Document Layout Segmentation Results Analysis

Chenrui Fan

March 16, 2024

1 Using UNet Only

Here the input is the whole image[960, 640], without resizing.

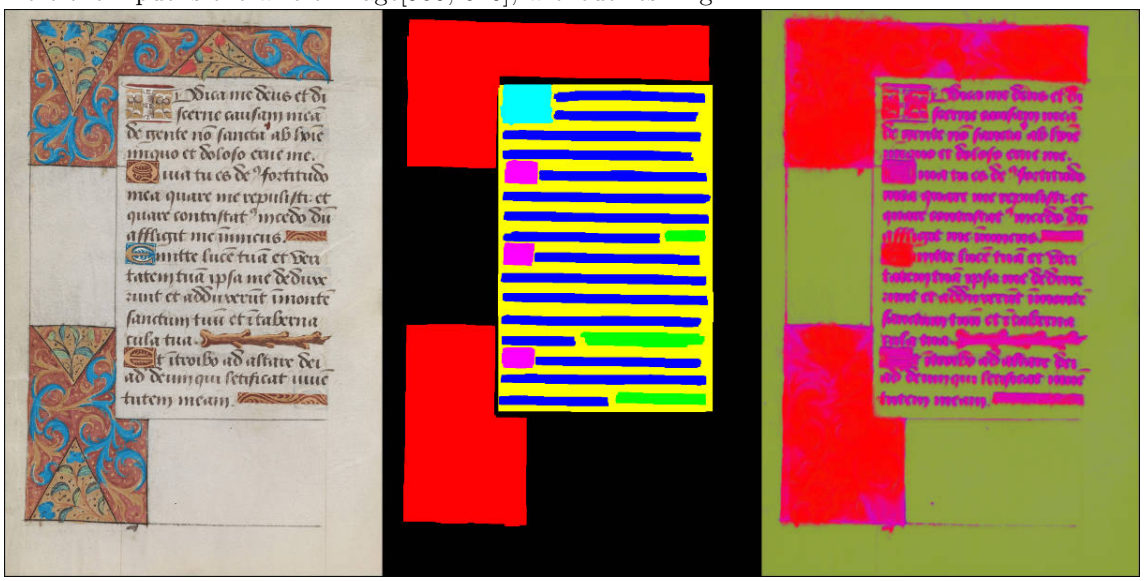


Figure 1: The effect at the very beginning of the process.

The trickiest part of this split is that the color of the text box is the same as the background color. If the text box is not processed, the result will be very poor, as shown in the figure:

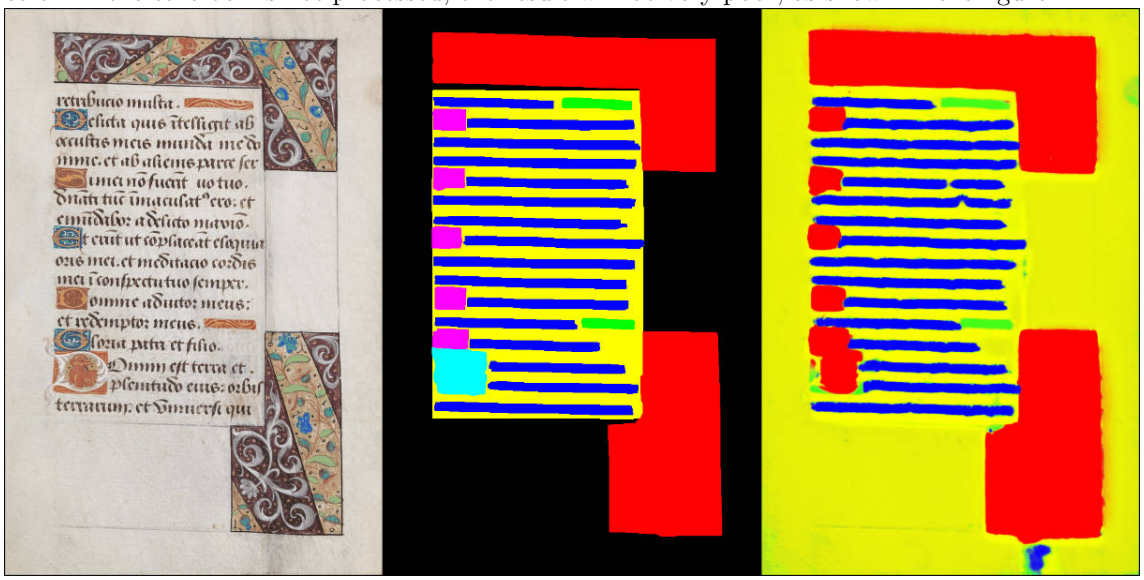
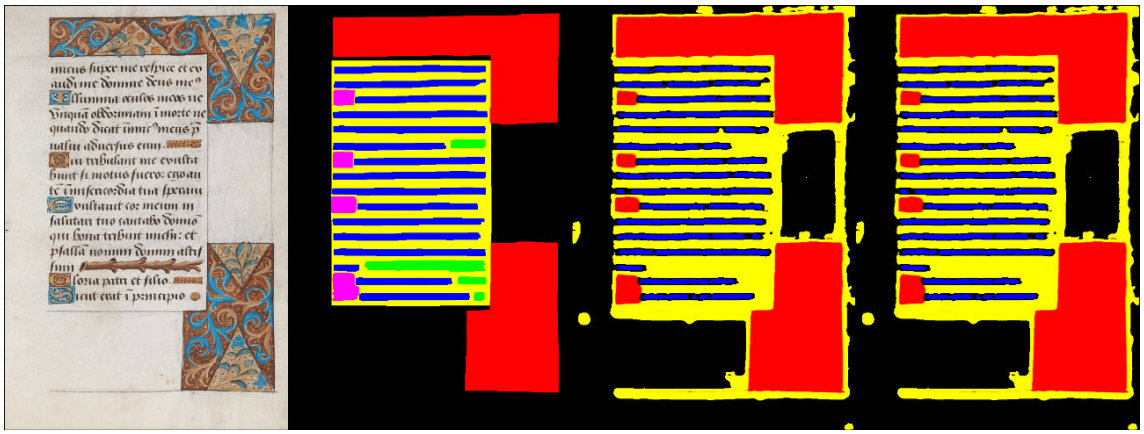


