

动手学OCR

Dive into OCR

OCR

Dive into OCR

**Chenxia Li, Weiwei Liu, Ruoyu Guo, Xiaoting Yin
Kaitao Jiang, Yongkun Du, Yuning Du
Lingfeng Zhu, Runjie Jin, Keying Liu, Yehua Yang
Ran Bi, Xiaoguang Hu, Dianhai Yu, Yanjun Ma**

Aug 12, 2022

CONTENTS

1 Preface	1
1.1 About the Book	1
1.1.1 Content and Structure	2
1.1.2 Intended Audience	3
1.2 Community	3
1.3 Acknowledge	3
2 Course Prerequisites	5
2.1 Preliminary Knowledge	5
2.2 Basic Environment Preparation	5
2.3 Code Acquisition and Running	6
2.4 Access to Information	6
2.5 Asking for Help	6
3 Introduction to OCR Technology	7
3.1 Technical Background of OCR	7
3.1.1 Application Scenarios	7
3.1.2 Technical Challenges	9
3.2 OCR Cutting-edge Algorithms	10
3.2.1 Text Detection	10
3.2.2 Text Recognition	12
3.2.3 Document Structured Recognition	14
3.2.4 Other Technologies	18
3.3 Industrial Practice of OCR Technology	19
3.3.1 Difficulties in Industrial Practice	19
3.3.2 Industry-level OCR Development Kit —PaddleOCR	20
3.4 Summary	28
3.5 Reference	28
4 Text Detection	31
4.1 Introduction to Text Detection Methods	32
4.1.1 Regression-based Text Detection	33
4.1.2 Segmentation-based Text Detection	37
4.2 Practice of a Text Detection Algorithm DBNet	41
4.2.1 Quick Start	42
4.2.2 Detailed Implementation of DB Text Detection Algorithm	45
4.2.3 Training the DB Text Detection Model	51
4.2.4 Text Detection FAQ	71
4.2.5 Assignment	76
4.3 Summary	76

4.4	Reference	77
5	Text Recognition	79
5.1	Introduction to Text Recognition Methods	81
5.1.1	Regular Text Recognition	81
5.1.2	Irregular Text Recognition	84
5.2	Practice of a Text Recognition CRNN	87
5.2.1	Quick Start	88
5.2.2	Detailed Implementation of CRNN Text Recognition Algorithm	88
5.2.3	Training the CRNN Text Recognition Model	102
5.2.4	Text Recognition FAQ	111
5.2.5	Assignment	116
5.3	Summary	116
5.4	Reference	117
6	PP-OCR System and Strategy	119
6.1	Introduction of PP-OCR	119
6.1.1	Introduction to PP-OCR System and Optimization Strategies	119
6.1.2	Introduction to PP-OCRv2 System and Optimization Strategies	120
6.2	PP-OCR Optimization Strategies	121
6.2.1	Text Detection	122
6.2.2	Direction Classifier	133
6.2.3	Text recognition	139
6.3	Interpretation of PP-OCRv2 Optimization Strategies	148
6.3.1	Detailed Explanation of Text Detection Model Optimization	149
6.3.2	Detailed Explanation of Text Recognition Model Optimization	161
6.4	Summary	171
6.5	Assignment	171
7	Inference and Deployment of PP-OCRv2	173
7.1	Overview of Inference and Deployment	173
7.1.1	Introduction	173
7.1.2	Environment Preparation	174
7.2	Python Inference Based on Paddle Inference	175
7.2.1	Introduction	175
7.2.2	PP-OCRv2 Text Detection Model Inference	176
7.2.3	Inference of PP-OCRv2 Direction Classifier Model	187
7.2.4	Inference of PP-OCRv2 Text Recognition Model	190
7.2.5	End-to-end Inference of PP-OCRv2 System	193
7.2.6	inference using WHL Package in PP-OCRv2	199
7.3	C++ Inference Based on Paddle Inference	202
7.3.1	Preparaing Model	202
7.3.2	Compiling OpenCV Library	202
7.3.3	Downloading the Inference Library of Paddle Inference	203
7.3.4	Compiling the Inference Code of PaddleOCR	204
7.3.5	Running the PP-OCRv2 System	204
7.4	Service Deployment using Paddle Serving	205
7.4.1	Intorduction to Paddle Serving	206
7.4.2	Preparaing Inference Data and Deployment Environment	206
7.4.3	Preparing for model deployment	207
7.4.4	Deploying Paddle Serving pipeline	208
7.4.5	FAQ	210
7.5	End-to-side inference based on Paddle Lite	210
7.5.1	Enviornment Preparation	211

7.5.2	Preparaing model	211
7.5.3	Compiling	211
7.5.4	Uploading to mobile terminals such as mobile phones	211
7.5.5	Running	212
7.5.6	FAQ	212
7.6	Homework	213
8	Document Analysis Technology	215
8.1	Introduction to Document Analysis Technology	215
8.1.1	Layout Analysis	215
8.1.2	Table Recognition	220
8.1.3	Document VQA	228
8.2	OCR Table Recognition Practice	236
8.2.1	Quick Start	236
8.2.2	Explanation of Prediction Principle:	238
8.2.3	Training	249
8.2.4	Summary	254
8.2.5	Assignment	254
8.3	DOC-VQA SER Practice	254
8.3.1	Quick Start	254
8.3.2	Explanation of The Principle	257
8.3.3	Training	264
8.3.4	Assignment	270
9	End-to-end algorithm	271
9.1	Background	271
9.2	Algorithms	272
9.2.1	End-to-end Regular Text Recognition Algorithms	272
9.2.2	End-to-end Arbitrary-shaped Text Recognition Algorithms	274
9.3	Summary	284
9.4	References	285
10	Pre-processing Algorithm	287
10.1	Background	287
10.2	Data Augmentation	288
10.2.1	Standard Data Augmentation	289
10.2.2	Image Transformation Techniques	296
10.2.3	Image Cropping Techniques	297
10.2.4	Image Mixing Techniques	300
10.3	Image Binarization	301
10.3.1	Global Thresholding	301
10.3.2	Local Thresholding	301
10.3.3	Techniques Based on Deep Learning	302
10.4	Denoising	303
10.4.1	Spatial Domain Filtering	303
10.5	Transform Domain Filtering	306
10.5.1	BM3D	306
10.5.2	Methods based on Deep Learning	307
10.6	Summary	307
10.7	References	307
11	Data Synthesis Algorithms	309
11.1	Background	309
11.2	Data Synthesis Algorithms	309
11.3	summary	315

11.4 References	315
---------------------------	-----

**CHAPTER
ONE**

PREFACE

In recent years, with the development of technology, Optical Character Recognition (OCR) has been widely used in various scenarios. The text detection and recognition algorithms based on deep learning framework has been widely used in daily life, such as license plate recognition, bank card information recognition, ID card information recognition, train ticket information recognition, etc. In addition, general OCR technology is also widely used, such as content security monitoring, or combined with visual features to complete tasks like video comprehension and video search.

On the one hand, we see the wide application space of OCR, but on the other hand, we also find that there are few books on the world that comprehensively introduce OCR from theory to practice, which leads to many algorithm engineers need to take many detours to get familiar with and understand this field. At the same time, in practical applications, especially in a wide range of general scenarios, OCR problems still need to face some challenges, such as affine transformation, scale problems, insufficient lighting, shooting blur and other technical difficulties; in addition, OCR applications are often docked to massive amounts of data, but require data to be processed in real time; and OCR applications are often deployed on mobile or embedded hardware, and the end-side has limited storage space and computational power, so there are high requirements on the size and prediction speed of OCR models. Sharing these contents, which are very crucial for practitioners, can undoubtedly accelerate the upgrading of the industry related to OCR and the industrial implementation of OCR deep learning technology.

Based on the above motivation, around the key content of OCR industry application, pay tribute to the world renowned book “Dive into Deep Learning”, the book named “Dive into OCR”, through the joint creation of universities, enterprises, community developers, open source all the content and code of the book in Github, and provide a supporting video course for developers to learn to use.

As the book is completed by many developers, the later editors try to unify the style as much as possible, but it is inevitable to miss one, if there are omissions and mistakes, readers are welcome to criticize and correct in the GitHub discussion, and you are also welcome to submit a Pull Request directly to revise and participate in the common construction.

1.1 About the Book

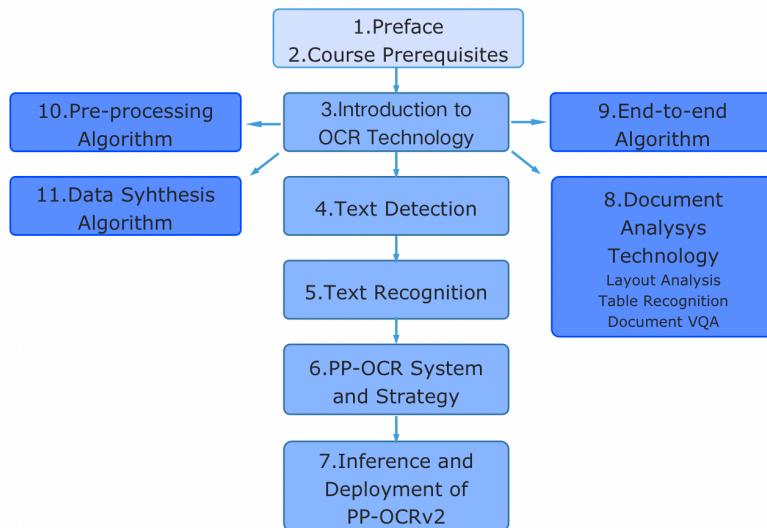
The workflow of this book is also in the form of GitHub submission and maintenance code + jupyter notebook integration code, formulas and images, containing the latest methods and applications of OCR, and constantly updated; in terms of content, it mainly introduces deep learning technology based on and considers the feasibility of practical applications; in terms of content, we dare not claim that this is a rigorous textbook, but it is indeed a vivid tutorial with executable code that can help developers quickly implement OCR projects.

We strongly believe in the importance of hands-on learning for deep learning, and we also present as much as possible how to implement a given method by code, as well as explanations of the ideas and implementation details of the algorithm design. This book, not only for OCR beginners to quickly understand the basic concepts and some key algorithms in OCR area, but also for algorithm engineers who want to quickly get their projects off the ground based on example code and inference and deployment programs.

About this project, we hope to achieve the following goals.

1. Free online access to code source files for all.
2. Use this book as a bridge to attract OCR researchers to build and share, publish their recent research results, expand the technical breadth as much as possible, and become a scientific research overview textbook for OCR technology books;
3. All algorithms contain executable code, showing OCR algorithm engineers how to solve problems in practice, and becoming a tutorial document for OCR industry implementation.
4. The book content is co-built and shared by the entire community, with constantly update iterations, keeping pace with the still rapidly developing field of deep learning.
5. Questions and answers about technical details can be discussed in PaddleOCR's GitHub issues and discussion, allowing people to answer each other's questions and exchange experiences.

1.1.1 Content and Structure



- The first part is the preface and preliminary knowledge of the book, including the knowledge index and resource links needed in the process of using the book.
- The second part of the book, chapters 3-7, introduces the concepts, applications and industrial practices related to the core detection and recognition capabilities of OCR. In “Introduction to OCR Technology”, we explain in general the application scenarios and challenges of OCR, the basic concepts of the technology, and the pain points in industrial applications. Then two basic tasks of OCR are introduced in the chapters “Text Detection” and “Text Recognition”, and each chapter is accompanied by an algorithm explanation to code details and practical exercises. Chapters 6 and 7 are about the detailed introduction of PP-OCR series models. PP-OCR is an OCR system for industrial applications, based on the detection and recognition models through a series of optimization strategies to achieve a General industrial SOTA models, and at the same time to support a variety of inference and deployment solutions to enable enterprises to quickly implement OCR applications.
- The third part of the book, chapters 8-11, introduces applications beyond the two-stage OCR engine, including data synthesis, pre-processing algorithms, and end-to-end models, focusing on the OCR’s capabilities in document

scenarios, including layout analysis, table recognition, and visual document Q&A, again through a combination of algorithms and code that enables readers to understand and apply them in depth.

1.1.2 Intended Audience

This book is for students, researchers and engineers who wish to learn and apply OCR knowledge in depth. It is a practical book in the OCR field and requires basic knowledge of deep learning, machine learning, and computer vision.

1.2 Community

There are two main ways to discuss this book in PaddleOCR

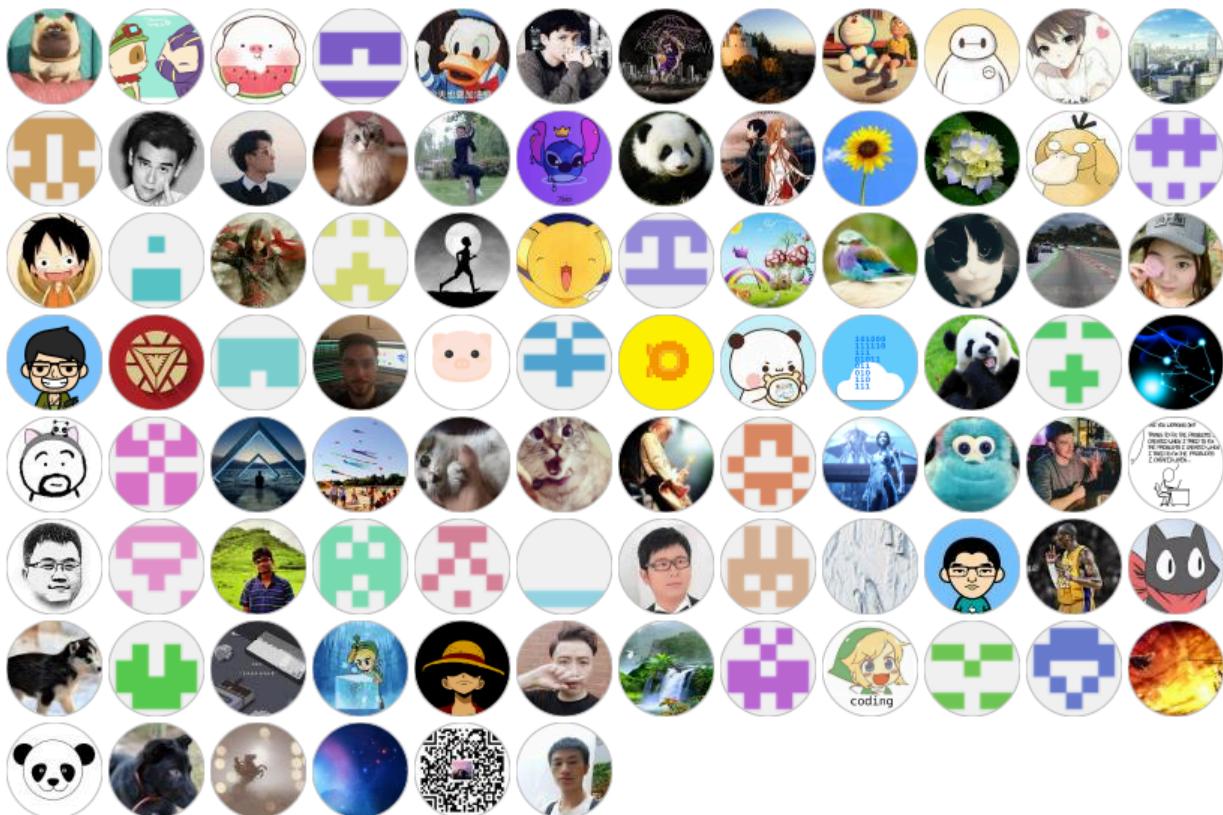
1. [i18n] PaddleOCR's GitHub discussions is mainly for international developers for technical discussions and exchanges, including but not limited to theoretical algorithms, technologies and applications.
2. [Chinese] PaddleOCR Community regular season is a Chinese community activity with OCR as the core, providing multi-level and multi-dimensional open tasks to different types of developers, and giving multiple material and spiritual rewards to excellent community projects.

1.3 Acknowledge

We are extremely grateful to the contributors to the Chinese and English editions of this book, including but not limited to adding content, correcting errata, improving the structure, and providing valuable feedback. In particular, we would like to thank every developer who has contributed to the project. The GitHub username or name of these contributors is (in no particular order): LDOUBLEV, WenmuZhou, dyning, tink2123, MissPenguin, littletomatodonkey, Evezerest, andyjpaddle, D-DanielYang, Topdu, weisy11, BeyondYourself, JetHong, Intsigstephon, xmy0916, cuicheng01, bjjwwang, ZhangXinNan, hysunflower, d2623587501, Wei-JL, xxxpsyduck, Yipeng-Sun, TingquanGao, tangmq, MrCuiHao, authorfu, HexToString, GreatV, neonhuang, xiangyubo, Huntersdeng, iamyo, butplihang, Lovely-Pig, OliverLPH, YukSing12, bingo00, fengxiaoshuai, lilinxiong, SibiAkkash, linkecoding, kjf4096, Sunny-wong, butp906, XiaoguangHu01, Nikhil-Sawant-141, xxlyu-2046, znssoftm, xiaoyangyang2, sdc, ly120117, daassh, PeterH0323, before31, zhiqu, zhangyingying520, DannyIsFunny, ufoym, ITerydh, fushall, baiyfbupt, OneYearIsEnough, tirkarthi, Zhouzd21, karlhorky, lgcy, raoyutian, ronny1996, light1003, JimEverest, Justus-Jonas, Jane-Ding, sushant1212, mengfu188, Channingss, edencfc, mymagicpower

In addition, we would like to give a special thanks to the developers in the OCR community RangeKing, HustBestCat, v3fc, 1084667371, livingbody, haigang1975, fansong1983, Kongsea, fanruinet, thunderstudying, WZMIAOMIAO. They have made an outstanding contribution to the Chinese and English documents of the e-book.

Besides the e-book, it is worth mentioning that 'Dive into OCR' also has a supporting [Chinese video course](#), which is also deeply loved by OCR developers. It has attracted more than 8000 people to sign up for study. **The English courses will be launched later.**



PaddleOCR Contributor

COURSE PREREQUISITES

The OCR model involved in this course is based on deep learning, so its related basic knowledge, environment configuration, project engineering and other materials will be introduced in this section. Readers who are new to deep learning can make use of this part.

2.1 Preliminary Knowledge

The “learning” of deep learning is derived from the content of neurons, perceptrons, and multilayer neural networks in machine learning. Therefore, understanding the basic machine learning algorithms is of great help to the understanding and application of deep learning. The “deepness” of deep learning is embodied in the vector-based mathematical operations such as convolution and pooling used in processing a large amount of information. If not familiar with the theoretical foundation of the two, you can learn it from teacher Li Hongyi’s [Linear Algebra](#) and [Machine Learning](#) courses.

To understand deep learning itself, you can refer to the basic course of Bi Ran, an outstanding architect of Baidu: [Baidu Architect Guides You through Deep Learning Practice](#), which includes the development history of deep learning and introduces all its components with one classic case. It is a practice-oriented course.

To learn the practice of theoretical knowledge, it is essential to take the course of [Basic Knowledge of Python](#). At the same time, in order to quickly reproduce the deep learning model, the framework used in this course is: PaddlePaddle. If you used other frameworks before, you can quickly learn how to use it with [Quick Start Document](#).

2.2 Basic Environment Preparation

If you want to run the code in this course in a local environment and have not built a Python environment before, you can follow the [Beginner’s Guide to Prepare the Operational Environment](#), and install Anaconda or docker environment based on your operational system.

If you don’t have local resources, you can run the code at the AI Studio training platform. Each item in it is presented in a notebook, convenient for developers to learn. If you don’t know how to use Notebook, refer to [AI Studio Project Description](#).

2.3 Code Acquisition and Running

This course relies on the formation of PaddleOCR's code repository. First, clone the whole project of PaddleOCR:

```
# [recommended]
git clone https://github.com/PaddlePaddle/PaddleOCR

# If you cannot pull due to network problems, you can also choose to use the hosting on Gitee:
git clone https://gitee.com/paddlepaddle/PaddleOCR
```

Note: Gitee's code hosting service may not be able to synchronize the update of this github project in real-time, and there is a delay of 3~5 days. Please use the recommended method first.

If strange to git operations, you can download the compressed package in the [Code](#) on the homepage of PaddleOCR

Then install third-party libraries:

```
cd PaddleOCR
pip3 install -r requirements.txt
```

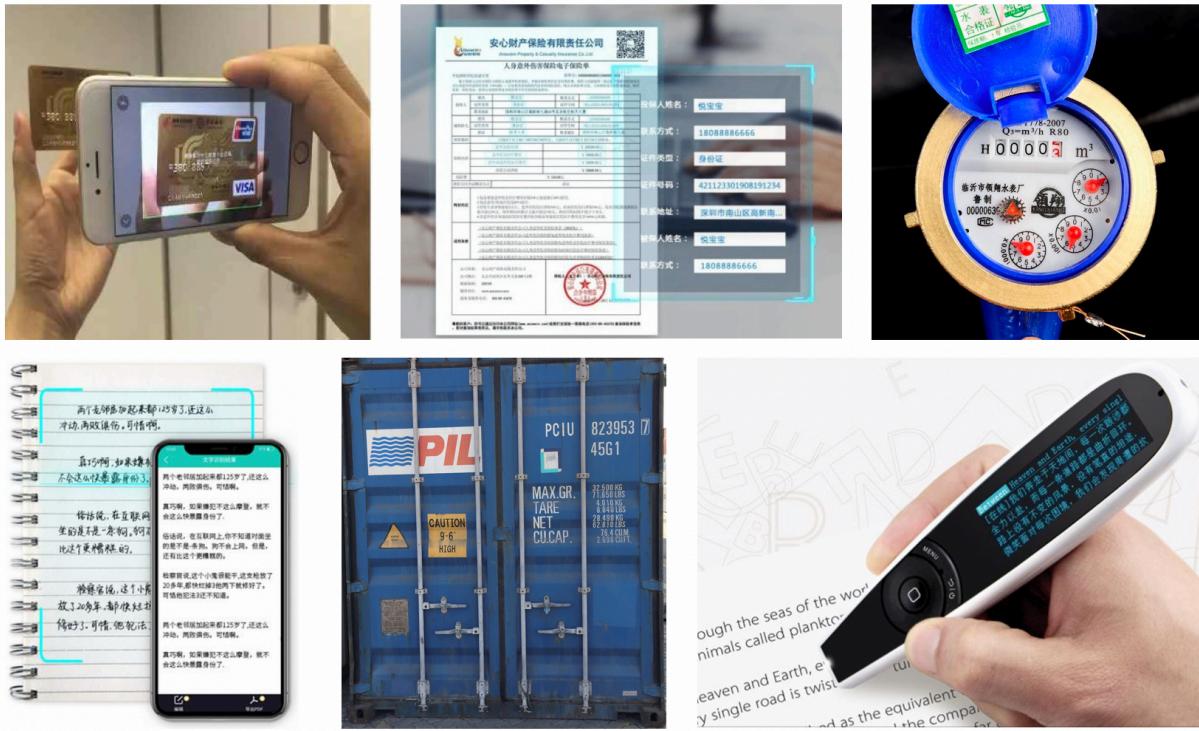
2.4 Access to Information

[PaddleOCR Usage Document](#) elaborate on how to use PaddleOCR to realize model application, training and deployment. The document is informative. Most of the users'questions are covered in it or FAQ, especially in [FAQ](#), which gathers the common issues by following the application process of deep learning. It is recommended that you read it carefully.

2.5 Asking for Help

If you encounter a bug, usability or document-related issues while using PaddleOCR, you can contact the office via [Github issue](#). Please follow the issue template to provide as much information as possible so that official personnel can quickly locate the problem. Also, our WeChat group is a daily communication position for PaddleOCR users, especially for consultation. Besides the PaddleOCR team members, there are also some enthusiastic developers answering your questions.

INTRODUCTION TO OCR TECHNOLOGY



Note: The above pictures are from the Internet

3.1 Technical Background of OCR

3.1.1 Application Scenarios

- What is OCR?

OCR (Optical Character Recognition) is a key field in computer vision. Traditional OCR is usually used to scan documents. Now it often refers to scene text recognition (STR), mainly for natural scenes such as signs shown in the figure below and other texts in various natural scenes.



VS.

Figure 1: Document text recognition VS. Scene text recognition

- What are the application scenarios of OCR?

OCR technology has abundant application scenarios. One typical scenario is the recognition of structured texts of particular areas, which is widely used in daily life, such as license plate recognition, bank card information recognition, ID card information recognition, train ticket information recognition, and so on. The common feature of these verticals is that they have fixed formats. And it is very suitable to use OCR technology for automation, labor-saving and efficiency.

This kind of recognition is currently the scene that is most extensively used and relatively mature in technology in OCR.



Figure 2: Application scenarios of OCR technology

In addition to the recognition of structured texts of particular areas, the general OCR technology also has various application scenarios and is often used to complete multi-modal tasks with other technologies. For example, in the video scene, OCR technology is often used for subtitle automatic translation, content security monitoring, and so on, or to finish tasks like video understanding and video search with visual features.



Figure 3: General OCR in a multi-modal scene

3.1.2 Technical Challenges

There are two kinds of technical challenges: the algorithmic challenges and application challenges.

- **Algorithmic challenges**

OCR enjoys rich application scenarios, leading to its multiple technological difficulties. Here are eight common problems:



Figure 4: Technical challenges of OCR algorithms

These problems have brought huge technological challenges to text detection and recognition. It can be seen that these challenges are mainly generated in natural scenes. At present, most academic research focuses on natural scenes, and so do the academic datasets in OCR. There are many studies concentrating on these issues. But, recognition is more challenging than detection.

- **Application challenges**

In application, especially in various general scenarios, OCR technology also faces two major difficulties in addition to those algorithmic ones summarized above such as affine transformation, scale problems, insufficient lighting, and shooting blur:

1. **Massive data requires OCR to achieve real-time processing.** OCR is often applied to deal with massive data, so real-time data processing is demanded. But it is quite challenging to improve the model speed to meet its standard.

- The end-side application requires that the OCR model is light enough and its recognition speed is fast enough. OCR is often deployed on mobile terminals or embedded hardware. There are generally two modes for end-side OCR applications: uploading to the server vs. terminal-side direct recognition. Considering that the previous method has higher requirements for the network and perform not well in real-time, that the server is under high pressure with large request volumes, and that there may be security risks in data transmission, we hope to adopt the latter method. However, the storage space and computing power of the terminal side are limited, so there are high requirements for the size and inference speed of the OCR model.

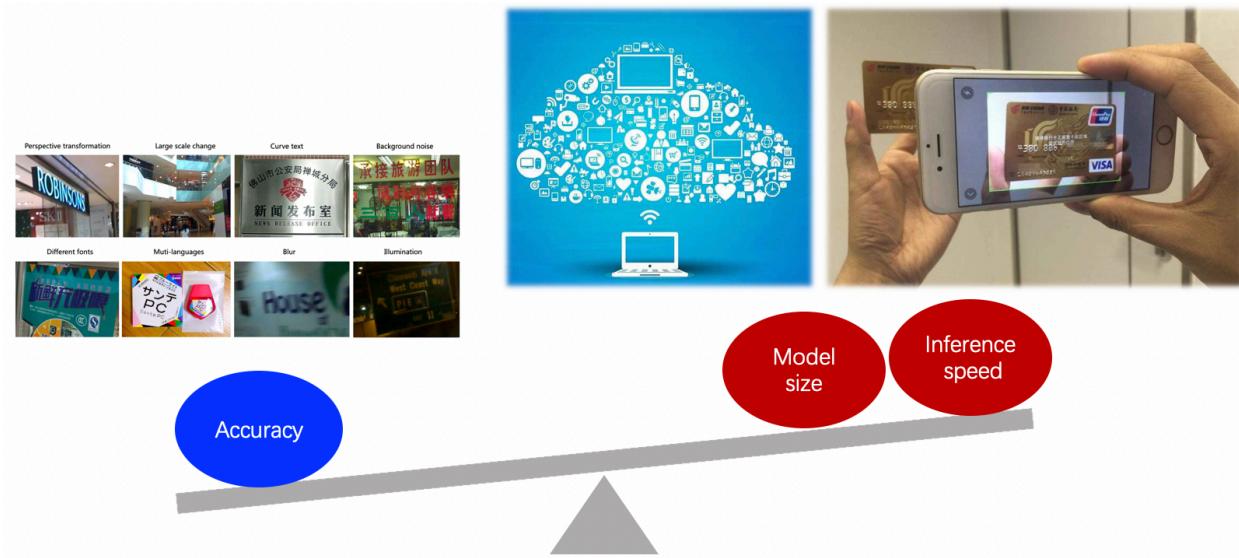


Figure 5: Technical challenges of OCR application

3.2 OCR Cutting-edge Algorithms

Although OCR is relatively specific, it involves many aspects of technologies, including text detection, text recognition, end-to-end text recognition, document analysis, and so on. Academic research on related technologies of OCR flourishes. The following part will briefly introduce some several key technologies in the OCR task.

3.2.1 Text Detection

The text detection task is to locate text regions of the input image. In recent years, there are much academic research on text detection. A class of methods regard text detection as a specific scene in target detection, and modify general target detection algorithms for text detection. For example, TextBoxes[1] is based on one-stage target detector SSD. The algorithm [2] adjusts the target frame to fit text lines with extreme aspect ratios, while CTPN [3] is developed from the Faster RCNN [4]. However, there are still some differences between text detection and target detection in the target information and the task itself. For instance, texts are often quite long and look like “stripes”, line space is small, texts are curved, etc. Therefore, many algorithms especially for text detection have been derived, such as EAST[5], PSENet[6], DBNet[7] and so on.



Figure 6: Example of text detection task

At present, some popular text detection algorithms can be roughly divided into two categories: **Regression-based Algorithms** and **Segmentation-based Algorithms**. There are also some algorithms combining the two. Algorithms based on regression draw on general object detection algorithms, realize detection box regression by setting the anchor, or directly perform pixel regression. This type of methods perform well on discerning regularly-shaped texts, but badly on irregularly-shaped texts. For example, CTPN [3] is good at the recognition of horizontal texts, but poor in the detection of twisted and curved texts. SegLink [8] is more suitable for long texts, but does not fit in detecting sparsely-distributed texts. Algorithms based on segmentation introduced Mask-RCNN [9], this type of algorithms can perform better in the detection in various scenes and texts of various shapes, but the disadvantage is that the post-processing is complicated, so it may be slow in speed and cannot detect overlapped texts.

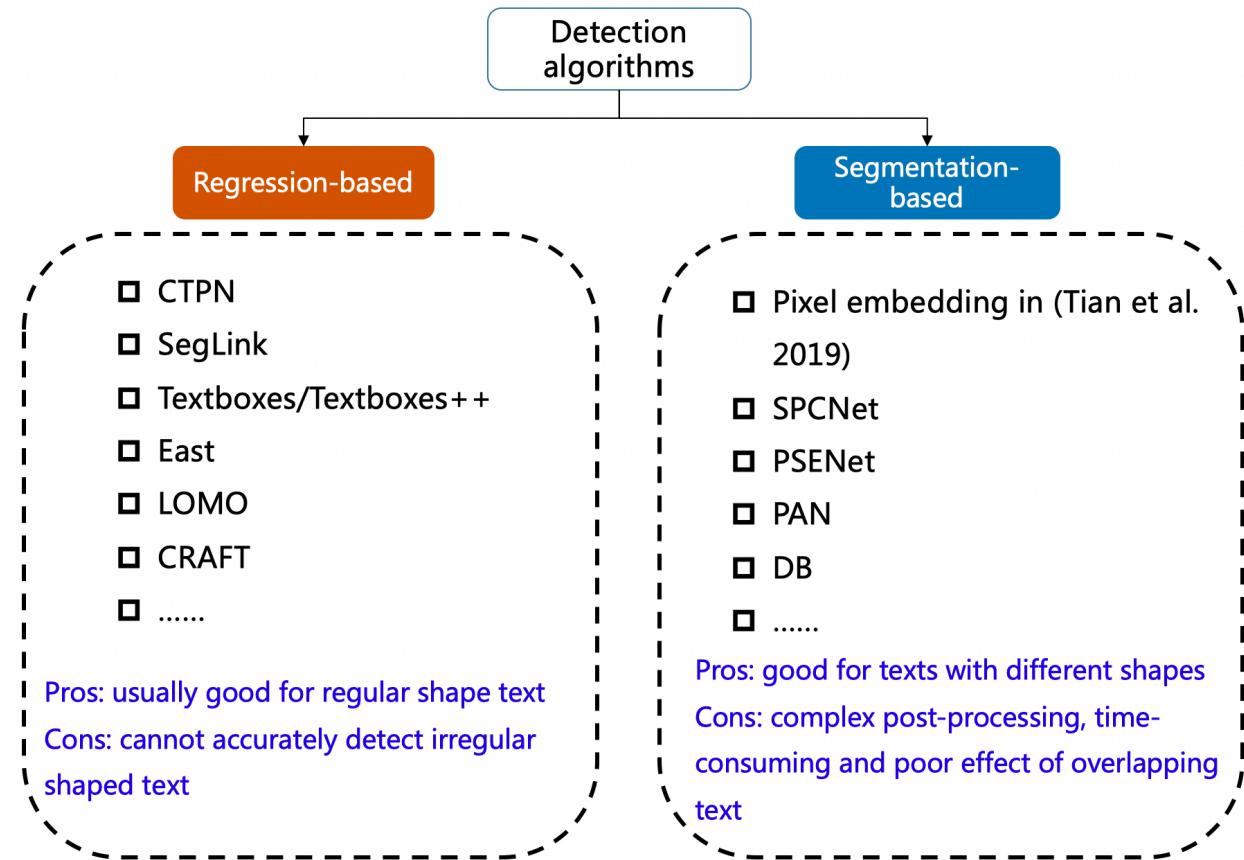


Figure 7: Overview of text detection algorithms

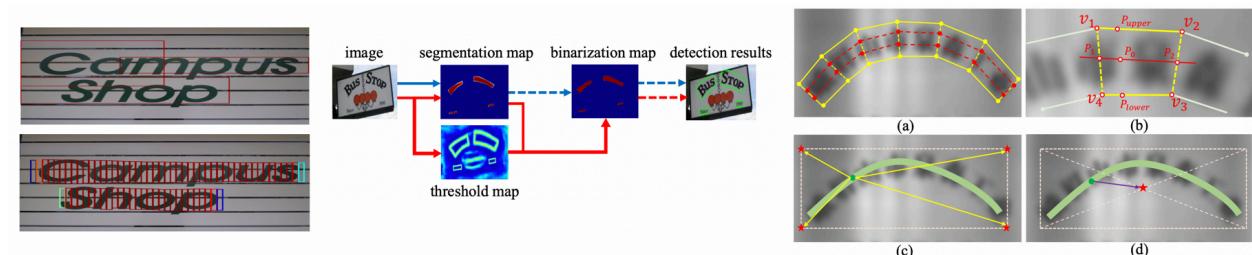


Figure 8: (left) Anchor optimization of CTPN[3] algorithm based on regression (middle) Optimized post-processing of DB[7] algorithm b

The technologies related to text detection will be interpreted and practiced in Chapter 2.

3.2.2 Text Recognition

Text recognition is to recognize the text content in the image, and the input generally comes from the textual area of the image cut out by the text box generated by text detection. Text recognition can generally be divided into two categories: **Regular Text Recognition** and **Irregular Text Recognition** according to the contour of the text to be recognized. Regular text mainly refers to printed fonts, scanned text, and so on which is roughly horizontal. Irregular text is often not in a horizontal position, and often curved, covered, and blurred. Irregular text scenes are very challenging, and it is also the main research direction in text recognition.



Figure 9: (Left) Regular texts VS. (Right) Irregular texts

The algorithms of regular text recognition can be divided into two types according to the different decoding methods: CTC-based algorithm and Sequence2Sequence-based algorithm. They differ in the way to convert the sequence features learned by the network into the final recognition result. A representative of the algorithm based on CTC is classic CRNN [11].

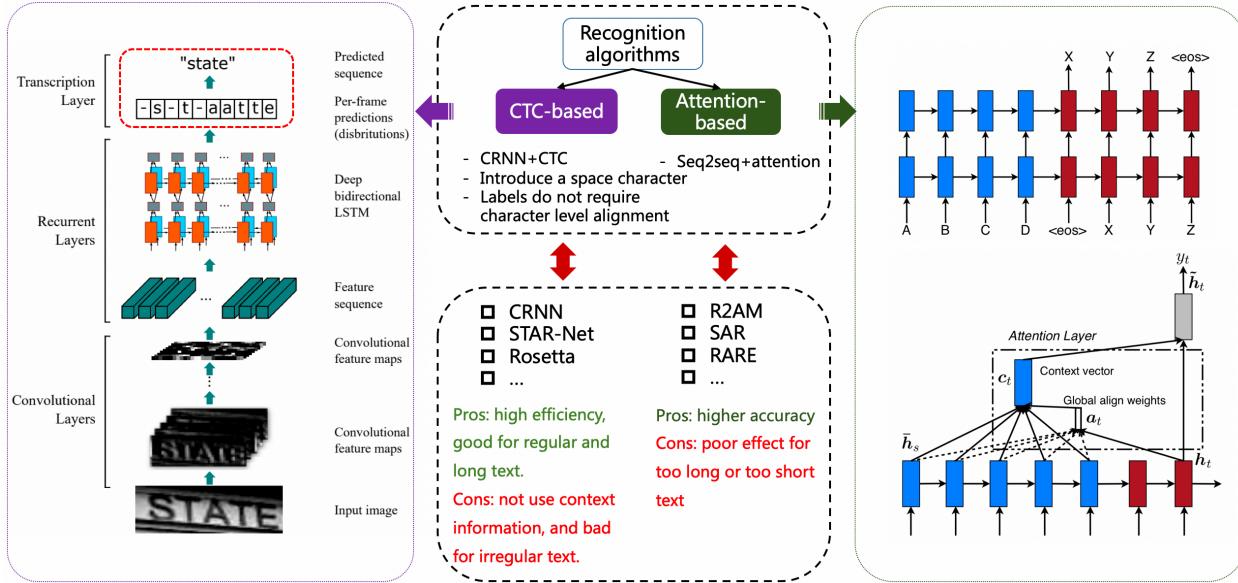


Figure 10: CTC-based recognition algorithm VS. Attention-based recognition algorithm

The recognition algorithms for irregular texts are more abundant. Methods like STAR-Net [12] correct contours of irregular texts into regular rectangles by adding correction modules such as TPS before performing recognition. Attention-based methods like RARE [13] pay more attention to the correlation of parts between sequences. The segmentation-based methods treat each character of a text line as a single unit, making it easier to recognize a segmented character than to recognize the entire text line after correction. In addition, with the rapid development of Transformer [14] and its effectiveness verified in various tasks in recent years, a number of transformer-based text recognition algorithms have flourished. This kind of solutions use the transformer structure to solve the long-term dependency on modeling of CNN and have achieved good results.

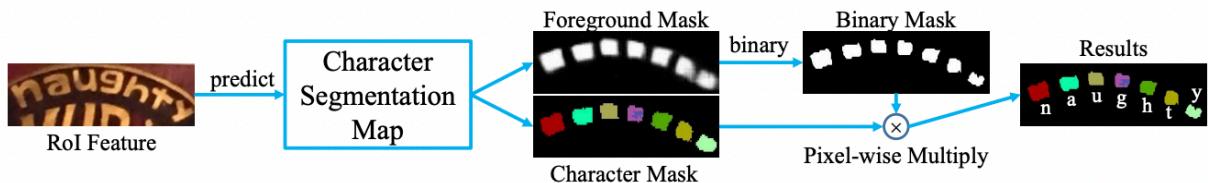


Figure 11: Recognition algorithm based on character segmentation [15]

The related technologies of text recognition will be interpreted and actual combat in detail in Chapter 3.

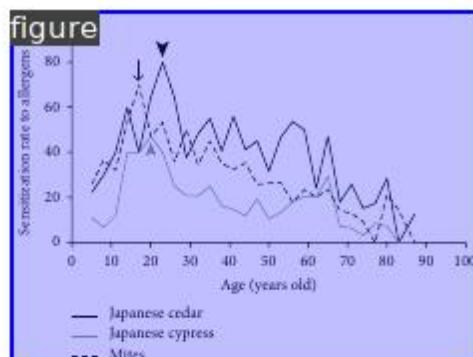
3.2. OCR Cutting-edge Algorithms

3.2.3 Document Structured Recognition

Traditionally, OCR technology can meet the demand for text detection and recognition. However, in practical scenarios, what we need usually is structured information, such as extraction of ID cards and invoices, structured identification of tables, and so on. The application scenarios of OCR technology are mostly express document extraction, contract content comparison, financial factoring document information comparison, and logistics document identification. OCR's result + post-processing is a commonly used structuring scheme, but it is complicated, and post-processing needs to be carefully designed and is poor in generalization. As OCR technology continues to prosper and the demand for structured information extraction is growing, various technologies concerning intelligent document analysis, like layout analysis, table recognition, and key information extraction, have gained increasing attention .

- **Layout Analysis**

Layout analysis is made to classify the content of document images into categories like plain texts, titles, tables, pictures, etc. Current methods generally detect or segment them respectively. For example, Soto Carlos [16] uses contextual information and the inherent position of the document content to improve the region detection performance based on the target detection algorithm Faster R-CNN. Sarkar Mausoom et al.[17] propose a priori-based segmentation mechanism to train the document segmentation model with high-resolution images, solving the problem that different structures in dense regions cannot be distinguished and merged due to excessive reduction of the original image.



text The rate of sensitization (determined by RAST) to Japanese cedar, Japanese cypress, and mites was affected by the patient's age. Black arrow head, gray arrow head, and black arrow show the peaks of each rate, respectively.

table

	Total IgE (IU/mL)	Eosinophil cell proportion (%)
Only spring pollens	118 ± 16	4.5 ± 0.4
Only fall pollens	172 ± 93	3.7 ± 1.4
Only perennial allergens	288 ± 51	3.2 ± 0.4
Spring and fall pollens	174 ± 30	5.2 ± 0.9
Spring pollens and perennial allergens	391 ± 67	5.4 ± 0.5
Fall pollens and perennial allergens	—	—
Spring and fall pollens and perennial allergens	878 ± 213	6.1 ± 0.6
No sensitization	120 ± 15	3.1 ± 0.2

text Range of total serum IgE levels was highest in 8-17-year-olds and decreased with age (Figure 3(a)).

text *Blood Cell Eosinophil Count.* The blood cell eosinophil count was also compared between groups. The eosinophil cell proportion was $4.5 \pm 0.4\%$ in patients sensitized only to spring pollens, while it was significantly higher ($5.7 \pm 0.4\%$) in patients sensitized to both perennial allergens and spring pollens ($P = 0.0146$, Mann-Whitney U test) (Figure 2(b)) (Table 2). The blood cell eosinophil count showed the same reducing tendency (Figure 3(b)).

text *Allergic Sensitization in Asthma.* Fifty-nine patients (46 adults, 13 children) had been previously diagnosed with asthma. The remaining 593 patients had not been diagnosed with asthma. Sensitization to any allergen was detected in 58% of patients with asthma (34/59). Twenty-six (44%) of 59 patients were sensitized to spring pollens (Table 3). Approximately half of the asthma patients (51%; 30/59) were sensitized to perennial allergens. Seven percent of patients with asthma (4/59) were sensitized only to spring

TABLE 3: Allergic sensitization in asthma

table

All pollens	4
Only fall pollens	0
Only perennial allergens	7
Spring and fall pollens	0
Spring pollens and perennial allergens	14
Fall pollens and perennial allergens	1
Spring and fall pollens and perennial allergens	8
No sensitization	25

text while 16% (94/593) in patients without asthma were sensitized exclusively to these allergens. Thirty-seven percent of patients with a previous asthma diagnosis (22/59) were sensitized to both spring and perennial allergens, which was significantly higher than that observed in patients without asthma (20%; 117/593) ($P = 0.0017$, chi-square test).

text Total serum IgE levels in patients with asthma were 177 ± 89 IU/mL, while those in patients without asthma were 224 ± 27 IU/mL ($P = 0.0001$ compared to patients with asthma, Mann-Whitney U test). Blood eosinophil cell proportion in patients with asthma was $5.4 \pm 0.6\%$. In patients without asthma, the proportion was $3.9 \pm 0.2\%$. Blood eosinophil cell proportion in patients with asthma was significantly higher than those in patients without asthma ($P = 0.008$, Mann-Whitney U test).

4. Discussion

text Sensitization, as diagnosed by the serum allergen-specific IgE level, does not always correspond with the patient's symptoms. We found that approximately twice as many patients were sensitized to both spring pollens and perennial allergens compared to patients sensitized only to spring pollens. However, many patients were asymptomatic to perennial allergens. Exposure to perennial allergens, such as house dust mite and cat and dog dandruff, is an important predisposing risk factor for asthma [4]. Previous diagnosis of asthma was largely related to serum IgE levels and blood eosinophil counts [5-7]. Even in nonasthmatic patients airway responsiveness (assessed using methacholine [8]) is increased in some cases of allergic rhinitis, indicating an increased risk for asthma [9-11]. Sensitization to cat dandruff, dust mite, cockroach, and ragweed is an important predictor of airway hyperresponsiveness [12]. Airway hyperresponsiveness is strongly related to elevated total serum IgE levels even in asymptomatic patients [5, 13]. In other words, total serum IgE level is considered an indicator of probable airway hyperresponsiveness or asthma. In our study, total serum IgE levels and blood cell eosinophil counts were significantly elevated in patients sensitized to both spring pollens and perennial allergens, as compared to patients sensitized only to spring pollens. Therefore, patients sensitized to both spring pollens and perennial allergens might be at greater risk of developing airway hyperresponsiveness or asthma.

text Compared to adults, fewer children were sensitized only to spring pollens. Most children (approximately 80%) had

Figure 12: Layout analysis

- Table Recognition

Table recognition is to identify and transfer the table information of the document into an excel file. There are diverse

3.2. OCR Cutting-edge Algorithms

types and styles of tables in text images, such as various rowspans and colspans and different text types. In addition, the style of the document and the light environment during shooting have brought great challenges to table recognition, making table recognition a research difficulty in document understanding.

Methods	Ext	R	P	F	FPS
TextSnake [18]	Syn	85.3	67.9	75.6	-
CSE [17]	MLT	76.1	78.7	77.4	0.38
LOMO[40]	Syn	76.5	85.7	80.8	4.4
ATRR[35]	Sy-	80.2	80.1	80.1	-
SegLink++ [28]	Syn	79.8	82.8	81.3	-
TextField [37]	Syn	79.8	83.0	81.4	6.0
MSR[38]	Syn	79.0	84.1	81.5	4.3
PSENet-1s [33]	MLT	79.7	84.8	82.2	3.9
DB [12]	Syn	80.2	86.9	83.4	22.0
CRAFT [2]	Syn	81.1	86.0	83.5	-
TextDragon [5]	MLT+	82.8	84.5	83.6	-
PAN [34]	Syn	81.2	86.4	83.7	39.8
ContourNet [36]	-	84.1	83.7	83.9	4.5
DRRG [41]	MLT	83.02	85.93	84.45	-
TextPerception[23]	Syn	81.9	87.5	84.6	-
Ours	-	80.57	87.66	83.97	12.08
Ours	Syn	81.45	87.81	84.51	12.15
Ours	MLT	83.60	86.45	85.00	12.21

Methods	Ext	R	P	F	FPS
TextSnake [18]	Syn	85.3	67.9	75.6	-
CSE [17]	MLT	76.1	78.7	77.4	0.38
LOMO[40]	Syn	76.5	85.7	80.8	4.4
ATRR[35]	Sy-	80.2	80.1	80.1	-
SegLink++ [28]	Syn	79.8	82.8	81.3	-
TextField [37]	Syn	79.8	83.0	81.4	6.0
MSR[38]	Syn	79.0	84.1	81.5	4.3
PSENet-1s [33]	MLT	79.7	84.8	82.2	3.9
DB[12]	Syn	80.2	86.9	83.4	22.0
CRAHT [2]	Syn	81.1	86.0	83.5	-
TextDragon 1	MLT+	82.8	84.5	83.6	-
PAN [31]	Syn	81.2	86.4	83.7	39.8
ContourNet [36]	-	84.1	83.7	83.9	4.5
DRRG [Δ1]	MLT	83.02	85.93	84.45	-
TextPerception[23]	Syn	81.9	87.5	84.6	-
Ours	-	80.57	87.66	83.97	12
Ours	Syn	81.45	87.81	84.51	12.15
Ours	MLT	83.60	86.45	85.00	12.21

# Pre-training Data	# Pre-training Epochs	Precision	Recall	F1
500K	1 epoch	0.5779	0.6955	0.6313
	2 epochs	0.6217	0.705	0.6607
	3 epochs	0.6304	0.718	0.6713
	4 epochs	0.6383	0.7175	0.6756
	5 epochs	0.6568	0.734	0.6933
	6 epochs	0.665	0.7355	0.6985
1M	1 epoch	0.6156	0.7005	0.6552
	2 epochs	0.6545	0.737	0.6933
	3 epochs	0.6794	0.762	0.7184
	4 epochs	0.6812	0.766	0.7211
	5 epochs	0.6863	0.7625	0.7224
	6 epochs	0.6909	0.7735	0.7299
2M	1 epoch	0.6599	0.7355	0.6957
	2 epochs	0.6938	0.759	0.7249
	3 epochs	0.6915	0.7655	0.7266
	4 epochs	0.7081	0.781	0.7427
	5 epochs	0.7228	0.7875	0.7538
	6 epochs	0.7377	0.782	0.7592
11M	1 epoch	0.7464	0.7815	0.7636
	2 epochs	0.7597	0.8155	0.7866
	3 epochs			
	4 epochs			

# Pre-training Data	# Pre-training Epochs	Precision	Recall	F1
500K	1 epoch	0.5779	0.6955	0.6313
	2 epochs	0.6217	0.705	0.6607
	3 epochs	0.6304	0.718	0.6713
	4 epochs	0.6383	0.7175	0.6756
	5 epochs	0.6568	0.734	0.6933
	6 epochs	0.665	0.7355	0.6985
1M	1 epoch	0.6156	0.7005	0.6552
	2 epochs	0.6545	0.737	0.6933
	3 epochs	0.6794	0.762	0.7184
	4 epochs	0.6812	0.766	0.7211
	5 epochs	0.6863	0.7625	0.7224
	6 epochs	0.6909	0.7735	0.7299
2M	1 epoch	0.6599	0.7355	0.6957
	2 epochs	0.6938	0.759	0.7249
	3 epochs	0.6915	0.7655	0.7266
	4 epochs	0.7081	0.781	0.7427
	5 epochs	0.7228	0.7875	0.7538
	6 epochs	0.7377	0.782	0.7592
11M	1 epoch	0.7464	0.7815	0.7636
	2 epochs	0.7597	0.8155	0.7866

Figure 13: Table recognition

There are many table recognition methods. For example, in early days, there were traditional algorithms based on heuristic rules, such as the T-Rect algorithm proposed by Kieninger [18] et al. which generally use manual design rules and connected domain detection and analysis. In recent years, as deep learning continue to develop, some CNN-based table structure recognition algorithms have emerged, such as DeepTabStR proposed by Siddiqui Shaob Ahmed [19] et al. and TabStruct-Net proposed by Raja Sachin [20] et al. In addition, with the rise of *Graph Neural Network*, some researchers try to apply *Graph Neural Network* to table structure recognition and regard table recognition as a graph reconstruction issue on the basis of the *Graph Neural Network*. This is the way that TGRNet proposed by Xue Wenyuan [21] et al. works. What's more, there are end-to-end solutions which output the table structure in HTML with the network. Most of these adopt Seq2Seq to predict the table structure such as those based on attention or transformer, including TableMaster [22].

