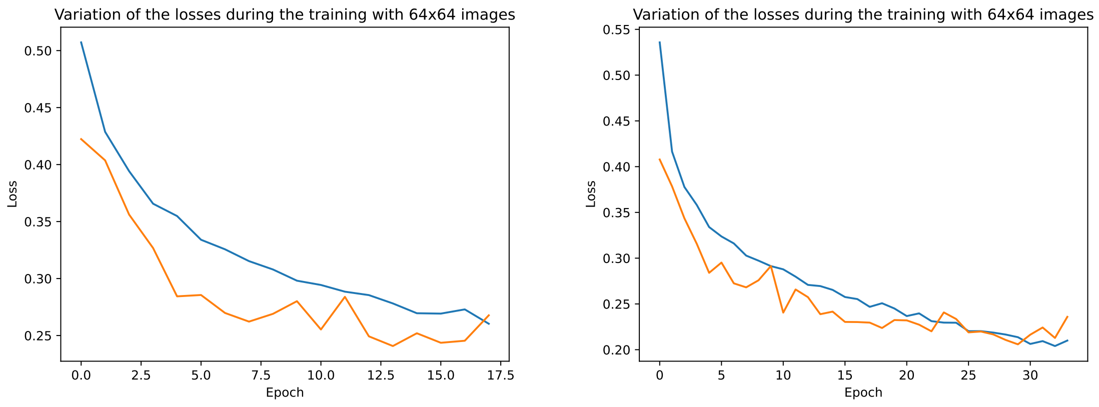


Figure 3: UNet vs UNet small

We can see on the table below that accuracy and average IoU of the small UNet clearly outperform the big one.

	Accuracy			Average IoU		
	Training test	Validation Set	Test set	Training test	Validation Set	Test set
UNet	0.8814	0.8910	0.8907	0.7340	0.7572	0.7558
UNet small	0.9151	0.9092	0.9107	0.8118	0.8041	0.8059

We can see that both algorithm takes almost the same computation time before reaching overfitting. We can see that the smallest UNet is capable of computing an epoch faster, obviously because it has less weights to compute

	UNet	UNet small
Computation time	12 min 0s	13 min 29s
Average time per epoch	38s	23s

Table 1: Training time comparison

We can also make a qualitative comparison between both structure that confirm our analysis :

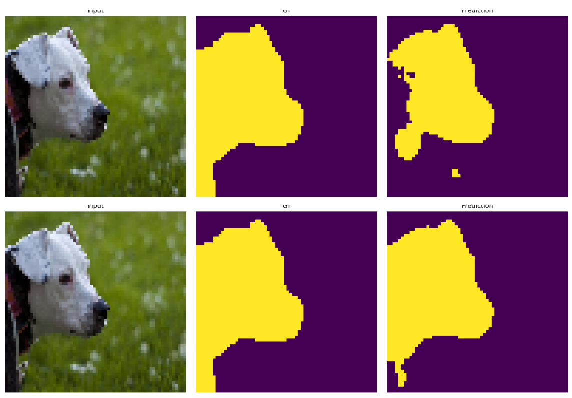


Figure 4: UNet(top) vs UNet small(bottom)

Trough this schema below, you can clearly see the dimension of the intermediate layer, and the transitions between them :