Figure 2: No background processing.

2 Add background proccessing

So I tried to manipulate the background. It didn't work very well, there was a lot of noise.

Figure 3: Design a background color convolution kernel.



There are also cases like this one, which are very hard to solve using only convolutional kernels.

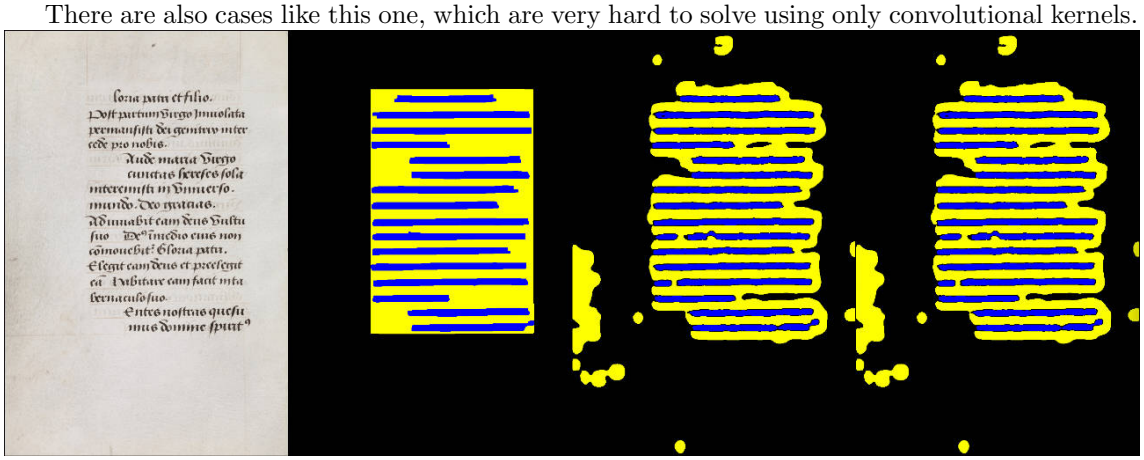


Figure 4: Difficult case.

I started visualizing the convolution process out. The image before the truth value is what I get by setting pixels below a certain threshold to zero.

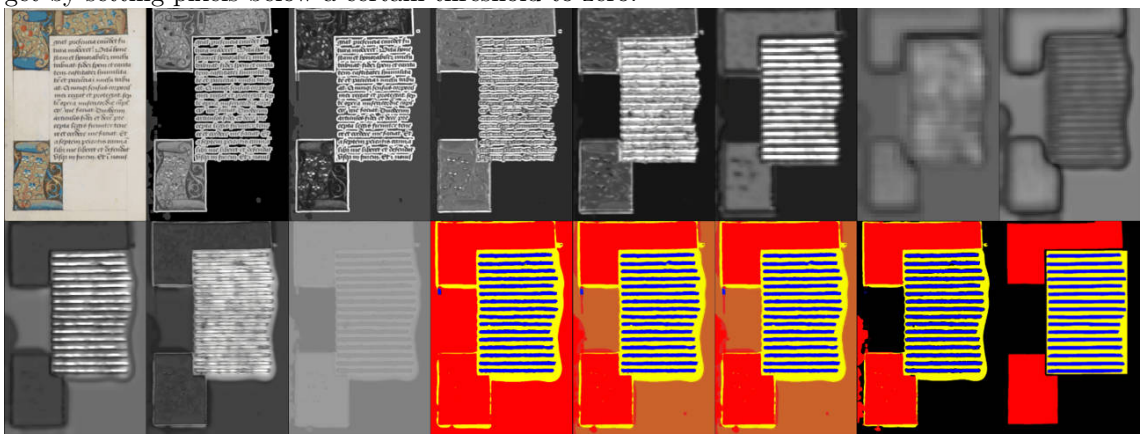


Figure 5: Difficult case.

You can see that a big problem is that during downsampling, the color "leaks" out with the depth of the network.

