ID Card Detection

1 Description

We want to solve a problem that occurs at buildings/companies' receptions. Currently, a receptionist needs to input all the information manually and it takes a long time. Visitors need to wait for a long time and sometimes it creates a long queue in front of the reception. We want to build a visitor sign-in system (visitor sign-in system) for company receptions. This system takes a photo of the Visitor's Photo ID card by a tablet placed on the reception and extracts the following information to identify the visitor.

- Face Photo
- Name
- ID Number
- Any other information as much as possible

And we cannot predict what type of ID card the visitor provides, it can be ...

- Passport of any countries
- The national ID card of any countries - something else

So, we need to support unpredictable formats. And also, the data extraction may happen on demand, so we need to complete the process within a reasonable duration.

2 Common used methods

The Optical Character Recognition (OCR) systems have been widely used in various of application scenarios, such as office automation (OA) systems, ID card detection, factory automations, online educations, map productions etc. However, OCR is still a challenging task due to the various of text appearances and the demand of computational efficiency. Let's look at the most commonly used methods for ID card detection problem.

2.1 PP-OCR

PP-OCR architecture is based on lightweight neural networks that have been improved. The proposed framework consists of three steps: text border detection, text corner correction, and text recognition. All three modules use lightweight backbones to speed things up. This allows the trained models to be used in embedded devices.

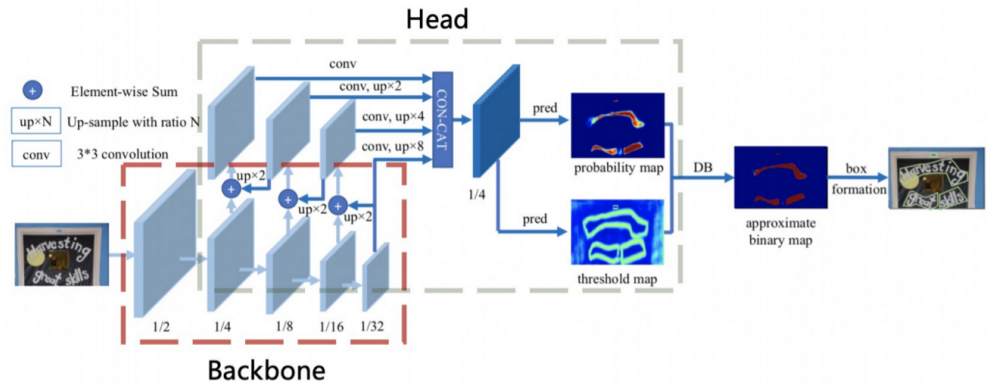

Figure 1: The architecture of the text detector DB.

PP-OCR uses a segmentation-based DB text detection algorithm. In order to improve the text recognition effect in the detection frame and maintain the consistency of the text, it is generally desired that the text frame to be recognized is in a positive horizontal direction. Since the DB text detection result is represented by a polygon with 4 points, it is easy to transform the detection result in a horizontal direction after an affine transformation.

PP-OCR uses CRNN, a commonly used method for text recognition. Although the CRNN text recognition method was proposed in 2016 and has some history, it is the most commonly used and effective text recognition method for Chinese recognition.

Figure 2: Demonstration PP-OCR on ID card

2.2 MORAN

They proposed a multi-object rectified attention network (MORAN) for general scene text recognition. The MORAN consists of a multi-object rectification network and an attention-based sequence recognition network. The multi-object rectification network is designed for rectifying images that contain irregular text. It decreases the difficulty of recognition and enables the attention-based sequence recognition network to more easily read irregular text. It is trained in a weak supervision way, thus requiring only images and corresponding text labels. The attentionbased

sequence recognition network focuses on target characters and sequentially outputs the predictions. Moreover, to improve the sensitivity of the attentionbased sequence recognition network, a fractional pickup method is proposed for an attention-based decoder in the training phase. With the rectification mechanism, the MORAN can read both regular and irregular scene text. Extensive experiments on various benchmarks are conducted, which show that the MORAN achieves state-of-the-art performance.