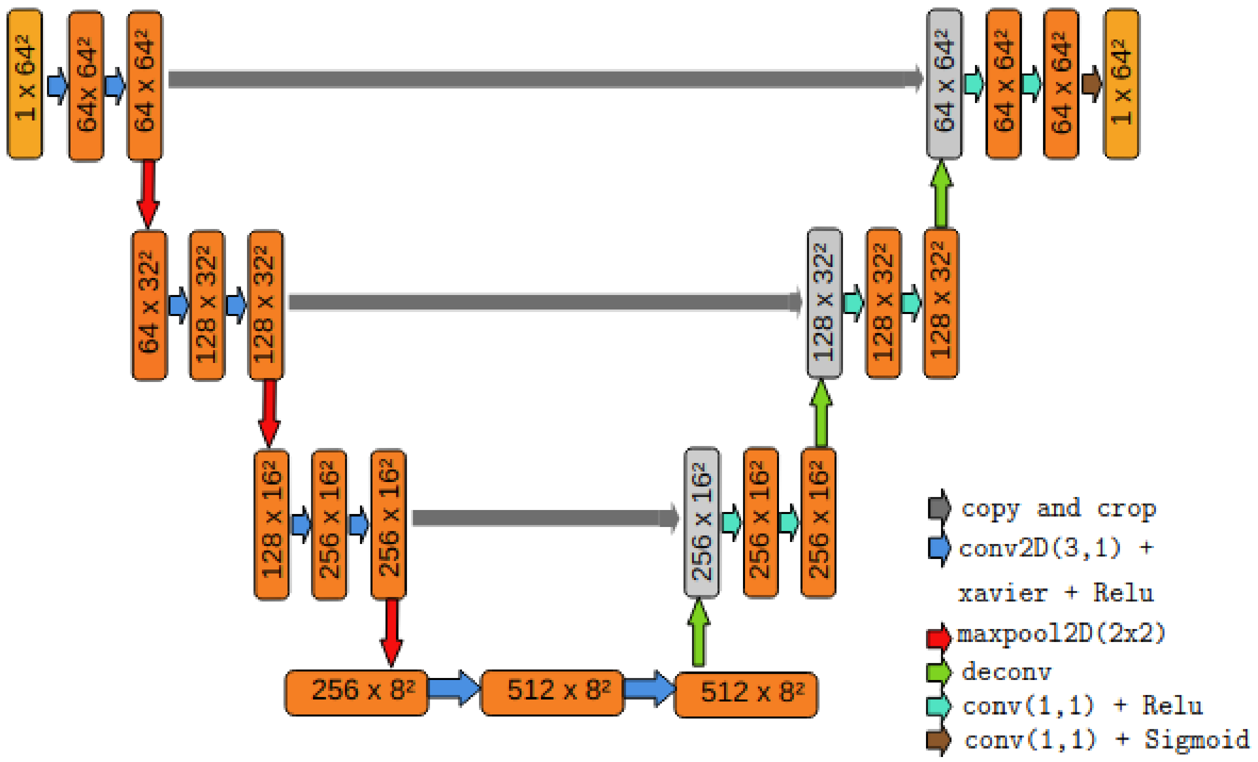


Figure 5: Network representation with layers dimension

2 Network training

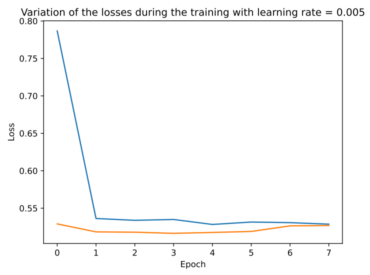
2.1 Loss-function

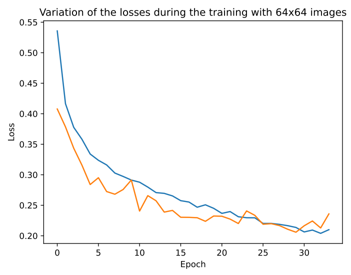
To train the model we have used the the Binary Cross-Entropy (BCE) loss function. The formula of the BCE loss can be found in equation 1 where y_i is the true label and $\hat{y}_i$ is the predicted probability. This formula represents the average negative log likelihood of the true labels y given the predicted probabilities across all samples in the dataset (N).

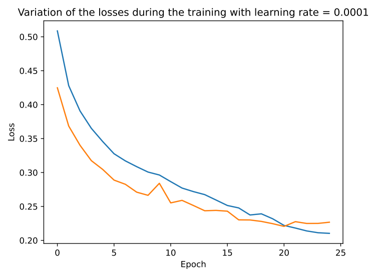
$$\text{BCE}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)] \tag{1}$$

2.2 Optimizer parameters

The training parameters are the learning rate in the Adam optimizer that we have set to 0.001 and the batch size that we have set to 16. As we can see on the following graphs, we could choose a learning rate of 0.0001 that gives equivalent results.