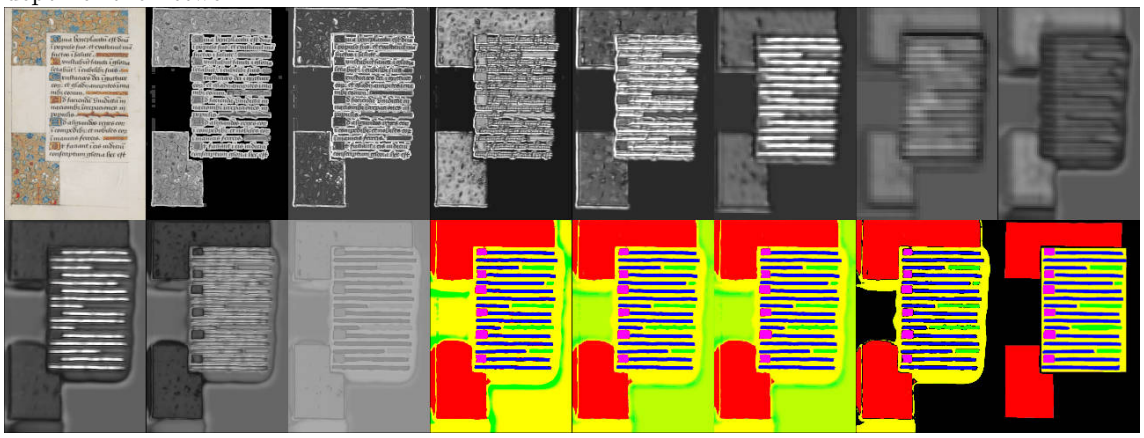


Figure 6: Leaking problem.

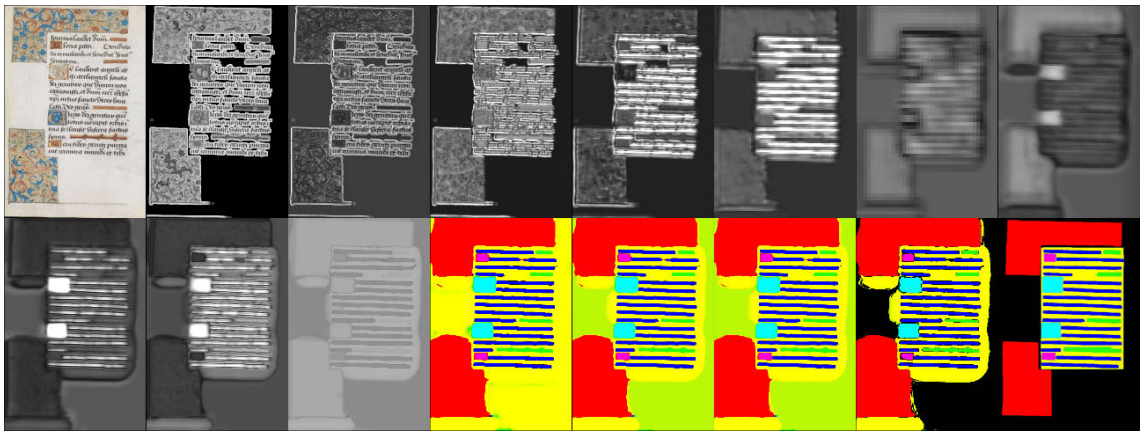


Figure 7: Leaking problem.

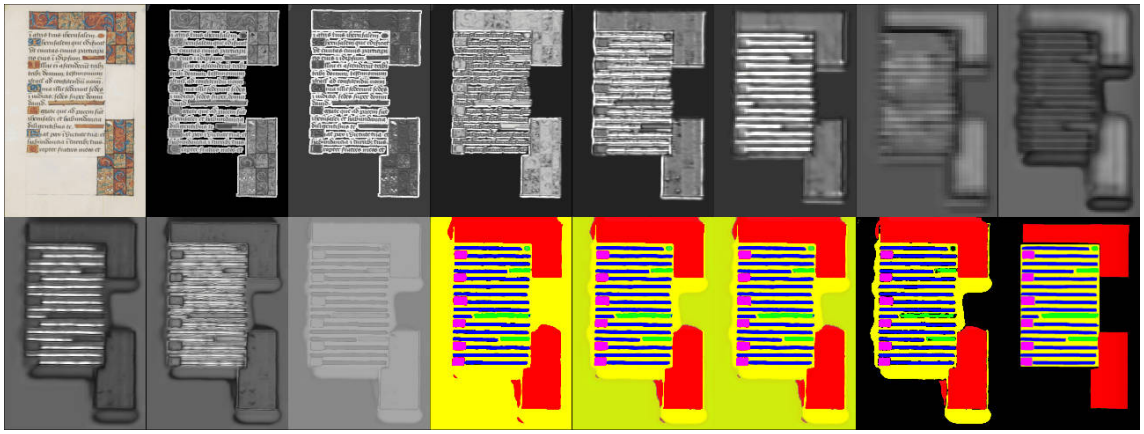


Figure 8: Leaking problem.

Here's another problem: I didn't handle the background cleanly, causing a ring of yellow around the red color when upsampling. This of course has to do with the difficulty in delineating the text box in the original image.

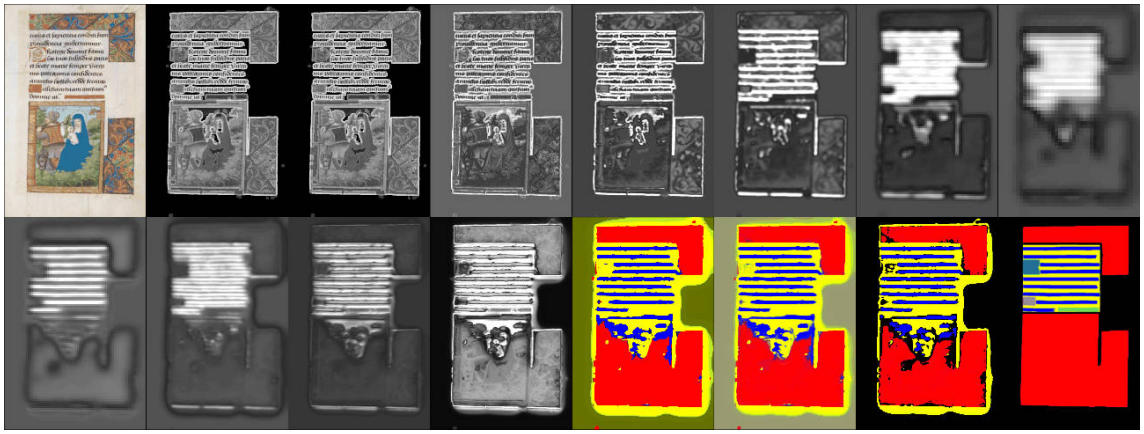


Figure 9: Yellow around problem.