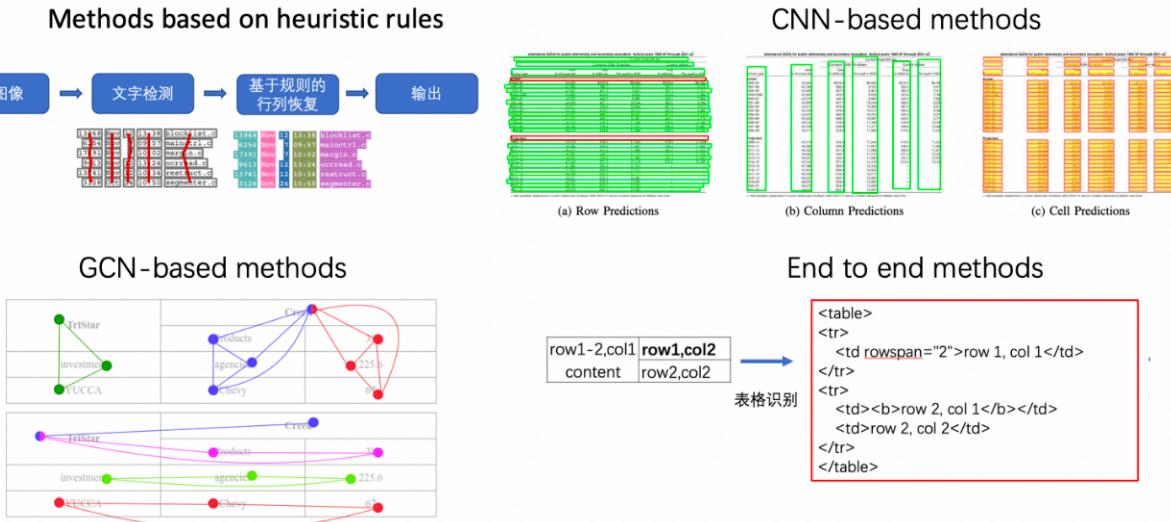


Figure 14: Table recognition methods

- Key Information Extraction

Key information extraction (KIE) is an important task in Document VQA. It refers to the extraction of the needed information from images, such as that of name and ID number from ID cards. Such information is fixed in one task, but is different between different tasks.

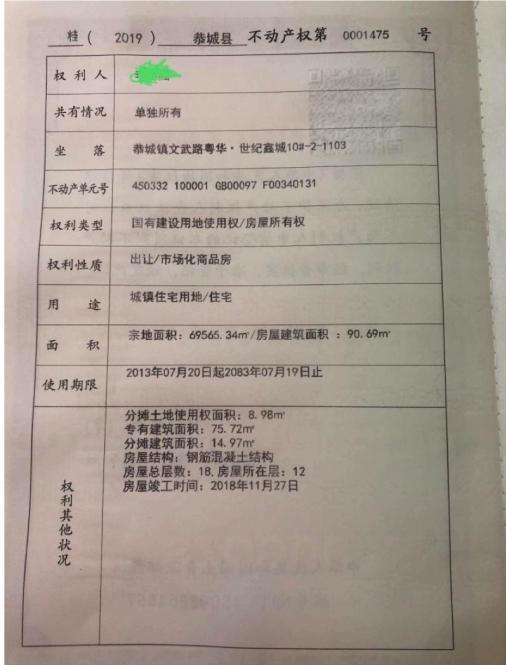


Figure 15: DocVQA tasks

KIE is usually divided into two sub-tasks for research:

- SER: It refers to semantic entity recognition which classifies each detected text. For example, it divides texts into name and ID cards like the black and red boxes in the figure below.

3.2. OCR Cutting-edge Algorithms

- RE: It refers to relation extraction, which classifies each detected text. For example, it may categorize texts into questions and answers, and then find the corresponding answer for each question. As shown in the figure below, the red and black boxes represent questions and answers respectively, and the yellow arrows show the correspondence between questions and answers.



Figure 16: SER and RE tasks

The general KIE method is developed based on the named entity recognition (NER) [4], but this kind of method only uses the text information in the image without employing visual and structural information. Therefore, it is not so accurate. In this way, in recent years, many solutions have begun to merge visual and structural information with text information. Due to the adoption of different principles in fusing multi-modal information, these methods can be divided into four types:

- Grid-based method
- Token-based method
- GCN-based method
- End-to-end method

Relevant technologies of Document analysis will be demonstrated and practiced in Chapter 6.

3.2.4 Other Technologies

Three key technologies in the OCR field are introduced above: text detection, text recognition, document structured recognition, and other cutting-edge technologies related to OCR like end-to-end text recognition, image preprocessing technology, and OCR data synthesis. For more details, please refer to Chapter 7 and Chapter 8.

3.3 Industrial Practice of OCR Technology



If you were Xiao Wang, what would you do?

1. I know nothing about this. I'd give it up.
2. I'd recommend the boss to find an outsourcing company or propose a commercial project. Anyway, it is the boss's bill.
3. I'd find similar projects online, and perform Github-oriented programming.

OCR technology aims to be applied in industrial practice. Although there is a lot of academic research on OCR technology and its commercial application is more mature than other AI technologies, there are still some difficulties in industrial application. The following section will analyze the difficulties in technology and industrial practice.

3.3.1 Difficulties in Industrial Practice

In industrial practice, developers often need to rely on open-source community resources to start or promote projects. But they often encounter three problems in the use of open-source models:



Figure 17: Three major problems in the industrial practice of OCR technology

1. Hard to find & hard to select

The open-source community enjoys rich resources, but information asymmetry will hinder developers from solving pain points efficiently. On one hand, resources of the open-source community are too rich for developers to quickly find projects matching the business requirement in a huge code repository when facing a requirement. This is the “hard to find” problem. On the other hand, in the selection of algorithms, developers have to verify them one by one for indicators on English public datasets cannot provide direct references to the Chinese scenarios that they encounter a lot, which is time-wasting and labor-consuming and even cannot guarantee that the selected ones are the most suitable. And this is the “hard to select” problem.

2. Not applicable to industrial scenarios

The work in the open-source community focuses more on effect optimization such as open source or reproduction of codes in academic papers, and algorithmic effects instead of the model size and speed. But the two indicators are as important as the model effect and cannot be ignored in industrial practice. No matter on the mobile side or the server side, the number of images to be recognized is so large that the model is hoped to be smaller, more accurate, and faster in inference. But GPU is too expensive, so it is more economical to use CPU. On the premise of meeting business demands, the lighter the model is, the fewer resources it will take.

3. Hard to optimize & problematic to train and deploy

Solely using open-source algorithms or models cannot meet business needs directly. In actual business scenarios, OCR needs to be used to deal with various problems. The personalization of business scenarios often requires the retraining of custom datasets. It is expensive to experiment with optimization solutions in the current open-source projects. In addition, OCR has been applied to a lot of scenarios for there is a wide range of application demands on the server and mobile devices. Therefore, diverse hardware environments should support various deployment methods. But the open-source community’s projects focus more on algorithms and models, and lack support in inference and deployment. To apply OCR technology of the algorithms in papers, it has high requirements for algorithmic and engineering abilities of developers.

3.3.2 Industry-level OCR Development Kit —PaddleOCR

OCR industry practice demands a set of full-process solutions to speed up research and development process and save some time. In other words, the ultra-lightweight model and its full-process solutions are a rigid demand especially for mobile terminals and embedded devices with limited computing power and storage.

So, the industry-level OCR development kit [PaddleOCR](#) has come into being.

The construction of PaddleOCR starts from user portraits and needs, selects and reproduces diverse cutting-edge algorithms with the core framework of PaddlePaddle, develops PP models that are more suitable for industrialization based on recurred algorithms, and integrates training and inference to provide various inference and deployment methods and meet different demand scenarios in application.

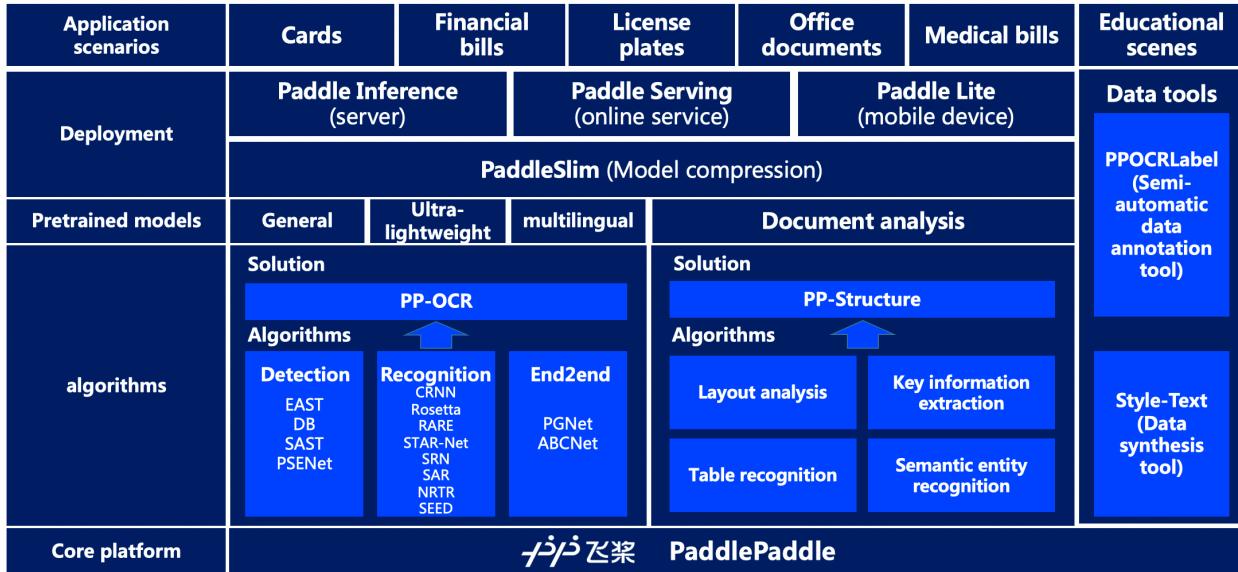


Figure 18: Panorama of the PaddleOCR development kit

It can be seen from the panorama that PaddleOCR provides abundant solutions in model algorithms, pre-training model libraries, and industry-level deployment with the help of the core framework of PaddlePaddle, and provides tools of data synthesis and semi-automatic data annotation to promote developers' data production.

As for model algorithms, PaddleOCR provides solutions for **text detection and recognition** and **document structured analysis** respectively. In terms of text detection and recognition, PaddleOCR has reproduced or open-sourced four text detection algorithms, eight text recognition algorithms, and one end-to-end text recognition algorithm. On this basis, a universal text detection and recognition solution of the PP-OCR series has been developed. In terms of document structured analysis, PaddleOCR provides algorithms such as layout analysis, table recognition, key information extraction, and named entity recognition, and has proposed a PP-Structure document to analyze solutions on this basis. A variety of select algorithms can meet the needs of developers in different business scenarios. The unification of the code framework also facilitates the optimization and performance comparison of different algorithms for them.

At for pre-training model libraries, with PP-OCR and PP-Structure solutions, PaddleOCR has developed and open-sourced PP series models fitting industrial practice, including universal, ultra-lightweight and multilingual text detection and recognition models, and complex-document analysis models. The PP series models are thoroughly optimized based on the original algorithms to make their effect and performance meet the standards in industrial practice. Developers can easily develop a “practical model” for their own business needs by either directly applying the models to business scenarios or using business data for finetuning.

As for industry-level deployment, PaddleOCR provides a server-side inference solution based on Paddle Inference, a service-based deployment solution based on Paddle Serving, and an end-side deployment solution based on Paddle-Lite to meet the deployment needs under different hardware environments. Also, it provides a model compression scheme based on PaddleSlim, which can further compress the model size. The above deployment methods have got through the whole process of training and inference to ensure that developers can make deployment efficiently, stably, and reliably.

As for data tools, PaddleOCR provides a semi-automatic data annotation tool—PPOCRLLabel and a data synthesis tool Style-Text to help developers produce the datasets and annotation information required for model training more conveniently. PPOCRLLabel, as the first open-source semi-automatic OCR data annotation tool in the industry, is aimed at solving the problems of the tedious and mechanical labeling process, massive demands for manual labeling of training data, and high costs of time and money. And it has introduced the built-in PP-OCR model to realizes the mode of pre-labeling + manual verification, which can greatly improve labeling efficiency and save labor costs. The data synthesis tool Style-Text focuses on the solution to serious shortage of real data in actual scenes and the failure to synthesize text styles (fonts, colors, spacing, background) of traditional synthesis algorithms. Only with a few target scene images can a large number of text images similar to target scene in style be synthesized.



Figure 19: Schematic of using PPOCRLLabel

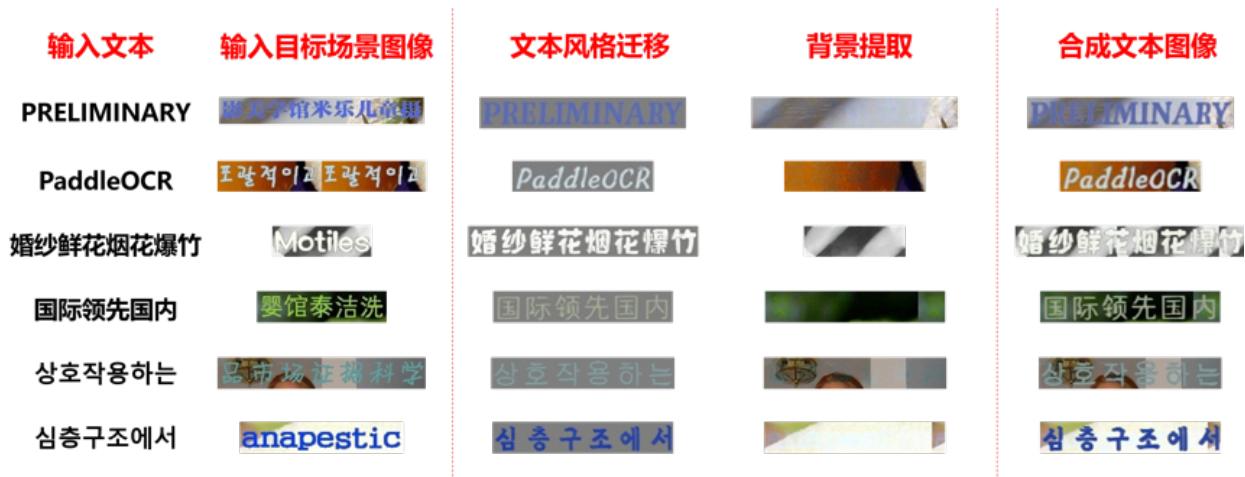


Figure 20: Examples of Style-Text synthesis results

PP-OCR and PP-Structre

The PP series models are thoroughly optimized to meet needs of industrial practice by visual development kits of PaddlePaddle, aiming to strike a balance between speed and accuracy. The PP series models in PaddleOCR include PP-OCR series models for text detection and recognition and PP-Structure series models for document analysis.

(1) Chinese and English model of PP-OCR



Figure 21: Examples of Chinese and English model of PP-OCR on recognition results

The typical two-stage OCR algorithm adopted by the Chinese and English models of PP-OCR is in the paradigm of detection model + recognition model. And its concrete algorithm framework is as follows:

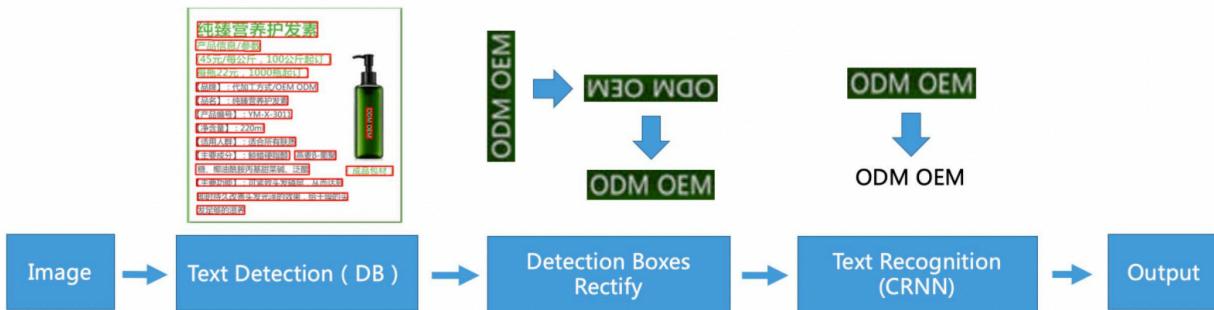


Figure 22: Schematic of the PP-OCR system pipeline

It can be seen that in addition to input and output, the core framework of PP-OCR contains three modules: text detection, detection frame correction, and text recognition.

- Text detection module: Its core is a text detection model trained on the [DB](#) detection algorithm, used to detect the text area in the image.
- Detection frame correction module: Input the detected text box into the detection frame correction module. At this stage, the irregular text box is corrected into a rectangular frame, preparing for text recognition. Also, the text direction will be judged and corrected. For example, if the text line is judged to be upside down, it will be corrected. This function relies on training a text direction classifier.
- Text recognition module: Finally, the module performs text recognition on the corrected detected box to discern the text and obtain the content. The classic text recognition algorithm used in PP-OCR is [CRNN](#).

PaddleOCR has introduced PP-OCR[23] and PP-OCRv2[24] models.

PP-OCR model has a mobile version (lightweight version) and a server version (universal version). The mobile version model is mainly optimized based on the lightweight backbone network MobileNetV3, and its optimized model (detection model + text direction classification model + recognition model) is only 8.1M in size, takes 350ms to predict a single image on the CPU, and takes about 110ms on T4 GPU. After cropping and quantization, its size can be further compressed to 3.5M with the same accuracy, convenient for end-side deployment. The previous model inference test only takes 260ms on the Snapdragon 855 processor. For more evaluation data of PP-OCR , please refer to [benchmark](#).

PP-OCRv2 keeps using the overall framework of PP-OCR, and performs policy optimization of effects, mainly in 3 aspects:

- Its model effect increases by more than 7% over the PP-OCR mobile version;
- Its speed grows by 220% compared with the PP-OCR server version;
- Its model size of 11.6M makes it easy to be deployed on both server and mobile terminals.

The optimization policies of PP-OCR and PP-OCRv2 will be detailed in Chapter 4.

In addition to the Chinese and English models, PaddleOCR has also trained and open-sourced the English digital model and the multi-language recognition model with different datasets. All of these are ultra-lightweight and suitable for different language scenarios.

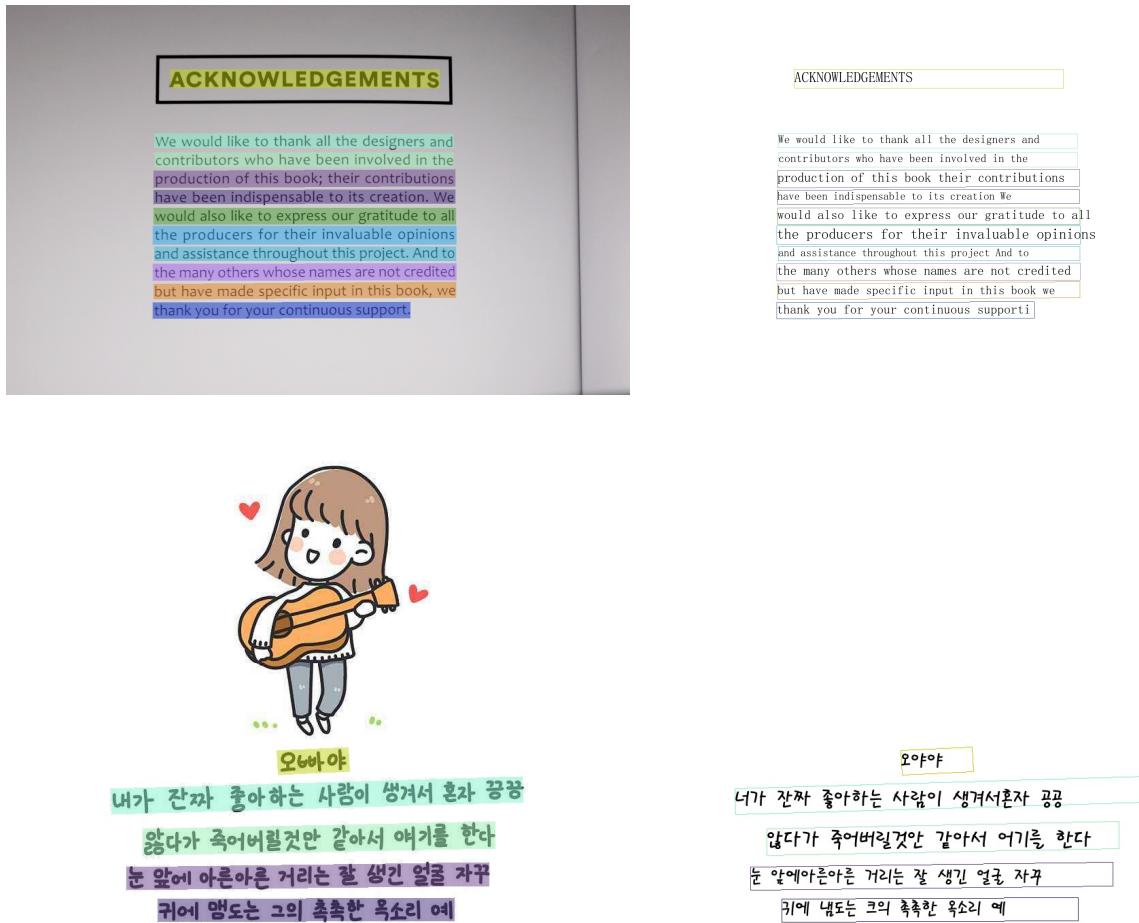


Figure 23: Schematic of the recognition result of the English digital model and multilingual model of PP-OCR

(2) PP-Structure document analysis model

PP-Structure supports three subtasks: layout analysis, table recognition, and DocVQA.

There are six core functions of PP-Structure:

- Performing layout analysis of documents in the form of pictures, which can be divided into 5 types of areas: texts, titles, tables, pictures and lists (used together with Layout-Parser)
- Extracting texts, titles, pictures and lists as text fields (used together with PP-OCR)
- Conducting structured analysis for tables, and the final result is output in an Excel file
- Supporting the Python whl package and the command line, simple and user-friendly
- Supporting custom training for two types of tasks: layout analysis and table structuring
- Supporting VQA tasks— SER and RE



Figure 24: Schematic of PP-Structure system (this figure only contains layout analysis + table recognition)

The specific plan of PP-Structure will be explained in detail in Chapter 6.

Industrial-level Deployment Plan

It is available to conduct full-process and full-scene inference and deployment on PaddlePaddle, with three main sources of models. The first is the training by using a network structure built with the PaddlePaddle APIs. The second is a series of PaddlePaddle toolkits which provides multiple model libraries and concise and easy-to-use APIs, and can be used out-of-the-box, including the visual model library PaddleCV, intelligent speech library PaddleSpeech and natural language processing library PaddleNLP, etc. The third kind of models is generated from third-party frameworks (PyTorch, ONNX, TensorFlow, etc.) by using X2Paddle tools.

PaddlePaddle models can be compressed, quantified, and distilled by using PaddleSlim tools. They supports five deployment schemes, including Paddle Serving (service-based), Paddle Inference (server-side or cloud-side), Paddle Lite (mobile or edge-port), Paddle.js (front end of the web). Some hardware unavailable to such as MCU, Horizon, Kunyun and other domestic chips can be converted into a third-party framework supportive to ONNX with the help of Paddle2ONNX.

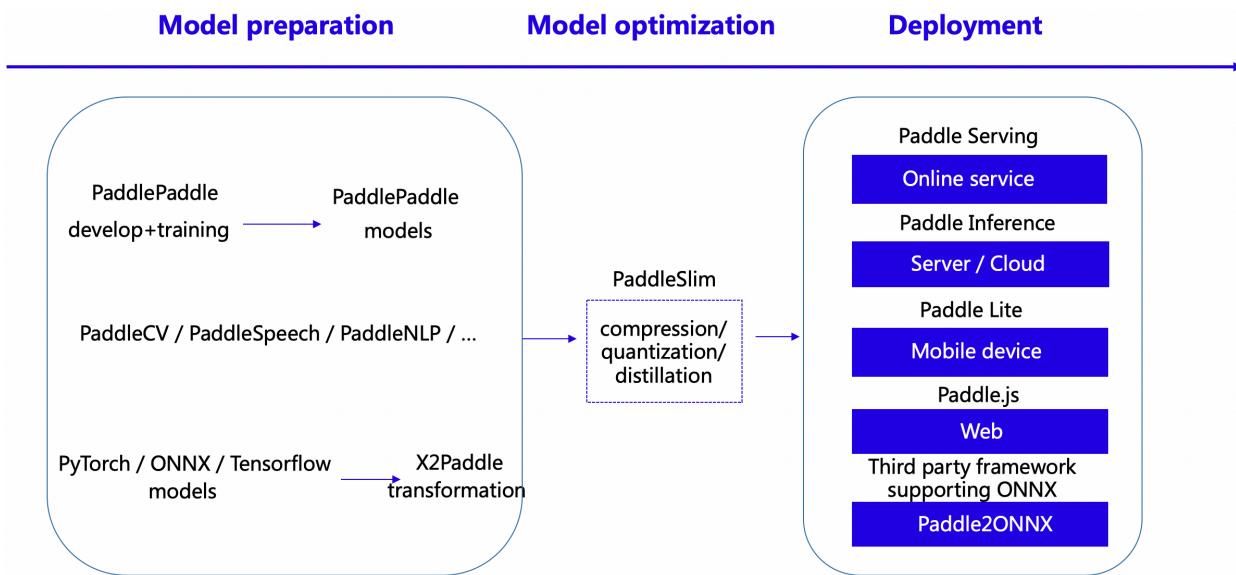


Figure 25: Deployment methods available on PaddlePaddle

Paddle Inference supports server-side and cloud deployment, with high performance and versatility. It has been thoroughly adapted and optimized for different platforms and different application scenarios. Paddle Inference is the native inference library for PaddlePaddle to ensure that models can be used as soon as they are trained on the server side and can be deployed quickly. It is suitable for high-performance hardware to use multiple language environments of applications to deploy models with complex algorithms. The hardware includes x86 CPUs, Nvidia GPUs, and AI accelerators such as Baidu Kunlun XPU and Huawei Shengteng.

Paddle Lite is an end-side inference engine featuring light weight and high performance. It has been configured and optimized in-depth for end-side devices and application scenarios. Currently it supports multiple platforms such as Android, IOS, embedded Linux devices, macOS, and so on. The hardware covers ARM CPU and GPU, X86 CPU and new hardware like Baidu Kunlun, Huawei Ascend and Kirin, Rockchip, etc.

Paddle Serving is a high-performance service framework designed to help users quickly deploy models in cloud services in a few steps. At present, Paddle Serving supports involves custom pre-processing, model combination, update of model hot reload, multi-machine multi-card multi-model, distributed inference, K8S deployment, security gateway and model encryption deployment, and multilingual and multi-client access. The official of Paddle Serving also provides deployment examples of more than 40 models including PaddleOCR, to help users get started faster.

Deployment	Feature	Scene	Hardware
Paddle Inference	General, High performance	<ul style="list-style-type: none"> Complex model algorithm Hardware with high performance 	<ul style="list-style-type: none"> X86 CPU NVIDIA GPU Loongson/Feiteng and other domestic CPUs AI acceleration chips such as Kunlun/Ascend/Haiguang DCU
Paddle Lite	Lite-weight	<ul style="list-style-type: none"> Lite-weight model Miscellaneous hardware, limited resources, low power consumption 	<ul style="list-style-type: none"> Arm CPU Arm / Qualcomm / Apple GPU AI acceleration hardware such as Kunlun / Ascend / Kirin / Rockchip / Yingmai / Cambrian / Bitmain
Paddle Serving	High concurrency	<ul style="list-style-type: none"> Large traffic, high concurrency, low latency, large throughput Resource elasticity regulation to cope with changes in service traffic Model combination, encryption, hot update, etc. 	<ul style="list-style-type: none"> X86 / Arm CPU NVIDIA GPU Kunlun / Shengteng, etc.
Paddle.js	Inference on the browser	<ul style="list-style-type: none"> Chrome, Safari, Firefox, etc 	
Paddle2ONNX	Open and compatible	<ul style="list-style-type: none"> Third-party inference framework Support deployment in more AI chips 	<ul style="list-style-type: none"> Horizon X3 Corairn X3 Allwinner R329 Other AI chips

Figure 26: Supported deployment mode of PaddlePaddle

The above deployment plans will be detailed and practiced based on the PP-OCRV2 model in Chapter 5.

3.4 Summary

This section first introduces the application scenarios and cutting-edge algorithms of OCR technology, and then analyzes the difficulties and three major challenges of OCR technology in its industrial practice.

The content of the follwings chapters is as follows:

- Chapter2 and Chapter3: text detection and recognition and their practice;
- Chapter 4: PP-OCR optimization policies;
- Chapter 5: Practice of inference and deployment;
- Chapter 6: Document structuring;
- Chapter 7: Other OCR-related algorithms such as end-to-end algorithm, data preprocessing, and data synthesis;
- Chapter 8: OCR-related datasets and data synthesis tools.

3.5 Reference

- [1] Liao, Minghui, et al. "Textboxes: A fast text detector with a single deep neural network." Thirty-first AAAI conference on artificial intelligence. 2017.
- [2] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//European conference on computer vision. Springer, Cham, 2016: 21-37.
- [3] Tian, Zhi, et al. "Detecting text in natural image with connectionist text proposal network." European conference on computer vision. Springer, Cham, 2016.

- [4] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[J]. Advances in neural information processing systems, 2015, 28: 91-99.
- [5] Zhou, Xinyu, et al. "East: an efficient and accurate scene text detector." Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2017.
- [6] Wang, Wenhai, et al. "Shape robust text detection with progressive scale expansion network." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.
- [7] Liao, Minghui, et al. "Real-time scene text detection with differentiable binarization." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 07. 2020.
- [8] Deng, Dan, et al. "Pixellink: Detecting scene text via instance segmentation." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 32. No. 1. 2018.
- [9] He K, Gkioxari G, Dollár P, et al. Mask r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2961-2969.
- [10] Wang P, Zhang C, Qi F, et al. A single-shot arbitrarily-shaped text detector based on context attended multi-task learning[C]//Proceedings of the 27th ACM international conference on multimedia. 2019: 1277-1285.
- [11] Shi, B., Bai, X., & Yao, C. (2016). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11), 2298-2304.
- [12] Star-Net Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In Advances in neural information processing systems, pages 2017–2025, 2015.
- [13] Shi, B., Wang, X., Lyu, P., Yao, C., & Bai, X. (2016). Robust scene text recognition with automatic rectification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4168-4176).
- [14] Sheng, F., Chen, Z., & Xu, B. (2019, September). NRTR: A no-recurrence sequence-to-sequence model for scene text recognition. In 2019 International Conference on Document Analysis and Recognition (ICDAR) (pp. 781-786). IEEE.
- [15] Lyu P, Liao M, Yao C, et al. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 67-83.
- [16] Soto C, Yoo S. Visual detection with context for document layout analysis[C]//Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019: 3464-3470.
- [17] Sarkar M, Aggarwal M, Jain A, et al. Document Structure Extraction using Prior based High Resolution Hierarchical Semantic Segmentation[C]//European Conference on Computer Vision. Springer, Cham, 2020: 649-666.
- [18] Kieninger T, Dengel A. A paper-to-HTML table converting system[C]//Proceedings of document analysis systems (DAS). 1998, 98: 356-365.
- [19] Siddiqui S A, Fateh I A, Rizvi S T R, et al. Deeptabstr: Deep learning based table structure recognition[C]//2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2019: 1403-1409.
- [20] Raja S, Mondal A, Jawahar C V. Table structure recognition using top-down and bottom-up cues[C]//European Conference on Computer Vision. Springer, Cham, 2020: 70-86.
- [21] Xue W, Yu B, Wang W, et al. TGRNet: A Table Graph Reconstruction Network for Table Structure Recognition[J]. arXiv preprint arXiv:2106.10598, 2021.
- [22] Ye J, Qi X, He Y, et al. PingAn-VCGroup's Solution for ICDAR 2021 Competition on Scientific Literature Parsing Task B: Table Recognition to HTML[J]. arXiv preprint arXiv:2105.01848, 2021.
- [23] Du Y, Li C, Guo R, et al. PP-OCR: A practical ultra lightweight OCR system[J]. arXiv preprint arXiv:2009.09941, 2020.

[24] Du Y, Li C, Guo R, et al. PP-OCRv2: Bag of Tricks for Ultra Lightweight OCR System[J]. arXiv preprint arXiv:2109.03144, 2021.

CHAPTER FOUR

TEXT DETECTION

Text detection is to find out the position of text in an image or video. Different from object detection, object detection is intended to solve not only the positioning problem, but also the problem of target classification.

The representation of text in images can be regarded as a kind of ‘object’, so object detection methods also fit into text detection. Find out their similarity and difference in terms of their tasks:

- Object detection: finds out the box of the target of the given image or video, and make classification;
- Text detection: finds out the text area of the given image or video, which can be a single character or a whole text line;

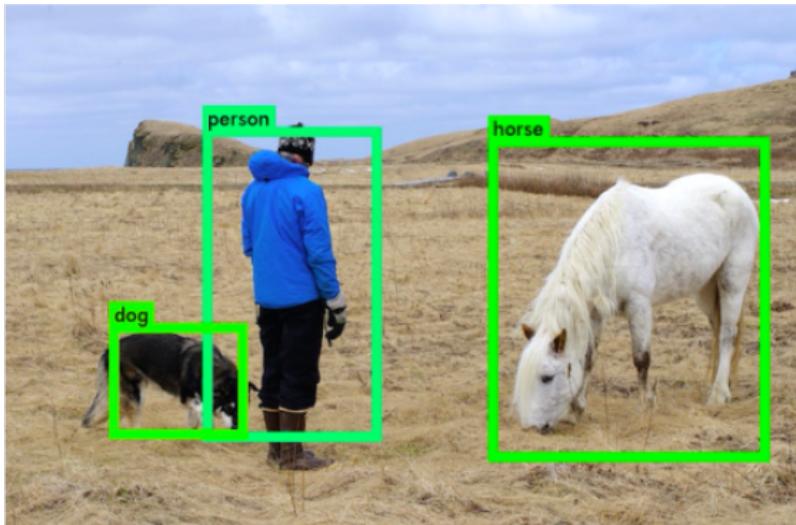


Figure 1: Schematic of object detection

代号	项目	结果	参考值	单位
ALT	谷丙转氨酶	25.6	0~40	U/L
TBIL	总胆红素	11.2	<20	umol/L
DBIL	直接胆红素	3.3	0~7	umol/L
IBIL	间接胆红素	7.9	1.5~15	umol/L
TP	总蛋白	58.9 ↓	60~80	g/L
ALB	白蛋白	35.1	33~55	g/L
GLO	球蛋白	23.8	20~30	g/L
A/G	白球比	1.5	1.5~2.5	
ALP	碱性磷酸酶	93	15~112	U/L
GGT	谷氨酰转肽酶	14.3	<50	U/L
AST	谷草转氨酶	16.3	8~40	U/L
LDH	乳酸脱氢酶	167	114~240	U/L
ADA	腺苷脱氨酶	12.6	4~24	U/L



Figure 2: Schematic diagram of text detection

Object detection and text detection both concern “location”. But the latter one does not need to classify the target, and the shape of the text is complex and diverse.

Currently, text detection usually refers to scene text detection, which encounters difficulties for:

1. Diversity of texts in natural scenes: text detection may be affected by text color, size, font, shape, direction, language, and length.
2. Complexity of backgrounds and interference: text detection may be affected by image distortion, blurring, low resolution, shadow, brightness and other factors.
3. Dense or overlapped texts.
4. Partial identity of texts: a small part of a text line can also be considered as an individual text.



Figure 3: Text detection scenes

Many text detection algorithms based on deep learning have emerged to solve the problems of text detection in natural scenes. These methods can be divided into regression-based and segmentation-based text detection methods.

The next section will briefly introduce the classic text detection algorithms based on deep learning technology.

4.1 Introduction to Text Detection Methods

In recent years, there are growing deep learning-based text detection algorithms. These methods can be roughly classified into two categories:

1. Regression-based methods
2. Segmentation-based methods

This section selects The methods commonly used from 2017 to 2021 are selected here and classified into the above two types of methods in the following table:

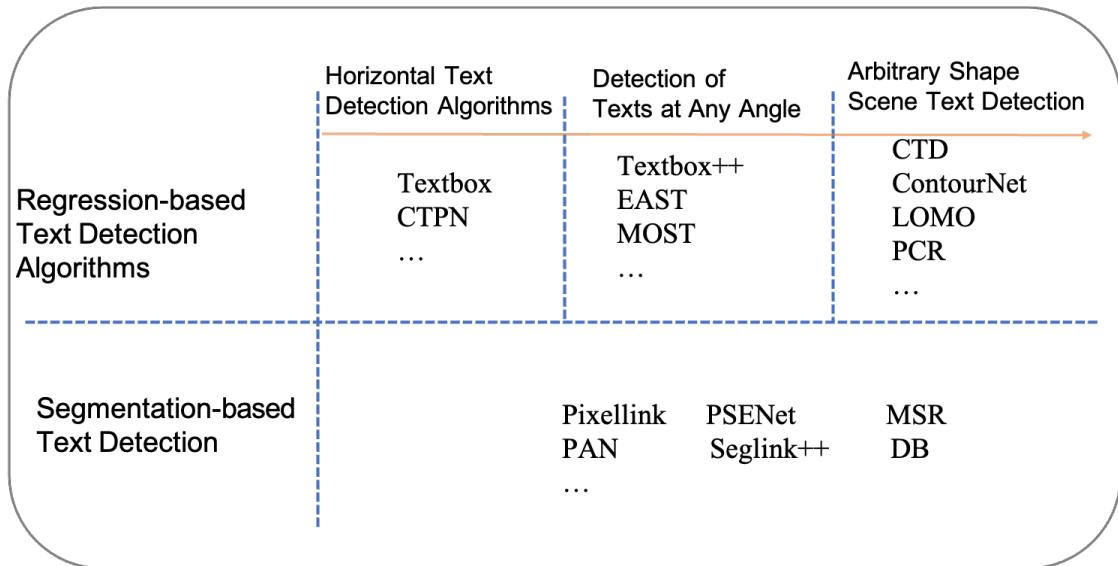


Figure 4: Text detection algorithms

4.1.1 Regression-based Text Detection

Regression-based algorithms are similar to object detection algorithms. And there are only two parts in text detection methods: the text of an image is the target to be detected, and the rest is the background.

Horizontal Text Detection Algorithms

In the early days, the methods based on deep learning are modified object detection algorithms, supporting horizontal text detection. For example, the TextBoxes algorithm is improved from the SSD algorithm, and the CTPN from Fast-RCNN, the two-stage object detection algorithm.

The TextBoxes[1] algorithm is adjusted according to the one-stage target detector SSD. The default text box is changed to a quadrilateral that fits the direction and aspect ratio of the text. The algorithm also provides an end-to-end training text detection method without complicated Post-processing.

- The pre-selection box is larger in the aspect ratio.
- The convolution kernel has been changed into 1x51 from 3x3, more suitable for detecting long texts.
- Multi-scale input is adopted

Based on the Fast-RCNN algorithm, CTPN[3] expands the RPN module and designs a CRNN-based module to enable the network to detect text sequences from convolutional features. The two-stage method can locate features more accurately through ROI Pooling. But TextBoxes and CTPN can only detect horizontal texts.

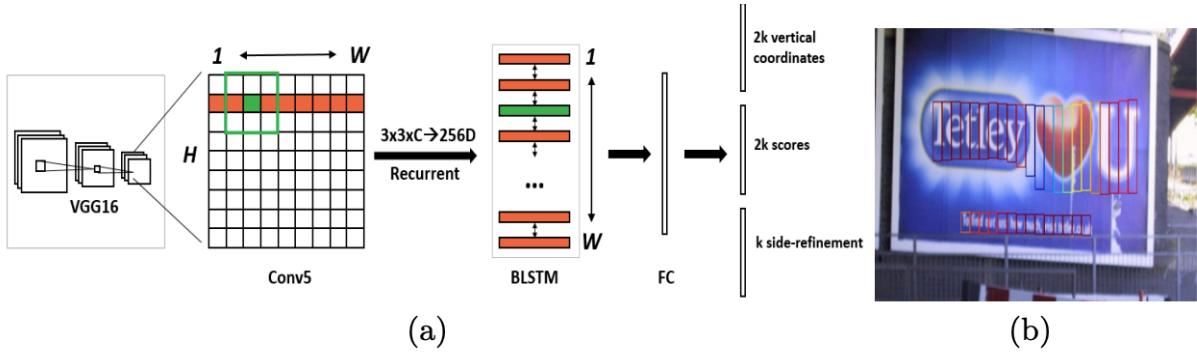


Figure 6: Frame diagram of CTPN

Detection of Texts at Any Angle

TextBoxes++[2] is modified from TextBoxes, and can detect texts at any angle. In terms of structure, unlike TextBoxes, TextBoxes++ is designed to detect multi-angle texts. First, it modifies the aspect ratio of the preselection box and adjusts the aspect ratio to 1, 2, 3, 5, 1/2, 1/3, and 1/5. Second, it changes the $1 * 5$ of convolution kernel to $3 * 5$ to better learn the characteristics of the tilted text. Finally, it outputs the represented information of the rotating box.

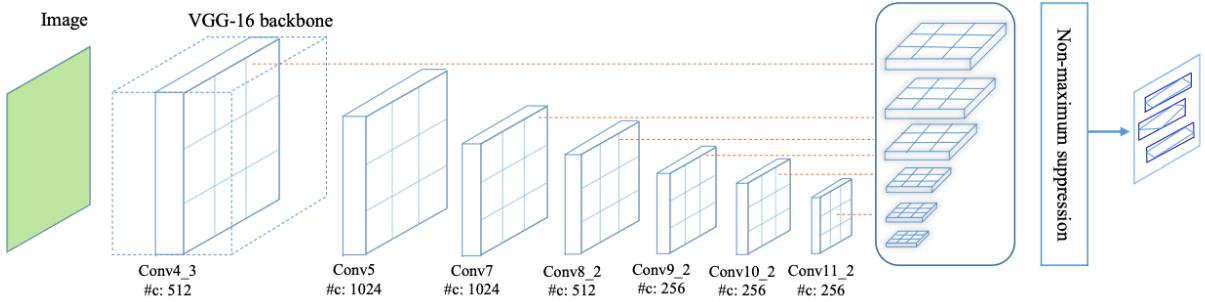


Figure 7: Frame diagram of TextBoxes++

EAST [4] adopts a two-stage text detection method for the location of tilted texts, including FCN feature extraction and NMS. EAST proposes a new text detection pipeline structure, which can conduct end-to-end training and detect texts in any orientation. It is also simple in structure and excellent in performance. FCN supports the output of inclined rectangular and horizontal boxes whose output formats can be decided by users.

- If the output shape is RBox, output the rotation angle of the box and the AABB text shape, AABB represents shifts of the box on the up, down, left, and right sides. RBox can rotate rectangular texts.
- If the output detection box is a four-point box, the last dimension of the output should be in 8 numbers, which means the text shift from the four vertices of the quadrilateral. This output method can predict texts in the shape of irregular quadrilaterals.

Text boxes output by FCN is redundant. For example, the box generated by the adjacent pixels of a text area is highly overlapped. But those derived from the same text do not go this way. Therefore, EAST proposes to merge the prediction boxes in rows, and then filter the remaining quads with the original NMS.

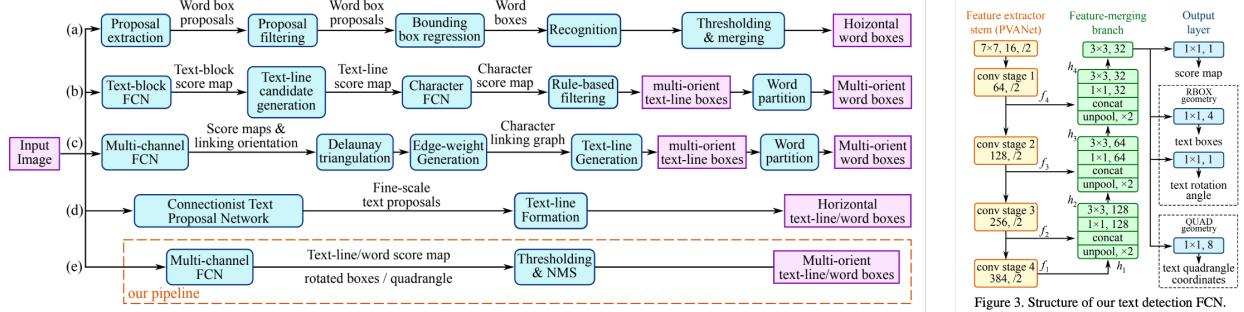


Figure 8: Frame diagram of EAST

MOST [15] has put forward the TFAM module to dynamically adjusts the receptive field according to the coarse-grained detection results, and also proposed PA-NMS to combine reliable detection and prediction results based on the location. In addition, the Instance-wise IoU loss function has also been put forward during training, which is used to balance training to handle text of different scales. This method can be combined with the EAST and thus obtains effects and performance in detecting texts with extreme aspect ratios and different scales.

