# ICDAR2019 RDCL competition

by ZANG Chendi, LI Hui, CHANG Xinfeng, WU Yaqiang from Lenovo Research

## method description

We use FCN(fully convolutional networks) to generate my results. With 15 original training images, we use data augmentation method, including cropping, gaussian blur, adding random noise as well as color jittering to generate 3960 images as my training set.

I use resnet18 to extract features(considering the efficiency), except that I added another resnet block at the end of resnet18, which enlarges the receptive field from original 224 to 448, since the cropped image is still large and 224 pixels can't cover large regions. We also tried to add SE Layer to resnet, but the result does'nt work well. This image feature extractor, or encoder, reduces the size of image by 64 times, and the final number of channel is 512.

For the decoder part, we used a modified FPN network[1]. Normally we have nine classes, namely background, text, image, graphic, table, chart, map, linedrawing and separator. But regions like tables are always full of nested text, which makes the true table region quite small, even unvisible in the label map. So we produce a final result of 10 channels, with the last two channels used to distinguish text and background, and other channels take charge of separating the other regions.

So the output of the network is actually two images with the same size of the original image, one image indicating whether each pixel is text or not, and the other image indicating whether each pixel belongs to the other 8 classes(no text class). These two images can produce really neat and promising results, but the number of samples of maps/charts/tables/graphic are so limited, and the classification are not very accurate. I believe that with more data added to the training set, the result with be far better than the one that I produce.

For post-process, we use opening operation to reduce small noises as well as splitting the paragraphs. Normally most paragrahs are well separated, but still some paragraphs may be connected by a small patch. We use an adaptive threshold for the opening operation to deal with that. But we still leave a lot of paragraphs connected since this kind of merger is of low priority.

We use connected components to determine the region area. For table and chart regions, we force these areas to be rectangles, and use these results to generate final xml.

[1]Feature Pyramid Networks for Object Detection by Tsung-Yi Lin et. al,   arXiv:1612.03144

# how to run the code

pytorch 1.0, python3

put the model file under `ckpt/` and run `bash test_RDCL.sh`

this code is based on the structure from `https://github.com/CSAILVision/semantic-segmentation-pytorch`, I do appreciate their great work which saves a lot of time for me.