Or maybe it's misclassifying red when there's no illustration available.

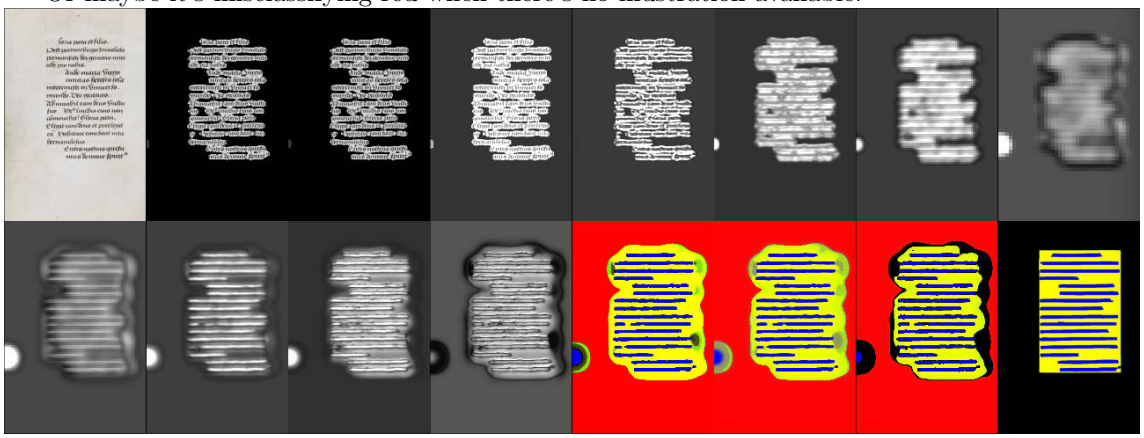


Figure 10: Misclassifying red.

3 Residual connection

I don't want the image to gradually blur out during the downsampling process, allowing the deeper network to learn features that aren't very useful. And the residual join idea in ResNet is just right, so I added this module to UNet.
I'm connecting between layers, meaning that each connection requires downscaling or upscaling, interpolation, deconvolution or pooling of features from the previous layer.

```
class DownResidual(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(DownResidual, self).__init__()
        self.conv1x1 = nn.Conv2d(in_channels, out_channels, kernel_size=1)
        self.leaky_relu = nn.LeakyReLU()
    def forward(self, x, x_ori):
        x_ori = self.conv1x1(x_ori)
        x_ori = F.avg_pool2d(x_ori, 2)
        x = x + x_ori
        x = self.leaky_relu(x)
        return x
```

Figure 11: DownResidual red.

```
class UpResidual(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(UpResidual, self).__init__()
        self.up_conv = nn.ConvTranspose2d(in_channels, out_channels, kernel_size=2, stride=2)
        self.leaky_relu = nn.LeakyReLU()
    def forward(self, x, x_ori):
        x_ori = self.up_conv(x_ori)
        x = x + x_ori
        x = self.leaky_relu(x)
        return x
```

Figure 12: UpResidual red.

The effect of using residual connections for both downsampling and upsampling is as follows:

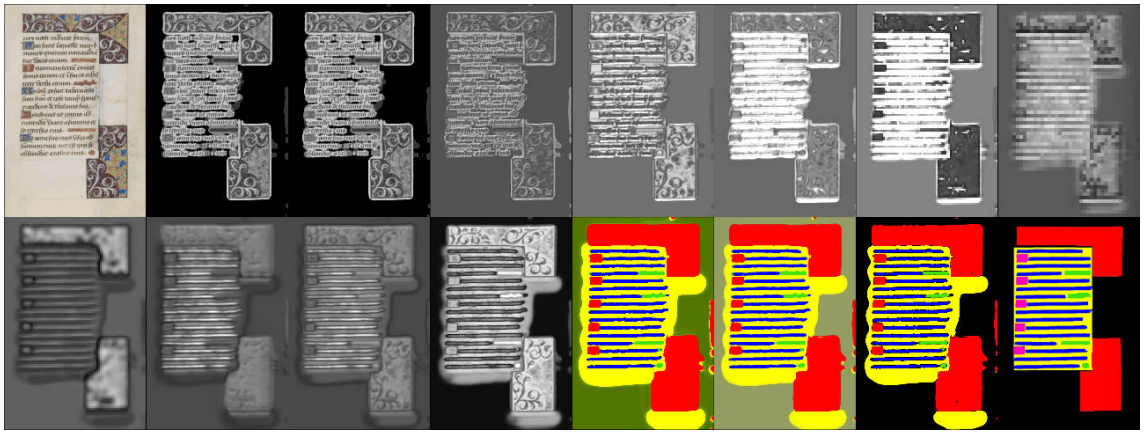


Figure 13: Both.