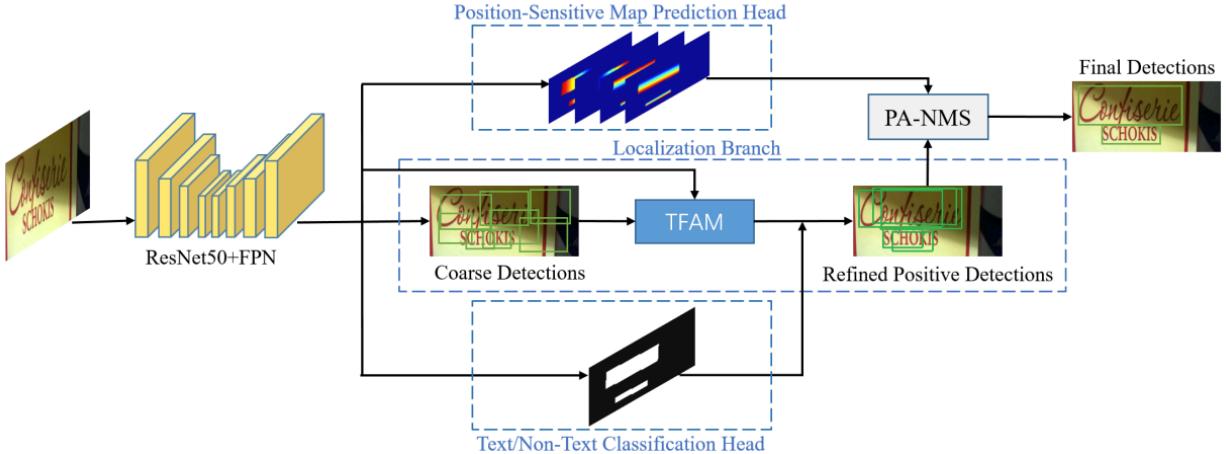


Figure 9: MOST frame diagram

Arbitrary Shape Scene Text Detection

Among ideas to use regression to detect curved texts, a simple one is to describe the polygons with boundaries of curved texts with multi-point coordinates, and then predict the vertex coordinates of the polygons.

CTD [6] has made a proposal to predict the polygons curved texts with 14 vertices. The network uses the Bi-LSTM [13] layer to refine the predicted coordinates of the vertices and detect curved texts based on regression.

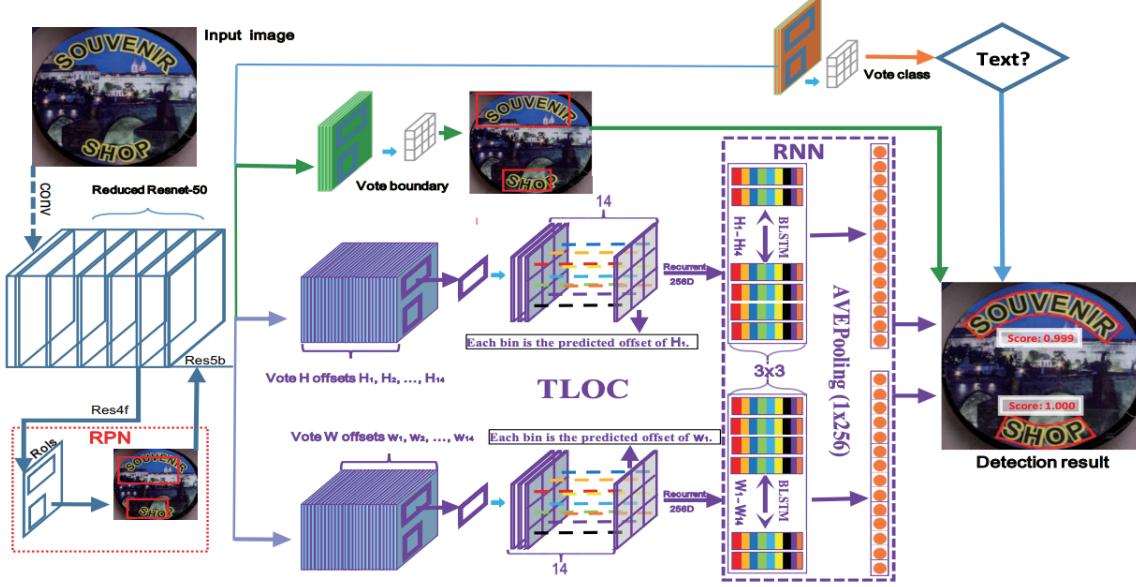


Figure 10: Frame diagram of CTD

For the detection of long texts and curved texts, LOMO [19] has proposed an iterative method for the optimization of text localization features to obtain more accurate text locations. LOMO consists of a direct regressor (DR), an iterative refinement module (IRM), and a shape expression module (SEM). Text areas are generated by DR, then IRM refines text localization features iteratively, and finally an SEM is introduced to predict the text region, the text center line and border offsets. Iterative optimization of text features can better solve long text localization and locate more accurate text regions.

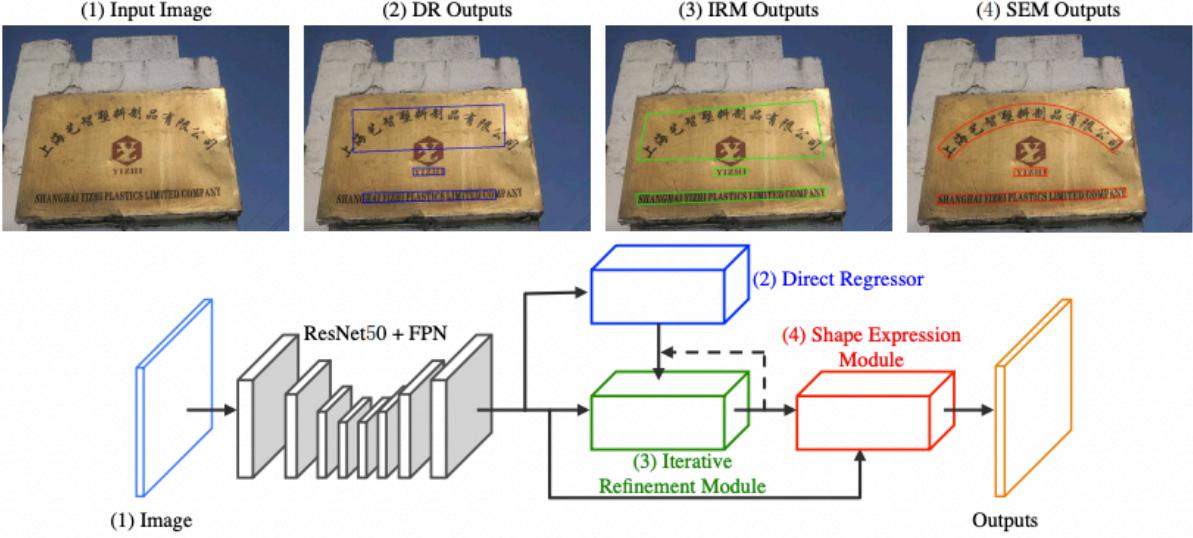


Figure 11: Frame diagram of LOMO

Contournet [18] suggests modeling text contour points to get a detection box of the curved text. First, an Adaptive-RPN is proposed to generate the proposal of the text region. Then, a Local Orthogonal Texture-aware Module (LOTM) learns horizontal and vertical texture features, and represents them with contour points. Finally, considering the feature responses in two orthogonal directions, the Point Re-Scoring algorithm is adopted to filter the prediction of the strong unidirectional or weakly orthogonal activation, and assure that the text contour can be represented with a group of high-quality contour points.

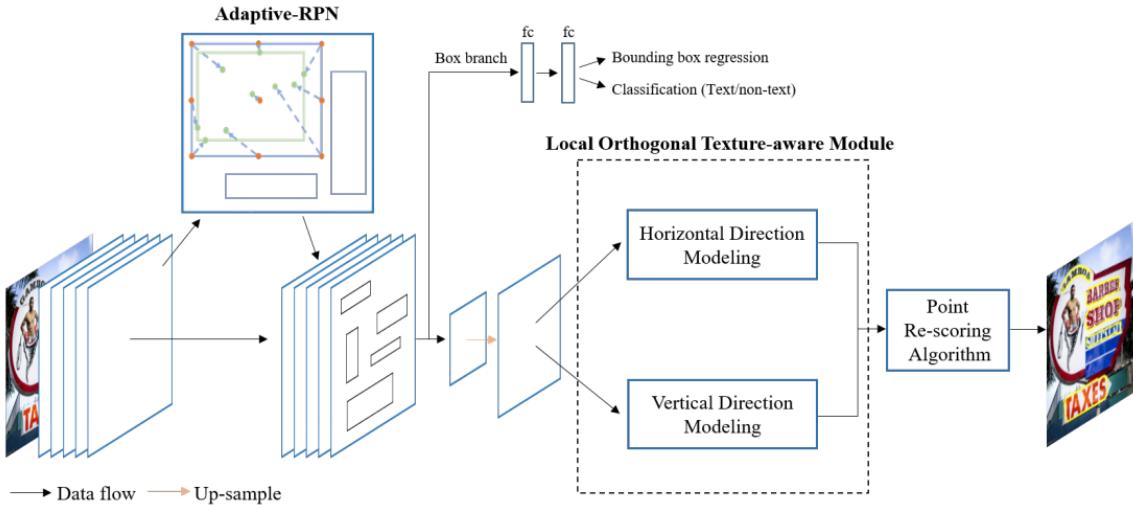


Figure 12: Frame diagram of Contournet

PCR [14] has advised to detect curved texts with progressive coordinate regression. The solution has three stages. First, detect the rough text region and get a text box. Second, the Contour Localization Mechanism is designed to predict the corners of the smallest bounding box of the text. Finally, pile CLM modules and RCLM modules to predict the curved text. The progressive contour regression method can help get more refined text representations, which can not only protect the coordinate regression from being affected by redundant noise, but also locate the text region more accurately.

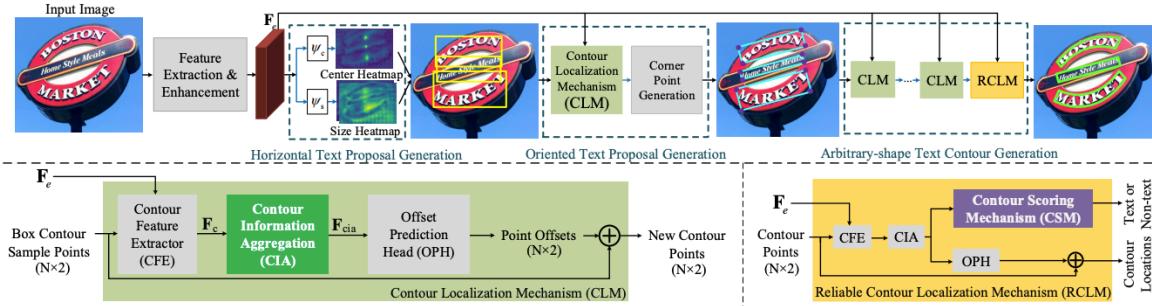


Figure 13: Frame diagram of PCR

4.1.2 Segmentation-based Text Detection

Although the regression-based methods work well in text detection, it is often difficult for them to outline a curved text with a smooth curve and their models are more complex and not superior in performance. Therefore, researchers have proposed a text segmentation method based on image segmentation. First, classify the pixels, determine whether each pixel matches one text target. Then obtain a probability graph of the text region, and get the curve outlining the segmented text through post-processing.

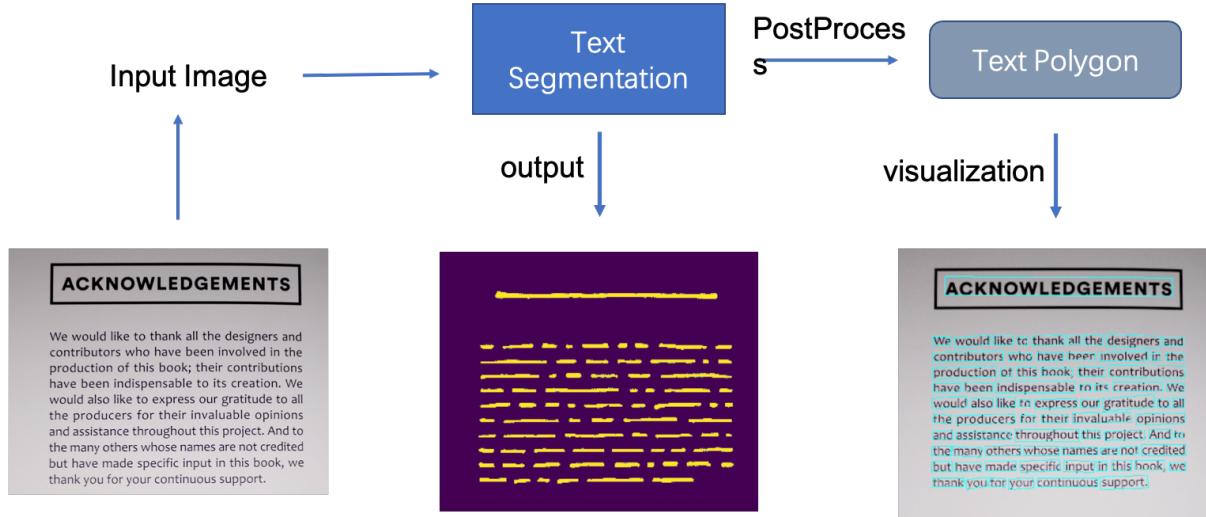


Figure 14: Schematic of text segmentation algorithm

Such methods are usually based on segmentation to achieve text detection, and segmentation-based methods have natural advantages for text detection with irregular shapes. The main idea of the segmentation-based text detection method is to obtain the text area in the image through the segmentation method, and then use OpenCV, polygon and other post-processing to obtain the minimum enclosing curve of the text area.

Pixellink [7] uses segmentation to detect texts. The segmented object is a text region. The pixels of the same text line (word) are linked together to segment the text, and the text box is extracted from segmentation without position regression. And the text detection result is as good as that of the regression-based ones. However, there is a problem with the segmentation-based method. For texts in similar positions, the segmented regions easily “adhere” to one another. Wu, Yue et al. [8] have proposed to separate texts and learn positions of the text boundaries to better distinguish text regions. In addition, Tian et al. [9] have proposed to map the pixels of the same text to the mapping space, where distances of the mapped vectors of the same text are close, and those of different texts stay far away from each other.

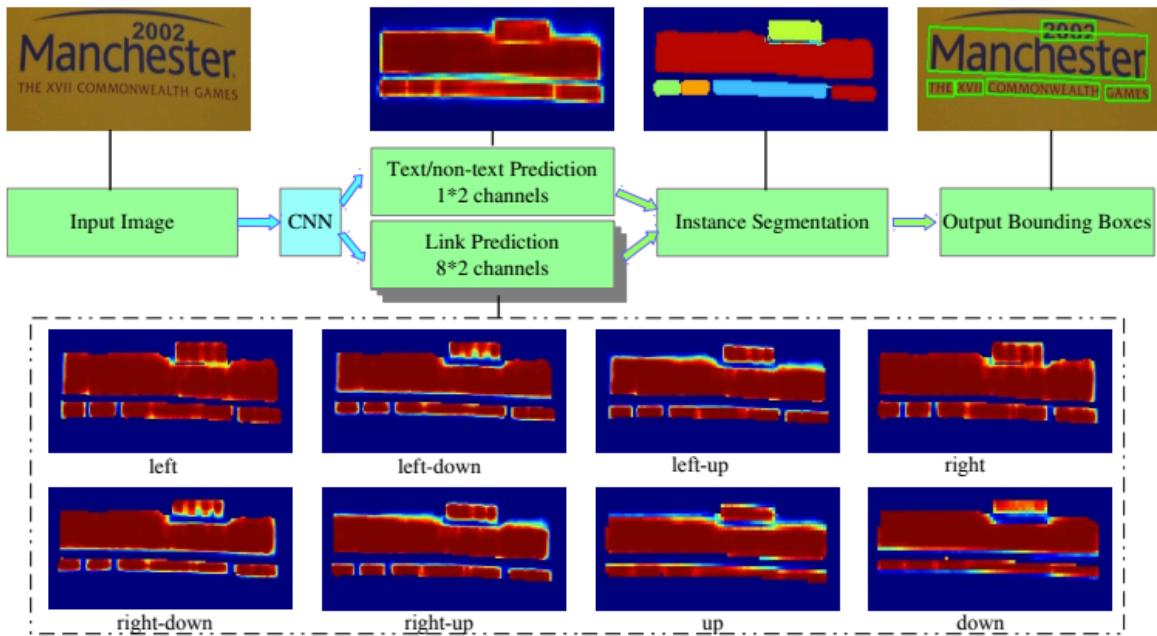


Figure 15: Frame diagram of PixelLink

For the multi-scale issue of text detection, MSR [20] recommends to extract multi-scale features from the same image, then merge these features and upsample them to the original image size. Finally, the multi-scale network predicts the text center area and the offsetX and offsetY of each point in the center area to the nearest boundary point. Finally, the coordinate set of the text region contour can be obtained.

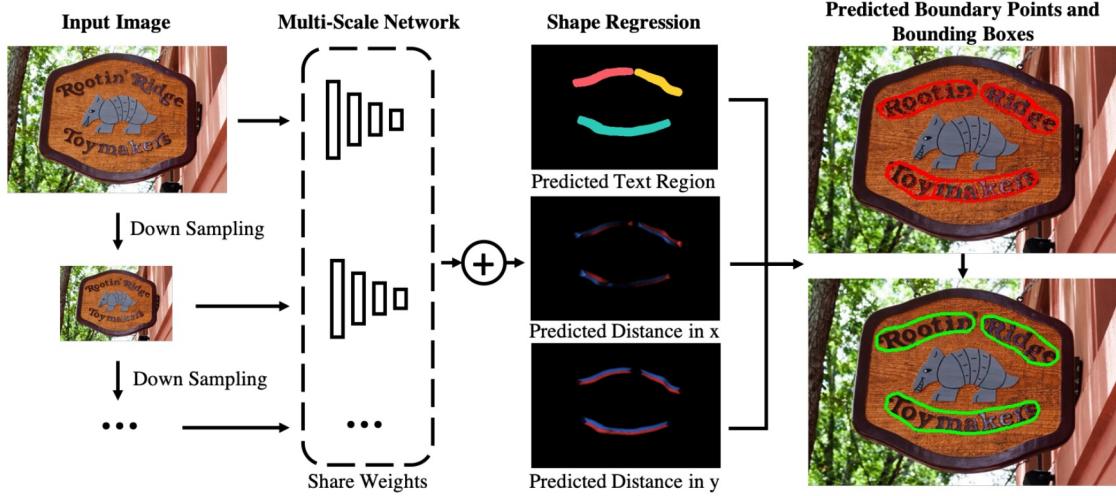


Figure 16: Frame diagram of MSR

Considering that segmentation-based text algorithms have difficulties in distinguishing adjacent texts, PSENet [10] adopts a novel progressive scale expansion network to learn segmented text regions, predict text regions with different shrinkage ratios, and expand the detected text regions. The essence of this method is a variant of the method of learning boundaries, effectively detecting adjacent texts of any shape.

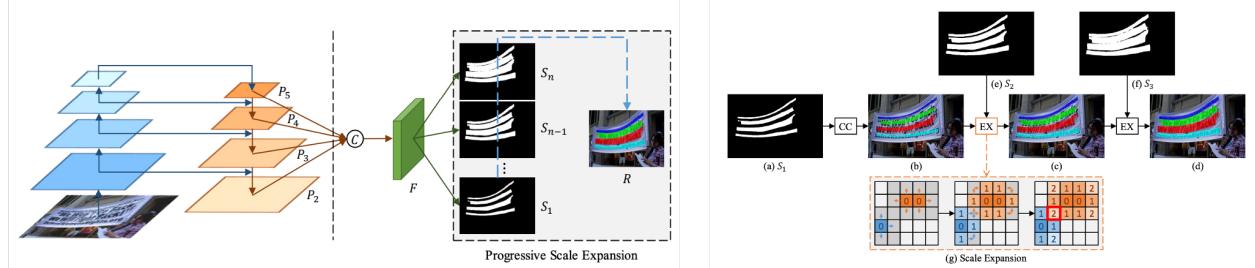


Figure 17: Frame diagram of PSENet

Assume that three kernels of different scales are used in PSENet post-processing, as shown in the above figure s_1 , s_2 , and s_3 . First, the minimum kernel s_1 calculates the connected domain of the text segmented area, gets (b), then expands the connected domain all around, and classifies the pixels of s_2 instead of s_1 in the expanded area. When there are conflicts, it will repeat the expansion under the principle of “first come first served”, and finally all text lines can be segmented into individual regions.

For curved texts and dense texts, Seglink++ [17] proposes a characterization of the attraction and repulsion between text segments, uses a minimum spanning tree algorithm to combine segments to get the text detection box and an instance-aware loss function to enable its end-to-end training.

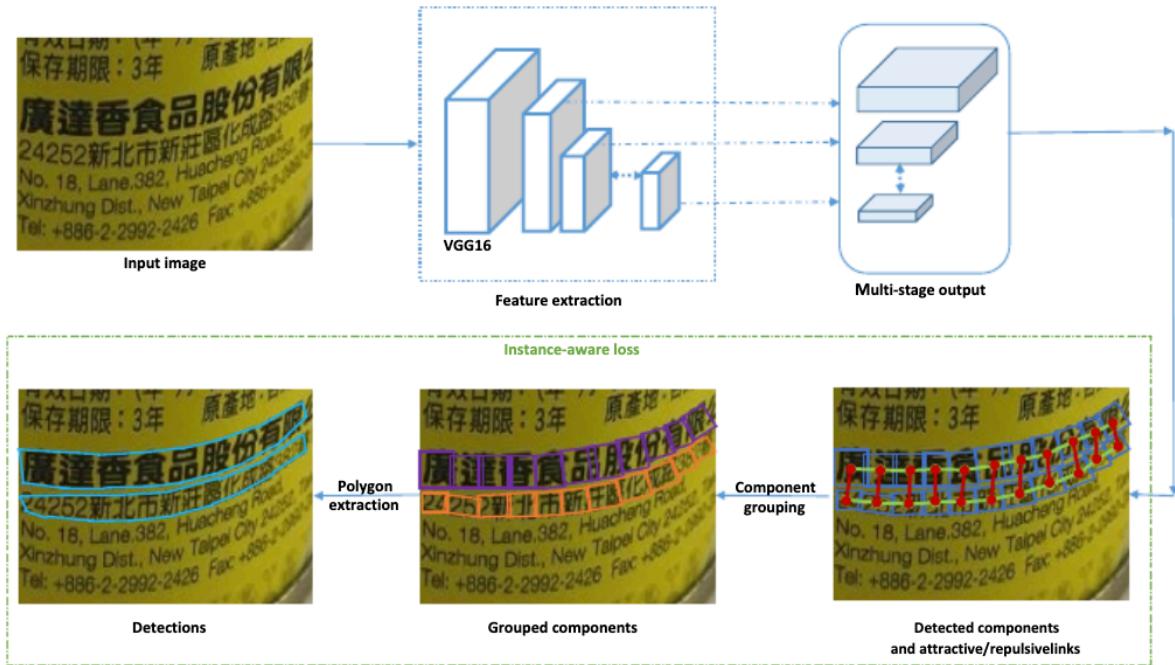


Figure 18: Frame diagram of Seglink++

Although segmentation realizes curved text detection, but complex post-processing logic and prediction speed also need to be optimized.

PAN [11] aims at speeding up text detection and prediction by improving the network design and post-processing to enhance the performance of the algorithm. First, PAN uses the lightweight ResNet18 as the backbone, and designs the lightweight feature enhancement module FPEM and feature fusion module FFM to improve features extracted by the backbone. In terms of post-processing, PAN adopts pixel clustering to merge pixels whose distance from the kernel is less than the threshold d around the predicted text center (kernel). This solution can guarantee both accuracy and fast prediction speed.

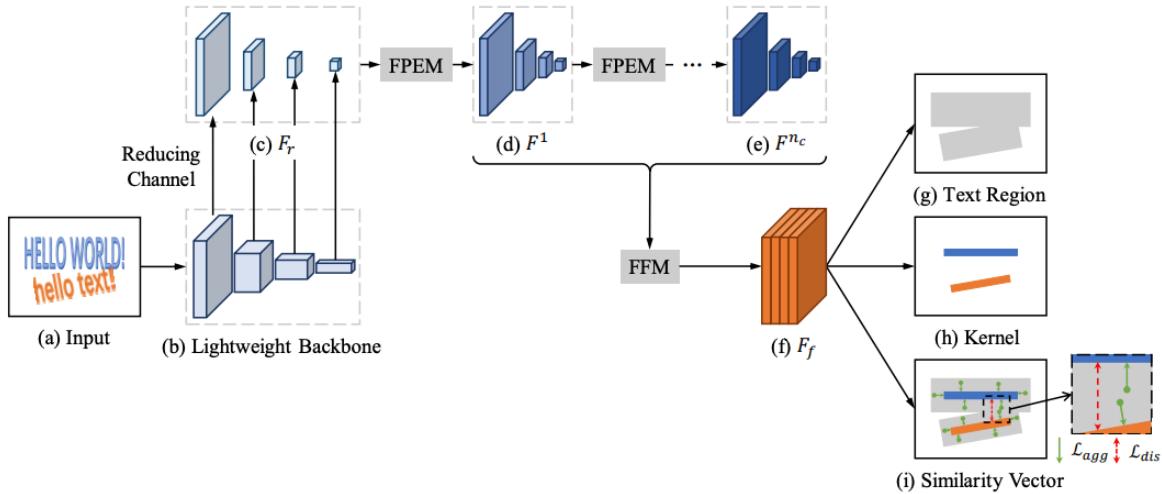


Figure 19: Frame diagram of PAN

DBNet [12] is aimed at optimizing the time-consuming post-processing in segment-based methods that require to use the threshold for binarization. It proposes a learnable threshold and a binarization function similar to the step function to ensure the segmentation network learns the threshold of segmentation end-to-end in the training. The automatic adjustment of the threshold not only improves accuracy, but also simplifies post-processing and improves the performance of text detection.

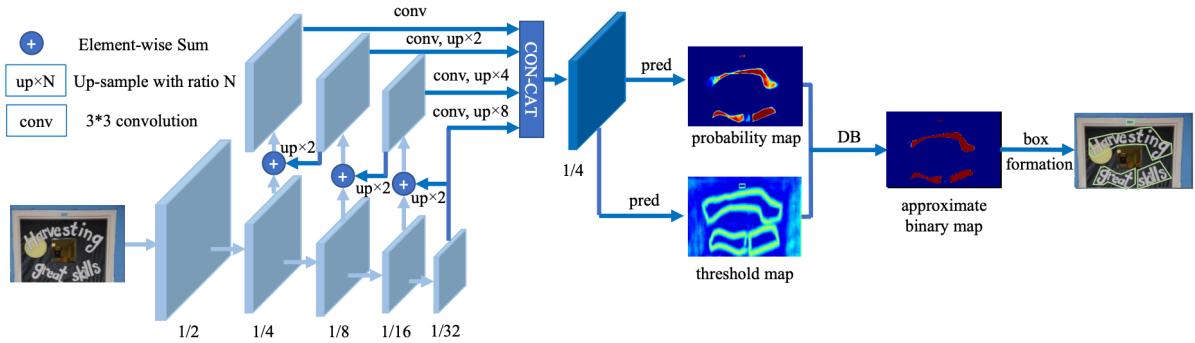


Figure 20: Frame diagram of DB

FCENet [16] expresses the curve outlining the text with Fourier transform parameters. Since the Fourier coefficient can theoretically fit any closed curve, FCENet designs a suitable model to predict the representation of the outlining curve in arbitrary shapes based on Fourier transform. In this way, the detection accuracy of highly curved texts in natural scenes can be improved.

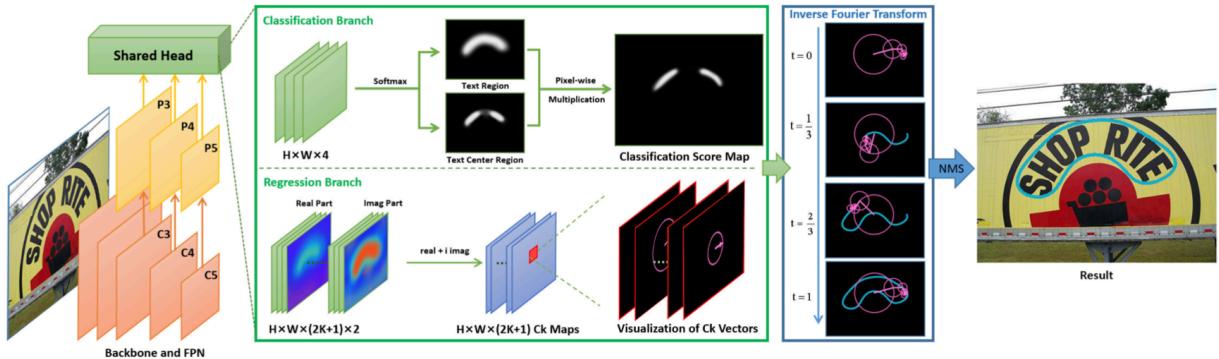


Figure 21: Frame diagram of FCENet

4.2 Practice of a Text Detection Algorithm DBNet

This section will introduce how to use PaddleOCR to complete the training and implementation of DB algorithm of text detection, including:

1. Quickly calling PaddleOCR package to try text detection
2. Understanding the principle of DB algorithm
3. Learning procedures of building the text detection model
4. Learning the training of the text detection model

Note: paddleocr refers to PaddleOCR whl package.

4.2.1 Quick Start

This section will take `paddleocr` as an example to introduce how to quickly implement text detection in three steps.

1. Install [PaddleOCR whl package](#)
2. Issue a command to run the DB algorithm to get the detection result
3. Visualize text detection result

Install PaddleOCR whl package

```
!pip install --upgrade pip  
!pip install paddleocr
```

Issue a command to implement text detection

At the first implementation, `paddleocr` will automatically download and operate [PP-OCRV2](#) lightweight model in PaddleOCR's github repository.

Inputting the image `./12.jpg` in the installed `paddleocr` will get the following result:



Figure 1: ./12.jpg

```
[[79.0, 555.0], [398.0, 542.0], [399.0, 571.0], [80.0, 584.0]]  
[[21.0, 507.0], [512.0, 491.0], [513.0, 532.0], [22.0, 548.0]]  
[[174.0, 458.0], [397.0, 449.0], [398.0, 480.0], [175.0, 489.0]]  
[[42.0, 414.0], [482.0, 392.0], [484.0, 428.0], [44.0, 450.0]]
```

The inference result has four text boxes, each of which contains four coordinates, namely a coordinate cluster of each text box, and it is arranged in clockwise order from the upper left.

The paddleocr command calls the text detection model to predict the image ./12.jpg ,which is shown as follows:

Dive into OCR

```
import os
# Modify the default directory where Aistudio code runs to /home/aistudio/
os.chdir("/home/aistudio/")
```

```
# ``--image_dir`` point to the image path to be predicted; ``--rec`` false means no  
+recognition is used, only text detection is performed  
!paddleocr --image_dir ./12.jpg --rec false
```

```
[2021/12/22 21:07:21] root INFO: *****/.12.jpg*****
[2021/12/22 21:07:23] root INFO: [[79.0, 555.0], [398.0, 542.0], [399.0, 571.0], [80.0, 584.0]]
[2021/12/22 21:07:23] root INFO: [[21.0, 507.0], [512.0, 491.0], [513.0, 532.0], [22.0, 548.0]]
[2021/12/22 21:07:23] root INFO: [[174.0, 458.0], [397.0, 449.0], [398.0, 480.0], [175.0, 489.0]]
[2021/12/22 21:07:23] root INFO: [[42.0, 414.0], [482.0, 392.0], [484.0, 428.0], [44.0, 450.0]]
```

In addition, paddleocr also provides a code calling method:

```
# 1. Import PaddleOCR class from paddleocr
from paddleocr import PaddleOCR

# 2. Declare the PaddleOCR class
ocr = PaddleOCR()
img_path ='./12.jpg'
# 3. Perform inference
result = ocr.ocr(img_path, rec=False)
print(f"The predicted text box of {img_path} are follows.")
print(result)
```

The predicted text box of ./12.jpg are follows.

```
[[[79.0, 555.0], [398.0, 542.0], [399.0, 571.0], [80.0, 584.0]], [[21.0, 507.0], [512.0, 491.0], [513.0, 532.0], [22.0, 548.0]], [[174.0, 458.0], [397.0, 449.0], [398.0, 480.0], [175.0, 489.0]], [[42.0, 414.0], [482.0, 392.0], [484.0, 428.0], [44.0, 450.0]]]]
```

Visualize results of text detection inference

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
# When using matplotlib.pyplot to draw in the notebook, you need to add this command
# to display
%matplotlib inline

# 4. Visual inspection results
image = cv2.imread(img_path)
boxes = [line[0] for line in result]
for box in result:
    box = np.reshape(np.array(box), [-1, 1, 2]).astype(np.int64)
    image = cv2.polylines(np.array(image), [box], True, (255, 0, 0), 2)

# Draw the read picture
```

(continues on next page)

(continued from previous page)

```
plt.figure(figsize=(10, 10))
plt.imshow(image)
```

4.2.2 Detailed Implementation of DB Text Detection Algorithm

The Principle of DB Algorithm

DB is a segmentation-based text detection algorithm, using a Differentiable Threshold Differentiable Binarization module (DB module) to distinguish the text region from the background with a dynamic threshold.

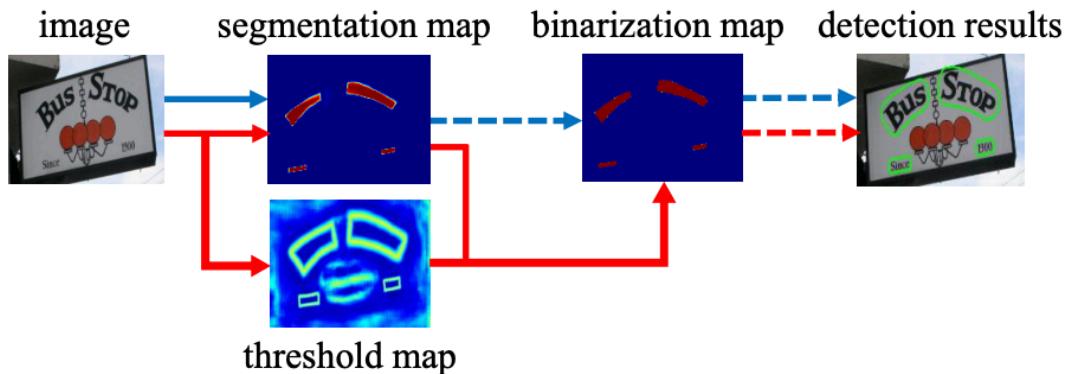


Figure 2: Traditional pipeline (blue flow) and our pipeline (red flow). Dashed arrows are the inference only operators; solid arrows indicate differentiable operators in both training and inference.

Figure 2: The difference between DB model and others

The blue arrows in the picture show the process of the common segmentation-based text detection algorithms. This kind of methods use a fixed threshold to gain the binarized segmentation map after segmentation, and then adopt heuristic algorithms such as pixel clustering are used to get the text region.

The red arrows in the picture show the flow of the DB algorithm. The biggest difference from common solutions is that DB has a threshold map, and it will predict the threshold at every pixel point of the picture through the network, instead of designating a fixed value. So it can better distinguish the text background and the foreground.

The DB algorithm is advantageous for:

1. Its algorithmic structure is simple and free of tedious post-processing
2. Its open-source data have good accuracy and performance

After gaining the probability map, the traditional algorithm based on image segmentation will set all the pixels below t to 0 and to 1 otherwise. The formula is:

$$B_{i,j} = \begin{cases} 1, & \text{if } P_{i,j} \geq t, \\ 0, & \text{otherwise.} \end{cases}$$

But the standard binarization method is not differentiable, thus making the network unable to be trained end-to-end. To solve this problem, the DB algorithm uses Differentiable Binarization (DB), which approximates the step function of standard binarization. It uses another formula:

$$\hat{B} = \frac{1}{1 + e^{-k(P_{i,j}-T_{i,j})}}$$

P refers to the probability map above, T refers to the threshold map, and k is the gain factor which is set to 50 under a rule of thumb in the experiment. The **Figure 2(a)** below shows the differences between standard binarization and differentiable binarization.

When using cross-entropy loss, the loss of positive and negative samples is l_+ and l_- respectively:

$$l_+ = -\log\left(\frac{1}{1 + e^{-k(P_{i,j}-T_{i,j})}}\right)$$

$$l_- = -\log\left(1 - \frac{1}{1 + e^{-k(P_{i,j}-T_{i,j})}}\right)$$

Inputting x to take the partial derivative may result in:

$$\frac{\delta l_+}{\delta x} = -kf(x)e^{-kx}$$

$$\frac{\delta l_-}{\delta x} = -kf(x)$$

It can be found that the gain factor will magnify the gradient of the error prediction, thereby optimizing the model to obtain better results. In **Figure 2(b)**, the part of $x < 0$ represents the case where positive samples are predicted to be negative samples. It can be seen that the gain factor k magnifies the gradient. **Figure 2(c)** shows $x > 0$ which refers to the case where negative samples are predicted to be positive samples, and the gradient is also magnified.

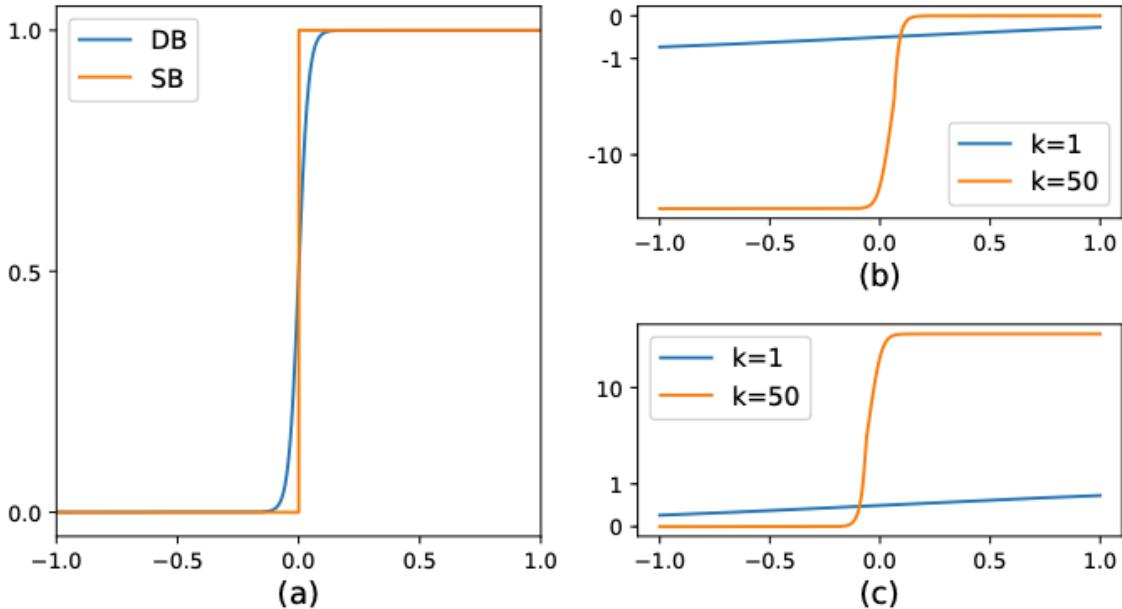


Figure 3: Schematic of DB algorithm

The overall structure of the DB algorithm is:

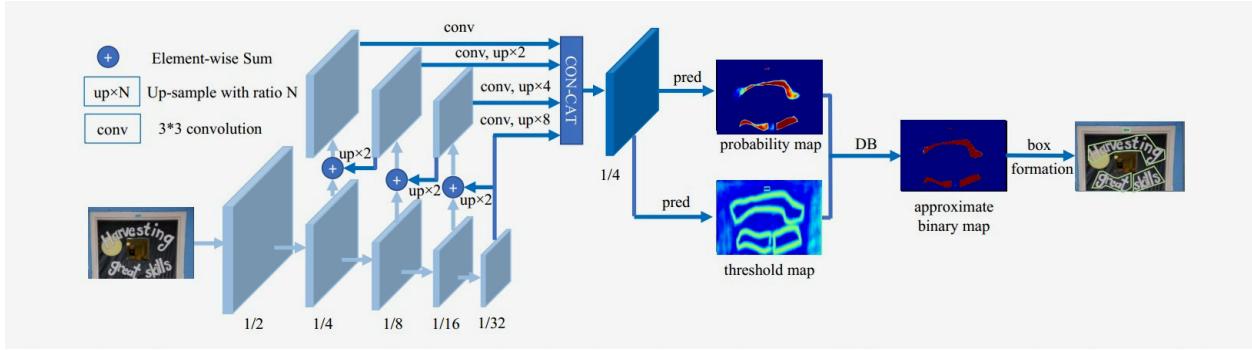


Figure 4: Schematic of DB model network structure

Features of the input image is extracted through the network Backbone and FPN, and they are cascaded together to get a feature whose size is a quarter of the original image. Then, the convolutional layer is used to obtain the inference probability map and the threshold map, and then get the outlining curve through the post-processing of DB.

Building the DB Text Detection Model

The DB text detection model consists of three parts:

- Backbone network for extraction of image features
- FPN (Feature Pyramid Network) for feature enhancement
- Head network for calculation of the probability map of the text region

In this section, PaddlePaddle will implement the three network modules and build a complete network.

Backbone Network

The backbone of the DB network uses an image classification network. ResNet50 is used in the paper. In this section, to speed up the training, the experiment will use the MobileNetV3 large as the backbone.

```
# For the first run, you need to open the comment on the next line and download the
# PaddleOCR code
#!git clone https://gitee.com/paddlepaddle/PaddleOCR
import os
# Modify the default directory where the code runs to /home/aistudio/PaddleOCR
os.chdir("/home/aistudio/PaddleOCR")
# Install PaddleOCR third-party dependencies
!pip install --upgrade pip
!pip install -r requirements.txt

# https://github.com/PaddlePaddle/PaddleOCR/blob/release%2F2.4/ppocr/modeling/
# backbones/det_mobilenet_v3.py
from pocr.modeling.backbones.det_mobilenet_v3 import MobileNetV3
```

If you want to use ResNet as Backbone, you can select ResNet in the PaddleOCR code, or select your own backbone model in PaddleClas.

DB's Backbone is used to extract the multi-scale features of the image, as shown in the following code. Assume the input shape is [640, 640], and the output of the backbone network has four features whose shapes are [1, 16, 160, 160], [1, 24, 80, 80], [1, 56, 40, 40], and [1, 480, 20, 20]. These features will be input to the FPN network for further enhancement.

```
import paddle

fake_inputs = paddle.randn([1, 3, 640, 640], dtype="float32")

# 1. Declare Backbone
model_backbone = MobileNetV3()
model_backbone.eval()

# 2. Perform inference
outs = model_backbone(fake_inputs)

# 3. Print network structure
print(model_backbone)

# 4. Print out shapes of the features
for idx, out in enumerate(outs):
    print("The index is ", idx, "and the shape of output is ", out.shape)
```

FPN

Feature Pyramid Network, or FPN, is commonly used in convolutional networks to efficiently extract features of each dimension of an image.

```
# https://github.com/PaddlePaddle/PaddleOCR/blob/release%2F2.4/ppocr/modeling/necks/
→db_fpn.py

import paddle
from paddle import nn
import paddle.nn.functional as F
from paddle import ParamAttr

class DBFPN(nn.Layer):
    def __init__(self, in_channels, out_channels, **kwargs):
        super(DBFPN, self).__init__()
        self.out_channels = out_channels

        # For detailed implementation of DBFPN, refer to: https://github.com/
→PaddlePaddle/PaddleOCRblob/release%2F2.4/ppocr/modeling/necks/db_fpn.py

    def forward(self, x):
        c2, c3, c4, c5 = x

        in5 = self.in5_conv(c5)
        in4 = self.in4_conv(c4)
        in3 = self.in3_conv(c3)
        in2 = self.in2_conv(c2)

        # Feature upsampling
        out4 = in4 + F.upsample(
            in5, scale_factor=2, mode="nearest", align_mode=1) # 1/16
        out3 = in3 + F.upsample(
            out4, scale_factor=2, mode="nearest", align_mode=1) # 1/8
        out2 = in2 + F.upsample(
            out3, scale_factor=2, mode="nearest", align_mode=1) # 1/4

        p5 = self.p5_conv(in5)
        p4 = self.p4_conv(out4)
```

(continues on next page)

(continued from previous page)

```

p3 = self.p3_conv(out3)
p2 = self.p2_conv(out2)

# Feature upsampling
p5 = F.upsample(p5, scale_factor=8, mode="nearest", align_mode=1)
p4 = F.upsample(p4, scale_factor=4, mode="nearest", align_mode=1)
p3 = F.upsample(p3, scale_factor=2, mode="nearest", align_mode=1)

fuse = paddle.concat([p5, p4, p3, p2], axis=1)
return fuse

```

The input of the FPN is the output of the backbone, and the height and width of the output feature image are a quarter of the original image in size. Assuming that the shape of the input image is [1, 3, 640, 640], the height and width of output feature in the FPN are [160, 160].

```

import paddle

# 1. Import DBFPN from PaddleOCR
from ppocr.modeling.necks.db_fpn import DBFPN

# 2. Obtain Backbone network output results
fake_inputs = paddle.randn([1, 3, 640, 640], dtype="float32")
model_backbone = MobileNetV3()
in_channels = model_backbone.out_channels

# 3. Declare FPN network
model_fpn = DBFPN(in_channels=in_channels, out_channels=256)

# 4. Print FPN network
print(model_fpn)

# 5. Calculate FPN result output
outs = model_backbone(fake_inputs)
fpn_outs = model_fpn(outs)

# 6. Print the shape of the FPN output feature
print(f"The shape of fpn outs {fpn_outs.shape}")

```

Head Network

Acquire the text region probability map, the text region threshold map and the binary map through calculation.

```

import math
import paddle
from paddle import nn
import paddle.nn.functional as F
from paddle import ParamAttr

class DBHead(nn.Layer):
    """
        Differentiable Binarization (DB) for text detection:
        see https://arxiv.org/abs/1911.08947
        args:
            params(dict): super parameters for build DB network
    """

```

(continues on next page)

(continued from previous page)

```

def __init__(self, in_channels, k=50, **kwargs):
    super(DBHead, self).__init__()
    self.k = k

    # For detailed implementation of DBHead, refer to: https://github.com/
    ↪PaddlePaddle/PaddleOCR/blob/release%2F2.4/ppocr/modeling/heads/det_db_head.py

def step_function(self, x, y):
    # Differentiable binarization can be realized, and text segmentation
    ↪binarization graph is calculated with the probability map and the threshold map
    return paddle.reciprocal(1 + paddle.exp(-self.k * (x - y)))

def forward(self, x, targets=None):
    shrink_maps = self.binarize(x)
    if not self.training:
        return {'maps': shrink_maps}

    threshold_maps = self.thresh(x)
    binary_maps = self.step_function(shrink_maps, threshold_maps)
    y = paddle.concat([shrink_maps, threshold_maps, binary_maps], axis=1)
    return {'maps': y}

```

The DB Head network will perform up-sampling on the basis of the FPN feature, and map the size of the feature from a quarter to the same size as the original image.

```

# 1. Import DBHead from PaddleOCR
from ppocr.modeling.heads.det_db_head import DBHead
import paddle

# 2. Calculate DBFPN network output results
fake_inputs = paddle.randn([1, 3, 640, 640], dtype="float32")
model_backbone = MobileNetV3()
in_channels = model_backbone.out_channels
model_fpn = DBFPN(in_channels=in_channels, out_channels=256)
outs = model_backbone(fake_inputs)
fpn_outs = model_fpn(outs)

# 3. Declare Head network
model_db_head = DBHead(in_channels=256)

# 4. Print DBhead network
print(model_db_head)

# 5. Calculate the output of the Head network
db_head_outs = model_db_head(fpn_outs)
print(f"The shape of fpn outs {fpn_outs.shape}")
print(f"The shape of DB head outs {db_head_outs['maps'].shape}")

```

4.2.3 Training the DB Text Detection Model

DB text detection algorithm is available on PaddleOCR, as well as two backbone networks, MobileNetV3 and ResNet50_vd. You can select the configuration file and start training according to your needs.

This section takes DB detection model whose backbone network is the icdar15 dataset or MobileNetV3 (the configuration used in the ultra-lightweight model) as an example to introduce how to train, evaluate and test text detection models of PaddleOCR.

Data Preparation

This experiment selects ICDAR2015, the most well-known and commonly-used dataset in Scene Text Detection and Recognition. The schematic of the ICDAR2015 dataset is shown below:



Figure 5: Schematic of the ICDAR2015 dataset

The ICDAR2015 dataset has been downloaded in this project and stored in /home/aistudio/data/data96799. You can run the following command to decompress the data set, or download it from the link.