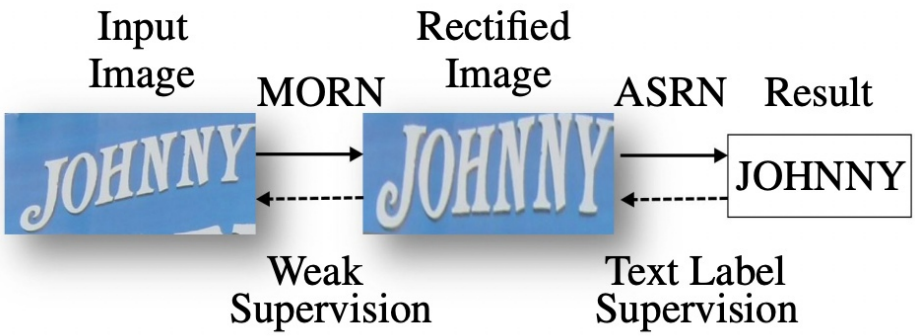

Figure 3: Overview of the MORAN. The MORAN contains a MORN and an ASRN.

The image is rectified by the MORN and given to the ASRN. The dashed lines show the direction of gradient propagation, indicating that the two sub-networks are jointly trained. The MORAN contains two parts. One is the MORN, which is trained in a weak supervision way to learn the offset of each part of the image. According to the predicted offsets, we apply sampling and obtain a rectified text image. The other one is ASRN, a CNN-LSTM framework followed by an attention decoder. The proposed fractional pickup further improves attention sensitivity. The curriculum learning strategy guides the MORAN to achieve stateof-the-art performance.

3 Datasets

3.1 Pre-training

First of all, we have to clarify the type of data in our task. These are, of course, ID cards, pass cards, driver's licenses, etc. So, we have to solve the problems of noisy photos, distorted photos, skewed photos. Also we have to solve text border detection problem and OCR, of course. There is not much open datasets on the Internet for the OCR problem, but some of them will be extremely useful. We need to get open data in order to pre-train our future model on large datasets.

The ICDAR2003 dataset is a dataset for scene text recognition. It contains 507 natural scene images (including 258 training images and 249 test images) in total. The images are annotated at character level. Characters and words can be cropped from the images. I think it will help a lot.

FUNSD. Form Understanding in Noisy Scanned Documents (FUNSD) comprises

199 real, fully annotated, scanned forms. The documents are noisy and vary widely in appearance, making form understanding (FoUn) a challenging task. The proposed dataset can be used for various tasks, including text detection, optical character recognition, spatial layout analysis, and entity labeling/linking.

TextOCR is a dataset to benchmark text recognition on arbitrary shaped scene-text. TextOCR requires models to perform text-recognition on arbitrary shaped scene-text present on natural images. TextOCR provides 1M high quality word annotations on TextVQA images allowing application of end-to-end reasoning on downstream tasks such as visual question answering or image captioning.

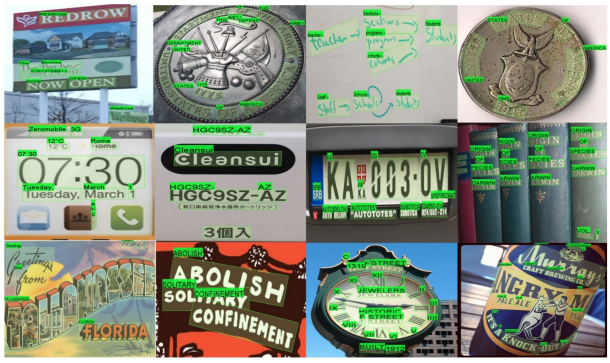

Figure 4: Data samples from TextOCR dataset.

3.2 Fine-tuning

After pre-training, the model needs to be fine-tuned on real data with which it will interact in the prediction. To do this, you need a small dataset of ID cards, driver licenses, ... We ask the customer for this. Also this can be found on the Internet, parsed from different sites or Google images, the more dataset we can get, the better the fine-tuning of the model for our task will be.

Each of these examples should have the maximum resolution in order to avoid incorrect recognition and have a high-quality annotation of the parts necessary for recognition.

Figure 5: Data sample of the high DPI for fine-tuning

So, machine learning requires labeled datasets so that the model can understand the input patterns easily and clearly. This is the Data Annotation process and is very important. The annotation process takes place on our sample of data used for fine tuning. We can use common services (for example, Scale) for data annotation and various NER models for feature extraction and named entities.