The effect of downsampling only using the residual connection is as follows:

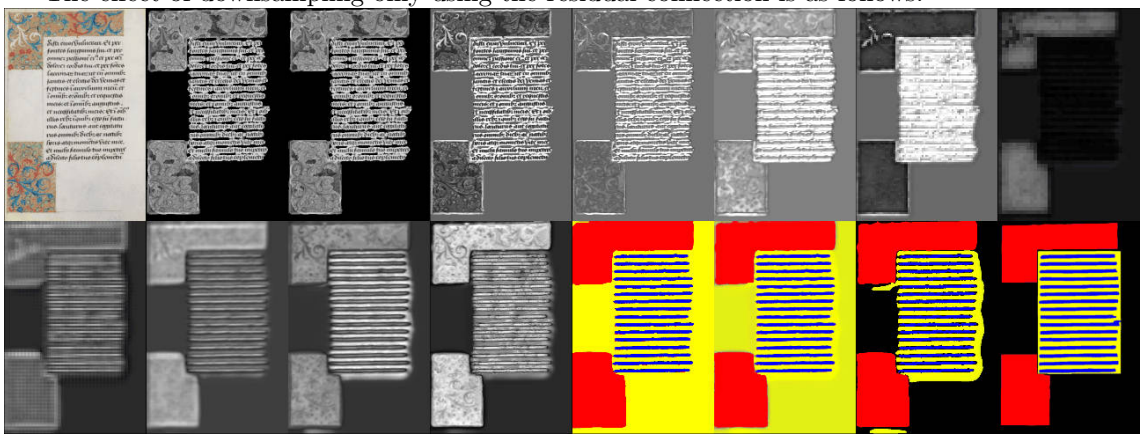


Figure 14: Only when downsampling.

You can see that it is better to use it only for downsampling. I think it's partly because the up-sampling is already concatenating with the previous feature map, so residual joining is a bit unnecessary, and partly because deeper feature maps may bring more fuzzy boundary information, which in turn leads to less good results.

A result that emerges from the training process that is nicely segmented except for the text boxes

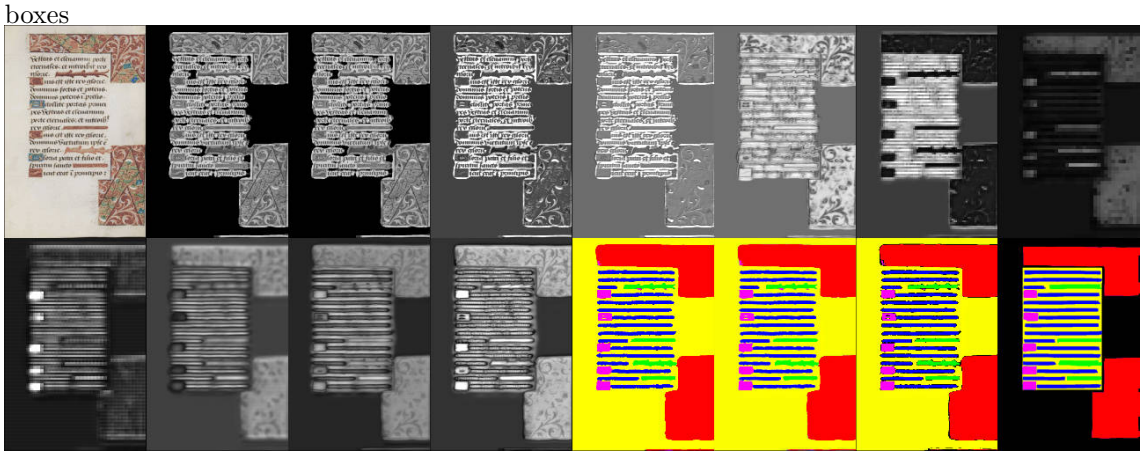


Figure 15: Nicely segmented except for the text boxes red.

I was also wondering if it was the Adam optimizer, so I tried SGD as well. It turns out it doesn't matter much.

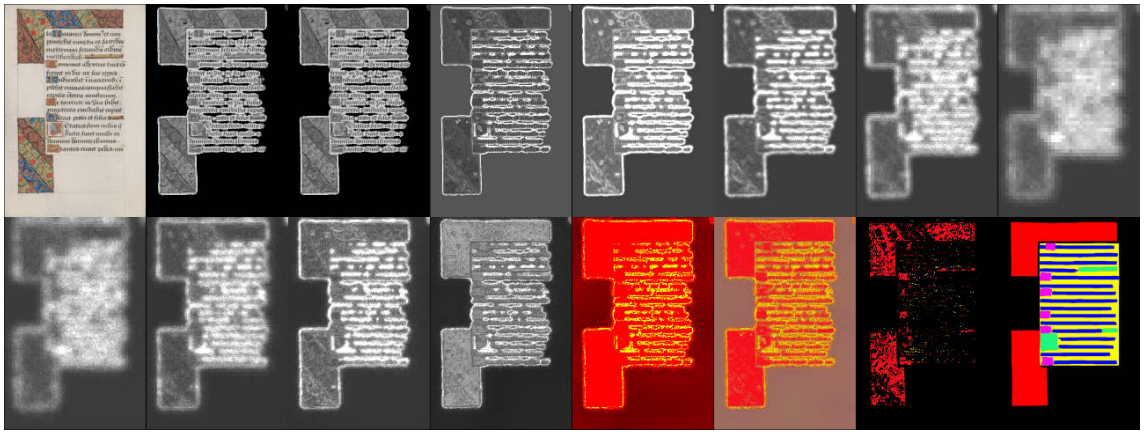


Figure 16: SGD.