```
! cd ~/data/data96799/ && tar xf icdar2015.tar
```

After running the command, there are two folders and two files in ~/train_data/icdar2015/text_localization :

```
~/train_data/icdar2015/text_localization
└─ Training data of icdar_c4_train_imgs/ icdar dataset
    └─ ch4_test_images/ test data of icdar dataset
    └─ train_icdar2015_label.txt icdar dataset training label
    └─ test_icdar2015_label.txt icdar dataset test label
```

The format of the annotation file provided is:

```
"Image file name json.dumps encoded image label information"
ch4_test_images/img_61.jpg [{"transcription": "MASA", "points": [[310, 104], [416, 141], [418, 216], [312, 179]], ...}]
```

The image label information before the encoding of json.dumps is a list containing multiple dictionaries. The points in the dictionaries represent the coordinates (x, y) of the four points of the text box, which are arranged clockwise from the point in the upper left. The field in transcription represents the text in the current text box, and this information is not needed in the text detection. If you want to train PaddleOCR on other datasets, you can construct the annotation file in the above format.

If the text in the “transcription” is ‘*’ or ‘###’, it means that the corresponding label can be ignored. Therefore, if there is no text label, the transcription field can be set to an empty string.

Data Preprocessing

During training, there are certain requirements for the format and size of the input image. At the same time, it is also necessary to obtain true labels of the threshold map and the probability map according to the label information. Therefore, before inputting the data to the model, the data needs to be preprocessed to make pictures and labels meet the needs of network training and inference. In addition, in order to expand the training dataset, suppress over-fitting, and improve the generalization ability of the model, we also use several basic data augmentation methods here.

The methods of preprocessing data in this experiment include:

- Image decoding: to convert the image to Numpy format;
- Label encoding: to parse the label information in txt files and save them in a unified format;
- Basic data augmentation: involving random horizontal flip, random rotation, random zoom, random crop, etc.;
- Acquisition of threshold map labels: to obtain the threshold map labels required in algorithm training through expansion;
- Acquisition of probability map labels: to obtain the probability map labels required in algorithm training through shrinking;
- Normalization: Through normalization, the input value distribution of any neuron in each layer of the neural network is changed to a standard normal distribution with a mean value of 0 and a variance of 1. So, the optimization of the optimal solution will obviously become smooth and the training process is easier to converge;
- Channel transformation: The image data format is [H, W, C] (height, width, and channel number), and the training data format used by the neural network is [C, H, W], so the image data needs to be rearranged, such as changing [224, 224, 3] to [3, 224, 224];

Image Decoding

```
import sys
import six
import cv2
import numpy as np
```

```
# https://github.com/PaddlePaddle/PaddleOCR/blob/release%2F2.4/ppocr/data/imaug/
→operators.py
class DecodeImage(object):
    """ decode image """
    def __init__(self, img_mode='RGB', channel_first=False, **kwargs):
        self.img_mode = img_mode
```

(continues on next page)

(continued from previous page)

```

    self.channel_first = channel_first

    def __call__(self, data):
        img = data['image']
        if six.PY2:
            assert type(img) is str and len(
                img) > 0, "invalid input 'img' in DecodeImage"
        else:
            assert type(img) is bytes and len(
                img) > 0, "invalid input 'img' in DecodeImage"
        # 1. Image decoding
        img = np.frombuffer(img, dtype='uint8')
        img = cv2.imdecode(img, 1)

        if img is None:
            return None
        if self.img_mode == 'GRAY':
            img = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
        elif self.img_mode == 'RGB':
            assert img.shape[2] == 3, 'invalid shape of image[%s]' % (img.shape)
            img = img[:, :, ::-1]

        if self.channel_first:
            img = img.transpose((2, 0, 1))
        # 2. The decoded image is placed in the dictionary
        data['image'] = img
    return data

```

Next, read the image from labels of the training data to demonstrate the use of the `DecodeImage` class.

```

import json
import cv2
import os
import numpy as np
import matplotlib.pyplot as plt
# When using matplotlib.pyplot to draw in the notebook, you need to add this command
# for displaying
%matplotlib inline
from PIL import Image
import numpy as np

label_path = "/home/aistudio/data/data96799/icdar2015/text_localization/train_"
    ↪icdar2015_label.txt"
img_dir = "/home/aistudio/data/data96799/icdar2015/text_localization/"

# 1. Read the first data of the training labels
f = open(label_path, "r")
lines = f.readlines()

# 2. Fetch the first data
line = lines[0]

print("The first data in train_icdar2015_label.txt is as follows.\n", line)
img_name, gt_label = line.strip().split("\t")

```

(continues on next page)

(continued from previous page)

```
# 3. Read the image
image = open(os.path.join(img_dir, img_name), 'rb').read()
data = {'image': image, 'label': gt_label}
```

Declare the DecodeImage class, decode the image, and return a new dictionary data.

```
# 4. Declare the DecodeImage class to decode the image
decode_image = DecodeImage(img_mode='RGB', channel_first=False)
data = decode_image(data)

# 5. Print the shape of the decoded image and visualize the image
print("The shape of decoded image is ", data['image'].shape)

plt.figure(figsize=(10, 10))
plt.imshow(data['image'])
src_img = data['image']
```

Label Decoding

Analyze the label information in txt files and save it in a unified format.

```
import numpy as np
import string
import json

# For detailed implementation, refer to: https://github.com/PaddlePaddle/PaddleOCR/
# blob/release%2F2.4/ppocr/data/imaug/label_ops.py#L38
class DetLabelEncode(object):
    def __init__(self, **kwargs):
        pass

    def __call__(self, data):
        label = data['label']
        # 1. Use json to read tags
        label = json.loads(label)
        nBox = len(label)
        boxes, txts, txt_tags = [], [], []
        for bno in range(0, nBox):
            box = label[bno]['points']
            txt = label[bno]['transcription']
            boxes.append(box)
            txts.append(txt)
            # 1.1 If the text label is * or ##, it means this label is invalid
            if txt in ['*', '##']:
                txt_tags.append(True)
            else:
                txt_tags.append(False)
        if len(boxes) == 0:
            return None
        boxes = self.expand_points_num(boxes)
        boxes = np.array(boxes, dtype=np.float32)
        txt_tags = np.array(txt_tags, dtype=np.bool)

        # 2. Get text, box, and other information
        data['polys'] = boxes
        data['texts'] = txts
```

(continues on next page)

(continued from previous page)

```
    data['ignore_tags'] = txt_tags
    return data
```

Run the following code to observe the difference between the pre-decoding and post-decoding of labels by the DetLabelEncode class.

```
# Import DetLabelEncode from PaddleOCR
from ppocr.data.imaug.label_ops import DetLabelEncode

# 1. Declare the class for label decoding
decode_label = DetLabelEncode()

# 2. Print the label before decoding
print("The label before decode are: ", data['label'])

# 3. Label decoding
data = decode_label(data)
print("\n")

# 4. Print the decoded label
print("The polygon after decode are: ", data['polys'])
print("The text after decode are: ", data['texts'])
```

Basic Data Augmentation

Data augmentation is commonly used to improve training accuracy and generalization of the model. Common data augmentation for text detection includes random horizontal flipping, random rotation, random scaling, and random cropping.

The code example of random horizontal flipping, rotation, and zoom: [Code](#). The code example of random cropping data augmentation: [Code](#).

Acquisition of Threshold Map Labels

Get the threshold map labels required in algorithm training through expansion;

```
import numpy as np
import cv2

np.seterr(divide='ignore', invalid='ignore')
import pyclipper
from shapely.geometry import Polygon
import sys
import warnings

warnings.simplefilter("ignore")

# Calculate the threshold map label class of the text region
# For detailed implementation code, refer to: https://github.com/PaddlePaddle/
# PaddleOCR/blob/release%2F2.4/ppocr/data/imaug/make_border_map.py
class MakeBorderMap(object):
    def __init__(self,
                 shrink_ratio=0.4,
                 thresh_min=0.3,
                 thresh_max=0.7,
                 **kwargs):
        self.shrink_ratio = shrink_ratio
        self.thresh_min = thresh_min
```

(continues on next page)

(continued from previous page)

```

self.thresh_max = thresh_max

def __call__(self, data):
    img = data['image']
    text_polys = data['polys']
    ignore_tags = data['ignore_tags']

    # 1. Generate an empty template
    canvas = np.zeros(img.shape[:2], dtype=np.float32)
    mask = np.zeros(img.shape[:2], dtype=np.float32)

    for i in range(len(text_polys)):
        if ignore_tags[i]:
            continue

        # 2. The draw_border_map function calculates the threshold map labels_
        # based on the decoded box information
        self.draw_border_map(text_polys[i], canvas, mask=mask)
        canvas = canvas * (self.thresh_max - self.thresh_min) + self.thresh_min

    data['threshold_map'] = canvas
    data['threshold_mask'] = mask
    return data

def draw_border_map(self, polygon, canvas, mask):
    polygon = np.array(polygon)
    assert polygon.ndim == 2
    assert polygon.shape[1] == 2

    polygon_shape = Polygon(polygon)
    if polygon_shape.area <= 0:
        return
    # Polygon indentation
    distance = polygon_shape.area * (
        1 - np.power(self.shrink_ratio, 2)) / polygon_shape.length
    subject = [tuple(l) for l in polygon]
    padding = pyclipper.PyclipperOffset()
    padding.AddPath(subject, pyclipper.JT_ROUND, pyclipper.ET_CLOSEDPOLYGON)
    # Calculate the mask
    padded_polygon = np.array(padding.Execute(distance)[0])
    cv2.fillPoly(mask, [padded_polygon.astype(np.int32)], 1.0)

    xmin = padded_polygon[:, 0].min()
    xmax = padded_polygon[:, 0].max()
    ymin = padded_polygon[:, 1].min()
    ymax = padded_polygon[:, 1].max()
    width = xmax - xmin + 1
    height = ymax - ymin + 1

    polygon[:, 0] = polygon[:, 0] - xmin
    polygon[:, 1] = polygon[:, 1] - ymin

    xs = np.broadcast_to(
        np.linspace(
            0, width - 1, num=width).reshape(1, width), (height, width))

```

(continues on next page)

(continued from previous page)

```

ys = np.broadcast_to(
    np.linspace(
        0, height - 1, num=height).reshape(height, 1), (height, width))

distance_map = np.zeros(
    (polygon.shape[0], height, width), dtype=np.float32)
for i in range(polygon.shape[0]):
    j = (i + 1) % polygon.shape[0]
    # Calculate the distance from point to line
    absolute_distance = self._distance(xs, ys, polygon[i], polygon[j])
    distance_map[i] = np.clip(absolute_distance / distance, 0, 1)
distance_map = distance_map.min(axis=0)

xmin_valid = min(max(0, xmin), canvas.shape[1] - 1)
xmax_valid = min(max(0, xmax), canvas.shape[1] - 1)
ymin_valid = min(max(0, ymin), canvas.shape[0] - 1)
ymax_valid = min(max(0, ymax), canvas.shape[0] - 1)
canvas[ymin_valid:ymax_valid + 1, xmin_valid:xmax_valid + 1] = np.fmax(
    1 - distance_map[ymin_valid - ymin:ymax_valid - ymax + height,
                      xmin_valid - xmin:xmax_valid - xmax + width],
    canvas[ymin_valid:ymax_valid + 1, xmin_valid:xmax_valid + 1])

```

```

# Import MakeBorderMap from PaddleOCR
from ppocr.data.imaug.make_border_map import MakeBorderMap

# 1. Declare the MakeBorderMap function
generate_text_border = MakeBorderMap()

# 2. Calculate bordermap information based on the decoded input data
data = generate_text_border(data)

# 3. Visualize the threshold map
plt.figure(figsize=(10, 10))
plt.imshow(src_img)

text_border_map = data['threshold_map']
plt.figure(figsize=(10, 10))
plt.imshow(text_border_map)

```

Acquisition of Probability Map Labels

Get the probability map labels needed in algorithm training through shrinking.

```

import numpy as np
import cv2
from shapely.geometry import Polygon
import pyclipper

# Calculate the probability map labels
# For the code example, refer to: https://github.com/PaddlePaddle/PaddleOCR/blob/
# release%2F2.4/ppocr/data/imaug/make_shrink_map.py
class MakeShrinkMap(object):
    r'''
    Making binary mask from detection data with ICDAR format.
    Typically following the process of class `MakeICDARData`.

```

(continues on next page)

(continued from previous page)

```

    ...

def __init__(self, min_text_size=8, shrink_ratio=0.4, **kwargs):
    self.min_text_size = min_text_size
    self.shrink_ratio = shrink_ratio

def __call__(self, data):
    image = data['image']
    text_polys = data['polys']
    ignore_tags = data['ignore_tags']

    h, w = image.shape[:2]
    # 1. Verify text detection labels
    text_polys, ignore_tags = self.validate_polygons(text_polys,
                                                       ignore_tags, h, w)
    gt = np.zeros((h, w), dtype=np.float32)
    mask = np.ones((h, w), dtype=np.float32)

    # 2. Calculate the probability map of the text region according to the text_
    ↪detection box
    for i in range(len(text_polys)):
        polygon = text_polys[i]
        height = max(polygon[:, 1]) - min(polygon[:, 1])
        width = max(polygon[:, 0]) - min(polygon[:, 0])
        if ignore_tags[i] or min(height, width) < self.min_text_size:
            cv2.fillPoly(mask,
                         polygon.astype(np.int32) [np.newaxis, :, :], 0)
            ignore_tags[i] = True
        else:
            # Polygon indentation
            polygon_shape = Polygon(polygon)
            subject = [tuple(l) for l in polygon]
            padding = pyclipper.PyclipperOffset()
            padding.AddPath(subject, pyclipper.JT_ROUND,
                            pyclipper.ET_CLOSEDPOLYGON)
            shrinked = []

            # Increase the shrink ratio every time we get multiple polygon_
    ↪returned back
            possible_ratios = np.arange(self.shrink_ratio, 1,
                                         self.shrink_ratio)
            np.append(possible_ratios, 1)
            # print(possible_ratios)
            for ratio in possible_ratios:
                # print(f"Change shrink ratio to {ratio}")
                distance = polygon_shape.area * (
                    1 - np.power(ratio, 2)) / polygon_shape.length
                shrinked = padding.Execute(-distance)
                if len(shrinked) == 1:
                    break

            if shrinked == []:
                cv2.fillPoly(mask,
                             polygon.astype(np.int32) [np.newaxis, :, :], 0)
                ignore_tags[i] = True
                continue

```

(continues on next page)

(continued from previous page)

```

# filling
for each_shrink in shrinked:
    shrink = np.array(each_shrink).reshape(-1, 2)
    cv2.fillPoly(gt, [shrink.astype(np.int32)], 1)

data['shrink_map'] = gt
data['shrink_mask'] = mask
return data

```

```

# Import MakeShrinkMap from PaddleOCR
from ppocr.data.imaug.make_shrink_map import MakeShrinkMap

# 1. Declare the generation of text probability map labels
generate_shrink_map = MakeShrinkMap()

# 2. Calculate the probability map of the text region based on the decoded labels
data = generate_shrink_map(data)

# 3. Visualize of Probability Map of text region
plt.figure(figsize=(10, 10))
plt.imshow(src_img)
text_border_map = data['shrink_map']
plt.figure(figsize=(10, 10))
plt.imshow(text_border_map)

```

Normalization

Through normalization, the input value distribution of any neuron in each layer of the neural network is changed to a standard normal distribution with a mean value of 0 and a variance of 1. So, the optimization of the optimal solution will obviously become smooth and the training process is easier to converge.

```

# Image normalization
class NormalizeImage(object):
    """
    normalize image such as subtract mean, divide std
    """

    def __init__(self, scale=None, mean=None, std=None, order='chw', **kwargs):
        if isinstance(scale, str):
            scale = eval(scale)
        self.scale = np.float32(scale if scale is not None else 1.0 / 255.0)
        # 1. Get the normalized mean and variance
        mean = mean if mean is not None else [0.485, 0.456, 0.406]
        std = std if std is not None else [0.229, 0.224, 0.225]

        shape = (3, 1, 1) if order == 'chw' else (1, 1, 3)
        self.mean = np.array(mean).reshape(shape).astype('float32')
        self.std = np.array(std).reshape(shape).astype('float32')

    def __call__(self, data):
        # 2. Get image data from dictionaries
        img = data['image']
        from PIL import Image
        if isinstance(img, Image.Image):
            img = np.array(img)
        assert isinstance(img, np.ndarray), "invalid input 'img' in NormalizeImage"

```

(continues on next page)

(continued from previous page)

```
# 3. Image normalization
data['image'] = (img.astype('float32') * self.scale - self.mean) / self.std
return data
```

Channel Transformation

The image data format is [H, W, C] (height, width, and channel number), and the training data format used by the neural network is [C, H, W], so the image data needs to be rearranged, such as changing [224, 224, 3] to [3, 224, 224].

```
# Change the channel order of the image, HWC to CHW
class ToCHWImage(object):
    """ convert hwc image to chw image
    """
    def __init__(self, **kwargs):
        pass

    def __call__(self, data):
        # 1. Get image data from dictionary
        img = data['image']
        from PIL import Image
        if isinstance(img, Image.Image):
            img = np.array(img)

        # 2. Change the channel order of the image by transposing
        data['image'] = img.transpose((2, 0, 1))
        return data

# 1. Declare the channel transformation class
transpose = ToCHWImage()

# 2. Print the image before transformation
print("The shape of image before transpose", data['image'].shape)

# 3. Image channel transformation
data = transpose(data)

# 4. Print the transformed image to the channel
print("The shape of image after transpose", data['image'].shape)
```

Building a Dataloader

The above code only shows how to read one picture and preprocess it. But in the actual model training, usually a batch of data are read and processed simultaneously.

This section uses `Dataset` and `DatasetLoader` APIs to build a dataloader.

```
# For detailed code for building dataloader, refer to: https://github.com/
→PaddlePaddle/PaddleOCR/blob/release%2F2.4/ppocr/data/simple_dataset.py

import numpy as np
import os
import random
from paddle.io import Dataset
```

(continues on next page)

(continued from previous page)

```

def transform(data, ops=None):
    """ transform """
    if ops is None:
        ops = []
    for op in ops:
        data = op(data)
        if data is None:
            return None
    return data

def create_operators(op_param_list, global_config=None):
    """
    create operators based on the config
    Args:
        params(list): a dict list, used to create some operators
    """
    assert isinstance(op_param_list, list), ('operator config should be a list')
    ops = []
    for operator in op_param_list:
        assert isinstance(operator,
                          dict) and len(operator) == 1, "yaml format error"
        op_name = list(operator)[0]
        param = {} if operator[op_name] is None else operator[op_name]
        if global_config is not None:
            param.update(global_config)
        op = eval(op_name)(**param)
        ops.append(op)
    return ops

class SimpleDataSet(Dataset):
    def __init__(self, mode, label_file, data_dir, seed=None):
        super(SimpleDataSet, self).__init__()
        # In the label file, use '\t' as a separator to distinguish the image name
        # from the label
        self.delimiter = '\t'
        # Dataset path
        self.data_dir = data_dir
        # Random number seed
        self.seed = seed
        # Get all the data and return them in a list
        self.data_lines = self.get_image_info_list(label_file)
        # Create a new list to store data index
        self.data_idx_order_list = list(range(len(self.data_lines)))
        self.mode = mode
        # If it is a training process, randomly shuffle the dataset
        if self.mode.lower() == "train":
            self.shuffle_data_random()

    def get_image_info_list(self, label_file):
        # Get all the data in the label file
        with open(label_file, "rb") as f:
            lines = f.readlines()
        return lines

```

(continues on next page)

(continued from previous page)

```

def shuffle_data_random(self):
    #Randomly shuffle the data
    random.seed(self.seed)
    random.shuffle(self.data_lines)
    return

def __getitem__(self, idx):
    # 1. Get the data whose index is idx
    file_idx = self.data_idx_order_list[idx]
    data_line = self.data_lines[file_idx]
    try:
        # 2. Get the image name and label
        data_line = data_line.decode('utf-8')
        substr = data_line.strip("\n").split(self.delimiter)
        file_name = substr[0]
        label = substr[1]
        # 3. Get the image path
        img_path = os.path.join(self.data_dir, file_name)
        data = {'img_path': img_path, 'label': label}
        if not os.path.exists(img_path):
            raise Exception("{} does not exist!".format(img_path))
        # 4. Read the picture and preprocess it
        with open(data['img_path'], 'rb') as f:
            img = f.read()
            data['image'] = img

        # 5. Complete the data augmentation
        outs = transform(data, self.mode.lower())

    # 6. If it fails to read the current data, read another randomly
    except Exception as e:
        outs = None
        if outs is None:
            return self.__getitem__(np.random.randint(self.__len__()))
    return outs

def __len__(self):
    # Return the size of the dataset
    return len(self.data_idx_order_list)

```

PaddlePaddle's Dataloader API can read data using multi-processing, and can set freely the number of threads. Using multiple threads to read data can accelerate data processing and model training. And its code is as follows:

```

from paddle.io import Dataset, DataLoader, BatchSampler, DistributedBatchSampler

def build_dataloader(mode, label_file, data_dir, batch_size, drop_last, shuffle, num_
_workers, seed=None):
    # Create data reading class
    dataset = SimpleDataSet(mode, label_file, data_dir, seed)
    # Define batch_sampler
    batch_sampler = BatchSampler(dataset=dataset, batch_size=batch_size,_
_shuffle=shuffle, drop_last=drop_last)
    # Use paddle.io.DataLoader to build a dataloader, and set batchsize, the number_
_of processes num_workers, and other parameters
    data_loader = DataLoader(dataset=dataset, batch_sampler=batch_sampler, num_
_workers=num_workers, return_list=True, use_shared_memory=False)

```

(continues on next page)

(continued from previous page)

```

    return data_loader

ic15_data_path = "/home/aistudio/data/data96799/icdar2015/text_localization/"
train_data_label = "/home/aistudio/data/data96799/icdar2015/text_localization/train_"
    ↪icdar2015_label.txt"
eval_data_label = "/home/aistudio/data/data96799/icdar2015/text_localization/test_"
    ↪icdar2015_label.txt"

# Define the training set dataloader, and the number of processes is set to 8
train_dataloader = build_dataloader('Train', train_data_label, ic15_data_path, batch_
    ↪size=8, drop_last=False, shuffle=True, num_workers=0)
# Define validation set dataloader
eval_dataloader = build_dataloader('Eval', eval_data_label, ic15_data_path, batch_
    ↪size=1, drop_last=False, shuffle=False, num_workers=0)

```

DB Model Post-processing

The output shape of the DB head network is the same as the original image. In fact, the three channel features output by the DB head network are the probability map, the threshold map and the binary map.

In the training, the three inference maps and the real labels together calculate the loss function and train the model.

In the inference, only the probability map is needed. Based on the response of the text area in the probability map, the DB post-processing function calculates the coordinates of the surrounding text box.

Since the probability map predicted by the network is the result of shrinking, in the post-processing, you can get the text box by expanding the predicted polygon area with the same offset value. The code example is shown below.

```

# https://github.com/PaddlePaddle/PaddleOCR/blob/release%2F2.4/ppocr/postprocess/db_
    ↪postprocess.py

import numpy as np
import cv2
import paddle
from shapely.geometry import Polygon
import pyclipper

class DBPostProcess(object):
    """
    The post process for Differentiable Binarization (DB).
    """

    def __init__(self,
                 thresh=0.3,
                 box_thresh=0.7,
                 max_candidates=1000,
                 unclip_ratio=2.0,
                 use_dilation=False,
                 score_mode="fast",
                 **kwargs):
        # 1. Obtain post-processing hyperparameters
        self.thresh = thresh

```

(continues on next page)

(continued from previous page)

```

self.box_thresh = box_thresh
self.max_candidates = max_candidates
self.unclip_ratio = unclip_ratio
self.min_size = 3
self.score_mode = score_mode
assert score_mode in [
    "slow", "fast"
], "Score mode must be in [slow, fast] but got: {}".format(score_mode)

self.dilation_kernel = None if not use_dilation else np.array(
    [[1, 1], [1, 1]])

# For detailed implementation of DB post-processing code, refer to: https://
#github.com/PaddlePaddle/PaddleOCR/blob/release%2F2.4/ppocr/postprocess/db_
#postprocess.py

def __call__(self, outs_dict, shape_list):

    # 1. Get network inference results from the dictionary
    pred = outs_dict['maps']
    if isinstance(pred, paddle.Tensor):
        pred = pred.numpy()
    pred = pred[:, 0, :, :]

    # 2. Greater than the post-processing parameter threshold self.thresh
    segmentation = pred > self.thresh

    boxes_batch = []
    for batch_index in range(pred.shape[0]):
        # 3. Get the shape of the original image and resize its ratio
        src_h, src_w, ratio_h, ratio_w = shape_list[batch_index]
        if self.dilation_kernel is not None:
            mask = cv2.dilate(
                np.array(segmentation[batch_index]).astype(np.uint8),
                self.dilation_kernel)
        else:
            mask = segmentation[batch_index]

        # 4. Use the boxes_from_bitmap function to calculate the text box from
        #the predicted text probability map
        boxes, scores = self.boxes_from_bitmap(pred[batch_index], mask,
                                                src_w, src_h)

        boxes_batch.append({'points': boxes})
    return boxes_batch

```

You can find that each word is surrounded by a blue box. These blue boxes are obtained by perform post-processing on the segmentation results output by the DB. Add the following code to line 177 of PaddleOCR/ppocr/postprocess/db_postprocess.py to visualize the segmentation map output by DB. The visualization result is saved as the image vis_segmentation.png.

```

_maps = np.array(pred[0, :, :] * 255).astype(np.uint8)
import cv2
cv2.imwrite("vis_segmentation.png", _maps)

```

```
# 1. Download the trained model
!wget -nc -P ./pretrain_models/ https://paddleocr.bj.bcebos.com/dygraph_v2.0/en/det_
→mv3_db_v2.0_train.tar
!cd ./pretrain_models/ && tar xf det_mv3_db_v2.0_train.tar && cd ../

# 2. Perform text detection inference to get the result
!python tools/infer_det.py -c configs/det/det_mv3_db.yml \
                           -o Global.checkpoints=./pretrain_models/det_mv3_db_v2.0_
                           →train/best_accuracy \
                           Global.infer_img=./doc/imgs_en/img_12.jpg
                           #PostProcess.unclip_ratio=4.0
# Note: For the introduction and usage of PostProcess parameters and Global_
→parameters, refer to: https://github.com/PaddlePaddle/PaddleOCR/blob/release%2F2.3/
→doc/doc_ch/config.md
```

Visualize the text probability map predicted by the predictive model and the final result of the predicted text box.

```
img = Image.open('./output/det_db/det_results/img_12.jpg')
img = np.array(img)

# Draw the read picture
plt.figure(figsize=(10, 10))
plt.imshow(img)

img = Image.open('./vis_segmentation.png')
img = np.array(img)

# Draw the read picture
plt.figure(figsize=(10, 10))
plt.imshow(img)
```

From the visualization results, it can be found that the output result of the DB is a binary image. As the response value of the text area is higher, that of the non-text background area is lower. The post-processing of DB is to find the minimum box of these response areas, and then get the coordinates of each text area. In addition, the size of the text box can be adjusted by modifying the post-processing parameters, or text boxes poor in detection can be filtered out.

There are four parameters for DB post-processing:

- thresh: The threshold for binarization of the segmentation map in DBPostProcess, and its default value is 0.3.
- box_thresh: The threshold for filtering the output box in DBPostProcess. Boxes below this threshold will not be output.
- unclip_ratio: The enlarging ratio of text boxes in DBPostProcess
- max_candidates: The maximum number of text boxes output in DBPostProcess, and its default value is 1000

```
# 3. Increase the unclip_ratio parameter of DB post-processing to 4.0, and the default_
→is 1.5. Change the size of the output text box, and the parameters perform text_
→detection inference to get the result
!python tools/infer_det.py -c configs/det/det_mv3_db.yml \
                           -o Global.checkpoints=./pretrain_models/det_mv3_db_v2.0_
                           →train/best_accuracy \
                           Global.infer_img=./doc/imgs_en/img_12.jpg \
                           PostProcess.unclip_ratio=4.0
# Note: For the introduction and usage of PostProcess parameters and Global_
→parameters, refer to: https://github.com/PaddlePaddle/PaddleOCR/blob/release%2F2.4/
→doc/doc_ch/config.md
```

(continues on next page)

(continued from previous page)

```



```

From the implementation results of the above code, it can be found that after increasing the unclip_ratio parameter of the DB post-processing, the predicted text box becomes much larger. Therefore, when the training result does not meet the expectation, it is feasible to adjust the post-processing parameters in order to refine the result. What's more, you can try adjusting the other three parameters, thresh, box_thresh, max_candidates, and compare the corresponding detection results.

Loss Function Definition

Since three inference maps are obtained in the training, in the loss function, it is also necessary to combine these three maps and their real labels to build three parts of the loss function respectively. The formula of the whole loss function is defined as follows:

$$L = L_b + \alpha \times L_s + \beta \times L_t$$

L is the total loss, L_s is the probability map loss. In this experiment, the Dice loss with OHEM (online hard example mining) is used. And L_t is the threshold map loss. The L_1 distance between the predicted value and the label is used in this experiment. L_b is the loss function of the text binary map. α and β are the weight coefficients, which are set to 5 and 10 here.

The three losses L_b , L_s , and L_t refer to Dice Loss, Dice Loss (OHEM), and MaskL1 Loss. Next, define these three parts:

- Dice Loss is to compare the similarity between the predicted text binary image and the label. It is often used in binary image segmentation. For code example, refer to [link](#). The formula is as follows:

$$\text{dice_loss} = 1 - \frac{2 \times \text{intersection_area}}{\text{total_area}}$$

- Dice Loss (OHEM) uses Dice Loss with OHEM to improve the imbalance of positive and negative samples. OHEM is a special automatic sampling method that can automatically select difficult samples for loss calculation, thereby improving the training effect of the model. Here, the sampling ratio of positive and negative samples is set to 1:3. For the code example, refer to [link](#).
- MaskL1 Loss is to calculate the L_1 distance between the predicted text threshold map and the label.

```

from paddle import nn
import paddle
from paddle import nn
import paddle.nn.functional as F

```

(continues on next page)

(continued from previous page)

```

# DB loss function
# For the code example, refer to: https://github.com/PaddlePaddle/PaddleOCR/blob/
˓→release%2F2.4/pocr/losses/det_db_loss.py
class DBLoss(nn.Layer):
    """
    Differentiable Binarization (DB) Loss Function
    args:
        param (dict): the super parameter for DB Loss
    """

    def __init__(self,
                 balance_loss=True,
                 main_loss_type='DiceLoss',
                 alpha=5,
                 beta=10,
                 ohem_ratio=3,
                 eps=1e-6,
                 **kwargs):
        super(DBLoss, self).__init__()
        self.alpha = alpha
        self.beta = beta
        # Declare different loss functions
        self.dice_loss = DiceLoss(eps=eps)
        self.l1_loss = MaskL1Loss(eps=eps)
        self.bce_loss = BalanceLoss(
            balance_loss=balance_loss,
            main_loss_type=main_loss_type,
            negative_ratio=ohem_ratio)

    def forward(self, predicts, labels):
        predict_maps = predicts['maps']
        label_threshold_map, label_shrink_map, label_shrink_
˓→mask = labels[
            1:]
        shrink_maps = predict_maps[:, 0, :, :]
        threshold_maps = predict_maps[:, 1, :, :]
        binary_maps = predict_maps[:, 2, :, :]
        # 1. For the text inference probability map, use the binary cross-entropy_
˓→loss function
        loss_shrink_maps = self.bce_loss(shrink_maps, label_shrink_map,
                                         label_shrink_mask)
        # 2. For the text inference threshold map, use the L1 distance loss function
        loss_threshold_maps = self.l1_loss(threshold_maps, label_threshold_map,
                                         label_threshold_mask)
        # 3. For text inference binary graph, use the dice loss loss function
        loss_binary_maps = self.dice_loss(binary_maps, label_shrink_map,
                                         label_shrink_mask)

        # 4. Multiply different loss functions by different weights
        loss_shrink_maps = self.alpha * loss_shrink_maps
        loss_threshold_maps = self.beta * loss_threshold_maps

        loss_all = loss_shrink_maps + loss_threshold_maps \
                  + loss_binary_maps

```

(continues on next page)

(continued from previous page)

```

losses = {'loss': loss_all, \
          "loss_shrink_maps": loss_shrink_maps, \
          "loss_threshold_maps": loss_threshold_maps, \
          "loss_binary_maps": loss_binary_maps}
return losses

```

Index Evaluation

Considering that the DB post-processing detection frame is diverse and not level, this experiment adopts a simple method of calculating IOU for evaluation. For the calculation code, refer to [Text detection evaluation method of icdar Challenges 4](#).

There are three calculation indicators for text detection: Precision, Recall and Hmean. The calculation logic of the three indicators is:

1. To create a matrix of size [n, m] called iouMat, where n is the number of GT (ground truth) boxes, m is the number of detected boxes, and n, m are the ### calibrated without the text Number of boxes.
2. To count the number of IOUs greater than the threshold 0.5 in iouMat, and divide this value by the number n of gt to get Recall.
3. To count the number of IOUs greater than the threshold 0.5 in iouMat, and divide this value by the number of detection frames m to get Precision.
4. Hmean's index calculation method is the same as that of F1-score, and the formula is as follows:

$$Hmean = 2.0 * \frac{Precision * Recall}{Precision + Recall}$$

The core code of text detection metric calculation is shown below. For the code example, refer to [link](#)

```

# The calculation method of the text detection metric is as follows:
# Complete code reference https://github.com/PaddlePaddle/PaddleOCR/blob/release%2F2.4/ppocr/metrics/det\_metric.py
if len(gtPols)> 0 and len(detPols)> 0:
    outputShape = [len(gtPols), len(detPols)]

    # 1. Create a matrix of size [n, m] to save the calculated IOU
    iouMat = np.empty(outputShape)
    gtRectMat = np.zeros(len(gtPols), np.int8)
    detRectMat = np.zeros(len(detPols), np.int8)
    for gtNum in range(len(gtPols)):
        for detNum in range(len(detPols)):
            pG = gtPols[gtNum]
            pD = detPols[detNum]

            # 2. Calculate the IOU between the inference box and the GT box
            iouMat[gtNum, detNum] = get_intersection_over_union(pD, pG)
    for gtNum in range(len(gtPols)):
        for detNum in range(len(detPols)):
            if gtRectMat[gtNum] == 0 and detRectMat[detNum] == 0 and gtNum not in gtDontCarePolsNum and detNum not in detDontCarePolsNum:
                # 2.1 Count the number of IOUs greater than the threshold 0.5
                if iouMat[gtNum, detNum]> self.iou_constraint:
                    gtRectMat[gtNum] = 1

```

(continues on next page)

(continued from previous page)

```

detRectMat [detNum] = 1
detMatched += 1
pairs.append({'gt': gtNum, 'det': detNum})
detMatchedNums.append(detNum)

# 3. Divide the number of IOUs greater than the threshold 0.5 by the number of GT
→boxes numGtCare to get the recall
recall = float(detMatched) / numGtCare

# 4. Divide the number of IOUs greater than the threshold 0.5 by the number of inference
→boxes numDetCare to get precision
precision = 0 if numDetCare == 0 else float(detMatched) / numDetCare

# 5. Calculate the Hmean indicator with formula
hmean = 0 if (precision + recall) == 0 else 2.0 * \
precision * recall / (precision + recall)

```

Questions:

1. In the below situation where the IOU of the GT box and the inference box is greater than 0.5, but some texts have not been detected, can the above indicator calculation accurately reflect the accuracy of the model?
2. When encountering such problems in the experiment, how to optimize the model?



Figure 6: Example of labeling of GT box and inference box

Model Training

After data processing, network and loss function defining, you can start training the model.

Training is based on PaddleOCR training, in the form of parameter configuration. For parameter files, refer to [link](#). Network structure parameters are as follows:

```
Architecture:  
  model_type: det  
  algorithm: DB  
  Transform:  
  Backbone:  
    name: MobileNetV3  
    scale: 0.5  
    model_name: large  
  Neck:  
    name: DBFPN  
    out_channels: 256  
  Head:  
    name: DBHead  
    k: 50
```

Parameters of the optimizer are:

```
Optimizer:  
  name: Adam  
  beta1: 0.9  
  beta2: 0.999  
  lr:  
    learning_rate: 0.001  
  regularizer:  
    name: 'L2'  
    factor: 0
```

Parameters of the post-processing are as follows:

```
PostProcess:  
  name: DBPostProcess  
  thresh: 0.3  
  box_thresh: 0.6  
  max_candidates: 1000  
  unclip_ratio: 1.5
```

...

For the complete parameter configuration file, refer to [det_mv3_db.yml](#).

```
!mkdir train_data  
!cd train_data && ln -s /home/aistudio/data/data96799/icdar2015 icdar2015  
!wget -P ./pretrain_models/ https://paddle-imagenet-models-name.bj.bcebos.com/dygraph/  
MobileNetV3_large_x0_5_pretrained.pdparams
```

```
!python tools/train.py -c configs/det/det_mv3_db.yml
```

The model after network training is saved in PaddleOCR/output/db_mv3/ by default. If you want to change the directory, you can set the parameter Global.save_model_dir during training, such as:

```
# Set Global.save_model_dir in the parameter file to change the model directory
python tools/train.py -c configs/det/det_mv3_db.yml -o Global.save_model_dir=../
➥output/save_db_train/"
```

Model Evaluation

During the training process, two models are saved by default: the latest trained model named latest, and the most accurate model named best_accuracy. Next, use the saved model parameters to evaluate the precision, recall, and hmean on the test set:

The text detection accuracy evaluation code is in PaddleOCR/ppocr/metrics/det_metric.py. You can call tools/eval.py to evaluate the accuracy of the trained model.

```
!python tools/eval.py -c configs/det/det_mv3_db.yml -o Global.checkpoints=../output/db_
➥mv3/best_accuracy
```

Model Inference

After training the model, you can also use the saved model to perform model inference on a certain picture or an image of a folder in the dataset, and observe the inference result.

```
!python tools/infer_det.py -c configs/det/det_mv3_db.yml -o Global.checkpoints=../
➥pretrain_models/det_mv3_db_v2.0_train/best_accuracy Global.infer_img=../doc/imgs_en/
➥img_12.jpg
```

The predicted image is saved in ./output/det_db/det_results/. And the visualization with PIL is:

```
import matplotlib.pyplot as plt
# When using matplotlib.pyplot to draw in the notebook, you need to add this command
➥for display
%matplotlib inline
from PIL import Image
import numpy as np

img = Image.open('../output/det_db/det_results/img_12.jpg')
img = np.array(img)

# Draw the read picture
plt.figure(figsize=(20, 20))
plt.imshow(img)
```

4.2.4 Text Detection FAQ

This section talks about problems that developers often encounter when using PaddleOCR's text detection model, and gives solutions or suggestions.

The FAQ is introduced in two parts:

- Questions related to model training in the text detection
- Questions related to model inference in the text detection

FAQ about Model Training in the Text Detection

1.1 What are the text detection algorithms provided by PaddleOCR?

A: PaddleOCR contains a variety of text detection models, including regression-based text detection methods like EAST and SAST, and segmentation-based text detection methods like DB, PSENet.

1.2 What datasets are used in the Chinese ultra-lightweight and generic models in the PaddleOCR project? How many samples are there? What is the configuration of GPUs? How many epochs were run, and how long did they run?

A: For the ultra-lightweight DB detection model, the training data includes open-source datasets lsvt, rctw, CASIA, CCPD, MSRA, MLT, BornDigit, iflytek, SROIE and synthetic datasets. The total data volume is 10W, The dataset is divided into 5 parts. In the training, samples are randomly picked. The training took about 500 epochs on a 4-card V100GPU, which took 3 days.

1.3 Does the text detection training label require specific text labeling? What does the “###” in the label mean?

A: In the model training, only coordinates of text regions are required. The label can be in four or fourteen points, arranged in the order of upper left, upper right, lower right, and lower left. The label file provided by PaddleOCR contains text fields. For unclear text in the text area, ### will be used instead. In the model training of the text detection, the text field in the label will not be used.

1.4 Is the training result of the text detection model bad when the layout of text lines is dense?

A: In using segmentation-based methods, such as DB, you'd better collect a batch of data for training to detect dense text lines and turn down [shrink_ratio] ([https://github.com/PaddlePaddle/PaddleOCR/blob/8b656a3e13631dfb1ac21d2095d4d4a4993ef710/ppocr/data/imaug/make_shrink_map_repo-pjax-container, div\[itemtype='http://schema.org/SoftwareSourceCode'\] main, \[data-pjax-container\]#L37](https://github.com/PaddlePaddle/PaddleOCR/blob/8b656a3e13631dfb1ac21d2095d4d4a4993ef710/ppocr/data/imaug/make_shrink_map_repo-pjax-container, div[itemtype='http://schema.org/SoftwareSourceCode'] main, [data-pjax-container]#L37)) which is to generate the binary map during the training. In addition, you can appropriately reduce unclipratio in model inference, for the larger the unclip_ratio parameter value is, the larger the detection frame is.

1.5 For some large-sized document images, using DB will tend to miss more texts in detection. How to avoid this problem?

A: First of all, you need to make sure that the missing detection is caused by the model training or image prediction. If the model is not well trained, input more data for training, or enhance data augmentation during training. If it is caused by the oversized predicted image, increase the longest side setting parameter [det_limit_side_len] ([https://github.com/PaddlePaddle/PaddleOCR/blob/8b656a3e13631dfb1ac21d2095d4d4a4993ef710/tools/infer/utility.py?pjx=#js-repo-pjax-container, div\[itemtype='http://schema.org/SoftwareSourceCode'\] main, \[data-pjax-container\]#L47](https://github.com/PaddlePaddle/PaddleOCR/blob/8b656a3e13631dfb1ac21d2095d4d4a4993ef710/tools/infer/utility.py?pjx=#js-repo-pjax-container, div[itemtype='http://schema.org/SoftwareSourceCode'] main, [data-pjax-container]#L47)), which is 960 by default. Also, you can observe whether the missed texts are segmented by visualizing the post-processed segmentated images. If there is no segmentation, it means the model is not well trained. If there is a complete segmentation area, it means that missing detection is due to the problem in the post-processing. In this case, it is recommended to adjust DB post-processing parameters [\[?\]](#)

**1.6 How to deal with the missing detection of curved texts (such as a slightly distorted document image) by using DB?

A: To calculate the average score of the text box in the DB post-processing is to calculate that of the rectangle area, but it is easy to cause missing detection of the curved texts. So the calculation of the average score of the polygon area has been added, which will be more accurate, but the speed may decrease. You can make a choice as you want. and you can see the [visualized comparison of effect] (<https://github.com/PaddlePaddle/PaddleOCR/pull/2604>) in the relevant pr. This function is selected by the parameter `det_db_score_mode`, and its value is optional [`fast` (default), `slow`], `fast` corresponds to the original rectangle mode, and `slow` corresponds to the polygon mode. Thank the user `buptlihang` for putting forward `pr` to help solve this problem.

1.7 In simple OCR tasks with low accuracy requirements, how many datasets do I need to prepare?

A: (1) The amount of training data is related to the complexity of the problem to be solved. The more difficult it is, the more accurate it requires, and more datasets are needed. Usually, if more data are trained, the result will be better.

(2) In scenarios with low accuracy requirements, the amount of data required for detection and recognition is different. In detection, 500 images can reach the basic detection standard. In recognition, it is necessary to ensure that the number of text images in which each character in the recognition dictionary appears in different scenes is greater than 200 (for example, if there are 5 characters in the dictionary, each word needs to appear in more than 200 pictures, then the minimum number of images should be between 200-1000), so that the basic recognition standard can be guaranteed.

1.8 How to get more data when the amount of training data is small?

A: When the amount of training data is small, you can try the following three ways to get more data: (1) The most direct and effective way is to collect more training data manually. (2) Perform basic image processing or transformation based on PIL and opencv. For example, write texts onto the background using ImageFont, Image, and ImageDraw in PIL, opencv's rotation and affine transformation, Gaussian filtering and so on. (3) Synthesize data using data generation algorithms, such as algorithms such as pix2pix.

1.9 How to replace the backbone of text detection/recognition?

A: No matter in text detection or text recognition, the choice of the backbone network is a trade-off between prediction effect and efficiency. Generally, if you choose a larger-scale backbone network, such as ResNet101_vd, the detection or recognition will be more accurate, but the inference time will increase accordingly. However, choosing a smaller-scale backbone network such as MobileNetV3_small_x0_35, will make inference faster, but the accuracy of detection or recognition will be greatly reduced. Fortunately, the detection or recognition effects of different backbone networks are positively correlated with the task of classifying images into 1000 classes in the ImageNet dataset. PaddleClas, an image classification suite of PaddlePaddle, includes 23 series of classification network structures such as ResNet_vd, Res2Net, HRNet, MobileNetV3, and GhostNet, the top1 recognition accuracy rate of the above image classification task, the inference time of GPU (V100 and T4) and CPU (Snapdragon 855), and the corresponding 117 pre-training model download addresses.

(1) The replacement of the text detection backbone network is mainly to determine 4 stages similar to ResNet to facilitate the integration of detection heads similar to FPN. In addition, in the text detection, using the classification trained by ImageNet to pre-train models can accelerate the convergence and improve the effect.

(2) The replacement of the backbone network for text recognition requires paying attention to the drop position of the network width and height. Since text recognition has a large ratio of width to height, the reduction of the height needs to be less than that of the width. You can refer to [Changes to the MobileNetV3 backbone network in PaddleOCR](#).

1.10 How to finetune the detection model, such as by freezing the previous layer or learning with a small learning rate for some layers?

A: If you want to freeze certain layers, you can set the stop_gradient of the variable to True, so that all the parameters before calculating this variable will not be updated, refer to: https://www.paddlepaddle.org.cn/documentation/docs/zh/develop/faq/train_cn.html#id4

If you want to learn with a smaller learning rate for some layers, it is not so convenient to carry out this in the static graph. So you can set a fixed learning rate for the weight attribute when the parameters are initialized. For this, please refer to: https://www.paddlepaddle.org.cn/documentation/docs/en/develop/api/paddle/fluid/param_attr/ParamAttr_cn.html#paramattr

In fact, our experiment has found that it is also feasible to load the model for finetuning without setting different learning rates of certain layers.

1.11 In the preprocessing of DB, why should the length and width of the picture be set to multiples of 32?

A: It is related to the stride of the network downsampling. Take the resnet backbone network of the detection as an example. After the image is input to the network, it needs to be downsampled by 2 for 5 times, 32 in total. Therefore, it is recommended that the input image size be a multiple of 32.

1.12 In the PP-OCR series models, why does the backbone network for text detection not use SEBlock?

A: The SE module is an important module of the MobileNetV3 network. It is to estimate the importance of each feature channel of the feature map, assign weights to each feature of the feature map, and improve the expressive ability of the network. However, for text detection, the resolution of the input network is large, usually 640*640, so it is difficult to use

the SE module to estimate the importance of each feature channel of the feature map. The network improvement ability is limited, but the module is relatively time-consuming. Therefore, in the PP-OCR system, the backbone network for text detection does not use the SE module. An experiment also shows that when the SE module is removed, the size of the ultra-lightweight model can be reduced by 40%, and the text detection effect is basically not affected. For more details, please refer to the PP-OCR technical article, <https://arxiv.org/abs/2009.09941>.

1.13 How to optimize the PP-OCR detection effect if it is not good?

A: It depends.

- If the detection effect is not available on your scene, the priority is to finetune your data;
- If the image is too large and the text is too dense, do not over-compress the image. You can try to modify the resize logic of the preprocessing of the detection to prevent the image from being over-compressed;
- If the size of the detection box is too close to the text or too large, you can adjust the db_unclip_ratio. Increasing the parameter can enlarge the detection box, and reducing the parameter can reduce the size of the detection box;
- If many detection boxes are missed in detection, you can reduce the threshold parameter det_db_box_thresh to prevent some detection boxes from being filtered out. You can also try to set det_db_score_mode to 'slow';
- You can also choose use_dilation as True to expand the output feature map of the detection. In general, the effect will be improved.

1.14 What should I do if I encounter part of the text is missed in detection like the figure below?

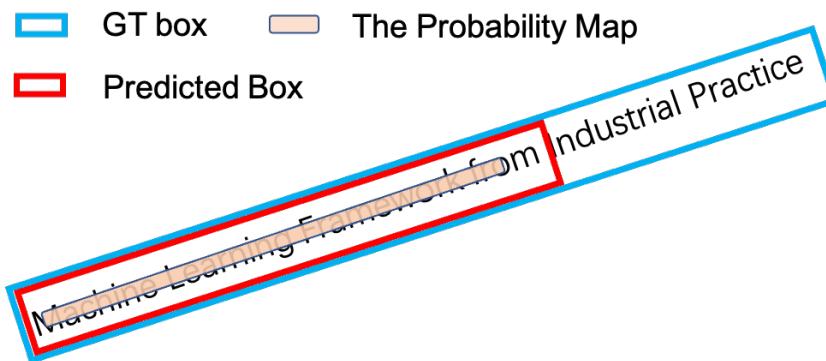


Figure 7: Missing detection

A: The above problem shows that part of the text is detected, but because the IOU of the text inference box and the GT box is greater than the threshold 0.5, the detection indicators cannot be fed back normally. If there are many such cases, increase the IOU threshold. In addition, the reason for missing detection is that the features of some texts do not respond. But ultimately, it is the network that has not learned features of the missed texts. Therefore, case-by-case analysis is feasible. Visualize the inference results to analyze the reason, and figure out whether it is caused by factors such as lighting, deformation, long text, and so on. Then use optimize the detection results through data augmentation, network adjustment, or post-processing adaptation.

FAQ about Model Inference in the Text Detection

2.1 In the DB solution, the boundaries of some boxes are too close to the text to make some corners of the text detected. Is there any solution?

A: Appropriately increase the post-processing parameter `unclip_ratio`. The larger the parameter is, the larger the text box will be.

2.2 Why does inference in the PaddleOCR detection only support one image for testing? That is, `test_batch_size_per_card=1`.

A: Scale the image in equal proportions, and the longest side is 960. The length and width of different images varies after the scaling, and they cannot form a batch, so set `test_batch_size` to 1.

2.3 Accelerate the model inference in PaddleOCR text detection on the CPU?

A: It is feasible to use mkldnn (OneDNN) for acceleration on x86 CPU . And start the `enable_mkldnn` parameter on the CPU supporting mkldnn. In addition, increasing `num_threads` used for inference on the CPU can speed up the inference speed on the CPU.

2.4 Accelerate PaddleOCR's text detection model prediction on GPU?

A: It is recommended to use TensorRT to accelerate inference on GPU.

- 1. Download the Paddle installation package or inference library with TensorRT from [link](#).
- 1. Download the [TensorRT](#) from the Nvidia official website. Note that the version of the downloaded TensorRT version is the same as that compiled in the Paddle installation package.
- 1. Set the environment variable `LD_LIBRARY_PATH` to point to the lib folder of TensorRT

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<TensorRT-${version}/lib>
```

- 1. Enable `tensorrt` option.