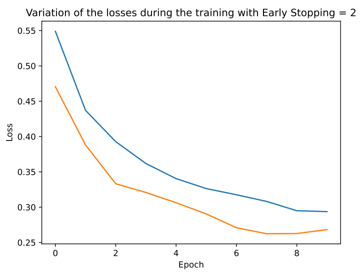


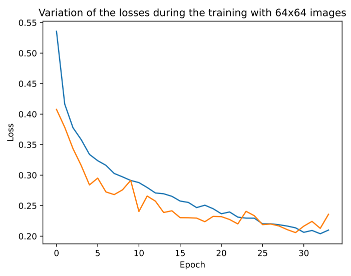


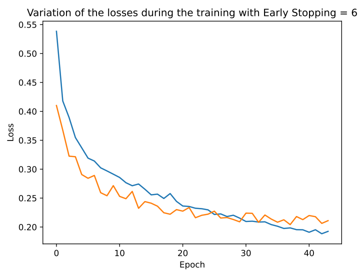


2.3 Preventing over-fitting

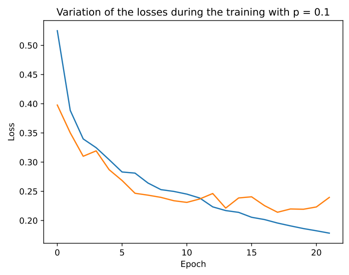
To mitigate the over-fitting we have implemented an early stop algorithm (inspired from [2]). It makes the training loop keep in memory the best neural network according to the loss on the validation set. Note that we can tune the *sensibility* of the early stop (param `patience` in the early stop function). We choose the *sensibilty* which gives uses the best results, i.e **??** :

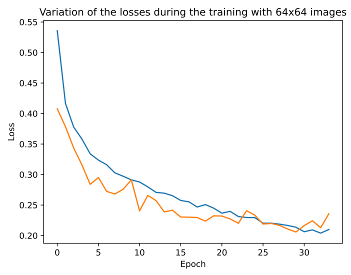


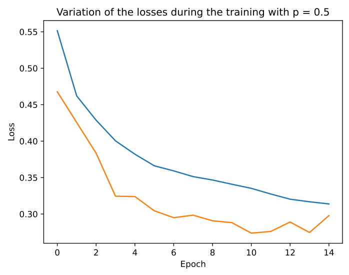




We also performed data augmentation, by applying transformations to our dataset with the library `albumentations`. We chose a probability we had the best results with, i.e. 0.2 :







On the following pictures we can see the evolution of the loss function for the validation and training set.

3 Evaluation

In this section, we make a quantitative and qualitative analysis of the results of our network. We take stock of the successes and failures and then quantify the results using various metrics. Images of sizes 64x64 and 256x256 are analysed.

3.1 Qualitative

On the figure **??**, we can see some smaples images from the test set.

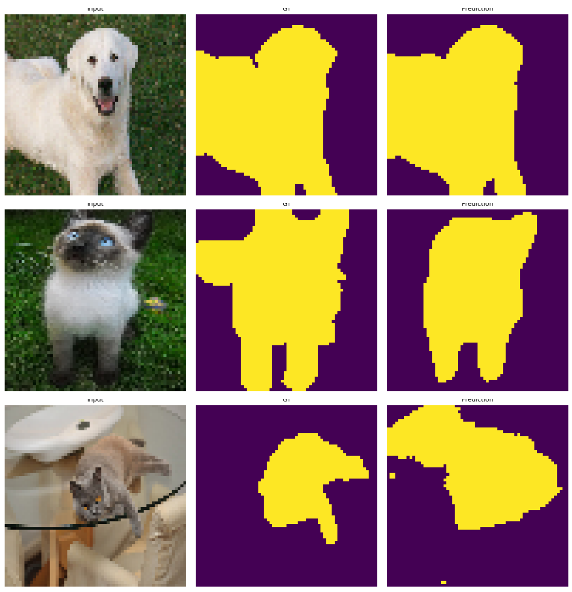


Figure 6: Results for 64x64 images

3.2 Quantitative

	Accuracy			Average IoU		
	Training test	Validation Set	Test set	Training test	Validation Set	Test set
UNet small	0.9151	0.9092	0.9107	0.8118	0.8041	0.8059

3.3 Transfer learning

Accuracy of the network on the test images: 63 Accuracy on cat 0.8825910931174089 and dog 0.13424657534246576

References

[1] Shervin Minaee et al. “Image Segmentation Using Deep Learning: A Survey”. In: *CoRR* abs/2001.05566 (2020). arXiv: `2001.05566`. URL: `https://arxiv.org/abs/2001.05566`.

[2] `https://github.com/Bjarten/early-stopping-pytorch/blob/master/MNIST_Early_Stopping_example.ipynb`. Last visited on 1/11/2023. 2023.