Document Layout Analysis

Chenrui Fan

May 6, 2024

1 Introduction

Document Layout Analysis is performed to determine the physical structure of a document, identifying document components. These components may include single connected regions—regions of pixels adjacent to form single areas, or groups of text lines. A text line consists of characters, symbols, and words that are adjacent and relatively close to each other, and through which a straight line can usually be drawn, typically with horizontal or vertical orientation.

2 Workflow

The workflow for this project is shown in Figure 1.

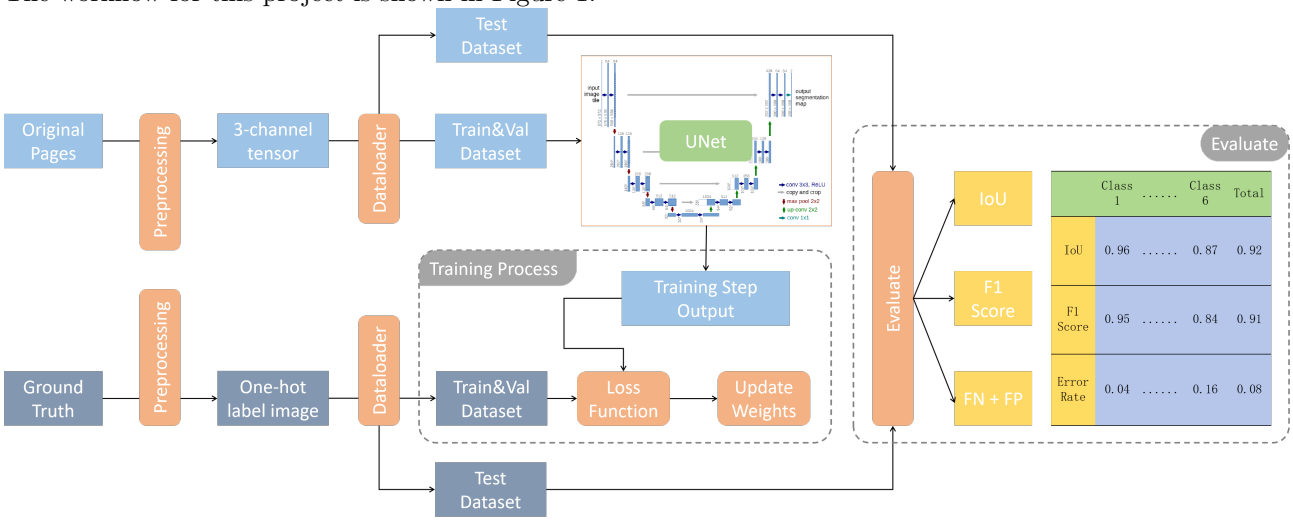

Figure 1: Workflow

3 Dataset Preprocessing

In the data preprocessing stage, I filter out the background part of the original image and remove the yellow (text box) part of the Mask. After that I will denoise the original image using Gaussian kernel or other methods. Since the image is too big (960x640), I will resize it to one half of the original (480x320).

4 Dataset and Dataloader

I have divided the dataset into training set, validation set and test set in the ratio of 8:1:1. Among them, the test set is used to apply evaluation metrics to measure whether the model performs well or not.

For document images in JPG format, I read them as three channels of image data. Note that for Mask images in GIF format, their colors should not be used as labels (e.g., red is labeled [1, 0, 0]), but rather they should be encoded according to how many colors there are in total.

Here both just a single-channel image can be used, where the integer value of each pixel directly represents the category label. For example, 0 for the background, 1 for the first category, 2 for the second category, etc.; or

a One-hot encoded Mask can be used: in this case, the label of each pixel is encoded as a vector whose length is equal to the number of possible labels. For example, if there are three categories (background, category 1, category 2), then the label of each pixel will be a three-dimensional vector (in our task, there are six categories, excluding the yellow text box).

5 Model

I tested the performance of the UNet model and its two variants, UNet++ and UNetDense, on this task.

Originally I proposed the idea of using cross-layer connections in ResNet to solve the "leakage" problem. However, after using the one-hot mask, the problem disappeared. But I still tested the performance of my modified model. The results are shown in the Experiments section.

6 Training Process

During the training process, I use both cross-entropy loss at the pixel level and boundary loss at the edge level of the segmented image, but of course, the specifics will depend on the training results.

I accumulate the gradients in the training set, update the model, and judge the results in the validation set, based on the results to determine whether to save the model, and then get the best performing model in the validation set.

7 Metrics

This is the most confusing part of this task: how do we know that the model is performing well? In other words, what are the most common metrics used on image segmentation tasks?

Pixel Accuracy refers to the percentage of pixels in an image that are correctly categorized. While it looks great, and after all, we do use it as an optimization goal for training, it's not a good metric. Let's say, for example, that the model gives a segmented image that is broadly very similar to Mask, and calculates a Pixel Accuracy of 95% or more. But in reality there are keywords or vignettes that need to be segmented that are not segmented at all - a phenomenon known as class imbalance, which suggests that high Pixel Accuracy doesn't always mean superior segmentation. This requires us to look for other metrics.

