Project: Semantic Edge Detection.

1. Choose a video from the DAVIS dataset, get frames, obtain the edges of the frames from the annotations of these frames.
2. Build an Encoder-Decoder architecture to train a neural network for semantic edge detection ***on a single image***. Take as encoder one of the *ResNet* (18, 50 or 101) architectures and initialize it with a pre-trained model.
3. Define a *loss function,* **overfit** the model with on the chosen video's frames.
4. As can be noticed, in the *ResNet* architecture there is a $MaxPool(x)$ layer. Define a function $F_W(x)$ as a mini-network (with weights W), such that for some W^* the function $F_{W^*}(x)$ is identical with the function $MaxPool(x)$. **The mini-network $F_W(x)$ must be composed of only convolutional layers and pointwise non-linearities**. Replace the $MaxPool(x)$ function with the $F_W(x)$ and initialize it with W^* .
5. Build an Encoder-Decoder architecture to train a neural network which will take into account the result of itself on the previous frame of the chosen video. More precisely, let $F_1, \ldots, F_T \in R^{BatchSize \times Height \times Width \times 3}$ be the frames of the chosen video, $NN()$ be a neural network and $O_0, O_1, \ldots, O_T$ are defined by the following way: O_0 is a tensor with shape $(BatchSize,\ Height,\ Width,\ 1)$ filled with zeros, $O_t = NN(F_t \| O_{t-1})$ for every $t = 1, 2, \ldots, T$, **where by $\|$ we denote the *concatenation operation of tensors along the last axis***.
6. Take as encoder of the above mentioned neural network NN the same architecture as in step 2. **(except for the first convolution layer, which will accept 4-channeled input instead of 3-channeled).** Initialize the encoder with the pre-trained resnet weights **except for the kernel of the first convolution.** For the kernel of the first convolution make the following initialization: the kernel is a tensor with the shape $(H, W, 4, OutChannels)$, initialize it's slice $[:, :, :3, :]$ with the pretrained resnet's first convolution kernel, and the slice $[:, :, 3, :]$ randomly.
7. Train the $NN()$ network on the $k - length$ **frame sequences** from the chosen video (try to overfit on the chosen video).
8. Add ***perceptual losses*** to the training.
9. Add *non-maximum suppression* after the output of the network. **The *non-maximum suppression* must be a part of the main graph and not just a post-processing function (to be able to put loss after the non-maximum suppression operation and train the network with this op.).**