2.5 How to deploy the PaddleOCR model on the mobile terminal?

A: PaddlePaddle has a special tool for mobile-device deployment: [PaddleLite](#), and PaddleOCR provides DB+CRNN as the demo code on android arm deployment. Refer to [link](#).

2.6 How to use multi-process inference on PaddleOCR?

A: PaddleOCR has recently added [Control parameters of multi-process inference](#). `use_mp` indicates whether there are multiple processes, and `total_process_num` represents the number of processes. For detailed usage, please refer to [document](#).

2.7 How to solve video memory explosion and memory leak during inference?

A: If it is the inference of the training model, it is because the video memory is not enough for the model or the input image is too large. You can refer to the code and add `paddle.no_grad()` before the main function runs to reduce the video memory usage. If the memory usage of the inference model is too high, you can add `config.enable_memory_optim()` to reduce the memory usage in configuration.

In addition, regarding the memory leak in the inference, please install the latest version of paddle where the memory leak has been fixed.

4.2.5 Assignment

Short-answer questions:

- 1. According to the size of the output feature maps of DB Backbone and FPN, determine the height and width of the input image of DB need to be multiples of: A: 32, B: 64

Experiment:

- 1. Use the DB algorithm configuration file configs/det/det_mv3_db.yml to train the text detection model on the dataset det_data_lesson_demo.tar, and optimize experimental accuracy.



Figure 8: Example of *det_data_lesson_demo* training data

4.3 Summary

This chapter introduces the theory and practice of text detection algorithms.

The first section introduces how to get started quickly with the PaddleOCR text detection model, and demonstrates the implementation process from data processing to text detection algorithm training with the example of the DB algorithm. The next section will talk about the text recognition algorithms.

The second one introduces the development of text detection in recent years, including regression-based and segmentation-based text detection methods. Also, the methodology and ideas of some classic papers are listed. The next section will take the PaddleOCR open-source library as an example to detail how to construct text detection algorithms from scratch and implement the training.

4.4 Reference

1. Liao, Minghui, et al. "Textboxes: A fast text detector with a single deep neural network." Thirty-first AAAI conference on artificial intelligence. 2017.
2. Liao, Minghui, Baoguang Shi, and Xiang Bai. "Textboxes++: A single-shot oriented scene text detector." IEEE transactions on image processing 27.8 (2018): 3676-3690.
3. Tian, Zhi, et al. "Detecting text in natural image with connectionist text proposal network." European conference on computer vision. Springer, Cham, 2016.
4. Zhou, Xinyu, et al. "East: an efficient and accurate scene text detector." Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2017.
5. Wang, Fangfang, et al. "Geometry-aware scene text detection with instance transformation network." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.
6. Yuliang, Liu, et al. "Detecting curve text in the wild: New dataset and new solution." arXiv preprint arXiv:1712.02170 (2017).
7. Deng, Dan, et al. "Pixellink: Detecting scene text via instance segmentation." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 32. No. 1. 2018.
8. Wu, Yue, and Prem Natarajan. "Self-organized text detection with minimal post-processing via border learning." Proceedings of the IEEE International Conference on Computer Vision. 2017.
9. Tian, Zhuotao, et al. "Learning shape-aware embedding for scene text detection." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.
10. Wang, Wenhai, et al. "Shape robust text detection with progressive scale expansion network." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.
11. Wang, Wenhai, et al. "Efficient and accurate arbitrary-shaped text detection with pixel aggregation network." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019.
12. Liao, Minghui, et al. "Real-time scene text detection with differentiable binarization." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 07. 2020.
13. Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.
14. Dai, Pengwen, et al. "Progressive Contour Regression for Arbitrary-Shape Scene Text Detection." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.
15. He, Minghang, et al. "MOST: A Multi-Oriented Scene Text Detector with Localization Refinement." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.
16. Zhu, Yiqin, et al. "Fourier contour embedding for arbitrary-shaped text detection." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.
17. Tang, Jun, et al. "Seglink++: Detecting dense and arbitrary-shaped scene text by instance-aware component grouping." Pattern recognition 96 (2019): 106954.
18. Wang, Yuxin, et al. "Contournet: Taking a further step toward accurate arbitrary-shaped scene text detection." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.
19. Zhang, Chengquan, et al. "Look more than once: An accurate detector for text of arbitrary shapes." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.
20. Xue C, Lu S, Zhang W. Msr: Multi-scale shape regression for scene text detection[J]. arXiv preprint arXiv:1901.02596, 2019.

TEXT RECOGNITION

This chapter is mainly about the theoretical knowledge of text recognition algorithms, including the background, classes of algorithms and some classic paper ideas.

In this chapter, you can learn:

1. The goal of text recognition
2. Types of text recognition algorithms
3. Typical ideas of various algorithms

Text recognition is a subtask of OCR (Optical Character Recognition), aimed at recognizing the content of one specific area. In the two-stage method of OCR, it comes after text detection to convert an image into a text.

Specifically, the model inputs a localized text line and predicts its content and confidence value. The visualization results are as follows:

 汉阳鹦鹉家居建材市场E区25-26号

Predicts of word_2.jpg:['汉阳鹦鹉家居建材市场E区25-26号', 0.9957605]

 原装电池

Predicts of word_3004.jpg:['原装电池', 0.99971426]

Figure 1: Visualization results of model prediction

There are many application scenarios for text recognition, such as document recognition, road sign recognition, license plate recognition, industrial number recognition, etc. In actual scenarios, the task of text recognition can be divided into two categories: **Regular text recognition** and **Irregular text recognition**.

- Regular text recognition: It mainly fits into the text which is considered mostly horizontal, like printed fonts, scanned texts, etc.
- Irregular text recognition: It is common in natural scenes in which texts are not horizontal and some are distorted, covered, or blurred, due to the large variance of its appearance including curvature, orientation, and distortion.

The figures below show the data patterns of IC15 and IC13, which respectively represent the irregular text and the regular text. Obviously, the irregular one often has problems such as distortion, blurring, and large font differences. It is closer to the natural scene but more challenging.

Therefore, the current major algorithms are trying to process irregular datasets with higher precision.

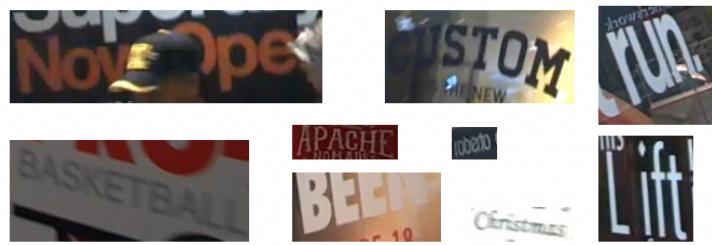


Figure 2: IC15 picture sample (irregular texts)



Figure 3: IC13 picture sample (regular texts)

The two public data sets are often involved in the comparison of algorithm capabilities. After multi-dimensional analysis, the common classification of English benchmark datasets is as follows:

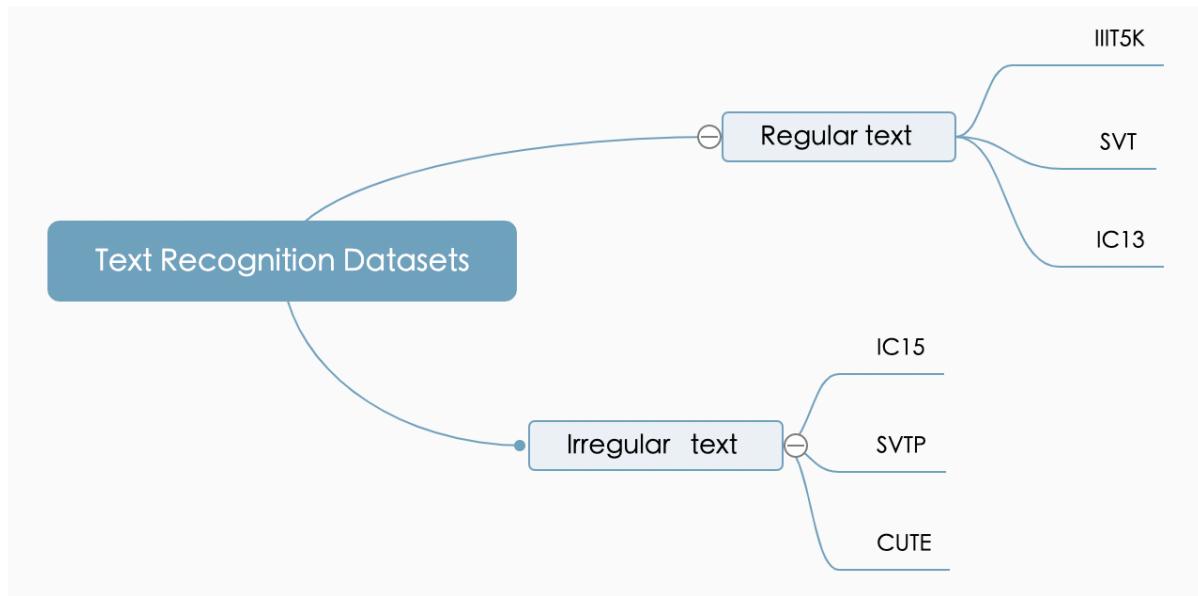
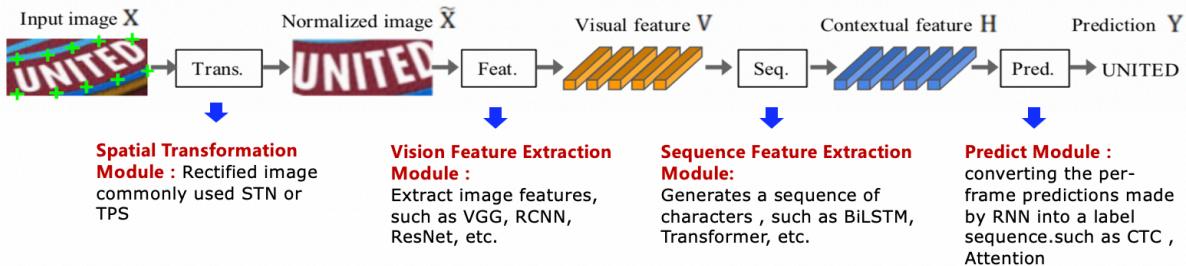


Figure 4: Common English benchmark datasets

5.1 Introduction to Text Recognition Methods

In the traditional text recognition method, the task is divided into 3 steps: image preprocessing, character segmentation and character recognition. In this way, it is necessary to model a specific scene, but the model will fail once the scene changes. In the face of complex text backgrounds and scene changes, methods based on deep learning can perform better.

Most recognition algorithms can be represented by the following framework, and every algorithm flow is divided into 4 stages:



We have sorted out the main algorithm types and major papers as below:

Algorithm type	Main ideas	Main papers
Traditional algorithm	Sliding window, character extraction, dynamic programming	-
CTC	Based on CTC, faster recognition can be realized without predefined alignment.	CRNN, Rosetta
Attention	Based on attention, the method can be applied to unconventional text.	RARE, DAN, PREN
Transformer	Transformer-based method	SRN, NRTR, Master, ABINet
Rectification	The rectification module learns the text boundary and corrects it along the horizontal axis	RARE, ASTER, SAR
Segmentation	The segmentation-based approach extracts the character position and then categorize the text	Text Scanner, Mask TextSpotter

5.1.1 Regular Text Recognition

There are two major algorithms for text recognition, namely the CTC (Connectionist Temporal Classification)-based algorithm and the Sequence2Sequence algorithm. They mainly differ in decoding.

The CTC-based algorithm puts the encoded sequence into CTC for decoding; the Sequence2Sequence-based method puts the sequence into the Recurrent Neural Network (RNN) module for cyclic decoding. The two methods have been verified to be effective and they are also the mainstream.

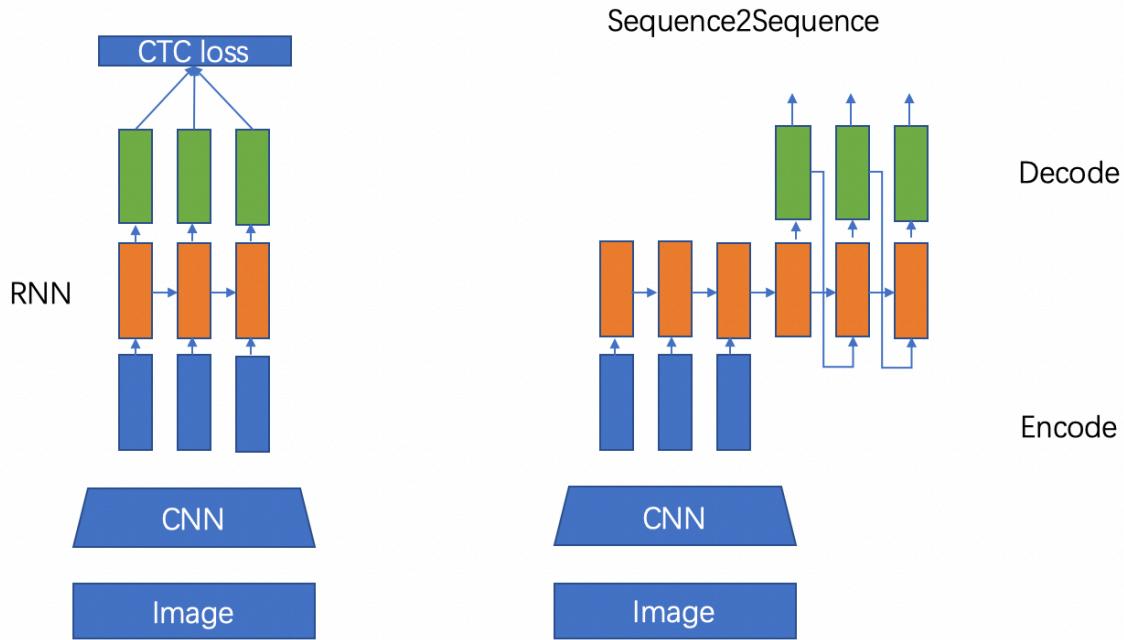


Figure 5: Left: CTC-based method, right: Sequence2Sequence-based method

Algorithms Based on CTC

The most typical CTC-based algorithm is CRNN (Convolutional Recurrent Neural Network) [1], which uses mainstream convolutional structures such as ResNet, MobileNet, VGG, etc. to extract features. Due to the particularity of text recognition tasks, there is a large amount of contextual information in the input data. The convolution kernel feature of the convolutional neural networks (CNN) leads to its focus on local information and lack of modeling long-term dependency, so it is difficult to dig into the contextual connections by only using CNN. To solve this problem, the CRNN text recognition algorithm introduces the bidirectional LSTM (Long Short-Term Memory) to enhance the context modeling. And experiments have proved that the bidirectional LSTM module can effectively extract the contextual information of the picture, and finally enter the output feature sequence into the CTC module, and decode the sequential result. This structure has been validated and widely used in text recognition. Rosetta [2] is a recognition network proposed by FaceBook, consisting of a fully convolutional model and CTC. Gao Y [3] et al. have used CNN convolution to replace LSTM for it has fewer parameters, better performance and the same accuracy.

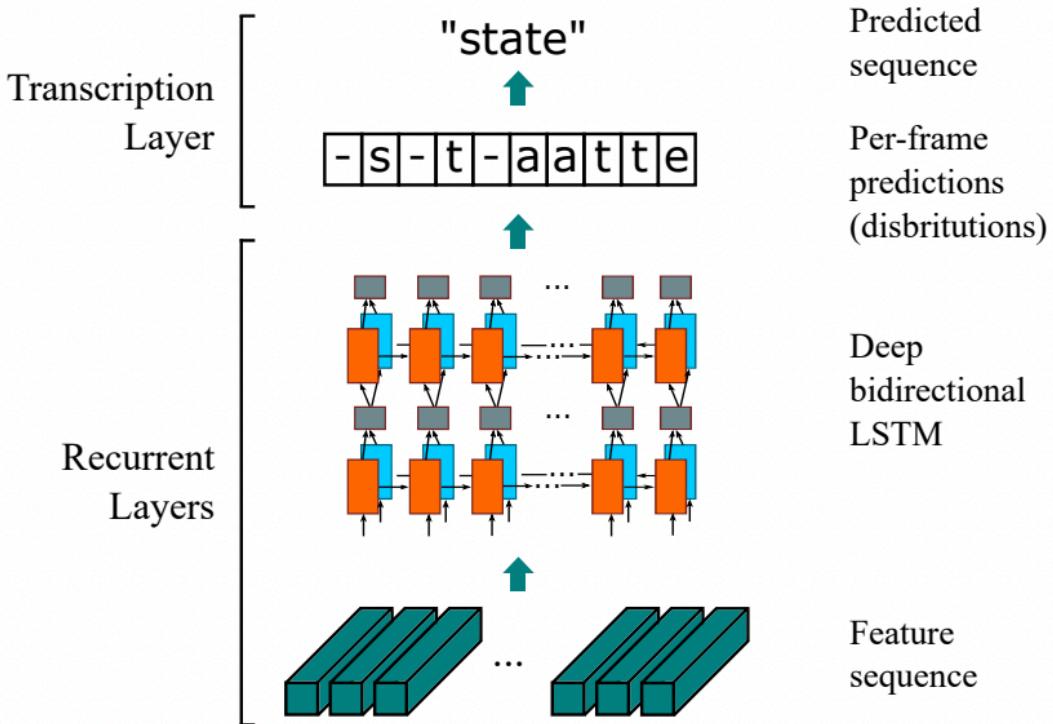


Figure 6: CRNN structure diagram

Sequence2Sequence Algorithms

In the Sequence2Sequence algorithms, the Encoder encodes all input sequences into unified semantic vectors, which are then decoded by the Decoder. In the decoding process, every output at the time step $t-1$ is continuously used as the input of the time step t , and the decoding is performed in a loop until the stop character is output. A general encoder is one RNN. For each input word, the encoder outputs a vector and hidden state, and uses the hidden state for the next input word to get the semantic vector in a loop; the decoder is another RNN, which receives output vectors of the encoder and output a series of words to create transformation. Inspired by Sequence2Sequence in translation, Shi [4] proposed an attention-based codec framework for text recognition. In this way, RNN can learn character-level language models hidden in strings from training data.

Figure 7: Sequence2Sequence structure diagram

The two kinds of algorithms perform well when it comes to regular texts, but limited by the network design, the kind of methods is hard to solve the curved or oriented irregular text. To solve such problems, some algorithmic researchers have proposed improved algorithms based on the above two.

5.1.2 Irregular Text Recognition

- Irregular text recognition algorithms can be divided into 4 categories: Rectification-based methods, Attention-based methods, Segmentation-based methods, and Transformer-based methods.

Rectification-based Methods

The Rectification-based method adopts some visual transformation modules to convert the irregular text into the regular text as much as possible, and then uses conventional recognition methods .

The RARE [4] model has first proposed a rectification scheme for irregular text. The network is of two main parts: an STN (Spatial Transformer Network) and a recognition network based on Sequence2Sequence. STN is the rectification module. An Irregular text image enters into STN and turns into a horizontal image through TPS (Thin-Plate-Spline). This transformation can rectify curved and transmissive texts to some extent, Then the rectify image will be sent to the sequence recognition network for decoding.

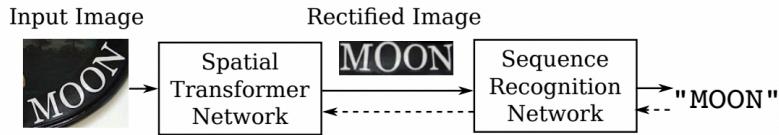


Figure 8: RARE structure diagram

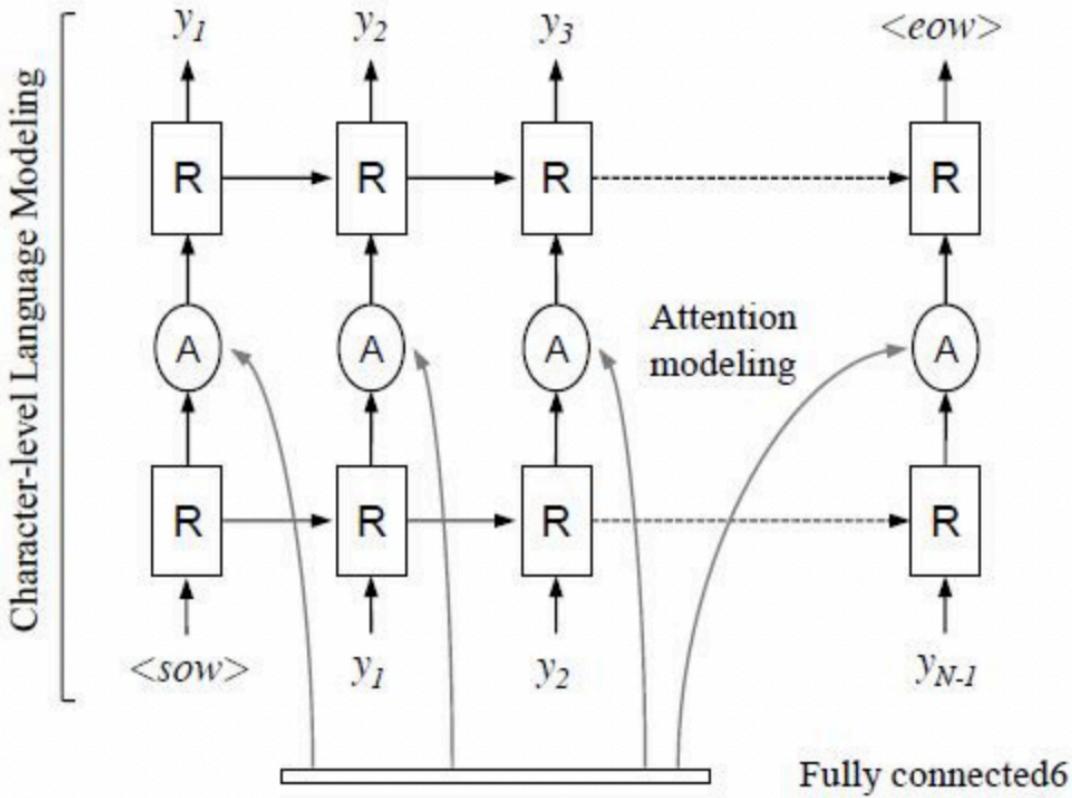
The paper of RARE points out that this method has greater advantages in irregular text datasets. The two datasets CUTE80 and SVTP are compared, and they are more than 5 percentage points higher than CRNN, proving that the rectification module works. Based on this, [6] also combines an STN with a text recognition system based on the attention sequence recognition network.

The rectification-based method is more flexible in migration. In addition to attention-based methods such as RARE, STAR-Net [5] applies the rectification module to the CTC-based algorithm, which has improved a lot compared with CRNN.

Attention-based Methods

The attention-based method concentrates on the correlation between sequences. This method was first proposed in machine translation. It is believed that the translation is mainly affected by certain words, and decisive words should be offered more weights. The same goes for text recognition. When decoding encoded sequences, each step selects the appropriate context to generate the next state, which can obtain more accurate results.

R²AM [7] is the first to introduced Attention into text recognition. The model first extracts the encoded image features from the input image through a recursive convolutional layer, and then uses the implicitly learned character-level linguistic statistics to decode the output character through a recurrent neural network. In the decoding process, for better utilization of image features, the attention mechanism is introduced to realize soft feature selection. This selective processing is closer to human intuition.

Figure 9: $R^2 AM$ structure drawing

A large number of algorithms have explored and updated in the field of attention. For example, SAR[8] have extended 1D attention to 2D attention. The RARE mentioned above is also based on attention. Experiments have also shown that the attention-based method has greater accuracy compared with the CTC method.

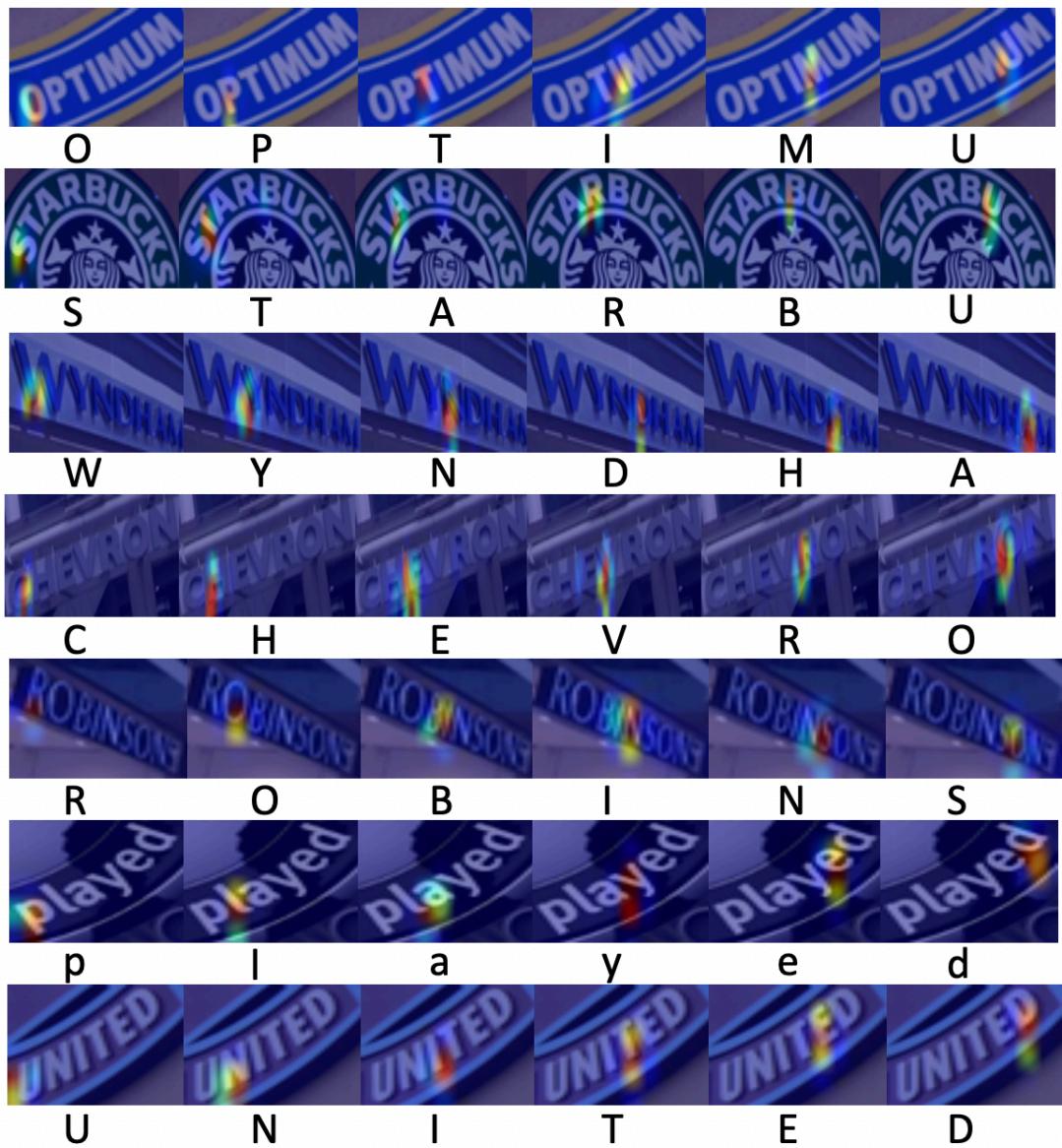


Figure 10: Attention diagram

Segmentation-based Methods

The Segmentation-based method is to treat each character of a text line as an individual unit, which can recognize the segmented characters more easily than to recognize the rectified entire text line. It tries to locate each character in the input text image, and uses a character classifier to obtain the recognition results, simplifying the complex global problem into a local problem and it works well in the irregular text scenes. However, this method requires character-level labeling which is relatively difficult in acquisition. Lyu [9] et al. propose an example-based word segmentation model to recognize words, which adopts the FCN (Fully Convolutional Network) based method in its recognition part. In [10] a character attention FCN is designed to solve the problem of text recognition from a two-dimensional perspective. When the text is curved or severely distorted, this method can perform better in positioning both regular and irregular texts.

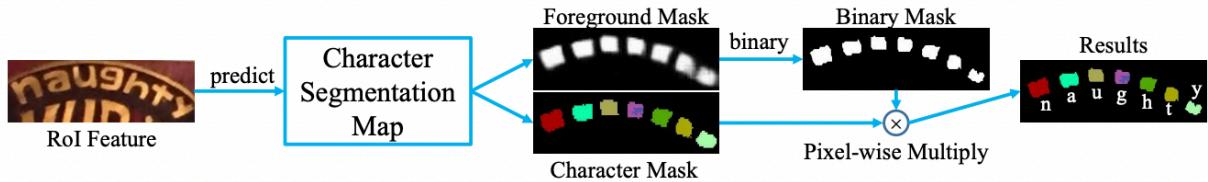


Figure 11: Mask TextSpotter structure diagram

Transformer-based Methods

With the rapid development of transformer, the effectiveness of transformer in visual tasks has been verified in classification and detection. As mentioned in the regular text recognition, CNN has limitations in modeling long-term dependency. The transformer structure can solve this problem. It can pay attention to global information in the feature extractor and replace additional context modeling modules (LSTM).

Some text recognition algorithms use the transformer's encoder structure and convolution to extract sequence features. The encoder is composed of multiple blocks stacked by MultiHeadAttention Layer and Positionwise Feedforward Layer. The self-attention in MulitHeadAttention uses matrix multiplication to simulate the time-series computation of RNN, liberating it from the long-term dependence on time series in RNN. There are also some algorithms using the transformer's decoder for decoding. It can obtain stronger semantic information than RNN and enjoys higher efficiency in parallel computing.

The SRN[11] algorithm connects the encoder of transformer to ResNet50 to enhance the 2D visual features. Also, it puts forward a parallel attention module, which uses the reading order as a query to make the calculation independent of time, and parallelly outputs aligned visual features of all time steps. In addition, SRN also uses transformer's encoder as a semantic module to integrate visual information and semantic information of the picture, more effective in dealing with the covered and blurred irregular texts.

NRTR [12] uses a complete transformer structure to encode and decode the input picture, and only uses a few simple convolutional layers for high-level feature extraction. The transformer structure has been proved effective in text recognition.

SRACN [13] uses transformer's decoder to replace LSTM, which once again verifies that parallel training is efficient and accurate.

5.2 Practice of a Text Recognition CRNN

The theory presented in the last section introduces the main methods of text recognition. And the CRNN is among the ones that have been early proposed and more widely used in the industrial sector. This chapter will elaborate on how to build, train, evaluate and predict the CRNN text recognition model based on PaddleOCR. The dataset is icdar 2015, in which there are 4468 training sets and 2077 test sets.

In this section, you can learn:

1. How to use PaddleOCR whl packages to quickly predict text recognition,
2. The basic principles and network structure of CRNN,
3. The necessary steps and parameter adjustment methods of model training,
4. How to use a custom dataset to train the network.

5.2.1 Quick Start

Installation of Related Dependencies and WHL Packages

First, confirm that paddle and paddleocr are installed. If it is done, skip this step.

```
# Install PaddlePaddle GPU version
!pip install paddlepaddle-gpu
# Install PaddleOCR whl package
!pip install -U pip
!pip install paddleocr
```

Quick Prediction of Content

The PaddleOCR whl package will automatically download the ppocr lightweight model as the default model.

The following shows how to use the whl package for recognition prediction:

Test picture:



```
from paddleocr import PaddleOCR

ocr = PaddleOCR() # need to run only once to download and load model into memory
img_path = '/home/aistudio/work/word_19.png'
result = ocr.ocr(img_path, det=False)
for line in result:
    print(line)
```

After executing the above code, the recognition result and recognition confidence will be returned.

```
('SLOW', 0.9776376)
```

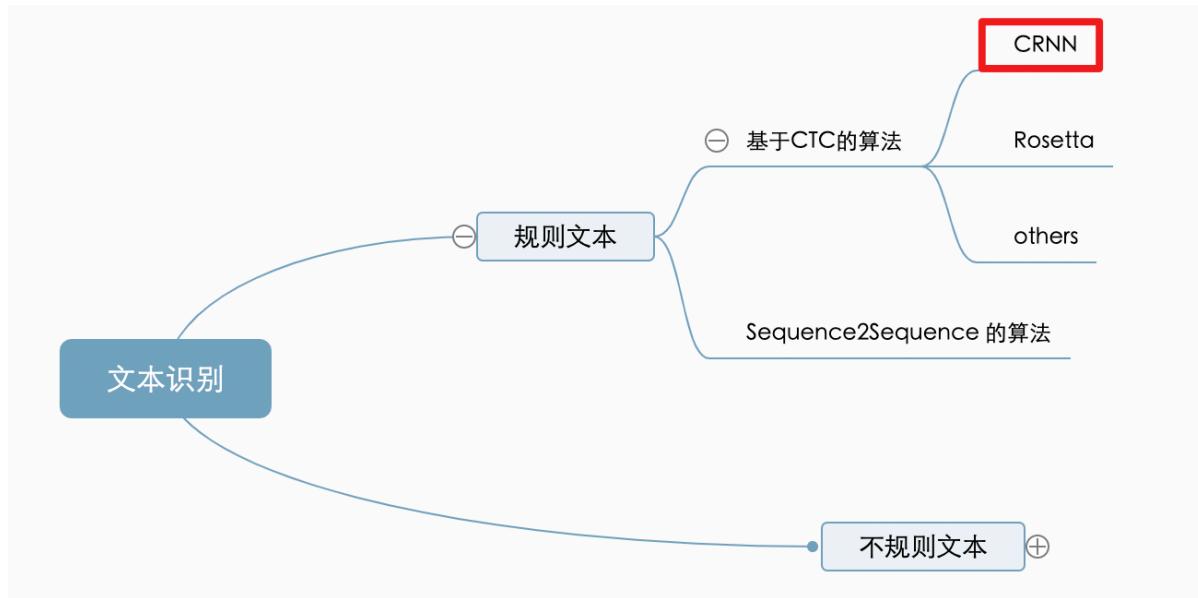
So far, you have learned how to use the PaddleOCR whl package to make predictions. There are more test pictures in the ./work/ path, so you can try other picture results.

5.2.2 Detailed Implementation of CRNN Text Recognition Algorithm

In the 4.2.1 section, paddleocr has loaded the trained CRNN recognition model for prediction. And this section will introduce the principles and process of CRNN in detail.

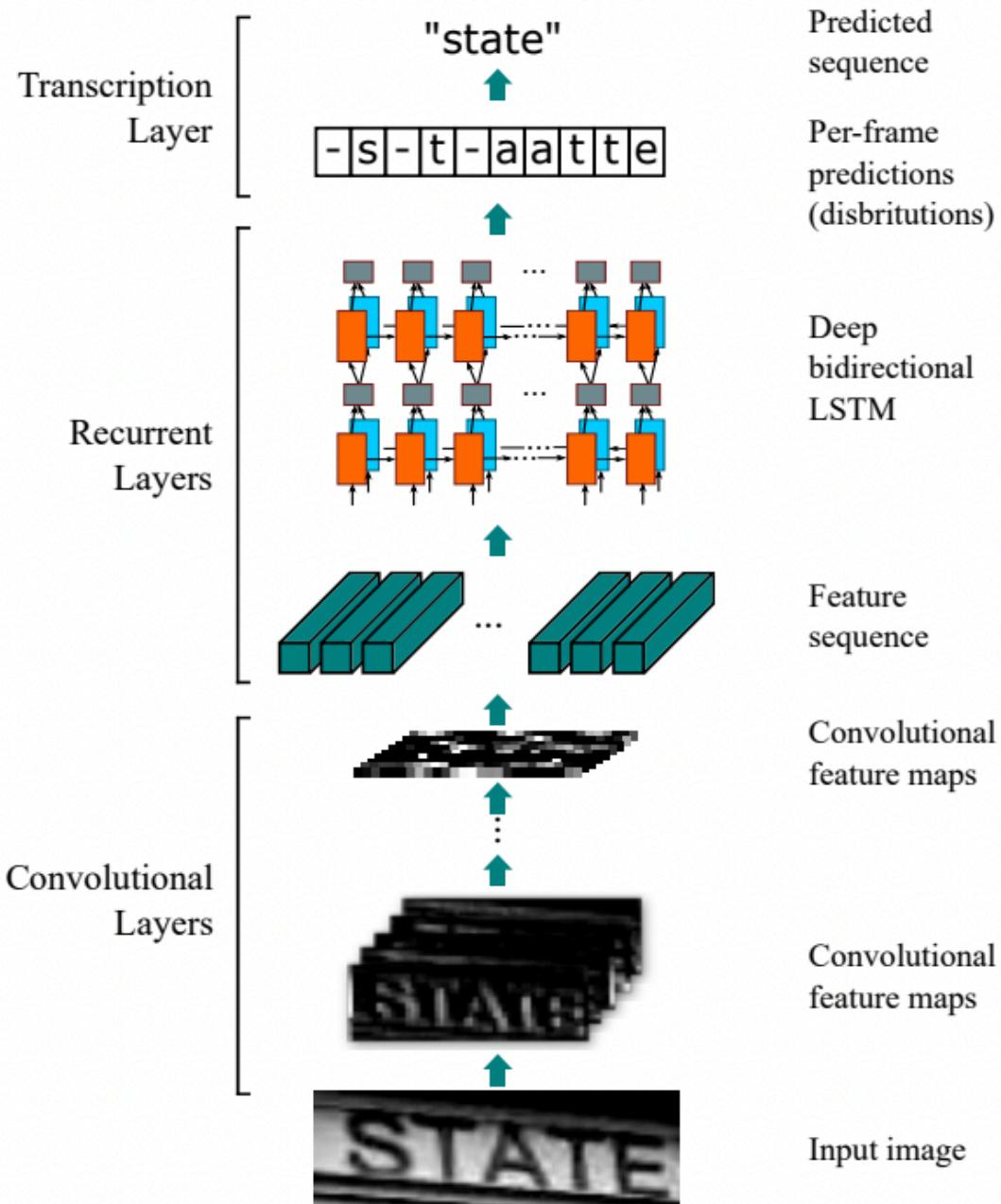
Category

CRNN is a CTC-based algorithm. Its position in the classification diagram presented in the theory part is as follows. It can be seen that CRNN is mainly used to deal with regular texts, and the CTC-based algorithm is faster in prediction speed and fits into long texts. Therefore, CRNN is chosen by PPOCR to recognize Chinese characters.



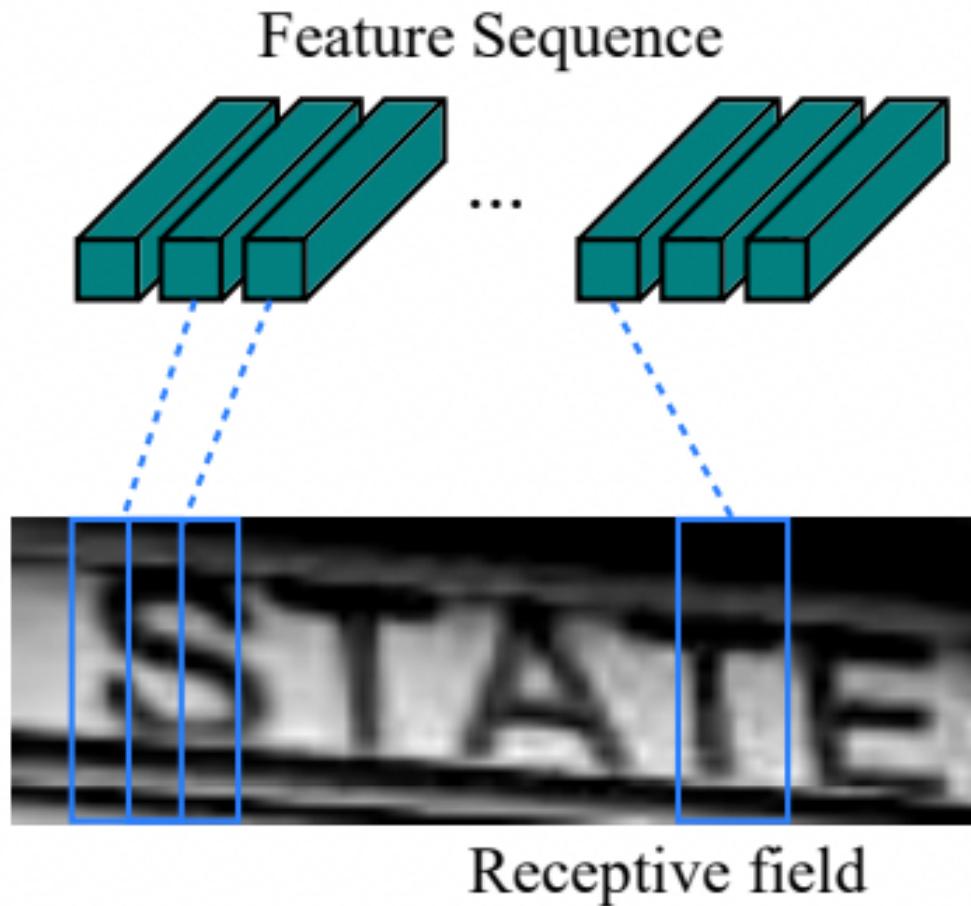
Detailed Introduction of the Algorithm

CRNN's network structure system is as follows. From the bottom to the top, there are three parts: convolutional layers, recurrent layers, and transcription layers:



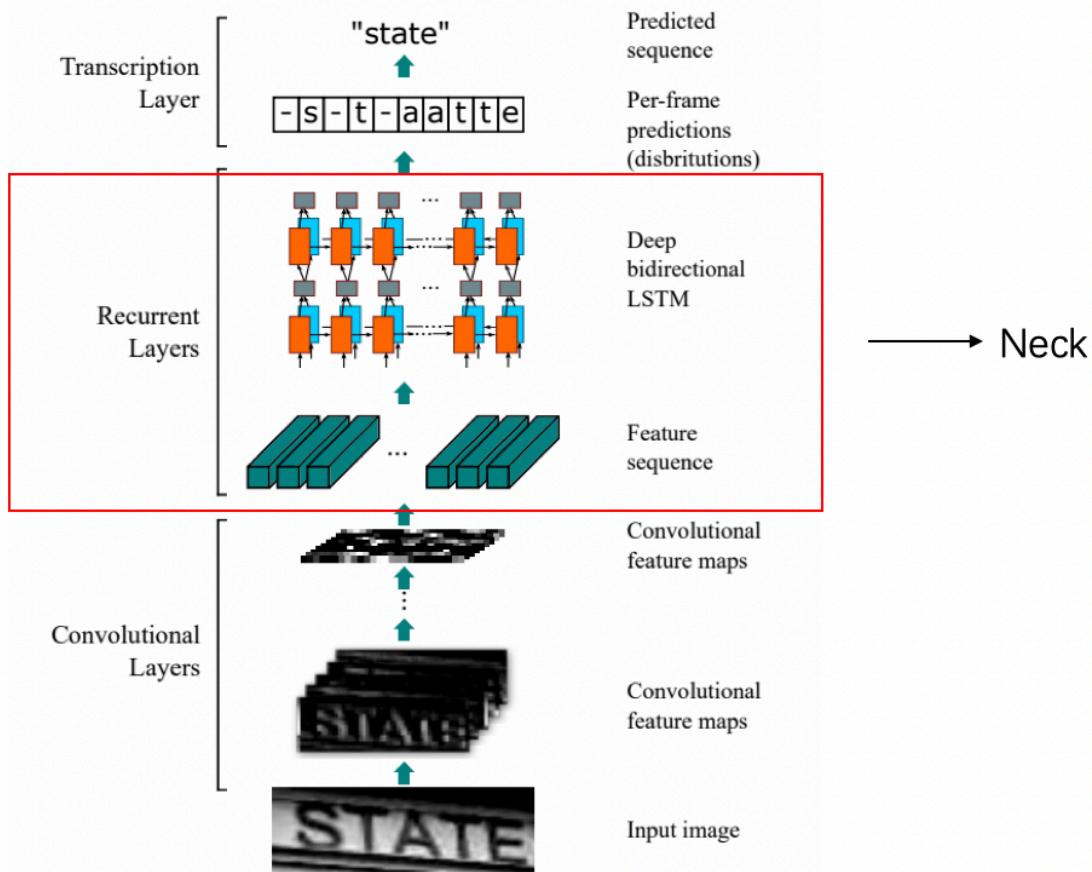
1. Backbone:

As an underlying backbone network, the convolutional network is used to extract feature sequences from the input image. Since `conv`, `max-pooling`, `elementwise` and activation functions all act on local areas, they are translation invariant. Therefore, each column of the feature map corresponds to a rectangular area (called a receptive field) of the original image, and these rectangular areas are in the same order from left to right as their corresponding columns on the feature map. Because in CNN needs to scale the input image to meet its fixed input dimensionality, it is not suitable for sequences that vary greatly in length. To better support sequences in variable lengths, CRNN sends the feature vector output from the last layer of backbone to the RNN layer and converts it into a sequence feature.



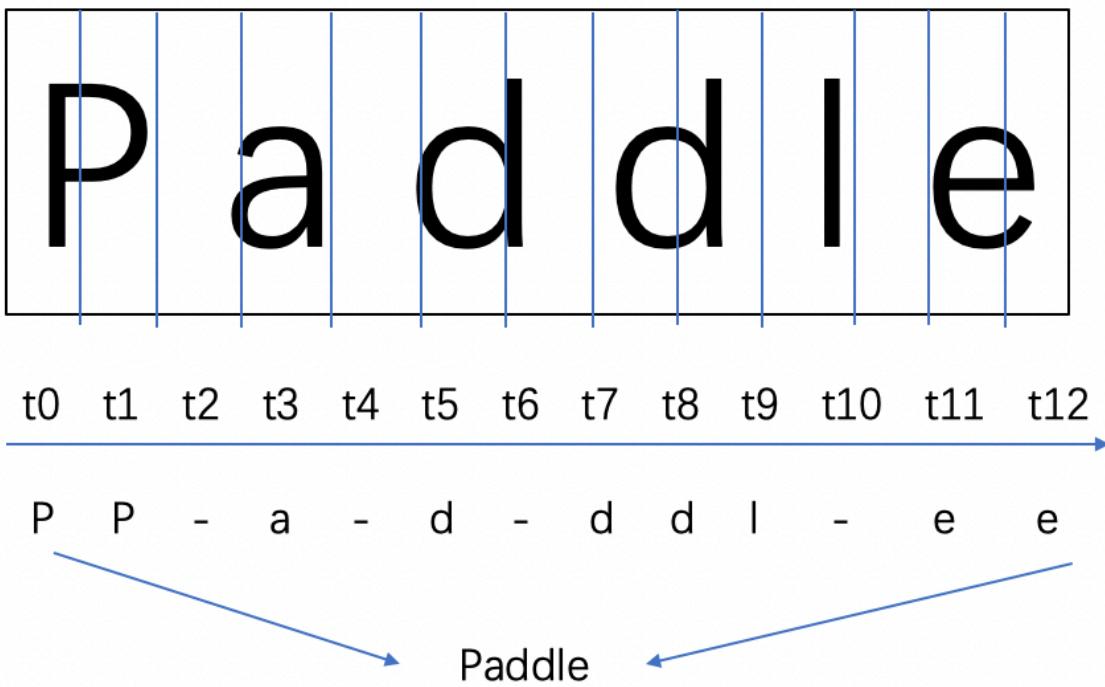
1. Neck:

Based on a convolutional network, the recurrent layer builds a recurrent network, converts image features into sequence features, and predicts the label distribution of each frame. RNN is very good at capturing context information of sequences. When it comes to Image-based sequence recognition, contextual cues are more useful than the individual processing of each pixel. Taking scene text recognition as an example, a wide character may require several consecutive frames for elaborate description. In addition, some ambiguous characters are easier to be clarified by observing the context. Second, RNN can propagate errors back to the convolutional layer, so that the network can be trained uniformly. Third, RNN can operate on sequences of any length, tackling the problem that text images become longer. And CRNN uses double-layer LSTM as the recurrent layer to solve the gradient disappearance and explosion in training long sequences.