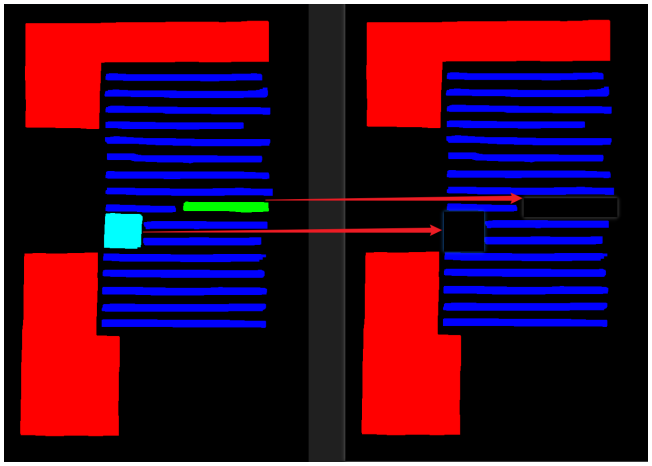

Figure 2: Class Imbalance

.

Intersection-Over-Union (IoU), one of the most commonly used metrics in semantic segmentation, is calculated as the area of overlap between the predicted segmentation and the true situation divided by the area of concatenation between the predicted segmentation and the true situation. For binary (two-class) or multi-class segmentation, the average IoU of an image is computed by averaging the IoUs of each class (as in our task). This will give more attention to image blocks that are not very large but need to be segmented.

Dice Coefficient (F1 Score) refers to the 2 * overlap area divided by the total number of pixels in the two images.Dice coefficients are very similar to IoU. They are positively correlated, both ranging from 0 to 1, where 1 indicates the greatest similarity between prediction and fact. But they are not the same thing: in general, the IoU metrics tend to quantitatively penalize individual instances of bad classification more than the F score, even if they both agree that the instance is bad. In the same way that L2 penalizes the largest errors more than L1, IoU metrics tend to have a "squaring" effect on errors relative to the F score. As a result, F scores tend to measure something closer to average performance, while IoU scores measure something closer to worst-case performance.

8 Experiment

I use a 4090 on my personal computer for training. The batchsize is 1 (for easy visualization during training) and num of epoch is 300. A full training session takes about half an hour. The results are shown in Table 1:

Table 1: IoU and F1 score for different models across various classes

Model	IoU						
	Cyan	Magenta	Green	Red	Blue	Black	Total
UNet	0.962	0.973	0.953	0.992	**0.967**	**0.988**	0.972
UNet++	**0.970**	0.972	0.952	**0.993**	0.964	**0.988**	**0.973**
UNetDense	0.631	0.959	0.929	0.989	0.961	0.985	0.909
UNetRes	0.400	**0.976**	**0.957**	0.993	**0.967**	0.986	0.880
	F1 Score						
	Cyan	Magenta	Green	Red	Blue	Black	Total
UNet	0.580	0.786	0.776	**0.996**	**0.983**	**0.994**	0.852
UNet++	**0.585**	0.786	0.775	0.996	0.982	0.994	**0.853**
UNetDense	0.574	0.779	0.763	0.995	0.980	0.993	0.847
UNetRes	0.000	**0.788**	**0.778**	0.996	0.983	0.993	0.756

You can see that UNet and UNet++ perform similarly, UNetDense doesn't perform that well, and UNetRes, which I designed myself, performs poorly in predicting the first label (probably a code problem) but performs basically equal to or better than UNet in predicting the other labels. This suggests that the introduction of residual concatenation helped the segmentation task to some extent.

The loss, IoU and F1 score during training UNet are shown in Fig 3:

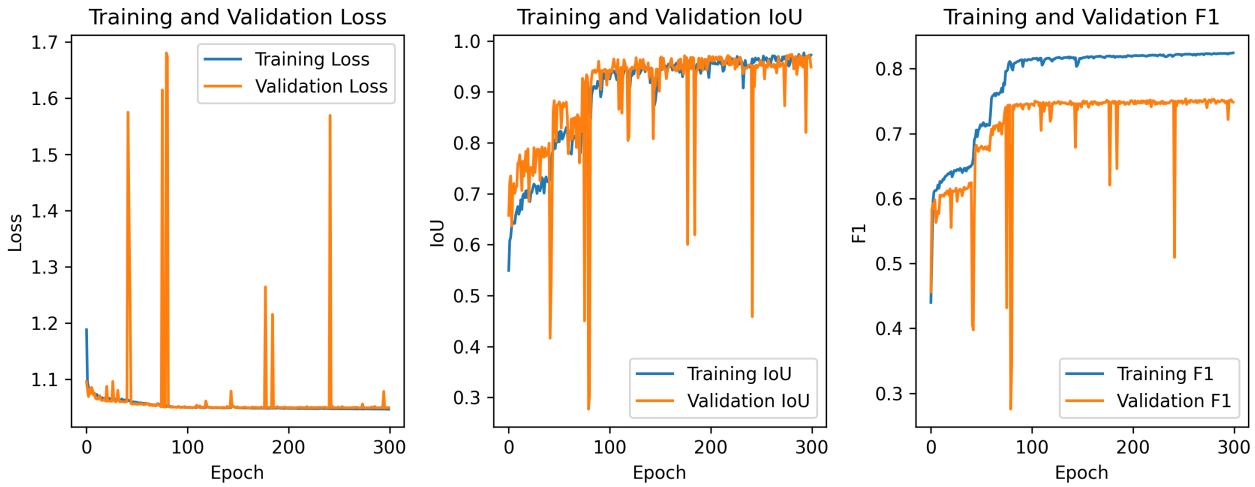

Figure 3: Training step

9 Result

I think the results are great and the visualization comes out as follows:

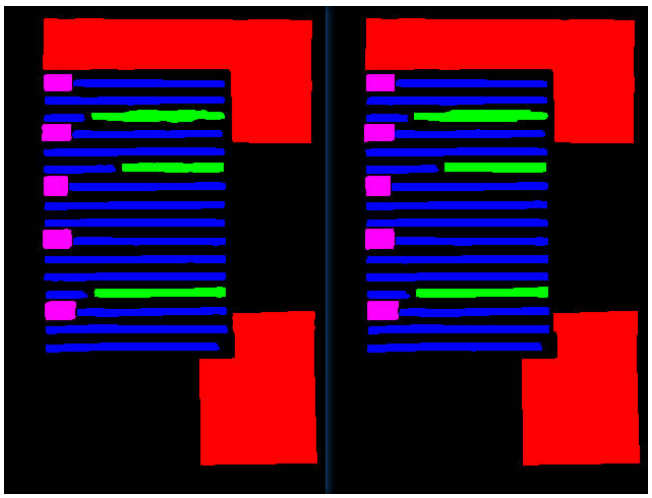

Figure 4: Prediction and groundtruth mask

10 Conclusion

This is the first time I've written all the code for a split task step by step. It mainly includes reproducing UNet, improving UNet, reproducing UNet++, writing Dataset classes for the dataset, writing training and testing code, and so on.

The most important part of this is actually how to label the Mask image. I've used RGB as a label before, and the result is a "leakage" problem, as shown in the figure:

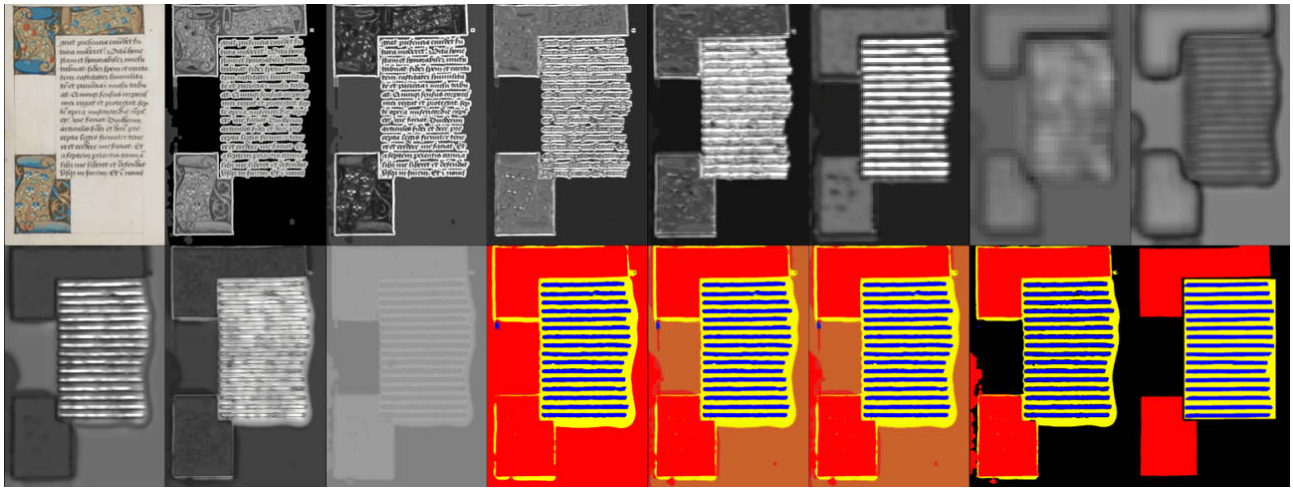

Figure 5: "Leakage" problem

Later on, I read other papers and realized that we can convert Mask into one-hot coded images. So I experimented, but the result is still very poor. After that, I restored the result and Mask to the original picture, and found that Mask could not be restored normally, and after checking for two days, I found that it was necessary to convert the GIF picture into RGB picture after reading it, in order to generate the one-hot label. After the modification and training, the IoU quickly increased from 28% to 97%.