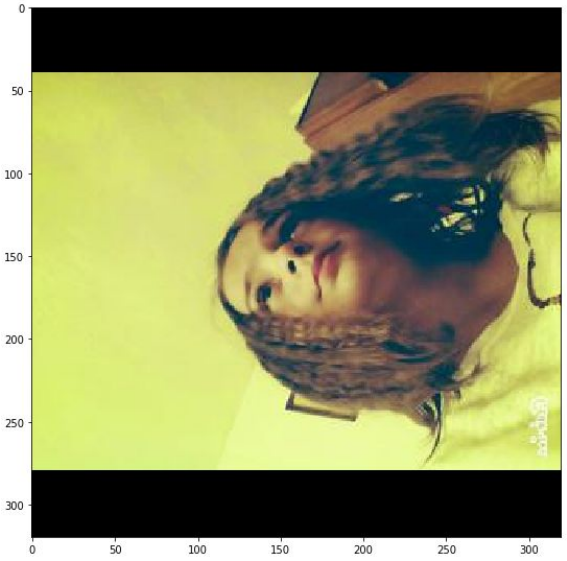
Technical report

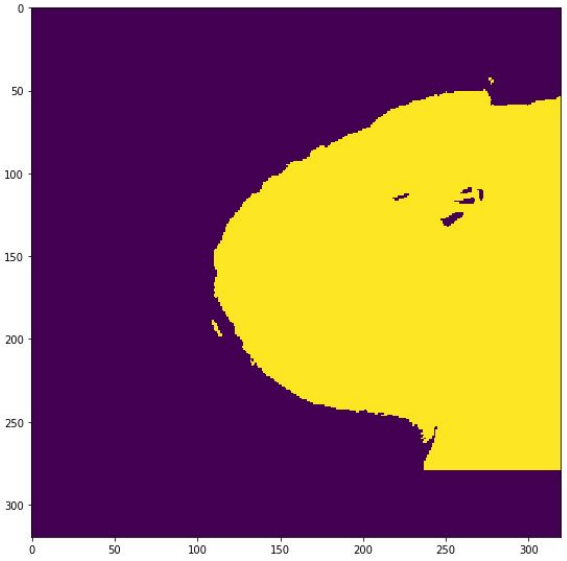
Task 1.

Problem description

Here the problem of multi-class face segmentation is introduced. We're given a dataset with around of 1500 images of people. The main task is to write a model that can separate a person from other thing on image.

For example:





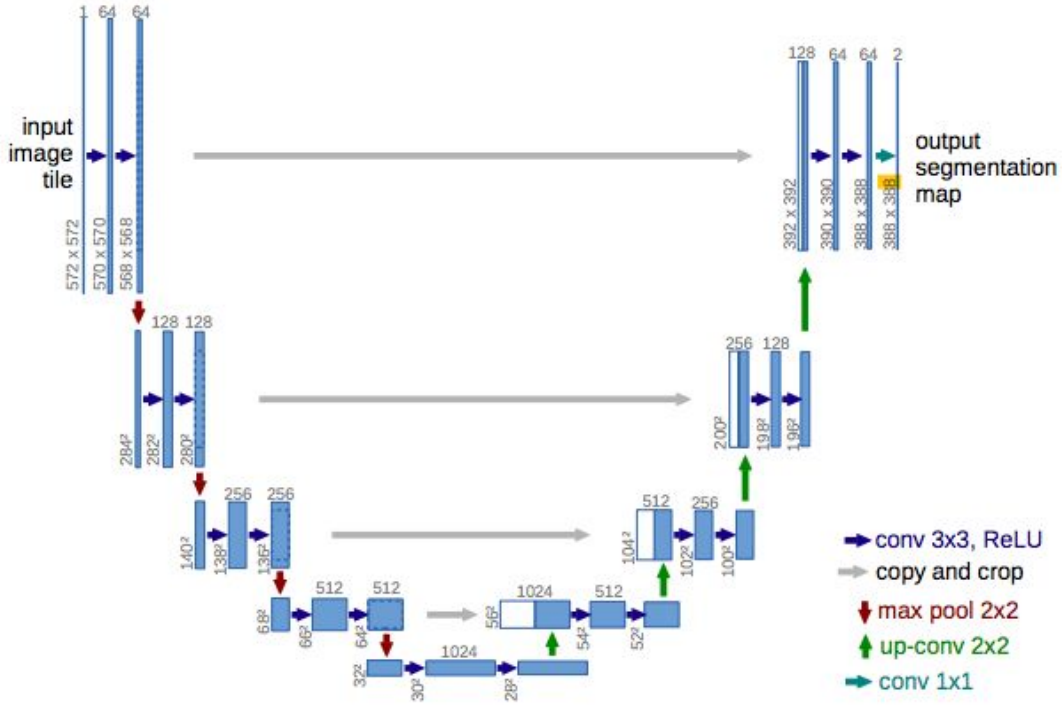
Challenges

Our model can encounter with some difficulties, because it has to learn how to find all endges and make a right conclusion. We should do some manipulations with the image like convolution and pooling to classify pixels. And also to get back to our initial resolution we have to do some unsampling techniques like interpolation, transposed convolution etc.

Model description

In our case we will use the U-net architecture. It is synonymous with an encoder-decoder architecture. Essentially, it is a deep-learning framework based on FCNs; it comprises two parts:

1. A contracting pasth similar to an encoder, to capture context via a compact feature map.
2. A symmetric expanding path similar to a decoder, which allows precise localization. This step is done to retain boundary information despite down sampling and max-pooling performed in the encoder stage.



Advantages of using U-Net:

1. Computationally efficient
2. Trainable with a small data-set
3. Trained end-to-end

As U-Net model I used ResNet34, because it is quite lightweight and in most of cases it is the baseline.

Training/validation strategy

We can read the original paper of UNet here - https://arxiv.org/pdf/1505.04597.pdf. However, I did one change to get better result. As seen below, the original paper used stochastic gradient descent optimizer, I just used an Adam Optimizer and Binary Cross Entropy with logits as our loss function.

3 Training

The input images and their corresponding segmentation maps are used to train the network with the stochastic gradient descent implementation of Caffe [6]. Due to the unpadded convolutions, the output image is smaller than the input

Also I used Dice and Jaccard loss which will give us a lot better understanding of how accurate our model is.

The main cause of my providing inaccuracy in this model can be a small number of training epochs. Due to the lack of time, I did only 30 epochs for each training.

Experiment log

I also did the experiments with Unet11 and Unet16 for this problem, but they takes more time for training and show worse results.

UNet16

```
train_loss: 0.540, train_dice: 0.690, train_jaccard: 0.557, val_loss: 0.446, val_dice: 0.736, val_jaccard: 0.610
```

UNet11

```
36868286
train_loss: 0.535, train_dice: 0.715, train_jaccard: 0.575, val_loss: 0.533, val_dice: 0.729, val_jaccard: 0.591
```

Also, I did some experiments with learning rate and the best number of learning rate is 0.001. When the number is bigger - our result is worse and vice versa the time of epoch is longer but the result is not better.

As well, I tried to do some experiments with other loss functions, decoder blocks etc, and it didn't give better results.

Conclusion

So, as we can see that the best choice is ResNet34 that gave us that result:

```
train_loss: 0.191, train_dice: 0.903, train_jaccard: 0.835, val_loss: 0.180, val_dice: 0.906, val_jaccard: 0.837
```

By the lack of time, it is the best result that we can achieve. Surely that for more epoches the result will be better and maybe other models can give us more accurate result, but then will be one big disadvantage - it is time.