1. Head:

The transcription layer converts the prediction of each frame into the final label sequence through the fully connected network and the softmax activation function. Finally, CTC Loss is used to complete the joint training of CNN and RNN without requiring sequence alignment. CTC has a special mechanism to merging sequences. After LSTM outputs the sequence, it needs to be classified in time sequence to obtain the prediction result. There may be multiple time steps corresponding to one category, so the same results need to be merged. In order to avoid merging the existing repeated characters, CTC introduced a blank character which is inserted between the repeated ones.



Code Example

The entire network structure is concise, and the code implementation is relatively simple. Modules can be built in sequence following the predicted process. In this section, four things needs to be finished: data input, backbone building, neck building, and head building.

[Data Input]

The data needs to be scaled to a uniform size (3,32,320) and normalized before being sent to the network. The data augmentation part required in the training is elided for brevity here, and the necessary steps of preprocessing are exemplified in a single [imagesource code location](#):

```
import cv2
import math
import numpy as np

def resize_norm_img(img):
    """
    Data scaling and normalization
    :param img: input picture
    """

    # Default input size
    imgC = 3
    imgH = 32
    imgW = 320

    # The real height and width of the picture
    h, w = img.shape[:2]
    # Picture real aspect ratio
```

(continues on next page)

(continued from previous page)

```

ratio = w / float(h)

# Scale
if math.ceil(imgH * ratio) > imgW:
    # If greater than the default width, the width is imgW
    resized_w = imgW
else:
    # If it is smaller than the default width, the actual width of the picture
    # shall prevail
    resized_w = int(math.ceil(imgH * ratio))
# Zoom in and out
resized_image = cv2.resize(img, (resized_w, imgH))
resized_image = resized_image.astype('float32')
# Normalize
resized_image = resized_image.transpose((2, 0, 1)) / 255
resized_image -= 0.5
resized_image /= 0.5
# For positions with insufficient width, add 0
padding_im = np.zeros((imgC, imgH, imgW), dtype=np.float32)
padding_im[:, :, 0:resized_w] = resized_image
# Transpose the image after padding for visualization
draw_img = padding_im.transpose((1, 2, 0))
return padding_im, draw_img

```

```

import matplotlib.pyplot as plt
# Read the picture
raw_img = cv2.imread("/home/aistudio/work/word_1.png")
plt.figure()
plt.subplot(2,1,1)
# Visualize the original image
plt.imshow(raw_img)
# Scale and normalize
padding_im, draw_img = resize_norm_img(raw_img)
plt.subplot(2,1,2)
# Visual network input diagram
plt.imshow(draw_img)
plt.show()

```

[Network Structure]

- Backbone

PaddleOCR adopts MobileNetV3 as the backbone network with a networking sequence consistent with the network structure. First, define the public modules in the network (source code location): ConvBNLayer, ResidualUnit, and make_divisible.

```

import paddle
import paddle.nn as nn
import paddle.nn.functional as F

class ConvBNLayer(nn.Layer):
    def __init__(self,
                 in_channels,
                 out_channels,
                 kernel_size,
                 stride,

```

(continues on next page)

(continued from previous page)

```

padding,
groups=1,
if_act=True,
act=None):
"""
Convolutional BN Layer
:param in_channels: number of input channels
:param out_channels: number of output channels
:param kernel_size: convolution kernel size
:para stride: stride size
:para padding: padding size
:param groups: the number of groups of the two-dimensional convolutional layer
:param if_act: whether to add activation function
:param act: activation function
"""
super(ConvBNLayer, self).__init__()
self.if_act = if_act
self.act = act
self.conv = nn.Conv2D(
    in_channels=in_channels,
    out_channels=out_channels,
    kernel_size=kernel_size,
    stride=stride,
    padding=padding,
    groups=groups,
    bias_attr=False)

self.bn = nn.BatchNorm(num_channels=out_channels, act=None)

def forward(self, x):
    # conv layer
    x = self.conv(x)
    # batchnorm layer
    x = self.bn(x)
    # whether to use the activation function
    if self.if_act:
        if self.act == "relu":
            x = F.relu(x)
        elif self.act == "hardswish":
            x = F.hardswish(x)
        else:
            print("The activation function({}) is selected incorrectly.".
                  format(self.act))
            exit()
    return x

class SEModule(nn.Layer):
    def __init__(self, in_channels, reduction=4):
        """
        SE module
        :param in_channels: number of input channels
        :param reduction: channel zoom ratio
        """
        super(SEModule, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2D(1)
        self.conv1 = nn.Conv2D(

```

(continues on next page)

(continued from previous page)

```

        in_channels=in_channels,
        out_channels=in_channels // reduction,
        kernel_size=1,
        stride=1,
        padding=0)
    self.conv2 = nn.Conv2D(
        in_channels=in_channels // reduction,
        out_channels=in_channels,
        kernel_size=1,
        stride=1,
        padding=0)

    def forward(self, inputs):
        # Average pooling
        outputs = self.avg_pool(inputs)
        # First convolutional layer
        outputs = self.conv1(outputs)
        # Relu activation function
        outputs = F.relu(outputs)
        # Second convolutional layer
        outputs = self.conv2(outputs)
        # Hardsigmoid activation function
        outputs = F.hardsigmoid(outputs, slope=0.2, offset=0.5)
        return inputs * outputs

    class ResidualUnit(nn.Layer):
        def __init__(self,
                     in_channels,
                     mid_channels,
                     out_channels,
                     kernel_size,
                     stride,
                     use_se,
                     act=None):
            """
            Residual layer
            :param in_channels: number of input channels
            :param mid_channels: number of intermediate channels
            :param out_channels: Number of output channels
            :param kernel_size: convolution kernel size
            :param stride: stride size
            :param use_se: whether to use se module
            :param act: activation function
            """
            super(ResidualUnit, self).__init__()
            self.if_shortcut = stride == 1 and in_channels == out_channels
            self.if_se = use_se

            self.expand_conv = ConvBNLayer(
                in_channels=in_channels,
                out_channels=mid_channels,
                kernel_size=1,
                stride=1,
                padding=0,
                if_act=True,

```

(continues on next page)

(continued from previous page)

```

        act=act)
    self.bottleneck_conv = ConvBNLayer(
        in_channels=mid_channels,
        out_channels=mid_channels,
        kernel_size=kernel_size,
        stride=stride,
        padding=int((kernel_size - 1) // 2),
        groups=mid_channels,
        if_act=True,
        act=act)
    if self.if_se:
        self.mid_se = SEModule(mid_channels)
    self.linear_conv = ConvBNLayer(
        in_channels=mid_channels,
        out_channels=out_channels,
        kernel_size=1,
        stride=1,
        padding=0,
        if_act=False,
        act=None)

    def forward(self, inputs):
        x = self.expand_conv(inputs)
        x = self.bottleneck_conv(x)
        if self.if_se:
            x = self.mid_se(x)
        x = self.linear_conv(x)
        if self.if_shortcut:
            x = paddle.add(inputs, x)
        return x

def make_divisible(v, divisor=8, min_value=None):
    """
    Make sure to be divisible by 8
    """
    if min_value is None:
        min_value = divisor
    new_v = max(min_value, int(v + divisor / 2) // divisor * divisor)
    if new_v < 0.9 * v:
        new_v += divisor
    return new_v

```

Use public modules to build backbone networks:

```

class MobileNetV3(nn.Layer):
    def __init__(self,
                 in_channels=3,
                 model_name='small',
                 scale=0.5,
                 small_stride=None,
                 disable_se=False,
                 **kwargs):
        super(MobileNetV3, self).__init__()
        self.disable_se = disable_se

```

(continues on next page)

(continued from previous page)

```

small_stride = [1, 2, 2, 2]

if model_name == "small":
    cfg = [
        # k, exp, c, se, nl, s,
        [3, 16, 16, True, 'relu', (small_stride[0], 1)],
        [3, 72, 24, False, 'relu', (small_stride[1], 1)],
        [3, 88, 24, False, 'relu', 1],
        [5, 96, 40, True, 'hardswish', (small_stride[2], 1)],
        [5, 240, 40, True, 'hardswish', 1],
        [5, 240, 40, True, 'hardswish', 1],
        [5, 120, 48, True, 'hardswish', 1],
        [5, 144, 48, True, 'hardswish', 1],
        [5, 288, 96, True, 'hardswish', (small_stride[3], 1)],
        [5, 576, 96, True, 'hardswish', 1],
        [5, 576, 96, True, 'hardswish', 1],
    ]
    cls_ch_squeeze = 576
else:
    raise NotImplementedError("mode[" + model_name +
                             "_model] is not implemented!")

supported_scale = [0.35, 0.5, 0.75, 1.0, 1.25]
assert scale in supported_scale, \
    "supported scales are {} but input scale is {}".format(supported_scale,_
scale)

inplanes = 16
# conv1
self.conv1 = ConvBNLayer(
    in_channels=in_channels,
    out_channels=make_divisible(inplanes * scale),
    kernel_size=3,
    stride=2,
    padding=1,
    groups=1,
    if_act=True,
    act='hardswish')
i = 0
block_list = []
inplanes = make_divisible(inplanes * scale)
for (k, exp, c, se, nl, s) in cfg:
    se = se and not self.disable_se
    block_list.append(
        ResidualUnit(
            in_channels=inplanes,
            mid_channels=make_divisible(scale * exp),
            out_channels=make_divisible(scale * c),
            kernel_size=k,
            stride=s,
            use_se=se,
            act=nl))
    inplanes = make_divisible(scale * c)
    i += 1
self.blocks = nn.Sequential(*block_list)

```

(continues on next page)

(continued from previous page)

```

    self.conv2 = ConvBNLayer(
        in_channels=inplanes,
        out_channels=make_divisible(scale * cls_ch_squeeze),
        kernel_size=1,
        stride=1,
        padding=0,
        groups=1,
        if_act=True,
        act='hardswish')

    self.pool = nn.MaxPool2D(kernel_size=2, stride=2, padding=0)
    self.out_channels = make_divisible(scale * cls_ch_squeeze)

def forward(self, x):
    x = self.conv1(x)
    x = self.blocks(x)
    x = self.conv2(x)
    x = self.pool(x)
    return x

```

Here, the definition of the backbone network is completed, and the entire network structure can be visualized through the paddle.summary structure:

```

# Define the network input shape
IMAGE_SHAPE_C = 3
IMAGE_SHAPE_H = 32
IMAGE_SHAPE_W = 320

# Visualize the network structure
paddle.summary(MobileNetV3(), [(1, IMAGE_SHAPE_C, IMAGE_SHAPE_H, IMAGE_SHAPE_W)])

```

```

# Input pictures in the backbone network
backbone = MobileNetV3()
# Convert numpy data to Tensor
input_data = paddle.to_tensor([padding_im])
# Output the Backbone network
feature = backbone(input_data)
# View the latitude of the feature map
print("backbone output:", feature.shape)

```

- Neck

The neck converts the visual feature map output by the backbone into a 1-dimensional vector input and sends it to the LSTM network, and outputs the sequence feature ([source code location](#)):

```

class Im2Seq(nn.Layer):
    def __init__(self, in_channels, **kwargs):
        """
        The image feature is converted into the sequence feature
        :param in_channels: number of input channels
        """
        super().__init__()
        self.out_channels = in_channels

```

(continues on next page)

(continued from previous page)

```

def forward(self, x):
    B, C, H, W = x.shape
    assert H == 1
    x = x.squeeze(axis=2)
    x = x.transpose([0, 2, 1]) # (NWC) (batch, width, channels)
    return x

class EncoderWithRNN(nn.Layer):
    def __init__(self, in_channels, hidden_size):
        super(EncoderWithRNN, self). __init__()
        self.out_channels = hidden_size * 2
        self.lstm = nn.LSTM(
            in_channels, hidden_size, direction='bidirectional', num_layers=2)

    def forward(self, x):
        x, _ = self.lstm(x)
        return x

class SequenceEncoder(nn.Layer):
    def __init__(self, in_channels, hidden_size=48, **kwargs):
        """
        Sequence encoding
        :param in_channels: number of input channels
        :param hidden_size: hidden layer size
        """
        super(SequenceEncoder, self). __init__()
        self.encoder_reshape = Im2Seq(in_channels)

        self.encoder = EncoderWithRNN(
            self.encoder_reshape.out_channels, hidden_size)
        self.out_channels = self.encoder.out_channels

    def forward(self, x):
        x = self.encoder_reshape(x)
        x = self.encoder(x)
        return x

```

```

neck = SequenceEncoder(in_channels=288)
sequence = neck(feature)
print("sequence shape:", sequence.shape)

```

- Head The prediction head is composed of a fully connected layer and softmax, which are used to calculate the label probability distribution on the time step of the sequence feature. This example only supports the model to recognize 36 categories of lowercase English letters and numbers (26+10) (source code location):

```

class CTCHead(nn.Layer):
    def __init__(self,
                in_channels,
                out_channels,
                **kwargs):
        """
        CTC prediction layer
        :param in_channels: number of input channels
        :param out_channels: number of output channels
        """

```

(continues on next page)

(continued from previous page)

```

"""
super(CTCHead, self).__init__()
self.fc = nn.Linear(
    in_channels,
    out_channels)

# Thinking: How much should be out_channels?
self.out_channels = out_channels

def forward(self, x):
    predicts = self.fc(x)
    result = predicts

    if not self.training:
        predicts = F.softmax(predicts, axis=2)
        result = predicts

    return result

```

When the network is randomly initialized, the output results are unordered. Through SoftMax, the prediction results with the highest probability at each time step can be obtained, where: `pred_id` represents the predicted tag ID, and `pre_scores` represents the confidence of the predicted result:

```

ctc_head = CTCHead(in_channels=96, out_channels=37)
predict = ctc_head(sequence)
print("predict shape:", predict.shape)
result = F.softmax(predict, axis=2)
pred_id = paddle.argmax(result, axis=2)
pred_socres = paddle.max(result, axis=2)
print("pred_id:", pred_id)
print("pred_scores:", pred_socres)

```

- Post-processing

The final result returned by the recognition network is the maximum index value at each time step, and the final expected output is the corresponding text result. Therefore, the post-processing of CRNN is decoding. The main logic is as follows:

```

def decode(text_index, text_prob=None, is_remove_duplicate=False):
    """ convert text-index into text-label. """
    character = "-0123456789abcdefghijklmnopqrstuvwxyz"
    result_list = []
    # Ignore tokens [0] represents the blank bit in ctc
    ignored_tokens = [0]
    batch_size = len(text_index)
    for batch_idx in range(batch_size):
        char_list = []
        conf_list = []
        for idx in range(len(text_index[batch_idx])):
            if text_index[batch_idx][idx] in ignored_tokens:
                continue
            # Merge the same characters between blanks
            if is_remove_duplicate:
                # only for predict
                if idx > 0 and text_index[batch_idx][idx - 1] == text_index[
                    batch_idx][idx]:
                    continue

```

(continues on next page)

(continued from previous page)

```

# Store the decoded result in char_list
char_list.append(character[int(text_index[batch_idx][
    idx])])
# Record confidence
if text_prob is not None:
    conf_list.append(text_prob[batch_idx][idx])
else:
    conf_list.append(1)
text = ''.join(char_list)
# Output the result
result_list.append((text, np.mean(conf_list)))
return result_list

```

Take the predicted result through the random initialization in the head as an example, and decode it to get:

```

pred_id = paddle.argmax(result, axis=2)
pred_socres = paddle.max(result, axis=2)
print(pred_id)
decode_out = decode(pred_id, pred_socres)
print("decode out:", decode_out)

```

Quick test: If the index of the input model is trained, is the decoding result correct?

```

# Replace the predicted result of the model
right_pred_id = paddle.to_tensor([[['xxxxxxxxxxxxxx']]])
tmp_scores = paddle.ones(shape=right_pred_id.shape)
out = decode(right_pred_id, tmp_scores)
print("out:", out)

```

The above steps have built the network and also realized a simple forward prediction.

The untrained network cannot predict the result correctly. Therefore, it is necessary to define the loss function and optimization strategy to run the entire network. The network training principle will be described in detail below.

5.2.3 Training the CRNN Text Recognition Model

Preparation of Data Training

PaddleOCR supports two data formats: -lmdb is used to train the dataset (LMDDBDataSet) stored in lmdb format; -General Data is used to train a dataset (SimpleDataSet) stored in a text file;

Here only introduces the reading of the general data format

The default storage path of training data is ./train_data, execute the following command to decompress the data:

```
!cd /home/aistudio/work/train_data/ && tar xf ic15_data.tar
```

After the decompression, the training images are in the same folder, and there is a txt file (rec_gt_train.txt) recording paths and labels of the images. The contents of the txt file are as follows:

"Image file name	Image annotation information"
train/word_1.png	Genaxis Theatre

(continues on next page)

(continued from previous page)

```
train/word_2.png      [06]
...
```

Note: In txt files, picture paths and labels are divided by \t by default. If they are divided by other methods, it will cause training errors.

The data set should have the following file structure:

```
|-train_data
  |-ic15_data
    |- rec_gt_train.txt
    |- train
      |- word_001.png
      |- word_002.jpg
      |- word_003.jpg
      | ...
    |- rec_gt_test.txt
    |- test
      |- word_001.png
      |- word_002.jpg
      |- word_003.jpg
      | ...
```

Confirm whether data paths in the configuration file are correct, take `rec_icdar15_train.yml` as an example:

```
Train:
  dataset:
    name: SimpleDataSet
    # Training the data root directory
    data_dir: ./train_data/ic15_data/
    # Training data labels
    label_file_list: ["./train_data/ic15_data/rec_gt_train.txt"]
    transforms:
      - DecodeImage: # load image
        img_mode: BGR
        channel_first: False
      - CTCLabelEncode: # Class handling label
      - RecResizeImg:
        image_shape: [3, 32, 100] # [3, 32, 320]
      - KeepKeys:
        keep_keys: ['image', 'label', 'length'] # dataloader will return list in
        ↪this order
    loader:
      shuffle: True
      batch_size_per_card: 256
      drop_last: True
      num_workers: 8
      use_shared_memory: False

Eval:
  dataset:
    name: SimpleDataSet
    # Evaluation of the data root directory
    data_dir: ./train_data/ic15_data
    # Evaluation of data labels
    label_file_list: ["./train_data/ic15_data/rec_gt_test.txt"]
```

(continues on next page)

(continued from previous page)

```

transforms:
    - DecodeImage: # load image
        img_mode: BGR
        channel_first: False
    - CTCLabelEncode: # Class handling label
    - RecResizeImg:
        image_shape: [3, 32, 100]
    - KeepKeys:
        keep_keys: ['image', 'label', 'length'] # dataloader will return list in_
        ↪this order
loader:
    shuffle: False
    drop_last: False
    batch_size_per_card: 256
    num_workers: 4
    use_shared_memory: False

```

Data Preprocessing

The training data sent to the network must be consistent in dimension within a batch. At the same time, in order to ensure features between different dimensions are comparable in numbers, the data needs to be uniformly **scaled** and **normalized**.

To increase the robustness of the model, suppress overfitting and improve generalization performance, a certain **data augmentation** needs to be implemented.

- Scaling and normalization

Related content has been introduced in the second section. This is the last step before pictures are sent to the network. Call `resize_norm_img` to complete image scaling, padding and normalization.

- Data augmentation

A variety of data augmentation methods are implemented in PaddleOCR such as: color inversion, random segmentation, affine transformation, random noise, etc. Here is an example of simple random cutting, more augmentation methods can be referred to: `rec_img_aug.py`

```

def get_crop(image) :
    """
    random crop
    """
    import random
    h, w, _ = image.shape
    top_min = 1
    top_max = 8
    top_crop = int(random.randint(top_min, top_max))
    top_crop = min(top_crop, h - 1)
    crop_img = image.copy()
    ratio = random.randint(0, 1)
    if ratio:
        crop_img = crop_img[top_crop:h, :, :]
    else:
        crop_img = crop_img[0:h - top_crop, :, :]
    return crop_img

```

```

# Read the picture
raw_img = cv2.imread("/home/aistudio/work/word_1.png")
plt.figure()
plt.subplot(2,1,1)
# Visualize the original image
plt.imshow(raw_img)
# Cut randomly
crop_img = get_crop(raw_img)
plt.subplot(2,1,2)
# Visualize the augmentation graph
plt.imshow(crop_img)
plt.show()

```

Training of the Main Program

The entry code of model training is `train.py`, which shows the various modules required in the training: build dataloader, build post process, build model, build loss, build optim, build metric. After connecting all the parts, you can start training:

- Building dataloader

The training model requires the data to be formed into a specified number of batches, which are sequentially yielded during the training process. The `SimpleDataSet` implemented in PaddleOCR is used in this example.

After slightly modifying the original code, the main logic of returning one piece of data is as follows:

```

def __getitem__(data_line, data_dir):
    import os
    mode = "train"
    delimiter = '\t'
    try:
        substr = data_line.strip("\n").split(delimiter)
        file_name = substr[0]
        label = substr[1]
        img_path = os.path.join(data_dir, file_name)
        data = {'img_path': img_path, 'label': label}
        if not os.path.exists(img_path):
            raise Exception("{} does not exist!".format(img_path))
        with open(data['img_path'], 'rb') as f:
            img = f.read()
            data['image'] = img
        # Pre-processing operation, comment out first
        # outs = transform(data, self.ops)
        outs = data
    except Exception as e:
        print("When parsing line {}, error happened with msg: {}".format(
            data_line, e))
        outs = None
    return outs

```

Suppose the current input label is `train/word_1.png` Genaxis Theatre and the path of the training data is `/home/aistudio/work/train_data/ic15_data/`. Then the analyzed result is a dictionary containing three fields: `img_path` `label` `image`.

```

data_line = "train/word_1.png      Genaxis Theatre"
data_dir = "/home/aistudio/work/train_data/ic15_data/"

item = __getitem__(data_line, data_dir)
print(item)

```

After the piece of data is returned, call `padde.io.Dataloader` to merge the data into a batch. For details, please refer to [build_dataloader](#).

- Build model

The build model is to build the main network structure. Its details are described in “2.3 Code Implementation”, and this section will not cover too much. For the code of each module, please refer to [modeling](#)

- Build loss

The loss function of the CRNN model is CTC loss, and PaddlePaddle has collected the commonly used loss functions. Implement them if needed:

```

import paddle.nn as nn
class CTCLoss(nn.Layer):
    def __init__(self, use_focal_loss=False, **kwargs):
        super(CTCLoss, self).__init__()
        # Blank is a meaningless connector for ctc
        self.loss_func = nn.CTCLoss(blank=0, reduction='none')

    def forward(self, predicts, batch):
        if isinstance(predicts, (list, tuple)):
            predicts = predicts[-1]
        # Transpose the prediction results of the head layer of the model and arrange
        # them along the channel layer
        predicts = predicts.transpose((1, 0, 2)) #[80, 1, 37]
        N, B, _ = predicts.shape
        preds_lengths = paddle.to_tensor([N] * B, dtype='int64')
        labels = batch[1].astype("int32")
        label_lengths = batch[2].astype('int64')
        # Calculate the loss function
        loss = self.loss_func(predicts, labels, preds_lengths, label_lengths)
        loss = loss.mean()
        return {'loss': loss}

```

- Build post process

The details are also introduced in “2.3 Code Implementation”, and the implementation logic is the same as before.

- Build optim

The optimizer uses Adam and also calls the API of PaddlePaddle: `paddle.optimizer.Adam`

- Build metric

The metric is used to calculate model indicators. In PaddleOCR’s text recognition, correct prediction of the whole sentence is equal to correct prediction. Therefore, the main logic of the accuracy rate calculation is as follows:

```

def metric(preds, labels):
    correct_num = 0
    all_num = 0
    norm_edit_dis = 0.0
    for (pred), (target) in zip(preds, labels):

```

(continues on next page)

(continued from previous page)

```

pred = pred.replace(" ", "")
target = target.replace(" ", "")
if pred == target:
    correct_num += 1
all_num += 1
correct_num += correct_num
all_num += all_num
return {
    'acc': correct_num / all_num,
}

```

```

preds = ["aaa", "bbb", "ccc", "123", "456"]
labels = ["aaa", "bbb", "ddd", "123", "444"]
acc = metric(preds, labels)
print("acc:", acc)
# Among the five prediction results, 3 are completely correct, so the accuracy rate
# should be 0.6

```

Combine the above parts and get the complete training process:

```

def main(config, device, logger, vdl_writer):
    # Init dist environment
    if config['Global']['distributed']:
        dist.init_parallel_env()

    global_config = config['Global']

    # Build dataloader
    train_dataloader = build_dataloader(config, 'Train', device, logger)
    if len(train_dataloader) == 0:
        logger.error(
            "No Images in train dataset, please ensure\n" +
            "\t1. The images num in the train label_file_list should be larger than\n" +
            "or equal with batch size.\n" +
            "\t2. The annotation file and path in the configuration file are provided\n" +
            "normally."
        )
        return

    if config['Eval']:
        valid_dataloader = build_dataloader(config, 'Eval', device, logger)
    else:
        valid_dataloader = None

    # Build post process
    post_process_class = build_post_process(config['PostProcess'],
                                             global_config)

    # Build model
    # For rec algorithm
    if hasattr(post_process_class, 'character'):
        char_num = len(getattr(post_process_class, 'character'))
        if config['Architecture']["algorithm"] in ["Distillation",
                                                   ]: # distillation model

```

(continues on next page)

(continued from previous page)

```
for key in config['Architecture']["Models"]:
    config['Architecture']["Models"][key]["Head"][
        'out_channels'] = char_num
else: # base rec model
    config['Architecture']["Head"]['out_channels'] = char_num

model = build_model(config['Architecture'])
if config['Global']['distributed']:
    model = paddle.DataParallel(model)

# Build loss
loss_class = build_loss(config['Loss'])

# Build optim
optimizer, lr_scheduler = build_optimizer(
    config['Optimizer'],
    epochs=config['Global']['epoch_num'],
    step_each_epoch=len(train_dataloader),
    parameters=model.parameters())

# Build metric
eval_class = build_metric(config['Metric'])

# Load pretrain model
pre_best_model_dict = load_model(config, model, optimizer)
logger.info('train dataloader has {} iters'.format(len(train_dataloader)))
if valid_dataloader is not None:
    logger.info('valid dataloader has {} iters'.format(
        len(valid_dataloader)))

use_amp = config["Global"].get("use_amp", False)
if use_amp:
    AMP RELATED FLAGS SETTING = {
        'FLAGS_cudnn_batchnorm_spatial_persistent': 1,
        'FLAGS_max_inplace_grad_add': 8,
    }
    paddle.fluid.set_flags(AMP RELATED FLAGS SETTING)
    scale_loss = config["Global"].get("scale_loss", 1.0)
    use_dynamic_loss_scaling = config["Global"].get(
        "use_dynamic_loss_scaling", False)
    scaler = paddle.amp.GradScaler(
        init_loss_scaling=scale_loss,
        use_dynamic_loss_scaling=use_dynamic_loss_scaling)
else:
    scaler = None

# Start training
program.train(config, train_dataloader, valid_dataloader, device, model,
              loss_class, optimizer, lr_scheduler, post_process_class,
              eval_class, pre_best_model_dict, logger, vdl_writer, scaler)
```

Starting Training

PaddleOCR recognition task is similar to the detection task, transmitting parameters through configuration files.

To perform a complete model training, first download the entire project and install related dependencies:

```
# Clone PaddleOCR code
#!git clone https://gitee.com/paddlepaddle/PaddleOCR
# Modify the default directory where the code runs to /home/aistudio/PaddleOCR
import os
os.chdir("/home/aistudio/PaddleOCR")
# Install PaddleOCR third-party dependencies
!pip install -r requirements.txt
```

Create a soft link and place the training data under the PaddleOCR project:

```
!ln -s /home/aistudio/work/train_data/ /home/aistudio/PaddleOCR/
```

Download the pre-trained model:

In order to speed up the convergence, it is recommended to download the trained model and finetune it on the icdar2015 dataset.

```
!cd PaddleOCR/
# Download the pre-trained model of MobileNetV3
!wget -nc -P ./pretrain_models/ https://paddleocr.bj.bcebos.com/dygraph_v2.0/en/rec_
_mv3_none_bilstm_ctc_v2.0_train.tar
# Decompress model parameters
!tar -xf pretrain_models/rec_mv3_none_bilstm_ctc_v2.0_train.tar && rm -rf pretrain_
_models/rec_mv3_none_bilstm_ctc_v2.0_train.tar
```

It is simple to start the training command by just specifying the configuration file. In addition, in the command line, you can use `-o` to modify the parameters in the configuration file. How to start the training command are shown below

in:

- `Global.pretrained_model`: loaded pretrained model path
- `Global.character_dict_path`: dictionary path (only 26 lowercase letters + numbers are supported here)
- `Global.eval_batch_step`: evaluation frequency
- `Global.epoch_num`: total number of training rounds

```
!python3 tools/train.py -c configs/rec/rec_icdar15_train.yml \
-o Global.pretrained_model=rec_mv3_none_bilstm_ctc_v2.0_train/best_accuracy \
Global.character_dict_path=ppocr/utils/ic15_dict.txt \
Global.eval_batch_step=[0,200] \
Global.epoch_num=40
```

According to the `save_model_dir` field set in the configuration file, the following parameters will be saved:

```
output/rec/ic15
├── best_accuracy.pdopt
├── best_accuracy.pdparams
├── best_accuracy.states
├── config.yml
└── iter_epoch_3.pdopt
```

(continues on next page)

(continued from previous page)

```
|── iter_epoch_3.pdparams
|── iter_epoch_3.states
|── latest.pdopt
|── latest.pdparams
|── latest.states
└── train.log
```

And `best_accuracy.*` is the optimum model on the evaluation set; `iter_epoch_x.*` is the model saved at intervals of `save_epoch_step`; `latest.*` is the last model of the epoch.

Summary:

Training your own data requires:

1. Training and evaluation of data paths (necessary)
2. Dictionary path (necessary)
3. Pre-trained model (optional)
4. Learning rate, image shape, and network structure (optional)

Model Evaluation

To evaluate datasets, you can modify the setting of `label_file_path` in `Eval` through `configs/rec/rec_icdar15_train.yml`.

The default evaluation set here is that of icdar2015 which is for loading weights of the newly trained model:

```
!python tools/eval.py -c configs/rec/rec_icdar15_train.yml -o Global.
    ↪checkpoints=output/rec/ic15/best_accuracy \
        Global.character_dict_path=ppocr/utils/ic15_dict.txt
```

After the evaluation, you can see the accuracy of the training model on the validation set.

PaddleOCR supports alternate training and evaluation. To make it, you can modify the evaluation frequency of `eval_batch_step` in `configs/rec/rec_icdar15_train.yml`. The default evaluation frequency is once every 2000 iter. In the evaluation process, the best acc model is saved as `output/rec/ic15/best_accuracy` by default.

If the validation set is large, the test will be more time-consuming. Under these circumstances, it is recommended to reduce evaluations or perform them after training.

Prediction

The model trained by PaddleOCR can go on quick predictions by the following script.

Prediction picture:



The default prediction image is stored in `infer_img`, and the trained parameter file is loaded through `-o Global.checkpoints`:

```
!python tools/infer_rec.py -c configs/rec/rec_icdar15_train.yml -o Global.
  ↪checkpoints=output/rec/ic15/best_accuracy Global.character_dict_path=ppocr/utils/
  ↪ic15_dict.txt
```

Get the prediction result of the input image:

```
infer_img: doc/imgs_words_en/word_19.png
  result: slow      0.8795223
```

5.2.4 Text Recognition FAQ

Universal Questions

Q1.1: What are the text recognition algorithms provided by PaddleOCR?

A: There are five major text recognition algorithms, including CRNN\StarNet\RARE\Rosetta and SRN. Among them, CRNN\StarNet\Rosetta are based on ctc, RARE is based on attention, and SRN is a Baidu's self-developed algorithm where semantic information is introduced and thus the accuracy has been improved. For details, please refer: text recognition algorithms.

Q1.2: What are the key CRNN technologies in text recognition?

A: There are three.(1) CNN image feature extraction, (2) deeply bi-directional LSTM network which further extracts sequence features of texts, (3) connectionist temporal classification(CTC), which can solve the problem of character alignment.

Q1.3 Which is better, CTC or Attention, to recognize text lines in Chinese?

A: (1) In terms of the effect, CTC performs better than Attention in common OCR scenarios. It is because that the dictionary to be recognized has many characters including more than 3,000 common Chinese characters. If lacking training samples, it will be difficult to figure out their sequences. Under these circumstances, the Attention model has no advantages. Also, the Attention model is more suitable for short sentence recognition instead of long sentence recognition.

(2) In terms of the speed of training and inference, compared with the serial decoder of Attention which limits the speed, the structure of CTC is more efficient and can achieve faster inference.

Q1.4 **How to recognize the curved text? What are the application scenarios of TPS? Does TPS work well?**

A: (1) In most cases, if the text is not severely curved, then detect its four vertexes and use affine transformation to rotate and recognize the image.

Q2 If this cannot meet your requirement, you can try TPS (Thin Plate Spline). TPS is an interpolation approach by using several control points to change images, usually used for image morphing. It is often adopted in the recognition of curved texts. When the text is detected to be irregular or curved by, for example, segment-based methods, TPS is the first choice to correct the text area into a rectangle and then recognize it. The TPS module has been introduced into STAR-Net and RARE algorithms.

Warning Though TPS works well in theory, it lacks the robustness in application and makes recognition more time-consuming. Therefore, think twice before using TPS.

The Application of Text Recognition in Vertical Scenarios

Q 2.1 **How to recognize texts blurred by the background (such as the signature on which the seal is stamped or the text of the seal) ?**

A (1) If the text is confirmed to be recognizable with the naked eye, you should first make sure that the detection box is correct. If not, it is necessary to consider pretraining images with color filtering and adding more relevant training data. During the recognition, add augmentation images where texts are blurred by the background to the training data.

(2) If the MobileNet cannot meet the demand, you can try larger models such as ResNet.

Q2.2 **Are there any image enhancement methods to handle some slightly blurred texts?**

A If the text is confirmed to be recognizable with the naked eye, you might consider adding blur data augmentation such as mean filter, the median filter, the gaussian filter or other fuzzy operators of image processing. You can also try to strengthen the robustness of models through data augmentation perturbation. It is also feasible to try adversarial training and super resolution (SR). But, there is no best solution which is widely accepted by professions. Therefore, it is recommended to set some limitations in data collection to improve the image quality.

Q2.3 **Are there any SR solutions to recognizing low-resolution texts or those with a small font size ?**

A Super resolution methods consist of traditional and deep-learning-based methods. Among DL-based ones, SRCNN is a classic solution, and there is related paper of CVPR 2020: Unpaired Image Super-Resolution using Pseudo-Supervision. But the paper is not proved by enough practice.

Q2.4 How to recognize long texts?

A  During the training of Chinese recognition models, instead of directly shrinking images to [3,32,320], you should scale them down in proportion to 32 in height. If their width is less than 320, filled it with 0. If the image ratio of width to height is more than 10, remove the image. In the inference, when it comes to the inference of a single image, scale down the image following the above instruction and don't set the limitation of the width; when it comes to the inference of several images, adopt the batch solution by choosing the maximum from the width range of every batch.

Q2.5 How to recognize English text lines with blanks?

A  There are two solutions to blank recognition:

(1)Optimize text detection algorithms. The text will be broken at blanks. This solution divides a text line containing blanks into many parts during the detection of data annotations.

(2)Optimize text recognition algorithms. Introduce space characters into the recognition dictionary, and then annotate the training data which uses a blank line. What's more, in data synthesis, produce texts with blanks by jointing training data.

Q2.6: Have curved texts been rectified by TPS of OpenCV?

A  The points of upper and lower boundaries are required to be labelled if you use TPS of OpenCV. The points can hardly be acquired by traditional or DL-based methods. In PaddleOCR, the TPS module of StarNet can learn the points and rectify texts automatically. You can have a try.

Q2.7: How to recognize the wordart on the signboard or the advertisement?

A: This is very challenging because the style of the wordart is very different from that of printed letters. If all the wordart is in the same dictionary list, you can turn each dictionary into an image template to be recognized, and then use the general image retrieval and recognition system. You can try the image recognition system of PaddleClas.

Q2.8: How to recognize the seal text?

A  1. Use the recognition network with TPS or ABCNet, 2. flatten the image through polar coordinate system transformation, and then use CRNN.

Q2.9: How to cope with poor performance of recognizing specific characters when using pretrained models for inference?

A: Since the provided recognition models are trained with universal large datasets, some characters might be rarely contained in the training set. You can build datasets for specific scenarios and fine-tune them based on the provided models. It is suggested the number of each character is no less than 300, and the number of different characters is balanced. For details, please refer to: fine-tuning.

Q2.10: How to handle the repeated characters in the inference of trained recognition models?

A: You can check whether the dimension of training is the same as that of inference. If the dimension of training is [3, 32, 320], and the dimension of inference is [3, 64, 640], there are many repeated characters being recognized.

Q2.11 How to recognize ancient Chinese characters on bamboo slips?

A: When they are common Chinese characters, just annotate the adequate amount of data and fine-tune models. If there is not enough data, you can try StyleText. But when they are special characters like inscriptions on oracle and hieroglyphs, it is necessary to make a specialized dictionary and train models.

Q2.12: How to recognize texts in videos with PaddleOCR?

A: Now PaddleOCR mainly deals with images. If you want to recognize texts in videos, you can first extract frames from videos and then use PaddleOCR for text recognition.

Q2.13: How to fine-tune the model when encountering characters incompatible with the Chinese-English recognition model?

A: You may need to update the recognition dictionary and fine-tune the model. If you also hope that the recognition model can cope with Roman numerals, please follow the steps here to fine-tune the model:

1. Prepare Chinese and English recognition data and Roman numerals for training, and ensure their quality;
 2. Rectify the default dictionary file by adding characters of Roman numerals at the end;
 3. Download the pretrained model offered by PaddleOCR, configurate the model and the data path, and then start the training.
-

Q2.14 [?] How to cope with poor recognition results of some special characters like some punctuation marks?

A [?] First you need to confirm whether the special characters are contained in the dictionary. If the characters are in the dictionary but the recognition results are still poor, you can increase relevant data to fine-tune the model since the poor performance might be caused by shortage of recognition data.

Q2.15 [?] How to handle an image with different kinds of texts (for example, there are printed texts and written words) ?

A [?] This is very common. Take an exam paper as an example. It has both printed texts and written words. Under this circumstance, you can adopt the solution of “1 detection model+ 1 N-class model+ N recognition models”. All types of texts share one detection model, and the N-class model which is a classifier in the extra training classifies the texts. If there are printed and written texts, the model is a two-class model; if there are N types of texts, the model is an N-class model. When it comes to recognition, every text type trains a recognition model. If there are printed and written texts in an image, two recognition models are needed, one for printed texts and the other for written texts. Then, when a text box is classified as a written text, then it will be recognized by the specialized model. The same goes for other cases.

Q2.16 Is there anything special about the dictionary containing different types of texts? How much loss of accuracy will the diversity cause?

A [?] It may lead to the oversized FC in the last layer and the model size increase. If you have any special requirements, you can merge dictionaries of different text types you need into one dictionary and use it to train the model. But if you introduce too many close words, there may be a loss of accuracy. And the issue of character balance also needs to be taken into account. You can separate the dictionaries in PaddleOCR for the time being.

Q2.17 [?] In some small languages like Thai, one word may take up two to three characters. So how to make a dictionary in such cases?

A [?] See characters of the same word as a whole. Make sure there is only one word in each line.

Training Process and Model Optimization

Q3.1: Increasing batch_size cannot largely improve the model training speed.

A [?] In this case, you can consider increase the value of the initial memory and set the environment variable before running the code: `export FLAGS_initial_cpu_memory_in_mb=2000 # Set the initial memory to about 2G.`

Q3.2: What to do if there are reminders of the oversized image and video and internal memory leaks?

A: You can relieve the leaks by following the PR. #2230

Q3.3 In the recognition training, though the precision of the training set reaches 90, the precision of the verification set is still 70. How to handle this case?

A: This may be due to overfitting. There are two solutions you can try: (1) Introduce more augmentation methods or increase the probability of augmentation, and the default value is 0.4; (2) increase the 12 decay value of the system.

5.2.5 Assignment

[Task 1]

Visualize the Data Augmentation results implemented in PaddleOCR: noise, jitter, and explain the effect in language .

Optional test picture:



[Task 2]

Replace the backbone in the configuration of configs/rec/rec_icdar15_train.yml with ResNet34_vd in PaddleOCR When the input image shape is (3, 32, 100), what is the final output feature size of the head layer?

[Task 3]

Download the 10W Chinese dataset rec_data_lesson_demo, modify the configs/rec/rec_icdar15_train.yml configuration file to train a recognition model and offer the training log.

Loadable pre-training model: https://paddleocr.bj.bcebos.com/dygraph_v2.0/en/rec_mv3_none_bilstm_ctc_v2.0_train.tar

5.3 Summary

This chapter introduces the theory and practice of text recognition algorithms.

The first section has introduced the theoretical knowledge and mainstream algorithms related to text recognition, including the CTC-based methods, the Sequence2Sequence-based methods, and the segmentation-based methods. The ideas and contributions of classic papers are presented respectively.

The next section is the practical course based on the CRNN algorithm, in which the whole training process will be illustrated from networking to optimization. For more functions and codes, please refer to PaddleOCR.

5.4 Reference

- [1] Shi, B., Bai, X., & Yao, C. (2016). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11), 2298-2304.
- [2] Fedor Borisuk, Albert Gordo, and Viswanath Sivakumar. Rosetta: Large scale system for text detection and recognition in images. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 71–79. ACM, 2018.
- [3] Gao, Y., Chen, Y., Wang, J., & Lu, H. (2017). Reading scene text with attention convolutional sequence modeling. arXiv preprint arXiv:1709.04303.
- [4] Shi, B., Wang, X., Lyu, P., Yao, C., & Bai, X. (2016). Robust scene text recognition with automatic rectification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4168-4176).
- [5] Star-Net Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In Advances in neural information processing systems, pages 2017–2025, 2015.
- [6] Baoguang Shi, Mingkun Yang, XingGang Wang, Pengyuan Lyu, Xiang Bai, and Cong Yao. Aster: An attentional scene text recognizer with flexible rectification. *IEEE transactions on pattern analysis and machine intelligence*, 31(11):855–868, 2018.
- [7] Lee C Y , Osindero S . Recursive Recurrent Nets with Attention Modeling for OCR in the Wild[C]// IEEE Conference on Computer Vision & Pattern Recognition. IEEE, 2016.
- [8] Li, H., Wang, P., Shen, C., & Zhang, G. (2019, July). Show, attend and read: A simple and strong baseline for irregular text recognition. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, No. 01, pp. 8610-8617).
- [9] P. Lyu, C. Yao, W. Wu, S. Yan, and X. Bai. Multi-oriented scene text detection via corner localization and region segmentation. In Proc. CVPR, pages 7553–7563, 2018.
- [10] Liao, M., Zhang, J., Wan, Z., Xie, F., Liang, J., Lyu, P., ... & Bai, X. (2019, July). Scene text recognition from two-dimensional perspective. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, No. 01, pp. 8714-8721).
- [11] Yu, D., Li, X., Zhang, C., Liu, T., Han, J., Liu, J., & Ding, E. (2020). Towards accurate scene text recognition with semantic reasoning networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 12113-12122).
- [12] Sheng, F., Chen, Z., & Xu, B. (2019, September). NRTR: A no-recurrence sequence-to-sequence model for scene text recognition. In 2019 International Conference on Document Analysis and Recognition (ICDAR) (pp. 781-786). IEEE.
- [13] Yang, L., Wang, P., Li, H., Li, Z., & Zhang, Y. (2020). A holistic representation guided attention network for scene text recognition. Neurocomputing, 414, 67-75.
- [14] Wang, T., Zhu, Y., Jin, L., Luo, C., Chen, X., Wu, Y., ... & Cai, M. (2020, April). Decoupled attention network for text recognition. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 07, pp. 12216-12224).
- [15] Wang, Y., Xie, H., Fang, S., Wang, J., Zhu, S., & Zhang, Y. (2021). From two to one: A new scene text recognizer with visual language modeling network. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 14194-14203).
- [16] Fang, S., Xie, H., Wang, Y., Mao, Z., & Zhang, Y. (2021). Read Like Humans: Autonomous, Bidirectional and Iterative Language Modeling for Scene Text Recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 7098-7107).
- [17] Yan, R., Peng, L., Xiao, S., & Yao, G. (2021). Primitive Representation Learning for Scene Text Recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 284-293).

PP-OCR SYSTEM AND STRATEGY

6.1 Introduction of PP-OCR

The first two chapters talk about the DBNet text detection algorithms and CRNN text recognition algorithms. However, in actual scene, it is impossible to obtain the text position and text content of an image simultaneously solely based on the text detection or recognition model. Therefore, we thread the text detection algorithms and the text recognition algorithms to build the PP-OCR text detection and recognition system. In the actual use, the detected text direction may be not what we expect, which will lead to errors in recognition. Therefore, a direction classifier has also been introduced in the PP-OCR system.

This chapter mainly introduces the PP-OCR text detection and recognition system and the involved optimization strategies. In this section, you can learn:

- PaddleOCR strategy refining skills
- Optimization techniques and methods for text detection, recognition, and direction classifier models

The PP-OCR system has undergone two phases of optimization. The following part will briefly introduce the system and its two optimizations.

6.1.1 Introduction to PP-OCR System and Optimization Strategies

In PP-OCR, if you want to extract the text from an image, you need to:

- Use text detection methods to obtain the polygon of the text area (DBNet is used in the text detection in PP-OCR, which will get four-point text boxes).
- Crop and apply perspective transformation correction to the polygon area, convert the text region into a rectangular, and then use the direction classifier to correct its direction.
- Recognize the text in the rectangular box and get the recognition result.

After all these, the text detection and recognition for an image are finished.

The frame diagram of PP-OCR is shown below.

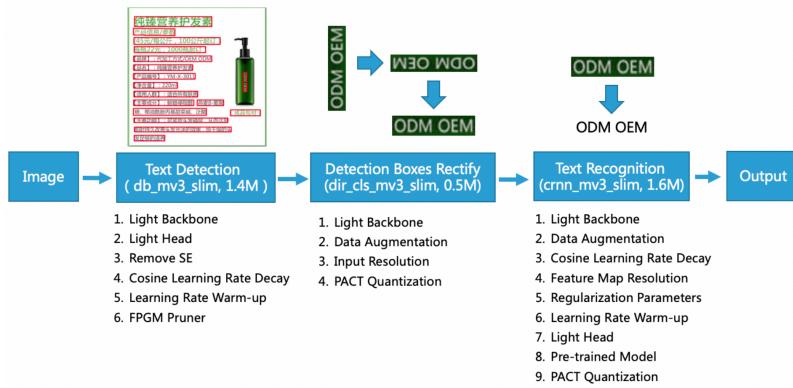


Figure 1: PP-OCR system frame diagram

Text detection adopts DBNet whose post-processing is relatively simple. Text region correction mainly uses geometric transformation and the direction classifier. Text recognition makes use of CRNN, a combination of convolution and sequence features, and applies CTC loss to solving the inconsistency of prediction results and labels.

PP-OCR uses 19 strategies in the backbone network, learning rate strategy, data augmentation, and model tailoring and quantification, to optimize and slim the model, and create a PP-OCR server system and a PP-OCR mobile system.

6.1.2 Introduction to PP-OCRV2 System and Optimization Strategies

Compared with PP-OCR, PP-OCRV2 further improves the backbone network, data augmentation, and loss function to tackle the poor end-to-end prediction efficiency, complex background, and misrecognition of similar characters. At the same time, it introduces the knowledge distillation training strategy to increase the model accuracy. Its improvement includes:

- Detection model optimization: (1) CML (Collaborative Mutual Learning) knowledge distillation strategy; (2) CopyPaste data augmentation strategy;
- Recognition model optimization: (1) PP-LCNet; (2) U-DML updated knowledge distillation strategy; (3) Enhanced CTC loss.

There are three main improvements in the effect:

- In terms of the model effect, PP-OCRV2 outperforms the PP-OCR mobile version by over 7%;
- In terms of the speed, PP-OCRV2 outperforms the PP-OCR server version by more than 220%;
- In terms of model size, PP-OCRV2 can be easily deployed on the server or mobile platforms with a total size of 11.6M.

The comparison of the accuracy, prediction time, and model size between the PP-OCRV2 model and the previous models of PP-OCR series is shown below.