It also didn't work well with large stains.

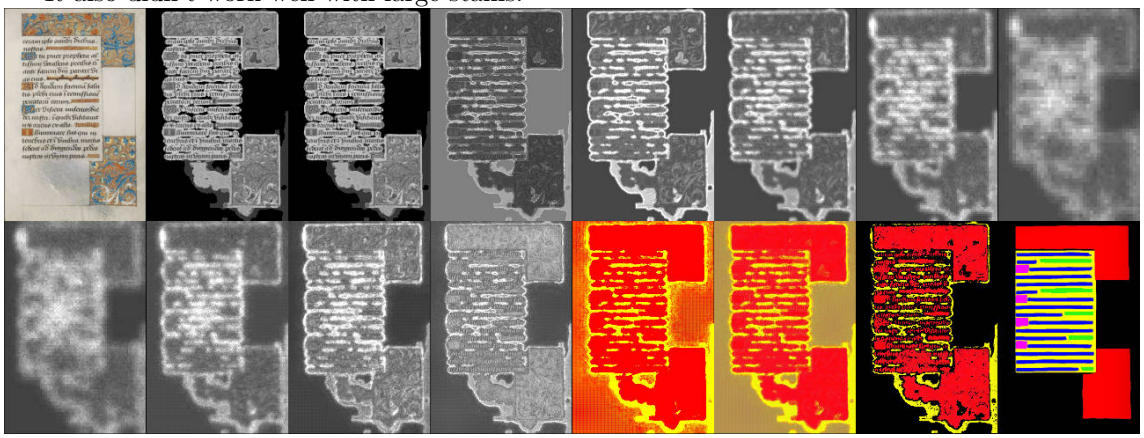


Figure 17: Large stains.

4 Using smaller kernel or simple threshold

I've tried again to continue to optimize the background processing part at the very beginning, hoping to eliminate the problem of a ring of white border around the illustration. I used a smaller convolution kernel, and a method that directly sets pixels larger than [170, 170, 170] to black. They do remove the background nicely, but don't help much with the delineation of the text box.

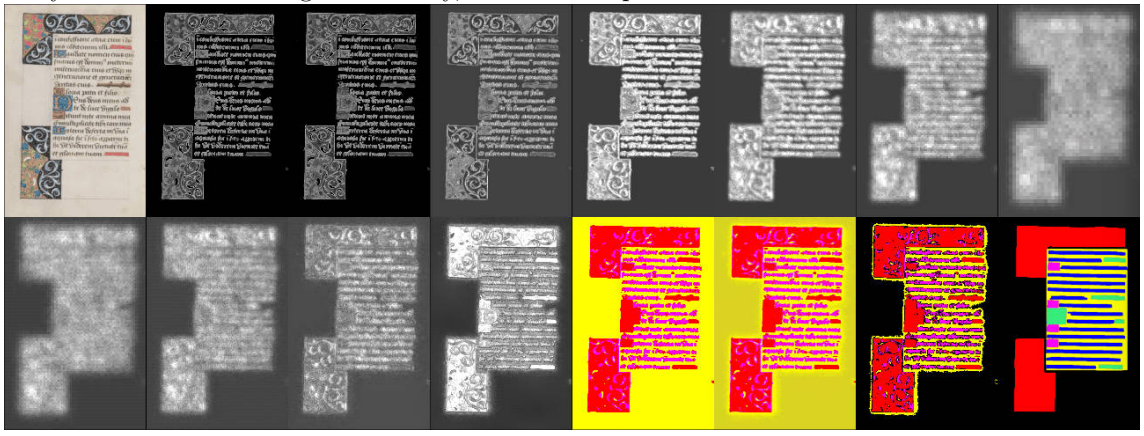


Figure 18: Smaller kernel.

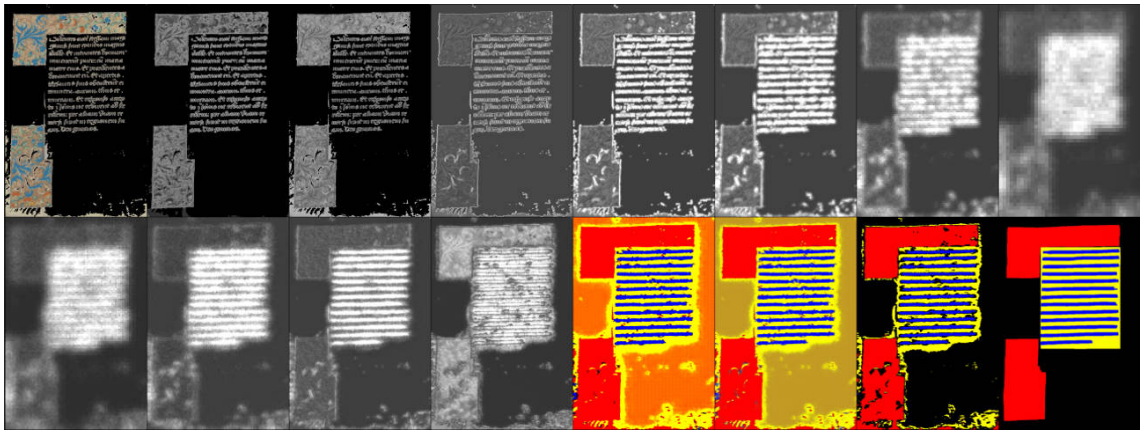


Figure 19: Simple threshold.

