# Project #3 : Encoder-Decoder implementation

In this project we are going to implement encoder-decoder architecture to practice segmentation on a dataset of images. The goal of this segmentation is to make a classification. The dataset is the development kit of Pascal VOC 2012 which contains 20 classes (+2 convenient classes) in 11530 images. First, we are going to make that segmentation using the original U-Net architecture. Secondly, we are going to modify U-Net’s architecture by replacing its encoder with a ResNet 50.

**Original U-Net**

In this first part I had to fill in a skeleton code of U-Net’s implementation in PyTorch. Using the instruction sheet, we can fill in the channel size for each convolution. Note that in the decoder’s convolution I had to consider the concatenations that occur in U-Net. For instance, for the first convolution layer of the decoder, the input is the tensor that is obtained by concatenating the tensors that have 1024 and 512 for the channel size. Therefore, the convolution is going to be operated on a tensor that has a channel size of 1536. Then, I had to implement the forward pass. The missing parts where the concatenations. To do that I used the PyTorch function **torch.cat((A, B), 1)** which concatenates 2 tensors A and B along the dimension of the channel (which is 1).

A black screen with white text

Description automatically generated Then in order to train the model I completed the modules\_skeleton.py script. I added the Torch instructions to operate the forward and backward pass. This script contains also a function to compute the loss over the training set. In this part we do a forward pass, but we don’t include the instructions for the backward pass since we just want to compute the loss without optimizing. Finally, to the implement the segmentation I added some instructions that gives a color to a pixel in function of the class. This is done for both labels and predictions.

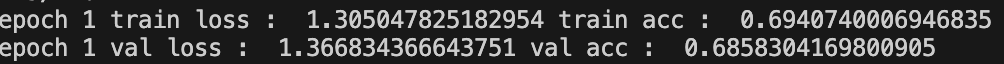
We can now start the training. I loaded a pre-trained model (a checkpoint) so I can only train the model for one more epoch. It took approximately 3 hours to get the following results using an Apple M2 CPU.



We can observe that the accuracies for the training and validation datasets are pretty good. We also notice a difference between the training and validation loss which proves that the model is not overfitted.

**U-Net with ResNet 50 encoder**

Now, we are going to implement an encoder-decoder architecture based on U-Net where the encoder is the ResNet 50 we previously implemented. As in the previous project, we first implement the residual block. Then, we implement the forward pass by adding the concatenations showed in the instruction sheet. Using the figure of the architecture in the instruction sheet I added the residual blocks with the corresponding input and output channel sizes. We can finally run the training.



We notice that the accuracies are pretty low compared to the original U-Net but it is acceptable. Moreover we observe that the difference between the training and validation loss is small, which can makes us wonder if the model is not overfitting.