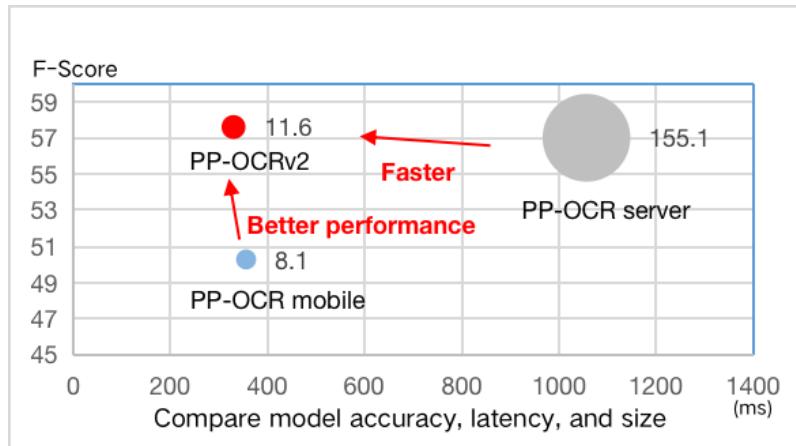


Figure 2: Comparison of speed, accuracy and model size between PP-OCRv2 and PP-OCR

The frame diagram of PP-OCRv2 is shown below.

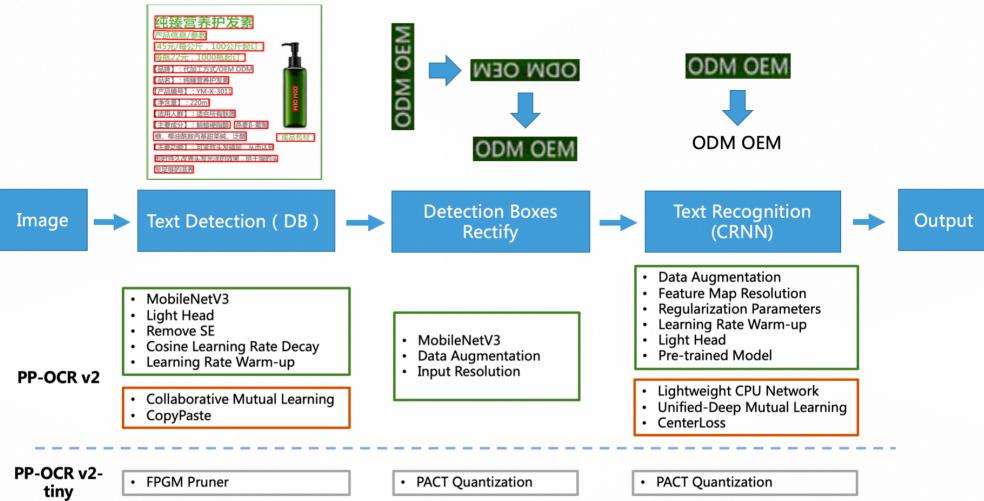


Figure 2: The framework of the proposed PP-OCRv2. The strategies in the green boxes are the same as PP-OCR. The strategies in the orange boxes are the newly added ones in the PP-OCRv2. The strategies in the gray boxes will be adopted by the PP-OCRv2-tiny in the future.

Figure 3: PP-OCRv2 system frame diagram

This chapter will give a detailed interpretation of optimization strategies of the PP-OCR and PP-OCRv2.

6.2 PP-OCR Optimization Strategies

The PP-OCR system includes a text detector, a direction classifier and a text recognizer. This section introduces the model optimization strategies in these three directions in detail.

6.2.1 Text Detection

The text detection in PP-OCR is based on the DBNet (Differentiable Binarization) model, which is based on the segmentation, and simple in post-processing. The figure below shows the model structure of DBNet.

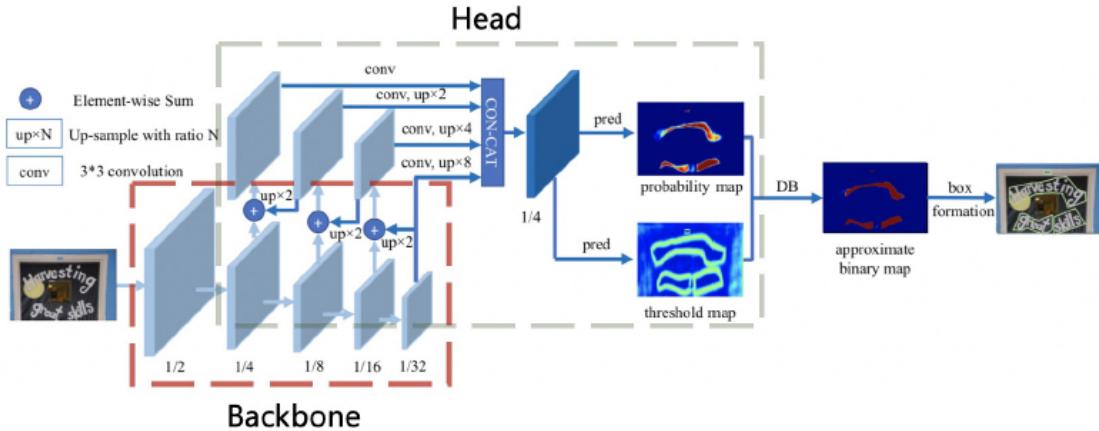


Figure 4: DBNet frame diagram

DBNet extracts features through the backbone, and uses the structure (neck) of DBFPN to fuse the features to obtain the combined features. The fused features are decoded by operations (head) such as convolution to generate a probability map and a threshold map. And an approximate binary map is calculated with the two maps combined. When calculating the loss function, the loss function is calculated for these three feature maps. Here, the binary supervision is also applied to the training so that the model can learn more accurate boundaries.

Six optimization strategies are used in DBNet to improve model accuracy and speed, including the backbone network, FPN, head structure, learning rate strategy, and model cropping. On the validation set, the conclusions of the ablation experiment2 for different modules are shown below.

inner_channel of the head	Remove SE	Cosine Learning Rate Decay	Learning Rate Warm-up	Precision	Recall	HMean	Model Size (M)	Inference Time (CPU, ms)
256				0.6821	0.5560	0.6127	7	406
96				0.6677	0.5524	0.6046	4.1	213
96	✓			0.6952	0.5413	0.6087	2.6	173
96	✓	✓		0.7034	0.5404	0.6112	2.6	173
96	✓	✓	✓	0.7349	0.5420	0.6239	2.6	173

Figure 5: DBNet Ablation Experiment

The detailed description is given below.

Lightweight Backbone Network

The size of the backbone network has an important influence on the model size of the text detector. Therefore, a lightweight backbone network should be selected to build an ultra-lightweight detection model. With the development of image classification technology, MobileNetV1, MobileNetV2, MobileNetV3 and ShuffleNetV2 series are often used as lightweight backbone networks. Each series has a different model size and performance. PaddeClas provides more than 20 kinds of lightweight backbone networks. Their Accuracy-Speed curve on ARM is shown below.

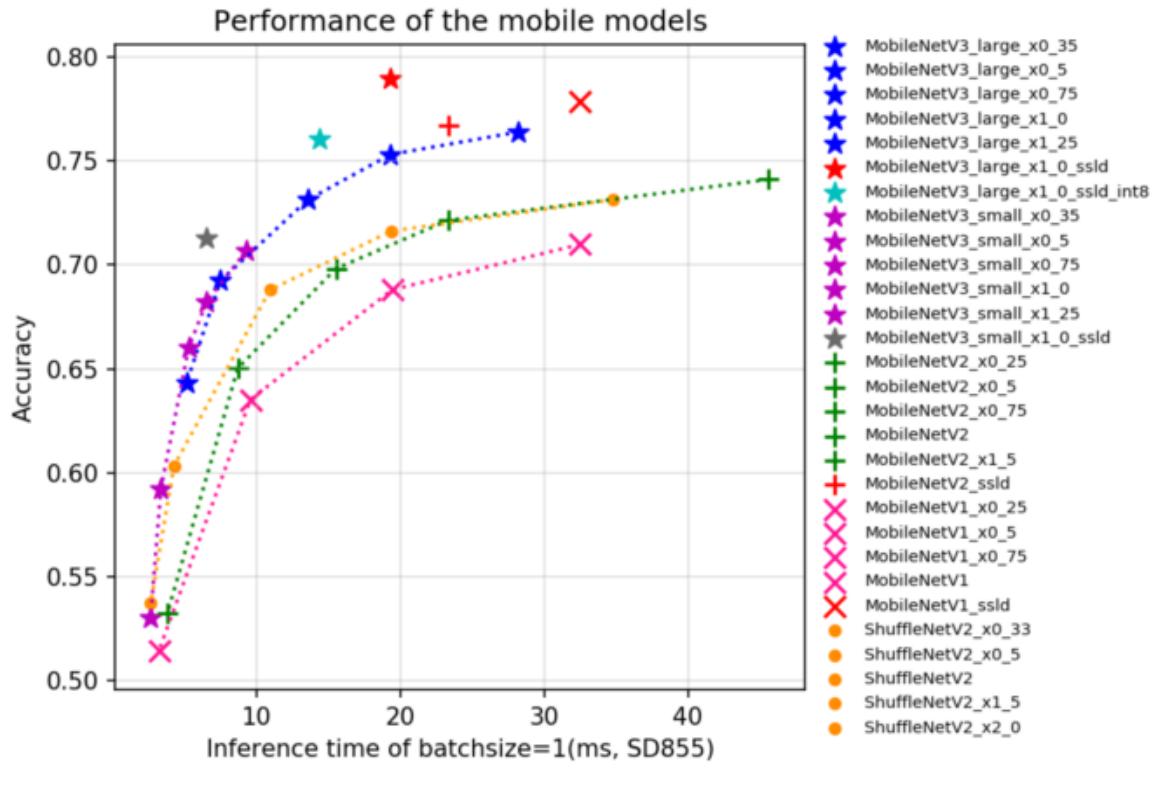


Figure 6: The "speed-accuracy" curve of the backbone network in PaddleClas

When the prediction time is the same, the MobileNetV3 series can achieve higher accuracy. In order to cover as many scenes as possible, the author uses the parameter scale to adjust the number of feature map channels in design. The standard value is 1x. If it is 0.5x, it means that the number of feature map channels in the network is 0.5 times of the 1x corresponding network. In order to balance the accuracy and efficiency, when choosing the size of V3, we adopt the structure of MobileNetV3_large 0.5x.

The feature map size of each stage of MobileNetV3 in DBNet is printed out below.

```
import os
import sys

# Download code
os.chdir("/home/aistudio/")
!git clone https://gitee.com/paddlepaddle/PaddleOCR.git
# Switch working directory
os.chdir("/home/aistudio/PaddleOCR/")
!pip install -U pip
!pip install -r requirements.txt

# For the detailed code implementation, refer to:
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/ppocr/modeling/backbones/
# det_mobilenet_v3.py
import numpy as np
import paddle

# Set random input
```

(continues on next page)

(continued from previous page)

```

inputs = np.random.rand(1, 3, 640, 640).astype(np.float32)
x = paddle.to_tensor(inputs)

# Import MobileNetV3 library
from ppocr.modeling.backbones.det_mobilenet_v3 import MobileNetV3

# Model definition
backbone_mv3 = MobileNetV3(scale=0.5, model_name='large')

# Model forward
bk_out = backbone_mv3(x)

# Model middle layer printing
for i, stage_out in enumerate(bk_out):
    print("the shape of ", i, 'stage: ', stage_out.shape)

```

```

the shape of 0 stage: [1, 16, 160, 160]
the shape of 1 stage: [1, 24, 80, 80]
the shape of 2 stage: [1, 56, 40, 40]
the shape of 3 stage: [1, 480, 20, 20]

```

Lightweight Feature Pyramid Network DBFPN Structure

The feature fusion (neck) part of the text detector, DBFPN, is similar in structure to the FPN in the target detection, It fuses feature maps of different scales to improve the detection effect of text regions of different scales.

To facilitate the merging of feature maps of different channels, a convolution of 1×1 is used to reduce the channel number of feature maps to the same amount.

The probability map and the threshold map are generated by the feature map fused by convolution, and the convolution is also associated with inner_channels. Therefore, inner_channels has a great influence on the model size. When inner_channels is reduced from 256 to 96, the model size is reduced from 7M to 4.1M, and the speed is increased by 48%, but the accuracy is only slightly reduced.

The structure of DBFPN and the fusion result of the backbone network feature map are printed below.

```

# For detailed code implementation, refer to:
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/ppocr/modeling/necks/db_
→fpn.py
from ppocr.modeling.necks.db_fpn import DBFPN

neck_bdfpn = DBFPN(in_channels=[16, 24, 56, 480], out_channels=96)
# Print DBFPN structure
print(neck_bdfpn)

# First reduce the number of original channels to 96, then to 24, and to 4 concat_
→feature maps
fpn_out = neck_bdfpn(bk_out)

print('the shape of output of DBFPN: ', fpn_out.shape)

```

```

DBFPN(
  (in2_conv): Conv2D(16, 96, kernel_size=[1, 1], data_format=NCHW)

```

(continues on next page)

(continued from previous page)

```

(in3_conv): Conv2D(24, 96, kernel_size=[1, 1], data_format=NCHW)
(in4_conv): Conv2D(56, 96, kernel_size=[1, 1], data_format=NCHW)
(in5_conv): Conv2D(480, 96, kernel_size=[1, 1], data_format=NCHW)
(p5_conv): Conv2D(96, 24, kernel_size=[3, 3], padding=1, data_format=NCHW)
(p4_conv): Conv2D(96, 24, kernel_size=[3, 3], padding=1, data_format=NCHW)
(p3_conv): Conv2D(96, 24, kernel_size=[3, 3], padding=1, data_format=NCHW)
(p2_conv): Conv2D(96, 24, kernel_size=[3, 3], padding=1, data_format=NCHW)
)
the shape of output of DBFPN: [1, 96, 160, 160]

```

SE Module Analysis in Backbone Network

SE is the abbreviation of squeeze-and-excitation (Hu, Shen, and Sun 2018).

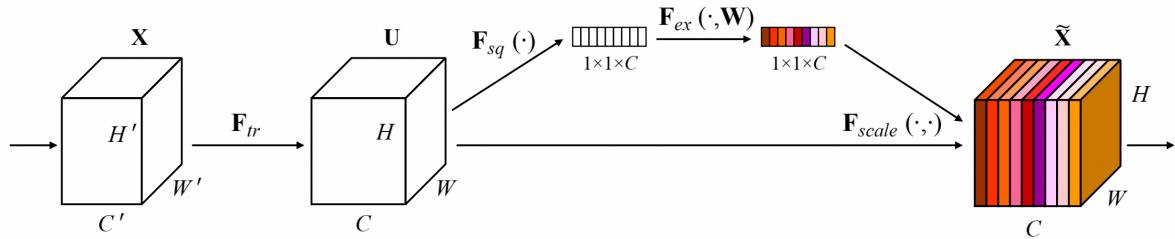


Figure 7: Schematic of SE module

The SE module explicitly models the interdependence between the channels and adaptively recalibrates the characteristic response of channels. The use of SE modules in the network can significantly improve the accuracy of vision tasks. Therefore, the search space of MobileNetV3 contains SE modules, so do the MobileNetV3. However, when the input resolution is large, such as 640×640 , it is difficult to estimate the characteristic response of channels using the SE module, and the accuracy improvement is limited, and it is very time-consuming. In DBNet, **we remove the SE module from the backbone network**, and reduce the model size from 4.1M to 2.6M, but the accuracy remains the same.

In PaddleOCR, you can remove the SE modules in the backbone network by setting `disable_se=True` as shown below.

```

# For detailed code implementation, refer to:
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/ppocr/modeling/backbones/
# det_mobilenet_v3.py

x = paddle.rand([1, 3, 640, 640])

from ppocr.modeling.backbones.det_mobilenet_v3 import MobileNetV3

# Define model
backbone_mv3 = MobileNetV3(scale=0.5, model_name='large', disable_se=True)

# Model forward
bk_out = backbone_mv3(x)
# Output
for i, stage_out in enumerate(bk_out):
    print("the shape of ", i, 'stage: ', stage_out.shape)

```

```

the shape of 0 stage: [1, 16, 160, 160]
the shape of 1 stage: [1, 24, 80, 80]
the shape of 2 stage: [1, 56, 40, 40]
the shape of 3 stage: [1, 480, 20, 20]

```

Learning Rate Strategy Optimization

- Cosine Learning Rate Reduction Strategy

In the gradient descent algorithm, we need to set a value to control the weight update amplitude, which is called the learning rate, a hyperparameter that controls the learning speed of the model. The smaller the learning rate is, the slower the loss function changes. Although a lower learning rate can ensure that no local minimum is missed, but the model will converge slowly.

Therefore, in the early stage of training, the weights are in random initialization, and we can set a relatively large learning rate to speed up the convergence. In the later stage, as the weight is close to the optimal value, and a small learning rate can prevent the model from oscillating during the convergence.

Therefore, the cosine learning rate strategy came into being. It refers to the learning rate that changes according to the cosine curve in the training. During the entire training process, the cosine learning rate decay strategy enables the network to maintain a relatively large learning rate in the initial stage, and the learning rate will gradually decay to 0 later. Its convergence speed is relatively slow, but the convergence accuracy is great. The figure below compares two different learning rate decay strategies: piecewise decay and cosine decay.

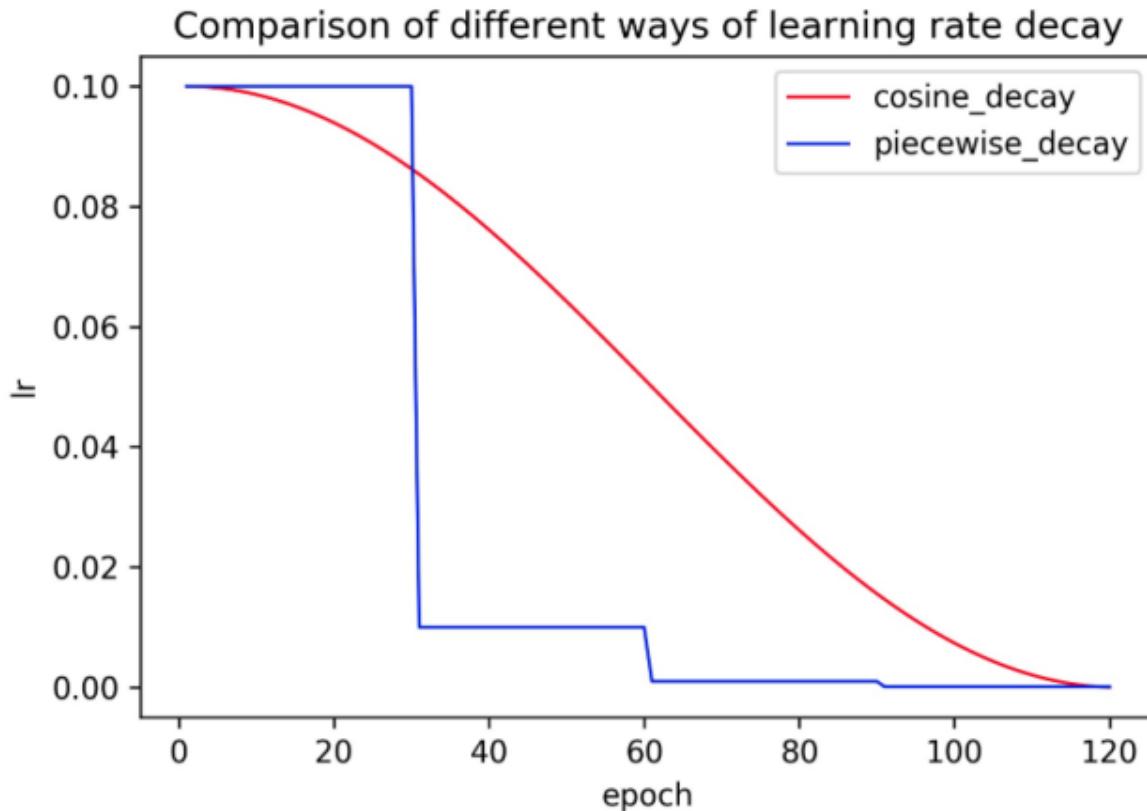


Figure 8: Cosine and Piecewise learning rate decay strategy

- Learning Rate Warm-up Strategy

When the model is first trained, the model weights are initialized randomly. At this time, if a larger learning rate is selected, the model training may be unstable. Therefore, the concept of **learning rate warm-up** is proposed to solve the problem of non-convergence in the early stage of model training.

Learning rate warm-up refers to gradually increasing the learning rate from a small value to a larger one at the beginning of training. It can ensure the stability of the model at the beginning of training. The strategy can improve the accuracy of image classification. Experiments show that this strategy is also effective in DBNet. When the learning rate warm-up strategy is combined with the cosine learning rate, the changing trend of the learning rate is shown in the following code.

```
# For detailed code implementation, refer to:
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/ppocr/optimizer/__init__.py
# Import the function built by the learning rate optimizer
from ppocr.optimizer import build_lr_scheduler
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
# Let's see the effect when warmup_epoch is 2
lr_config = {'name': 'Cosine', 'learning_rate': 0.1, 'warmup_epoch': 2}
epochs = 20 # config['Global']['epoch_num']
iters_epoch = 100 # len(train_dataloader)
lr_scheduler=build_lr_scheduler(lr_config, epochs, iters_epoch)

iters = 0
lr = []
for epoch in range(epochs):
    for _ in range(iters_epoch):
        lr_sduler.step() # ↪ https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/tools/program.py#L262
        iters += 1
        lr.append(lr_scheduler.get_lr())

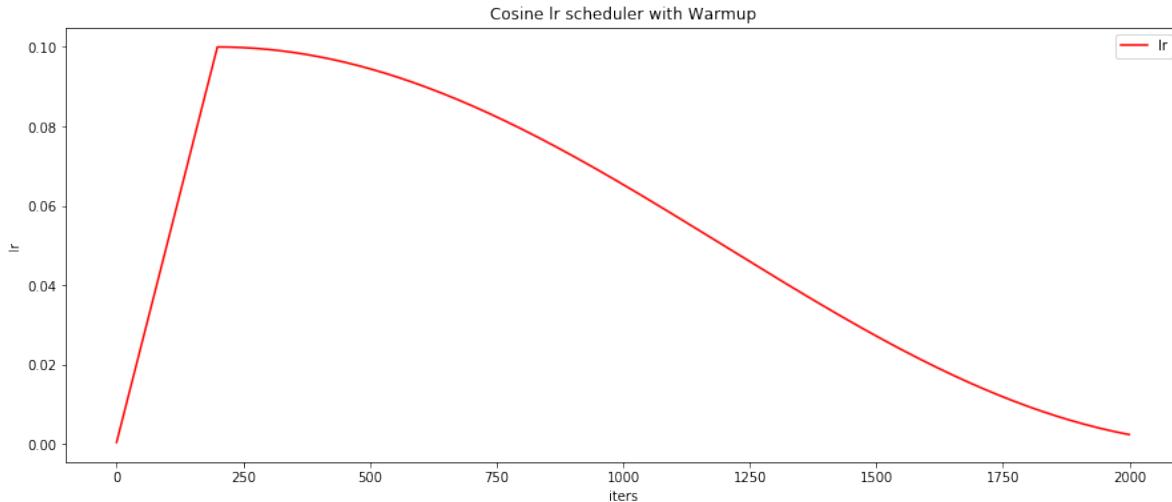
x = np.arange(iters,dtype=np.int64)
y = np.array(lr,dtype=np.float64)

plt.figure(figsize=(15, 6))
plt.plot(x,y,color='red',label='lr')

plt.title(u'Cosine lr scheduler with Warmup')
plt.xlabel(u'iters')
plt.ylabel(u'lr')

plt.legend()

plt.show()
```



Model Tailoring Strategy-FPGM

There are many redundant parameters in deep learning models. We can use some methods to remove redundant parts and improve the efficiency of model inference.

Model tailoring refers to the acquisition of a more lightweight network by removing redundant channels, filters, neurons, and so on on the premise of maintaining the accuracy of the model.

Compared with tailoring the channel or the feature map, tailoring the filter can get a more regular model, thus reducing the memory consumption and accelerating the model inference.

The previous versions are mostly based on the norm. That is, the filter with a smaller norm is considered to be less important, but this kind of methods require the minimum norm of the existing filter to be close to 0, otherwise it is difficult to be removed.

In this case, Filter Pruning via Geometric Median (FPGM) is proposed. FPGM treats each filter in the convolutional layer as one point in Euclidean space, and it introduces the concept of geometric median, which refers to the point with the smallest sum of distances from all sampling points. If a filter is close to this geometric median, then it is probable that the information of this filter overlaps with other filters and can be removed.

The comparison between FPGM and norm-based tailoring algorithm is shown in the figure below.

Figure 9: Schematic of FPGM tailoring

In PP-OCR, we use FPGM to tailor the detection model. In the end, the model accuracy of DBNet just decreases slightly, but the model size is reduced by 46%, and the prediction speed is accelerated by 19%.

For more details on the implementation of FPGM model tailoring, please refer to [PaddleSlim](#).

Notice:

1. Model tailoring requires to retrain the model. For this, please refer to [PaddleOCR Pruning Tutorial](#).
2. The code tailoring is adapted for DBNet. If you need to prune your own model, re-analyze the model structure and the sensitivity of parameters. Usually it is only recommended to cut the parameters of relatively low sensitivity, and skip sensitive ones.
3. The tailoring rate of each convolutional layer is also very important for the performance of the tailored model. Using the identical tailoring rate will usually result in significant performance degradation.

4. Model tailoring cannot be accomplished overnight, Only through repeated experiments can a model that meets the requirements be tailored to.

Description of Text Detection Configuration

A brief description of the training configuration of DBNet is given below, and for complete configuration file, refer to:[ch_det_mv3_db_v2.0.yml](#)

```
Architecture:                                # Define model structure
  model_type: det
  algorithm: DB
  Transform:
  Backbone:
    name: MobileNetV3                      # Configure backbone network
    scale: 0.5
    model_name: large
    disable_se: True                         # Remove SE modules
  Neck:
    name: DBFPN                            # Configure DBFPN
    out_channels: 96                         # Configure inner_channels
  Head:
    name: DBHead
    k: 50
  Optimizer:
    name: Adam
    beta1: 0.9
    beta2: 0.999
    lr:
      name: Cosine                         # Configure cosine learning rate decay strategy
      learning_rate: 0.001                  # Initial learning rate
      warmup_epoch: 2                      # Configure learning rate warm-up strategy
    regularizer:
      name:'L2'                           # Configure L2 regularizer
      factor: 0                            # The weight of the regularizer
```

Summary of PP-OCR Detection Optimization

The last section introduces the optimization strategies of the text detection algorithm in PP-OCR, and this part will review the ablation experiments and conclusions corresponding to different optimization strategies.

inner_channel of the head	Remove SE	Cosine Learning Rate Decay	Learning Rate Warm-up	Precision	Recall	HMean	Model Size (M)	Inference Time (CPU, ms)
256				0.6821	0.5560	0.6127	7	406
96				0.6677	0.5524	0.6046	4.1	213
96	✓			0.6952	0.5413	0.6087	2.6	173
96	✓	✓		0.7034	0.5404	0.6112	2.6	173
96	✓	✓	✓	0.7349	0.5420	0.6239	2.6	173

Figure 10: DBNet Ablation Experiment

Through strategies such as the lightweight backbone network, lightweight neck structure, SE module analysis and removal, learning rate adjustment, and model tailoring, the model size of DBNet is reduced from **7M** to **1.5M**. Its model accuracy has increased by more than **1%** through improving training strategies such as the learning rate strategy.

In PP-OCR, the ultra-lightweight DBNet detection result is as follows:

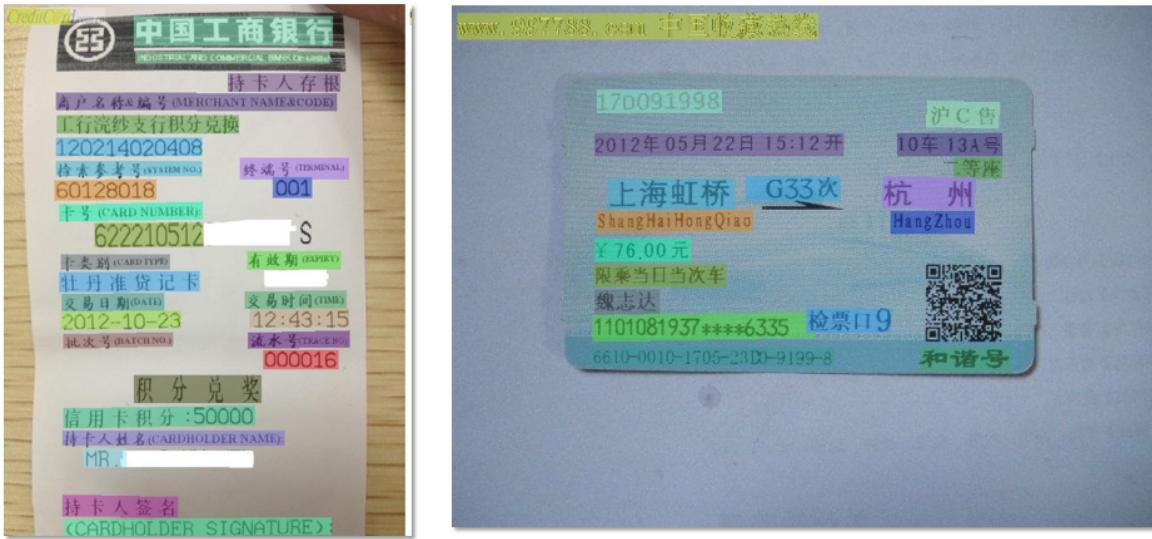


Figure 11: the ultra-lightweight DBNet detection result

The following shows the prediction result of quickly using the text detection model. The detailed prediction and inference code will be explained in Chapter 5.

```
!mkdir inference
!cd inference && wget https://paddleocr.bj.bcebos.com/PP-OCRV2/chinese/ch_PP-OCRV2_
->det_infer.tar -O ch_PP-OCRV2_det_infer.tar && tar -xf ch_PP-OCRV2_det_infer.tar
!python tools/infer/predict_det.py --image_dir="../doc/imgs/00111002.jpg" --det_model_
->dir="../inference/ch_PP-OCRV2_det_infer" --use_gpu=False
from PIL import Image
img_det = Image.open('../inference_results/det_res_00111002.jpg')

plt.figure(figsize=(14, 10)) # window size
plt.imshow(img_det)
plt.axis('on')
plt.title('Detection')
plt.show()
```

```
mkdir: cannot create directory 'inference': File exists
--2021-12-24 21:07:17-- https://paddleocr.bj.bcebos.com/PP-OCRV2/chinese/ch_PP-
->OCRV2_det_infer.tar
Resolving paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com)... 182.61.200.229, 182.
->61.200.195, 2409:8c04:1001:1002:0:ff:b001:368a
Connecting to paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com) |182.61.200.
->229|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3190272 (3.0M) [application/x-tar]
Saving to: 'ch_PP-OCRV2_det_infer.tar'

ch_PP-OCRV2_det_inf 100%[=====] 3.04M 4.13MB/s in 0.7s

2021-12-24 21:07:18 (4.13 MB/s) - 'ch_PP-OCRV2_det_infer.tar' saved [3190272/
->3190272]
```

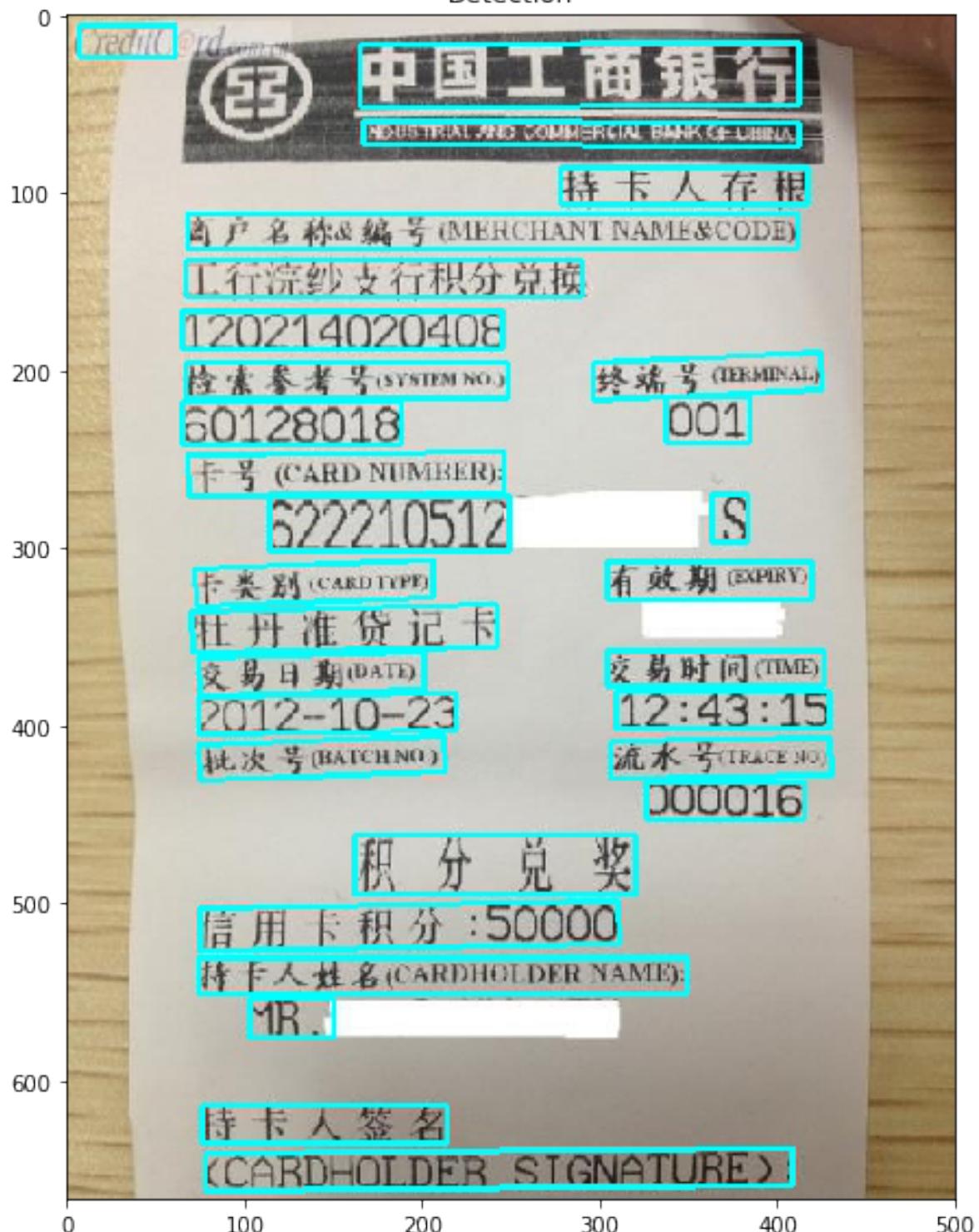
(continues on next page)

(continued from previous page)

```
[2021/12/24 21:07:22] root INFO: 00111002.jpg      [[[78, 641], [408, 638], [408,
↳ 659], [78, 662]], [[76, 614], [214, 614], [214, 635], [76, 635]], [[103, 554], ↳
↳ [150, 554], [150, 576], [103, 576]], [[74, 531], [349, 531], [349, 551], [74, ↳
↳ 551]], [[75, 503], [310, 499], [311, 523], [75, 527]], [[162, 462], [320, 462], ↳
↳ [320, 495], [162, 495]], [[326, 432], [415, 432], [415, 453], [326, 453]], [[306,
↳ 409], [429, 407], [430, 428], [306, 430]], [[74, 411], [212, 406], [213, 426], ↳
↳ [75, 431]], [[74, 384], [219, 382], [219, 403], [74, 405]], [[309, 381], [429, ↳
↳ 381], [429, 402], [309, 402]], [[74, 362], [201, 359], [201, 380], [75, 383]], ↳
↳ [[304, 358], [426, 358], [426, 378], [304, 378]], [[70, 336], [242, 332], [242, ↳
↳ 356], [71, 359]], [[72, 312], [206, 307], [206, 328], [73, 333]], [[304, 308], ↳
↳ [419, 308], [419, 329], [304, 329]], [[114, 271], [249, 271], [249, 302], [114, ↳
↳ 302]], [[363, 270], [383, 270], [383, 297], [363, 297]], [[68, 248], [246, 246], ↳
↳ [246, 269], [69, 271]], [[65, 218], [188, 218], [188, 242], [65, 242]], [[337, ↳
↳ 215], [384, 215], [384, 241], [337, 241]], [[67, 196], [248, 196], [248, 216], ↳
↳ [67, 216]], [[296, 196], [424, 190], [425, 211], [296, 217]], [[65, 167], [245, ↳
↳ 167], [245, 188], [65, 188]], [[67, 138], [290, 138], [290, 159], [67, 159]], ↳
↳ [[68, 112], [411, 112], [411, 132], [68, 132]], [[278, 86], [417, 86], [417, ↳
↳ 107], [278, 107]], [[167, 60], [412, 61], [412, 74], [167, 73]], [[165, 17], ↳
↳ [412, 16], [412, 51], [165, 52]], [[7, 6], [61, 6], [61, 24], [7, 24]]]

[2021/12/24 21:07:22] root INFO: The predict time of ./doc/imgs/00111002.jpg: 1.
↳ 7913281917572021
[2021/12/24 21:07:22] root INFO: The visualized image saved in ./inference_results/
↳ det_res_00111002.jpg
```

Detection



6.2.2 Direction Classifier

The aim of the direction classifier is to classify the direction of the text instance detected by the text, rotate the text to 0 degrees, and then send it to the text recognizer. In PP-OCR, we choose two directions: **0** degree and **180** degree. The following is a detailed introduction to the speed and accuracy optimization strategy for the direction classifier.

Figure 12: Directional classifier ablation experiment

Lightweight Backbone Network

Same as the text detector, we still use MobileNetV3 as the backbone network of the direction classifier. Because the task of direction classification is simpler, we use MobileNetV3 small 0.35x to balance model accuracy and prediction efficiency. Experiments show that using a larger backbone will not be further improved the accuracy.

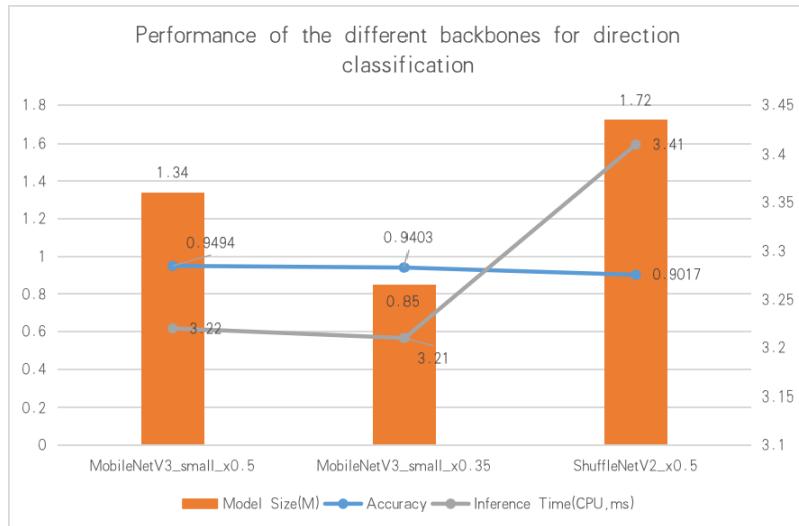


Figure 13: Comparison of the accuracy of direction classifiers with different backbone networks

Data Agumentation

Data augmentation refers to transforming the image and sending it to the network for training, which can improve the generalization performance of the network. Commonly used data augmentation methods includes rotation, perspective distortion and transformation, motion blur and Gaussian noise transformation. In PP-OCR, these data augmentation methods are categorized into BDA (Base Data Augmentation). It is verified that BDA can significantly improve the accuracy of the direction classifier.

The following shows the effect of some BDA data augmentation methods.

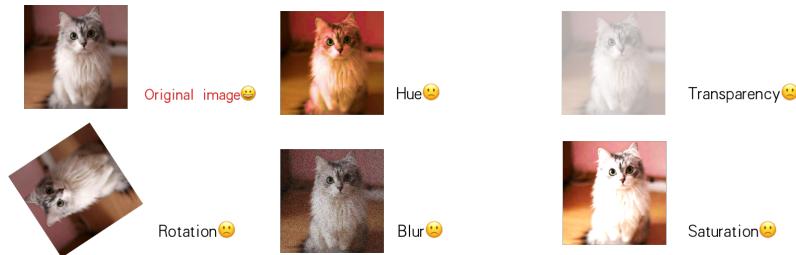


Figure 14: BDA data augmentation effect

Besides BDA, we also introduce some higher-level data augmentation methods to improve classification, such as AutoAugment (Cubuk et al. 2019), RandAugment (Cubuk et al. 2020), CutOut (DeVries and Taylor 2017), RandErasing (Zhong et al. 2020), HideAndSeek (Singh and Lee 2017), GridMask (Chen 2020), Mixup (Zhang et al. 2017) and Cutmix (Yun et al. 2019).

These data augmentations are divided into 3 categories:

- (1) Image transformation: AutoAugment, RandAugment
- (2) Image tailoring: CutOut, RandErasing, HideAndSeek, GridMask
- (3) Image mixing: Mixup, Cutmix

The visual comparison results of different high-level data augmentation are given:

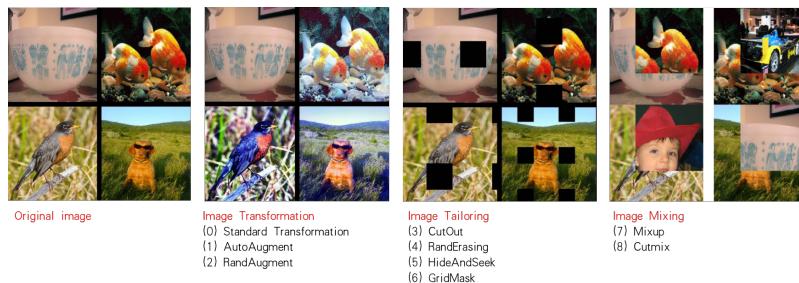


Figure 15: High-level data augmentation visualization effect

But experiments show that, except for RandAugment and RandErasing, most methods are not suitable for the direction classifier. The figure below also explains the changes in model accuracy with different data augmentation strategies.

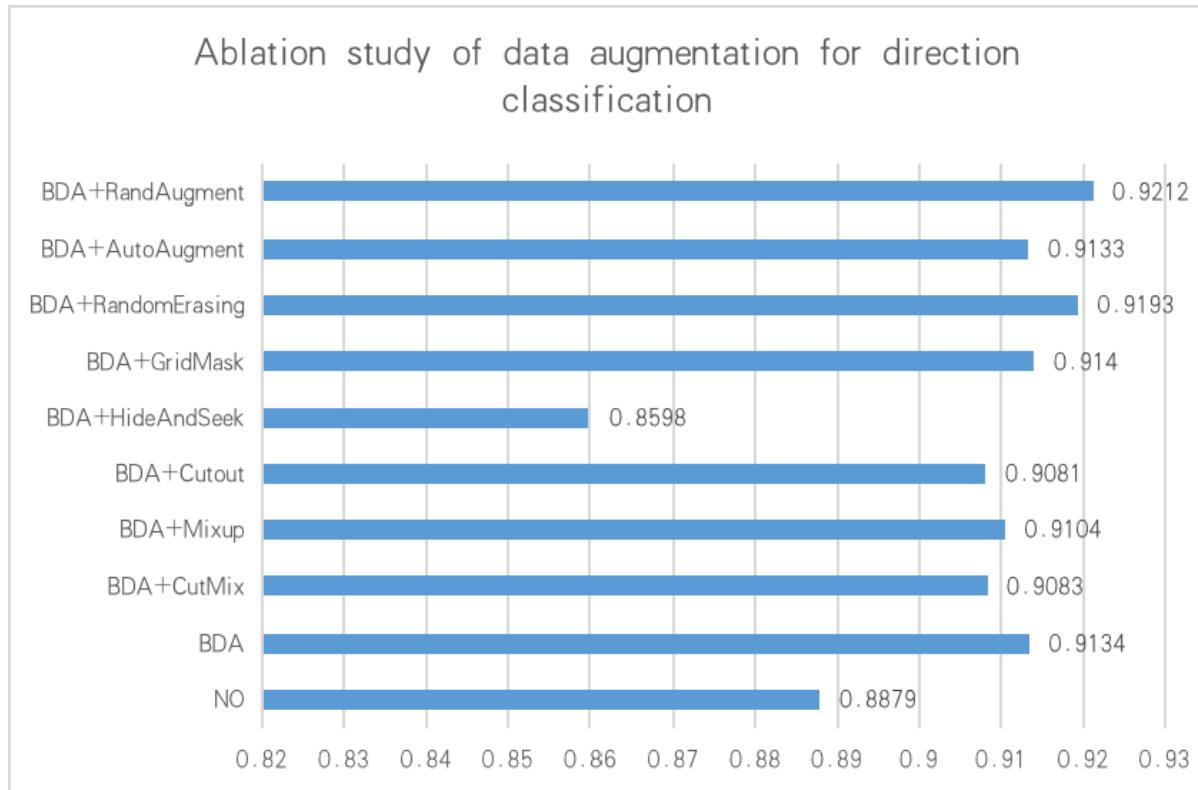


Figure 16: Ablation study of data augmentation for direction classification

Finally, we combine BDA and RandAugment as a data enhancement strategy for the direction classifier in the training.

- RandAugment code demo

```
# Reference Code:
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/ppocr/data/imaug/__init___.py
import random
from PIL import Image
from ppocr.data.imaug import DecodeImage, RandAugment, transform

np.random.seed(1)
random.seed(1)

img = Image.open('./doc/imgs_words/ch/word_4.jpg')

# Draw the original image
plt.figure("Image1") # Image window name
plt.imshow(img)
plt.axis('on') # Turn off the axis is off
plt.title('Before RandAugment') # image title
plt.show()

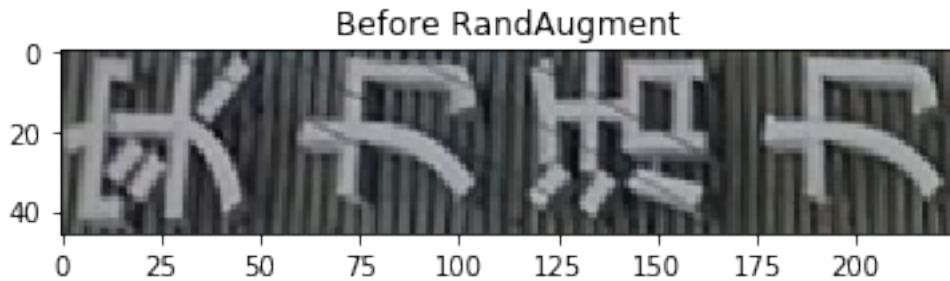
data = {'image':None}
with open('./doc/imgs_words/ch/word_4.jpg', 'rb') as f:
    img = f.read()
    data['image'] = img

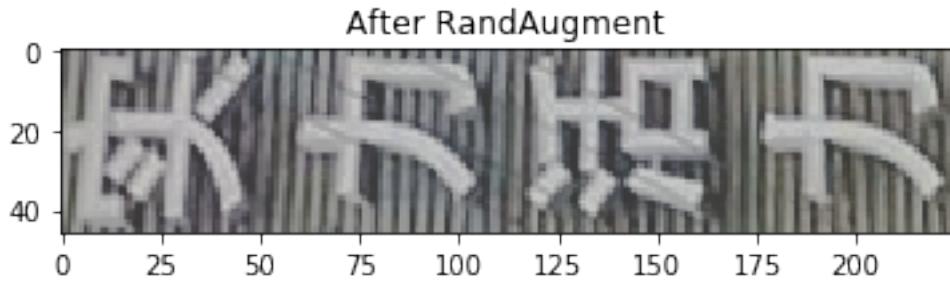
# Define transformation operator
ops_list = [DecodeImage(), RandAugment()]

# Data transformation
data = transform(data, ops_list)

img_auged = data['image']

# show
img_auged = Image.fromarray(img_auged, 'RGB')
plt.figure("Image") # Image window name
plt.imshow(img_auged)
plt.axis('on') # Turn off the axis is off
plt.title('After RandAugment') # image title
plt.title('After RandAugment') # Image title
plt.show()
```





The following shows the prediction effect of quickly using the direction classifier model. The specific predictive reasoning code will be explained in detail in Chapter 5.

```
# Reference Code:
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/tools/infer/predict_cls.py
!cd inference && wget https://paddleocr.bj.bcebos.com/dygraph_v2.0/ch/ch_ppocr_mobile_v2.0_cls_infer.tar -O ch_ppocr_mobile_v2.0_cls_infer.tar && tar -xf ch_ppocr_mobile_v2.0.tars_infer

# Direction classifier makes classification
!python tools/infer/predict_cls.py --image_dir=".doc/imgs_words/ch/word_1.jpg" --cls_model_dir=".inference/ch_ppocr_mobile_v2.0_cls_infer" --use_gpu=False

# Import image
import cv2
img = cv2.imread("./doc/imgs_words/ch/word_1.jpg")

plt.imshow(img[:, :, ::-1])
plt.show()

# Rotate 180 degrees
img = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
img = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
cv2.imwrite("./test.png", img)

# Use the direction classifier to classify the rotated image
!python tools/infer/predict_cls.py --image_dir="./test.png" --cls_model_dir=".inference/ch_ppocr_mobile_v2.0_cls_infer" --use_gpu=False

plt.imshow(img[:, :, ::-1])
plt.show()
```

```
--2021-12-24 21:19:04-- https://paddleocr.bj.bcebos.com/dygraph_v2.0/ch/ch_ppocr_mobile_v2.0_cls_infer.tar
Resolving paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com)... 182.61.200.195, 182.61.200.229, 2409:8c04:1001:1002:0:ff:b001:368a
Connecting to paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com)|182.61.200.195|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1454080 (1.4M) [application/x-tar]
Saving to: 'ch_ppocr_mobile_v2.0_cls_infer.tar'

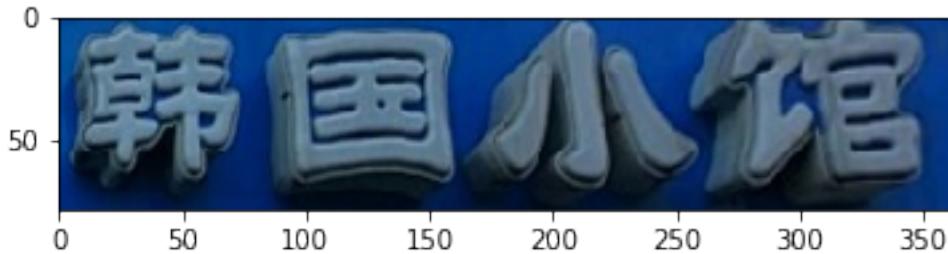
ch_ppocr_mobile_v2. 100%[=====] 1.39M --.-KB/s in 0.1s

2021-12-24 21:19:04 (14.3 MB/s) - 'ch_ppocr_mobile_v2.0_cls_infer.tar' saved
→ [1454080/1454080]
```

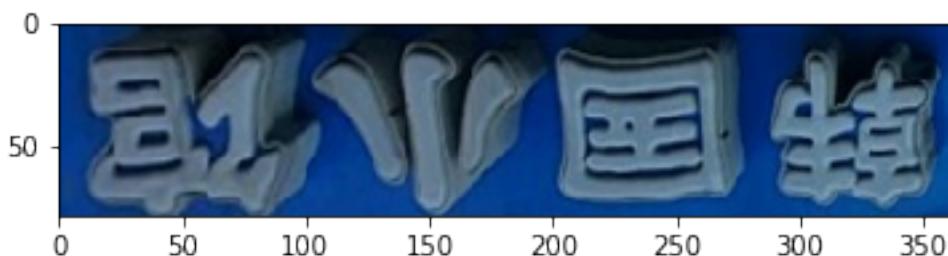
(continues on next page)

(continued from previous page)

```
[2021/12/24 21:19:06] root INFO: Predicts of ./doc/imgs_words/ch/word_1.jpg: ['0', 0.9998784]
```



```
[2021/12/24 21:19:09] root INFO: Predicts of ./test.png: ['180', 0.9999759]
```



Input Resolution Optimization

Generally speaking, when the input resolution of the image increases, the accuracy will also increase. Since the backbone network parameters of the direction classifier are very small, even if the resolution increases, the inference time will not increase significantly. The input image scale of the direction classifier is enlarged from 3x32x100 to 3x48x192, and then the accuracy of the direction classifier grows from 92.1% to 94.0%, but the prediction time only increases from 3.19ms to 3.21 ms.