During these steps we have to deal with a lot of GPU. I think we should look towards cloud GPUs, instead of using a real machine. There a lot of services which provide cloud GPU. The key benefits of using Cloud GPUs are:

1. **Highly Scalable.** If we want to expand our task, its workload will eventually increase. We will need a GPU that can scale with your increased workload.
2. **Minimizes Cost.** Instead of buying physical GPUs of high power that costs incredibly high, we can go with cloud GPUs on a rental that is available at a lower cost on an hourly basis.

4 Metrics

But what about metrics? The final step now is to assess how well our model has performed. Even if it gave high confidence scores, we need to measure performance with objective metrics. Since we cannot improve what we do not measure, these metrics serve as a vital benchmark for the iterative improvement of your OCR model. We will look at two metrics used to evaluate OCR output, namely Character Error Rate (CER) and Word Error Rate (WER).

A common intuition is to see how many characters were misspelled. While this is correct, the actual error rate calculation is more complex than that. This is because the OCR output can have a different length from the ground truth text.

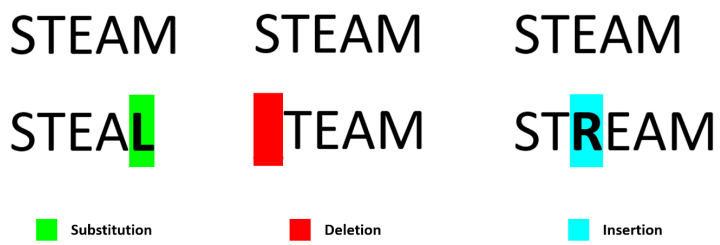

Figure 6: Examples of the three basic errors

4.1 Character Error Rate (CER)

CER calculation is based on the concept of Levenshtein distance, where we count the minimum number of character-level operations required to transform the ground truth text (aka reference text) into the OCR output. Good OCR accuracy: CER 1 - 2 % (i.e. 98 - 99 % accurate). It is represented with the formula:

$$CER = \frac{S + D + I}{N}$$

where S = Number of Substitutions, D = Number of Deletions, I = Number of Insertions, N = Number of characters in reference text (aka ground truth).

Also there should be CER Normalization. One thing to note is that CER values can exceed 100%, especially with many insertions. For example, the CER for ground truth 'ABC' and a longer OCR output 'ABC12345' is 166.67

4.2 Word Error Rate (WER)

If our project involves transcription of particular sequences (e.g., social security number, phone number, etc.), then the use of CER will be relevant. On the other hand, Word Error Rate might be more applicable if it involves the transcription of paragraphs and sentences of words with meaning (e.g., pages of books, newspapers).

$$WER = \frac{S_w + D_w + I_w}{N_w}$$

The formula for WER is the same as that of CER, but WER operates at the word level instead. It represents the number of word substitutions, deletions, or insertions needed to transform one sentence into another. For example: Ground Truth: 'my name is kenneth'. OCR Output: 'myy nime iz kenneth'. The CER is 16.67%, whereas the WER is 75%.

5 Timing

I see four main steps of developing this project.

- Firstly, we have to pre-train our model and test it on abstract data. It will take about 1 week.
- Secondly, we have to select our metrics and fine-tune our model (here I assume that it doesn't take time to collect data for additional training of our model, but in real life it may take additional time). The second step will take 2 weeks.
- The third step is final testing our model. Unit tests will carry out and metrics will check again, hyperparameters will save. The code is reproducible. It will take about 3 days.
- The final fourth step is deploy. Release of the model in real life and preliminary tests. About a day.

I think that the whole project from idea to release will take a little over a month, given the force majeure and discussion with the customer.

6 Conclusion

During the problem analysis, we considered the data type, training stages, the simplest pipeline for this project, as well as metrics and timing. The project is quite feasible, many companies are now doing similar projects, all of them were successful.

So, I can say that the project can be successful and completed with the achievement of the set goals and timing.

7 References

Here are the useful links.

- Datasets https://paperswithcode.com/datasets?task=optical-character-recognition
- MORAN https://arxiv.org/pdf/1901.03003.pdf
- PP-OCR https://arxiv.org/abs/2009.09941
- Metrics https://towardsdatascience.com/evaluating-ocr-output-quality-with-character-error-rate-cer-and-word-error-rate-wer-853175297510
- PadleOCR https://github.com/PaddlePaddle/PaddleOCR