5 Deeper UNet

I remembered that when I did the fundus vessel segmentation task before, the input image size was 224x224, meaning that the feature map size after four downsamplings was [1024, 14, 14]. However, I had a feature map size of [1024, 60, 40] after four downsamplings because I didn't resize the original image. So I added more layers.

First I added a layer, but kept the dimension at 1024 and didn't continue to double it to 2048.

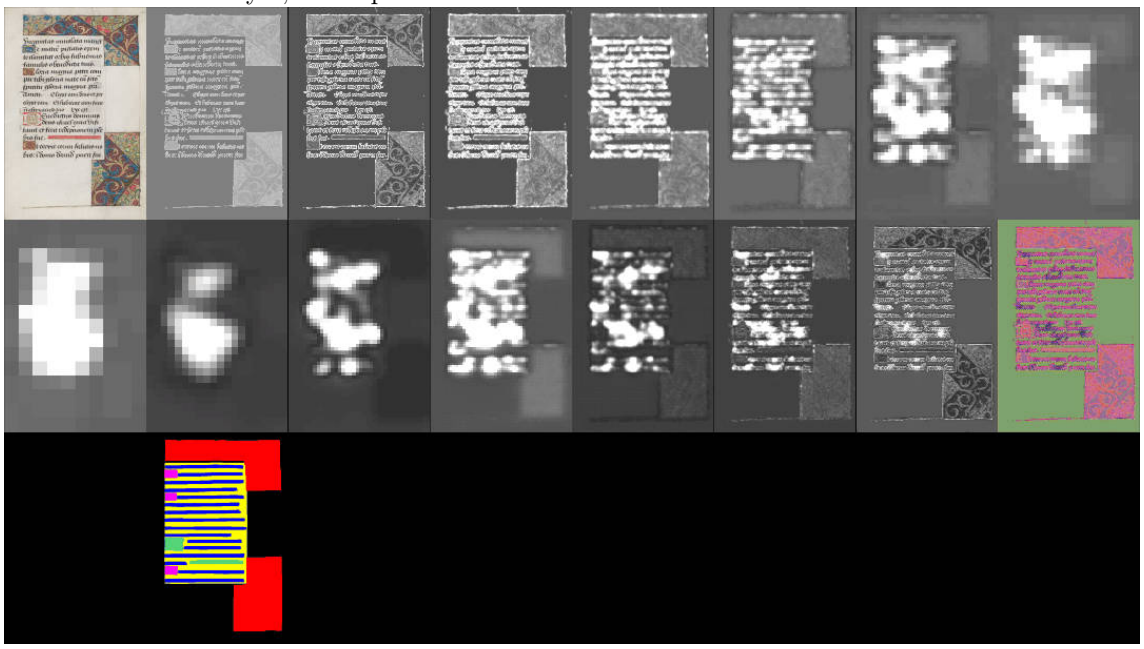


Figure 20: 5 layers.

Then I added another layer so that the size of the bottom feature map is 15x10, which is about the same as 14x14. But that was too big a parameter count, so I dropped the number of channels in all the layers to half of the original.

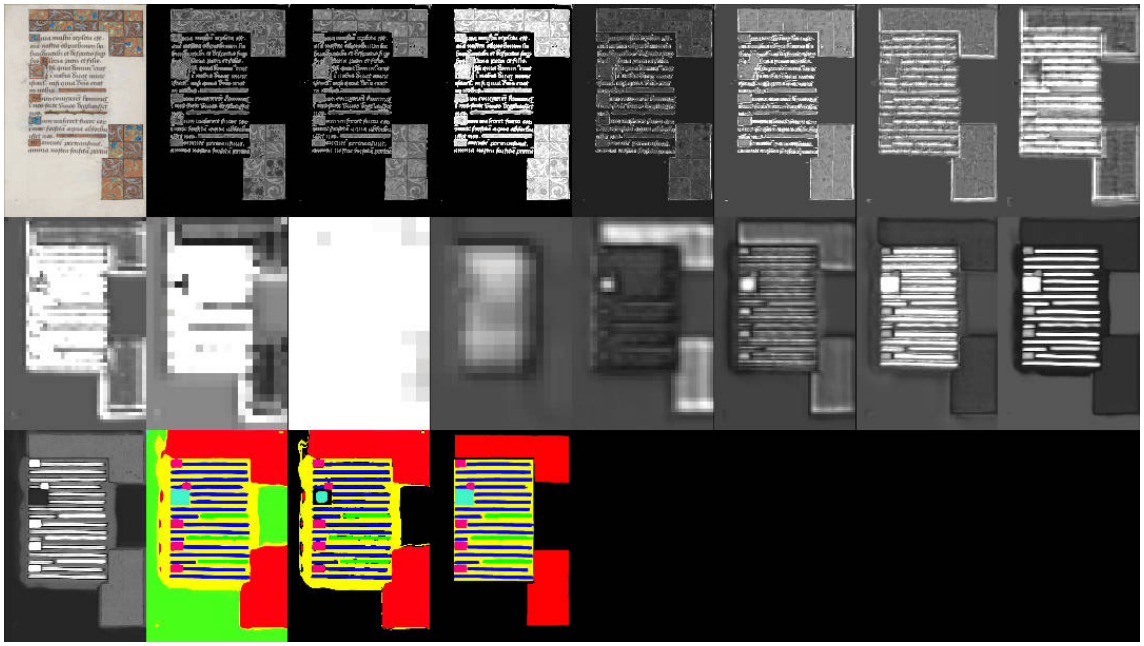


Figure 21: 6 layers.

On top of that I tried to optimize the processing of the background. I designed a kernel that determines that at least 0.9 of the pixels within the window have RGB three channels greater than 170 at the same time, and I hope that the text box can thus be separated from the background, as shown in the effect:

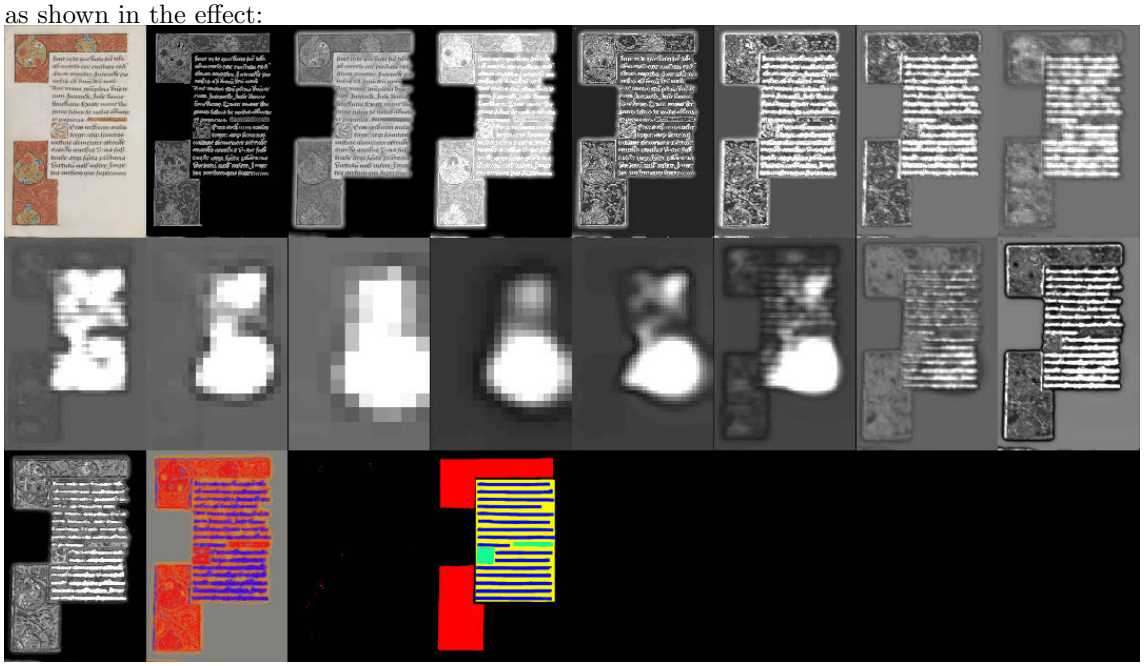


Figure 22: Back all reduced.

6 Without yellow

I let the model predict the true value without the text box, and it actually manages to perform pretty well.

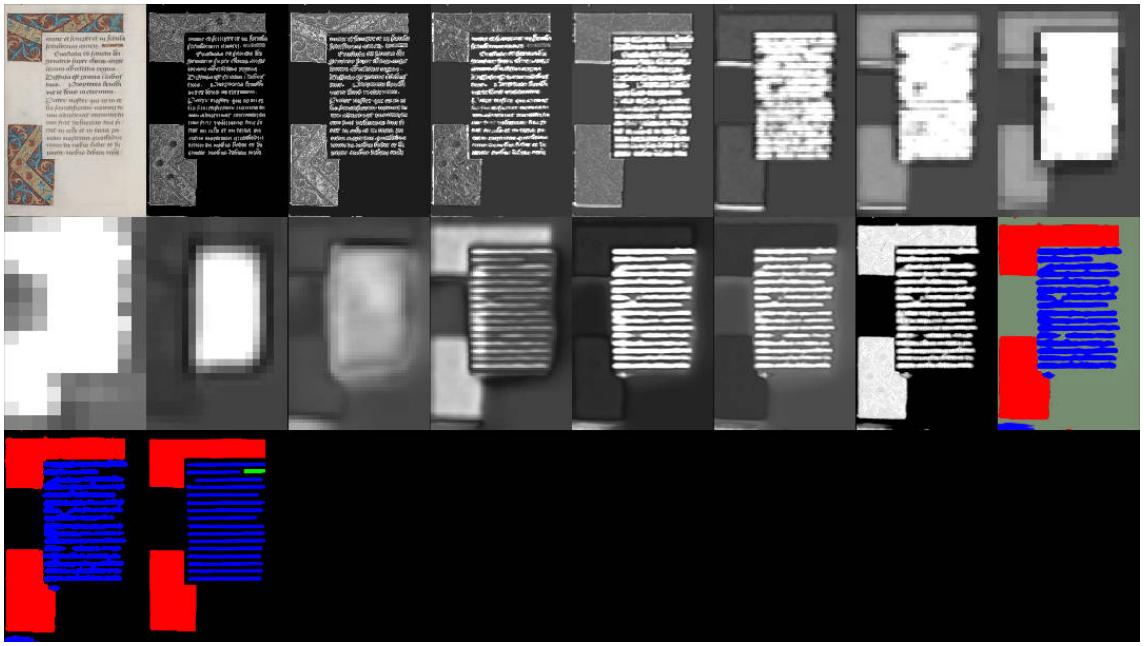


Figure 23: Without yellow.