Here is a comparison of the image sizes at the two scales.

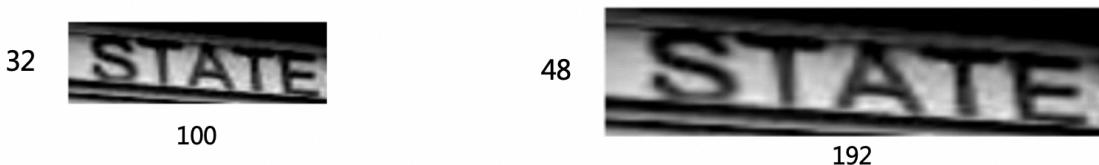


Figure 17: Image size comparison between 32x100 and 48x192

Model Quantization Strategy-PACT

Model quantization is to convert floating point calculations into low-rate specific point calculations, which can make neural network models lower in latency, smaller in size, and less in computational power.

Model quantization is mainly divided into offline quantization and online quantization. Offline quantization refers to a fixed-point quantization method that uses techniques such as KL divergence to determine quantization parameters, and does not require retraining after quantization; online quantization refers to the method that determines quantization parameters during the training process. Compared with offline quantization, its accuracy loss is smaller.

PACT (PArameterized Clipping acTivation) is a new online quantization method that can **remove some extreme values from the activation layer in advance**. After removal, the model can learn more appropriate quantization parameters. The preprocessing of the activation value of the ordinary PACT methods is based on the RELU function, and the formula is:

$$y = PACT(x) = 0.5(|x| - |x - \alpha| + \alpha) = \begin{cases} 0 & x \in (-\infty, 0) \\ x & x \in [0, \alpha) \\ \alpha & x \in [\alpha, +\infty) \end{cases}$$

All activation values greater than a certain threshold will be reset to a constant. However, the activation function in MobileNetV3 is not only ReLU, but also hardswish. Therefore, ordinary PACT quantization will lead to higher accuracy loss. To reduce the quantization loss, the formula of the activation function is modified to:

$$y = PACT(x) = \begin{cases} -\alpha & x \in (-\infty, -\alpha) \\ x & x \in [-\alpha, \alpha) \\ \alpha & x \in [\alpha, +\infty) \end{cases}$$

PaddleOCR provides quantization scripts for PP-OCR kits. For specific links, please refer to [PaddleOCR Model Quantization Tutorial](#).

Direction Classifier Configuration Instruction

When training the direction classifier, some key fields and descriptions in the configuration file are explained in the following. For the complete configuration file, refer to [cls_mv3.yml](#).

```
Architecture:
  model_type: cls
  algorithm: CLS
  Transform:
  Backbone:
    name: MobileNetV3 # Configure the classification model as MobileNetV3
    scale: 0.35
    model_name: small
  Neck:
  Head:
    name: ClsHead
    class_dim: 2

Train:
  dataset:
    name: SimpleDataSet
    data_dir: ./train_data/cls
    label_file_list:
      -./train_data/cls/train.txt
    transforms:
      -DecodeImage: # load image
```

(continues on next page)

(continued from previous page)

```



```

Summary of the Direction Classifier Experiment

When improving the direction classifier model, we use a lightweight backbone network and model quantization, and reduce the model size from 0.85M** to 0.46M**. Also, the combination of data augmentation, high resolution and other features improves the model accuracy by more than **2%**. The comparison in the ablation experiment is shown below.

Input Resolution	PACT Quantization	Accuracy	Model Size (M)	Inference Time (SD 855, ms)
3 × 32 × 100		0.9212	0.85	3.19
3 × 48 × 192		0.9403	0.85	3.21
3 × 48 × 192	✓	0.9456	0.46	2.38

Figure 18: Direction classifier ablation experiment

6.2.3 Text recognition

The text recognizer of PP-OCR is the CRNN model which uses CTC loss to solve the prediction of texts in various lengths.

The structure of the CRNN model is shown below.

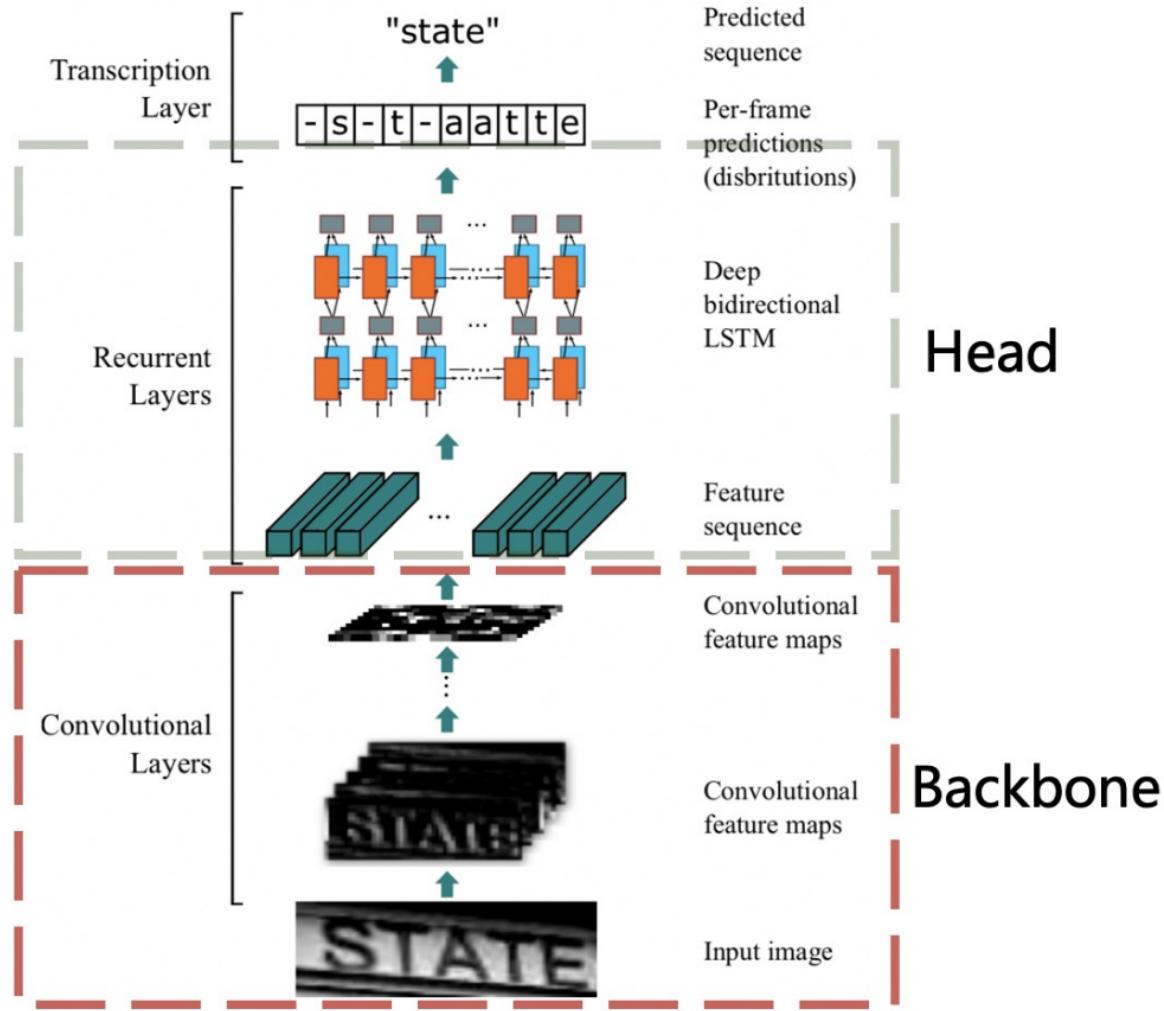


Figure 19: CRNN structure diagram

For the text recognizer, PP-OCR aims to optimize the model from the backbone network, head structure optimization, data enhancement, regularization, feature map downsampling strategy, and quantization. The ablation experiment is shown below.

Id	Ablation study	ACC	Size (M)
1	baseline	-	23
2	1 + Lightweight Backbone and Head	0.5193	4.6
3	2 + BDA	0.5505	4.6
4	3 + Cosine Learning Rate Strategy	0.5652	4.6
5	4 + Feature Map Downsampling Strategy	0.6179	4.6
6	5 + Regularization	0.6519	4.6
7	6 + Warmup	0.6581	4.6
8	7 + TIA	0.6670	4.6
9	8 + PACT	0.674	1.5
10	9 + Pretrain	0.69	1.5

Figure 20: Ablation experiment of the CRNN recognition model

The optimization strategies of the text recognition model are described in detail below.

Lightweight Backbone Network and Head Structure

- Lightweight backbone network

In text recognition, MobileNetV3 is still used as the backbone, the same as text detection. MobileNetV3_small_x0.5 is selected to further balance accuracy and efficiency. If there is no requirement for the model size, you can choose MobileNetV3_small_x1 which can significantly improve the accuracy with the model size only increased by 5M.

Backbone	Accuracy	Model Size (M)	Inference Time (CPU, ms)
MobileNetV3_small_x0.35	0.6288	22	17
MobileNetV3_small_x0.5	0.6556	23	17.27
MobileNetV3_small_x1	0.6933	28	19.15

Figure 21: Comparison of recognition model accuracy under different backbone networks

- Lightweight head structure

In CRNN, the lightweight head used for decoding is a fully connected layer used to decode sequence features into ordinary predicted characters. The dimension of the sequence feature has a great influence on the model size of the text recognizer, especially on scene of the recognition of over 6,000 Chinese characters (if the sequence feature dimension is set to 256, the model size of the head is **6.7M **). In PP-OCR, we have conducted experiments on the dimension of sequence features, and determine the dimension as 48, balancing accuracy and efficiency. The conclusions of some ablation experiments are as follows.

the number of channel	Accuracy	Model Size (M)	Inference Time (CPU, ms)
256	0.6556	23	17.27
96	0.6673	8	13.36
64	0.6642	5.6	12.64
48	0.6581	4.6	12.26

Figure 22: Accuracy comparison of feature dimensions of different sequences

Data Augmentation

Besides the aforementioned BDA (Basic Data Enhancement), which is often used in text recognition, TIA (Luo et al., 2020) is also another effective data augmentation method for text recognition. TIA is a data augmentation method for scene texts. It sets multiple reference points in the image, and then randomly moves the points to generate new images through geometric transformation, which can greatly improve data diversity and the generalization ability of the model. The basic flow chart of TIA is shown in the figure:

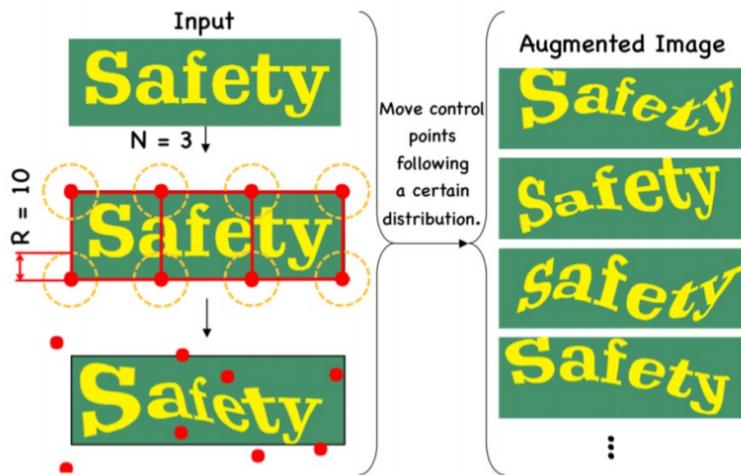


Figure 23: The basic flow chart of TIA

Experiments show that the use of TIA data augmentation can improve the accuracy of the text recognition model by 0.9% on a very high baseline.

Here is the visualization of the three types of data augmentation involved in TIA.

```
# Reference Code:
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/ppocr/data/imaug/text_
→image_aug/augment.py
import cv2
from ppocr.data.imaug.rec_img_aug import tia_distort, tia_stretch, tia_perspective
img = cv2.imread("./doc/imgs_words/ch/word_1.jpg")

img_out1 = tia_distort(img, 2.5)
```

(continues on next page)

(continued from previous page)

```



```



Learning Rate Strategy and Regularization

In the training of the recognition model, the learning rate reduction strategy is the same as the text detection, and the learning rate strategy of Cosine+Warmup is also used.

Regularization is widely used to avoid overfitting, and it includes L1 regularization and L2 regularization. In most usage scenarios, we use L2 regularization, which calculates the L2 norm of the weights in the network and add it to the loss function. With the help of L2 regularization, the weight of the network tends to be smaller, and finally the parameters in the entire network will be close to 0, thereby alleviating the over-fitting of the model and improving its generalization.

Our experiments have found that L2 regularization has a great influence on the recognition accuracy in text recognition.

Feature Map Downsampling Strategy

In downstream vision tasks such as detection, segmentation, and OCR, the backbone network is that used in the image classification, with the input resolution of 224x224. The width and height will be down-sampled at the same time.

However, in text recognition, since the input image is generally 32x100, with an extreme aspect ratio. In this case, downsampling the width and the height simultaneously will cause severe feature loss. Therefore, it is required to adapt the feature map downsampling in applying the backbone network in the image classification to text recognition (**If you change the backbone network by yourself, you also need to pay attention here.**).

In PaddleOCR, the height and width of the input image set by the CRNN Chinese text recognition model are set to 32 and 320. The original MobileNetV3 comes from the classification model. As mentioned above, the step size of downsampling needs to be adapted for the input resolution of the text image. Specifically, in order to retain more level information, the step size of the down-sampling feature map is modified from **(2,2)** to **(2,1)**, except for the first down-sampling. The final result is shown in the figure below.

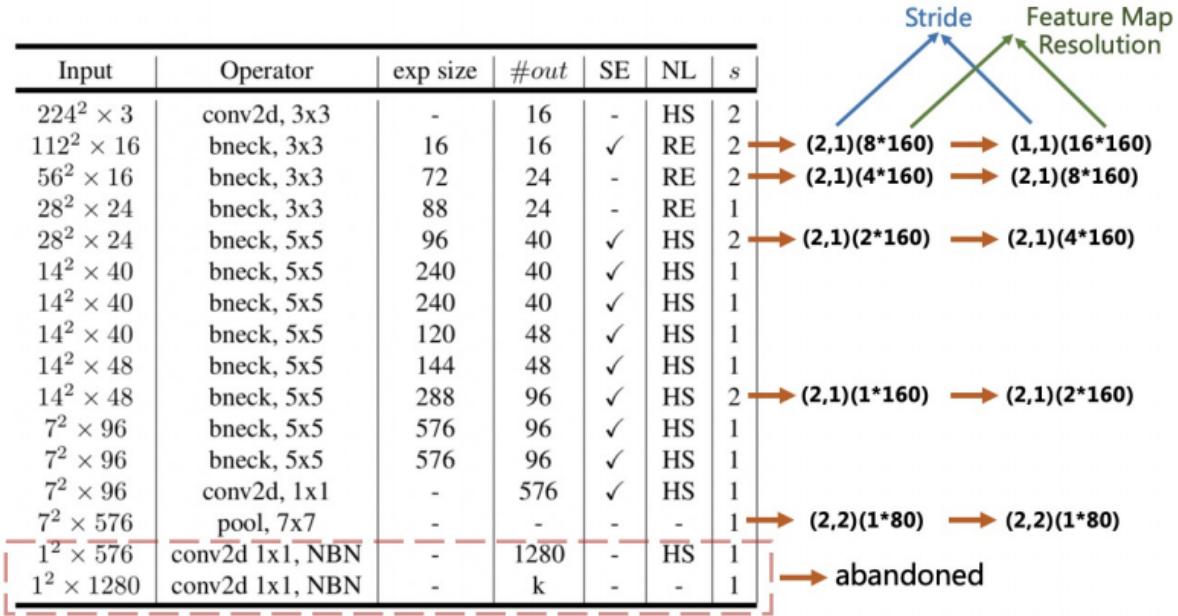


Figure 25: visualization of downsampling step strategy optimization

In order to retain more vertical information, the step size of the second downsampling feature map are modified from **(2,1)** to **(1,1)**. Therefore, the step size s2 of the second down-sampling feature map will significantly affect the resolution of the entire feature map and the accuracy of the text recognizer. In PP-OCR, s2 is set to (1,1) to gain better performance. At the same time, due to the increase of horizontal resolution, the inference time of the CPU increases from 11.84ms to 12.96ms.

The following shows the comparison of the feature map scales before and after stride optimization. Although the output feature maps have the same scale, after stride is modified from (2,1) to (1,1), the feature information is preserved better in encoding.

```
# Reference Code:
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/ppocr/modeling/backbones/
# rec_mobilenet_v3.py
from ppocr.modeling.backbones.rec_mobilenet_v3 import MobileNetV3

mv3_ori = MobileNetV3(model_name="small", scale=0.5, small_stride=[2, 2, 2, 2])
mv3_new = MobileNetV3(model_name="small", scale=0.5, small_stride=[1, 2, 2, 2])

x = paddle.rand([1, 3, 32, 320])

y_ori = mv3_ori(x)
y_new = mv3_new(x)

print(y_ori.shape)
print(y_new.shape)
```

```
[1, 288, 1, 80]
[1, 288, 1, 80]
```

PACT Online Quantization Strategy

We use a scheme similar to the quantization of the direction classifier to reduce the model size of the text recognizer. Due to the complexity of LSTM quantization, LSTM is not quantized in PP-OCR. Through the quantization strategy, the model size is reduced by 67.4%, the prediction speed increases by 8%, and the accuracy by 1.6%. Quantization can reduce model redundancy and enhance the expression ability of the model.

PACT Quantization	Accuracy	Model Size (M)	Inference Time (SD 855, ms)
	0.6581	4.6	12
✓	0.674	1.5	11

Figure 26: Ablation experiment of model quantitation

Text Recognition Pre-training Model

A suitable pre-trained model can speed up the convergence of the model. In real scenarios, the data used for text recognition is usually limited. In PP-OCR, tens of millions of levels of data are synthesized to train the model, and real data will be refined based on the model. Finally, recognition accuracy has increased from 65.81% to 69%.

Description of Text Recognition Configuration

A brief description of the training configuration of CRNN is given below. For the complete configuration file, refer to: [rec_chinese_lite_train_v2.0.yml](#).

```

Optimizer:
  name: Adam
  beta1: 0.9
  beta2: 0.999
  lr:
    name: Cosine # Configure Cosine learning rate reduction strategy
    learning_rate: 0.001
    warmup_epoch: 5 # Configure warmup learning rate
  regularizer:
    name:'L2' # Configure L2 regular
    factor: 0.00001

Architecture:
  model_type: rec
  algorithm: CRNN
  Transform:
  Backbone:
    name: MobileNetV3 # Configure Backbone
    scale: 0.5
    model_name: small
    small_stride: [1, 2, 2, 2] # Configure the stride for downsampling
  Neck:
    name: SequenceEncoder
    encoder_type: rnn
    hidden_size: 48 # Configure the dimension of the last fully connected layer
  Head:

```

(continues on next page)

(continued from previous page)

```
name: CTCHead
fc_decay: 0.00001

Train:
dataset:
    name: SimpleDataSet
    data_dir: ./train_data/
    label_file_list: ["./train_data/train_list.txt"]
    transforms:
        -DecodeImage: # load image
            img_mode: BGR
            channel_first: False
        -RecAug: # Configuration data enhancement BDA and TIA, TIA is used by default
        -CTCLabelEncode: # Class handling label
        -RecResizeImg:
            image_shape: [3, 32, 320]
        -KeepKeys:
            keep_keys: ['image', 'label', 'length'] # dataloader will return list in this order
loader:
    shuffle: True
    batch_size_per_card: 256
    drop_last: True
    num_workers: 8
```

Summary of Recognition Optimization

In terms of model size, PP-OCR reduces the model size from 4.5M to 1.6M through a lightweight backbone network, sequence dimension tailoring, and model quantization strategies. In terms of the accuracy, it improves the accuracy by 15.4% on the validation set by using optimization strategies such as TIA data augmentation, cosine-warmup learning rate strategy, L2 regularization, feature map resolution improvement, and pre-training model.

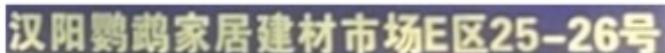
Some of the recognition results in PP-OCR are shown below.



Predicts of word_4.jpg:[‘实力活力’, 0.9954298]



Predicts of word_3004.jpg:[‘原装电池’, 0.99971426]



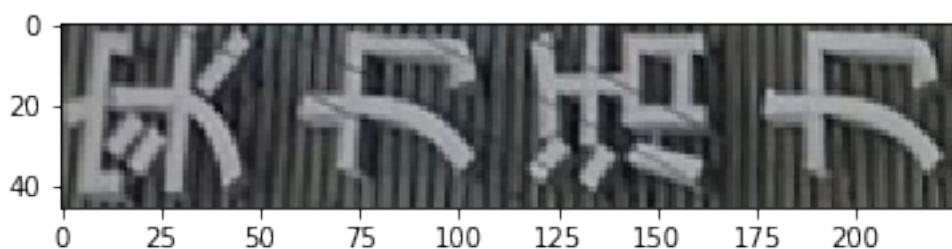
Predicts of word_2.jpg:[‘汉阳鹦鹉家居建材市场E区25-26号’, 0.9957605]

Figure 27: The recognition results in PP-OCR

The code demonstration of the text recognition model is as follows.

```
# Visualize the original image
img = cv2.imread("./doc/imgs_words/ch/word_4.jpg")
plt.imshow(img[..., ::-1])
plt.show()

!cd inference && wget https://paddleocr.bj.bcebos.com/PP-OCRV2/chinese/ch_PP-OCRV2_
->rec_infer.tar -O ch_PP-OCRV2_rec_infer.tar && tar -xf ch_PP-OCRV2_rec_infer.tar
!python tools/infer/predict_rec.py --image_dir="./doc/imgs_words/ch/word_4.jpg" --rec_
->model_dir="../inference/ch_PP-OCRV2_rec_infer" --use_gpu=False
```



```
--2021-12-24 21:50:31-- https://paddleocr.bj.bcebos.com/PP-OCRV2/chinese/ch_PP-
->OCRV2_rec_infer.tar
Resolving paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com) ... 182.61.200.195, 182.
->61.200.229, 2409:8c04:1001:1002:0:ff:b001:368a
Connecting to paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com) |182.61.200.
->195|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8875520 (8.5M) [application/x-tar]
```

(continues on next page)

(continued from previous page)

```

Saving to: 'ch_PP-OCRv2_rec_infer.tar'

ch_PP-OCRv2_rec_inf 100%[=====] 8.46M 24.2MB/s in 0.4s

2021-12-24 21:50:31 (24.2 MB/s) - 'ch_PP-OCRv2_rec_infer.tar' saved [8875520/
˓→8875520]

[2021/12/24 21:50:33] root INFO: Predicts of ./doc/imgs_words/ch/word_4.jpg: ('한국小館',
˓→ 0.9409585)

```

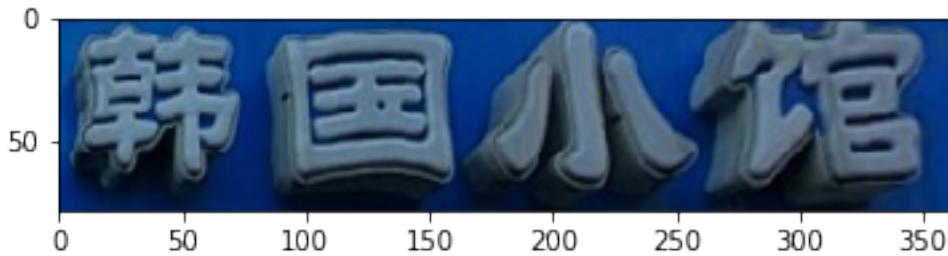
```

import cv2
# Rotate ./doc/imgs_words/ch/word_1.jpg 180 degrees to get
img = cv2.imread("./doc/imgs_words/ch/word_1.jpg")

plt.imshow(img[:, :, ::-1])
plt.show()

!python tools/infer/predict_rec.py --image_dir=". ./doc/imgs_words/ch/word_1.jpg" --
˓→rec_model_dir=". /inference/ch_PP-OCRv2_rec_infer" --use_gpu=False

```



```
[2021/12/24 21:52:00] root INFO: Predicts of . ./doc/imgs_words/ch/word_1.jpg: (
˓→'한국小館', 0.9967349)
```

6.3 Interpretation of PP-OCRv2 Optimization Strategies

The Section 2 mainly introduces PP-OCR and its 19 optimization strategies in detail.

Compared with PP-OCR, PP-OCRv2 further optimizes the three aspects of the backbone network, including data augmentation, and the loss function to solve the poor end-to-side prediction efficiency, complex background, and misrecognition of similar characters. It also introduces The knowledge distillation training strategy further improves the accuracy of the model. That is:

- Detection model optimization: (1) CML knowledge distillation strategy; (2) CopyPaste data augmentation strategy;
- Recognition model optimization: (1) PP-LCNet; (2) U-DML knowledge distillation strategy; (3) Enhanced CTC loss.

This section mainly interprets the optimization strategies of PP-OCRv2 based on the optimization process of text detection and recognition models.

6.3.1 Detailed Explanation of Text Detection Model Optimization

In the process of optimizing the text detection model, the CML knowledge distillation and CopyPaste data augmentation strategy are adopted. And the text detection model is increased from **0.759** to **0.795** with the size of the text detection model remaining unchanged. The ablation experiment is shown below.

Strategy	Precision	Recall	Hmean	Model Size (M)	Inference Time (CPU, ms)
PP-OCR det	0.718	0.805	0.759	3.0	129
PP-OCR det + DML	0.743	0.815	0.777	3.0	129
PP-OCR det + CML	0.746	0.835	0.789	3.0	129
PP-OCR det + CML + CopyPaste	0.754	0.840	0.795	3.0	129

Table 2: Ablation study of CML and CopyPaste for text detection.

Figure 28: Ablation experiment of the PP-OCRv2 detection model

CML Knowledge Distillation Strategy

The knowledge distillation method is commonly used in deployment. By using a large model to guide the learning of a small model, the accuracy of the small model can usually be further improved while the prediction time is the same, thereby further improving the actual deployment experience.

The standard distillation method is to use a large model as the teacher model to guide the student model to improve the effect. Later on, the DML mutual learning distillation method has been developed, which is to learn from each other through two models with the same structure. Compared with the former, DML does not depend on the large teacher model, making the distillation training process simpler, and improving the model output efficiency.

The PP-OCRv2 text detection model uses the CML (Collaborative Mutual Learning) distillation method between the three models, which involves mutual learning between two student models of the same structure, and introduces a larger model structure. Teacher model. The comparison between CML and other distillation algorithms is shown below.

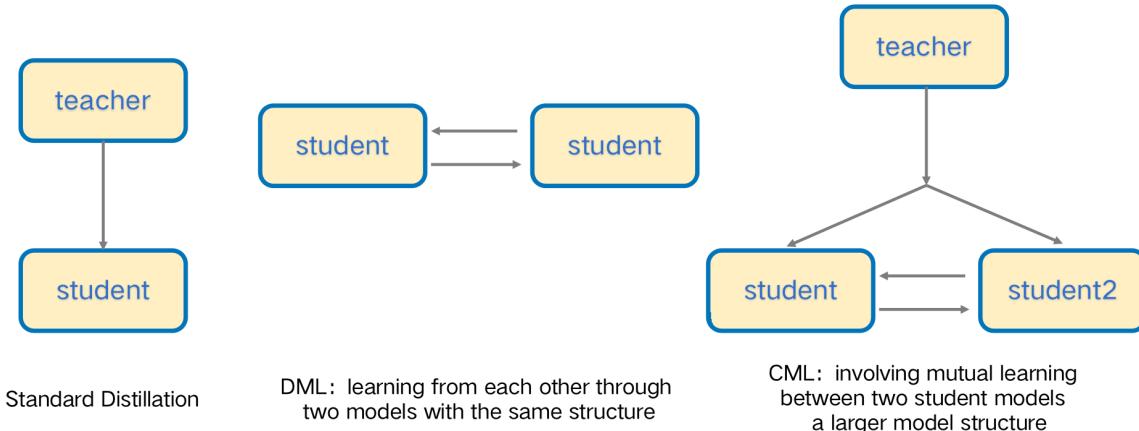


Figure 29: Comparison between CML and other knowledge distillation algorithms

In the text detection task, the structure diagram of CML is as follows. The response maps refer to the probability map output of the last layer of DBNet. In the whole training process, three loss functions are included.

- GT loss
- DML loss
- Distill loss

The backbone network of the teacher model here is ResNet18_vd, and that of the two Student models is MobileNetV3.

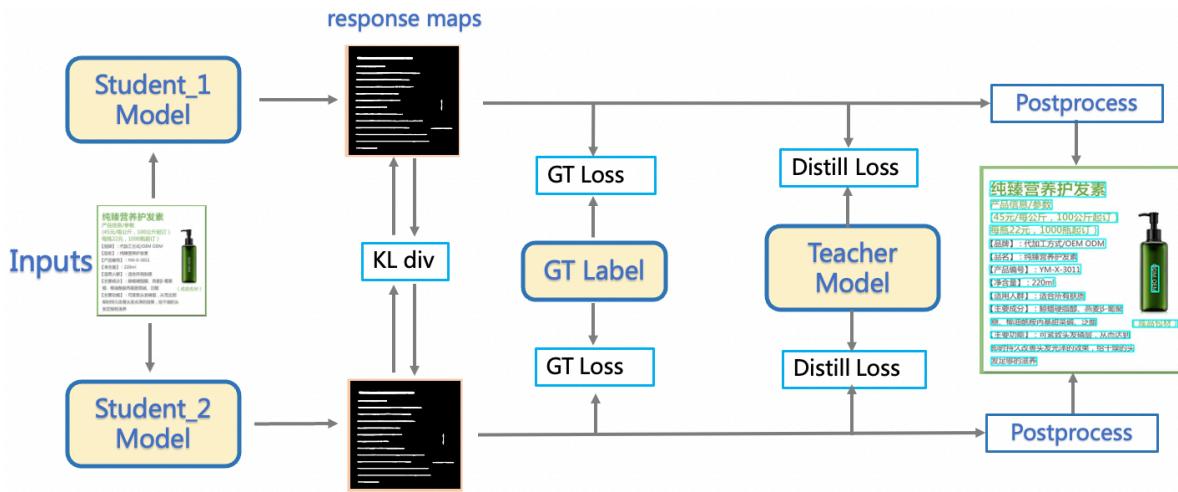


Figure 30: CML structure diagram

- GT loss

Most of the parameters in the two student models are initialized from scratch, so they need to be supervised by groundtruth (GT) information during the training. The pipeline of the DBNet training task is shown below. The output mainly contains three kinds of feature maps.

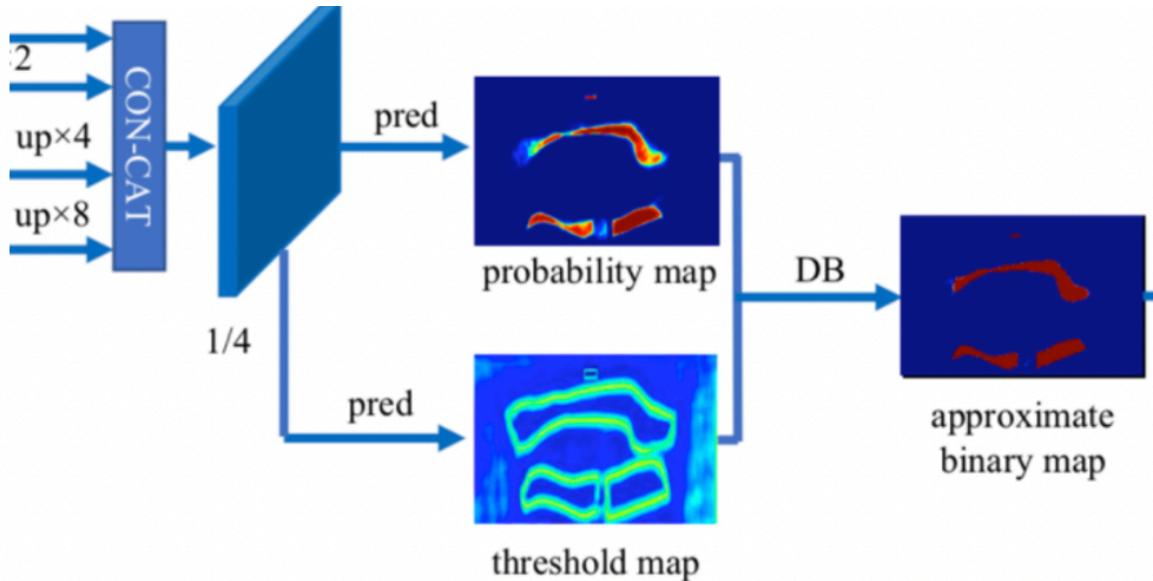


Figure 31: DBNet head structure

Use different loss functions to supervise these feature maps, as shown in the following table.

Feature map	Loss function	weight
Probability map	Binary cross-entropy loss	1.0
Binary map	Dice loss	α
Threshold map	L1 loss	β

The final GT loss can be expressed as below.

$$Loss_{gt}(T_{out}, gt) = l_p(S_{out}, gt) + \alpha l_b(S_{out}, gt) + \beta l_t(S_{out}, gt)$$

- DML loss

For two identical student models, they should have the same output for the same input. The final output of DBNet is probability maps (response maps). Based on the KL divergence, calculate the DML loss of two student models as follows.

$$Loss_{dml} = \frac{KL(S1_{pout} || S2_{pout}) + KL(S2_{pout} || S1_{pout})}{2}$$

Among them, $KL(\cdot | \cdot)$ is the calculation formula of KL divergence, and the calculated DML loss is symmetrical.

- Distill loss

In CML, the teacher model is introduced to supervise two Student models at the same time. In PP-OCRv2, only the feature Probability map is supervised by distillation. Specifically, for one of the student models, the calculation method is as follows. $l_p(\cdot)$ and $l_b(\cdot)$ represent binary cross-entropy loss and dice loss respectively. The loss calculation process of the other student model is exactly the same.

$$Loss_{distill} = \gamma l_p(S_{out}, f_{dila}(T_{out})) + l_b(S_{out}, f_{dila}(T_{out}))$$

Finally, by adding the above three losses, we can get the loss function used for CML training.

The detection configuration file is `ch_PP-OCRv2_det_cml.yml`, distillation structure part The configuration and part of the explanation are as follows.

```
Architecture:
  name: DistillationModel # Model name, this is the general distillation model
  ↪representation.
  algorithm: Distillation # Algorithm name,
  Models: # Models, including the configuration information of the subnet
    Teacher: # Teacher sub-network, including `pretrained` and `freeze_params` ↪
    ↪information and other parameters used to construct the sub-network
      freeze_params: true # Whether to fix the parameters of the Teacher network
      pretrained: ./pretrain_models/ch_ppocr_server_v2.0_det_train/best_accuracy # ↪
      ↪pretrained model
      return_all_feats: false # Whether to return all features, when it is True, the ↪
      ↪output of backbone, neck, head and other modules will be returned
      model_type: det # Model category
      algorithm: DB # Teacher network algorithm name
      Transform:
      Backbone:
        name: ResNet
        layers: 18
      Neck:
        name: DBFPN
        out_channels: 256
      Head:
        name: DBHead
        k: 50
  Student: # Student subnet
    freeze_params: false
    pretrained: ./pretrain_models/MobileNetV3_large_x0_5_pretrained
    return_all_feats: false
    model_type: det
    algorithm: DB
    Backbone:
```

(continues on next page)

(continued from previous page)

```

    name: MobileNetV3
    scale: 0.5
    model_name: large
    disable_se: True
Neck:
    name: DBFPN
    out_channels: 96
Head:
    name: DBHead
    k: 50
Student2: # Student2 subnet
    freeze_params: false
    pretrained: ./pretrain_models/MobileNetV3_large_x0_5_pretrained
    return_all_feats: false
    model_type: det
    algorithm: DB
Transform:
Backbone:
    name: MobileNetV3
    scale: 0.5
    model_name: large
    disable_se: True
Neck:
    name: DBFPN
    out_channels: 96
Head:
    name: DBHead
    k: 50

```

The implementation of the DistillationModel is in the `distillation_model.py` file, and the implementation and part of the explanation of DistillationModel are as follows.

```

class DistillationModel(nn.Layer):
    def __init__(self, config):
        """
        the module for OCR distillation.
        args:
            config (dict): the super parameters for module.
        """
        super().__init__()
        self.model_list = []
        self.model_name_list = []
        # According to each field in Models, extract the name of the subnet and the
        # corresponding configuration
        for key in config["Models"]:
            model_config = config["Models"][key]
            freeze_params = False
            pretrained = None
            if "freeze_params" in model_config:
                freeze_params = model_config.pop("freeze_params")
            if "pretrained" in model_config:
                pretrained = model_config.pop("pretrained")
            # According to the configuration of each sub-network, generate sub-
            # networks based on BaseModel
            model = BaseModel(model_config)
            # Determine whether to load the pre-trained model

```

(continues on next page)

(continued from previous page)

```

if pretrained is not None:
    load_pretrained_params(model, pretrained)
    # Determine whether it is necessary to fix the model parameters of the
    ↪sub-network
    if freeze_params:
        for param in model.parameters():
            param.trainable = False
    self.model_list.append(self.add_sublayer(key, model))
    self.model_name_list.append(key)

def forward(self, x):
    result_dict = dict()
    for idx, model_name in enumerate(self.model_name_list):
        result_dict[model_name] = self.model_list[idx](x)
    return result_dict

```

Use the following command to quickly the distillation model.

```

# Reference Code
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/ppocr/modeling/
↪architectures/_init__.py
from tools.program import load_config
from ppocr.modeling.architectures import build_model
config_path = "./configs/det/ch_PP-OCRv2/ch_PP-OCRv2_det_cml.yml"
config = load_config(config_path)
model = build_model(config['Architecture'])
print(model)

```

You can quickly experience the training process of CML distillation in the following way.

```

# Reference Code
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/tools/train.py
os.chdir("/home/aistudio/PaddleOCR/")
!mkdir train_data
!wget https://paddleocr.bj.bcebos.com/dataset/det_data_lesson_demo.tar -O det_data_
↪lesson_demo.tar && tar -xf det_data_lesson_demo.tar && rm det_data_lesson_demo.tar
!mkdir pretrain_models && wget https://paddleocr.bj.bcebos.com/dygraph_v2.0/ch/ch_
↪ppocr_server_v2.0_det_train.tar && tar -xf ch_ppocr_server_v2.0_det_train.tar
!mv ch_ppocr_server_v2.0_det_train pretrain_models/ && rm ch_ppocr_server_v2.0_det_
↪train.tar
# Training script
# Note: Only one epoch is trained here, which is only used for quick demonstration,
↪and the indicators will be very poor
!python tools/train.py -c configs/det/ch_PP-OCRv2/ch_PP-OCRv2_det_cml.yml \
-o Global.pretrained_model="" \
Train.dataset.data_dir="./det_data_lesson_demo/" \
Train.dataset.label_file_list=[("./det_data_lesson_demo/train.txt")] \
Train.loader.num_workers=0 \
Eval.dataset.data_dir="./det_data_lesson_demo/" \
Eval.dataset.label_file_list=[("./det_data_lesson_demo/eval.txt")] \
Eval.loader.num_workers=0 \
Optimizer.lr.learning_rate=0.00025 \
Global.epoch_num=1

```

Data Augmentation

Data augmentation is one of the important means to improve the generalization ability of the model. CopyPaste is a novel data augmentation technique that has been validated in target detection and segmentation. With CopyPaste, you can synthesize text instances to balance the ratio of positive to negative samples in the training image. In contrast, traditional image rotation, random flip and random cropping cannot achieve this.

The steps of CopyPaste include:

1. Randomly selecting two training images;
2. Random scale jittering;
3. flipping horizontally ;
4. Randomly selecting a target subset in an image;
5. Pasting it in a random location in another image.

In this way, the samples are more diverse, and the model is more robust. As shown in the figure below, the text cropped from the image in the lower left corner is randomly rotated and zoomed and then pasted into the upper left image. In this way, the background of the text is more diverse.



Figure 32: A `CopyPaste` result

If you want to use CopyPaste in model training, add CopyPaste in the Train.transforms configuration field.

```
Train:
dataset:
  name: SimpleDataSet
  data_dir: ./train_data/icdar2015/text_localization/
  label_file_list:
    - ./train_data/icdar2015/text_localization/train_icdar2015_label.txt
  ratio_list: [1.0]
transforms:
  - DecodeImage: # load image
    img_mode: BGR
    channel_first: False
  - DetLabelEncode: # Class handling label
  - CopyPaste: # Add CopyPaste
  - IaaAugment:
```

(continues on next page)

(continued from previous page)

```

augmenter_args:
    - { 'type': Fliplr, 'args': { 'p': 0.5 } }
    - { 'type': Affine, 'args': { 'rotate': [-10, 10] } }
    - { 'type': Resize, 'args': { 'size': [0.5, 3] } }
- EastRandomCropData:
    size: [960, 960]
    max_tries: 50
    keep_ratio: true
- MakeBorderMap:
    shrink_ratio: 0.4
    thresh_min: 0.3
    thresh_max: 0.7
- MakeShrinkMap:
    shrink_ratio: 0.4
    min_text_size: 8
- NormalizeImage:
    scale: 1./255.
    mean: [0.485, 0.456, 0.406]
    std: [0.229, 0.224, 0.225]
    order: 'hwc'
- ToCHWImage:
- KeepKeys:
    keep_keys: ['image', 'threshold_map', 'threshold_mask', 'shrink_map',
↳ 'shrink_mask'] # the order of the dataloader list
loader:
    shuffle: True
    drop_last: False
    batch_size_per_card: 8
    num_workers: 4

```

For example of CopyPaste, refer to [copy_paste.py](#).

In the following, CopyPaste will be demonstrated based on the icdar2015 detection dataset.

```

import os
import sys

os.chdir("/home/aistudio/PaddleOCR/")
!unzip -oq /home/aistudio/data/data46088/icdar2015.zip

```

```

# Reference Code:
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/ppocr/data/simple_
↳dataset.py
import logging
import random
import numpy as np

from ppocr.data.imaug import create_operators, transform

logger = logging.basicConfig()

# CopyPaste example
class CopyPasteDemo(object):
    def __init__(self, ):
        self.data_dir = "./icdar2015/text_localization/"

```

(continues on next page)

(continued from previous page)

```

    self.label_file_list = "./icdar2015/text_localization/train_icdar2015_label.
→txt"
    self.data_lines = self.get_image_info_list(self.label_file_list)
    self.data_idx_order_list = list(range(len(self.data_lines)))
    transforms = [
        {"DecodeImage": {"img_mode": "BGR", "channel_first": False}},
        {"DetLabelEncode": {}},
        {"CopyPaste": {"objects_paste_ratio": 1.0}},
    ]
    self.ops = create_operators(transforms)

# Select an image and copy the content to the current image
def get_ext_data(self, idx):
    ext_data_num = 1
    ext_data = []

    load_data_ops = self.ops[:2]

    next_idx = idx

    while len(ext_data) < ext_data_num:
        next_idx = (next_idx + 1) % len(self)
        file_idx = self.data_idx_order_list[next_idx]
        data_line = self.data_lines[file_idx]
        data_line = data_line.decode('utf-8')
        substr = data_line.strip("\n").split("\t")
        file_name = substr[0]
        label = substr[1]
        img_path = os.path.join(self.data_dir, file_name)
        data = {'img_path': img_path, 'label': label}
        if not os.path.exists(img_path):
            continue
        with open(data['img_path'], 'rb') as f:
            img = f.read()
            data['image'] = img
        data = transform(data, load_data_ops)
        if data is None:
            continue
        ext_data.append(data)
    return ext_data

# Get image information
def get_image_info_list(self, file_list):
    if isinstance(file_list, str):
        file_list = [file_list]
    data_lines = []
    for idx, file in enumerate(file_list):
        with open(file, "rb") as f:
            lines = f.readlines()
            data_lines.extend(lines)
    return data_lines

# Get a piece of data in the DataSet
def __getitem__(self, idx):
    file_idx = self.data_idx_order_list[idx]
    data_line = self.data_lines[file_idx]

```

(continues on next page)

(continued from previous page)

```

try:
    data_line = data_line.decode('utf-8')
    substr = data_line.strip("\n").split("\t")
    file_name = substr[0]
    label = substr[1]
    img_path = os.path.join(self.data_dir, file_name)
    data = {'img_path': img_path, 'label': label}
    if not os.path.exists(img_path):
        raise Exception("{} does not exist!".format(img_path))
    with open(data['img_path'], 'rb') as f:
        img = f.read()
        data['image'] = img
    data['ext_data'] = self.get_ext_data(idx)
    outs = transform(data, self.ops)
except Exception as e:
    print(
        "When parsing line {}, error happened with msg: {}".format(
            data_line, e))
    outs = None
if outs is None:
    return
return outs

def __len__(self):
    return len(self.data_idx_order_list)

copy_paste_demo = CopyPasteDemo()

idx = 1
data1 = copy_paste_demo[idx]
print(data1.keys())
print(data1["img_path"])
print(data1["ext_data"][0]["img_path"])

```

```

dict_keys(['img_path', 'label', 'image', 'ext_data', 'polys', 'texts', 'ignore_tags
          ''])
./icdar2015/text_localization/icdar_c4_train_imgs/img_603.jpg
./icdar2015/text_localization/icdar_c4_train_imgs/img_233.jpg

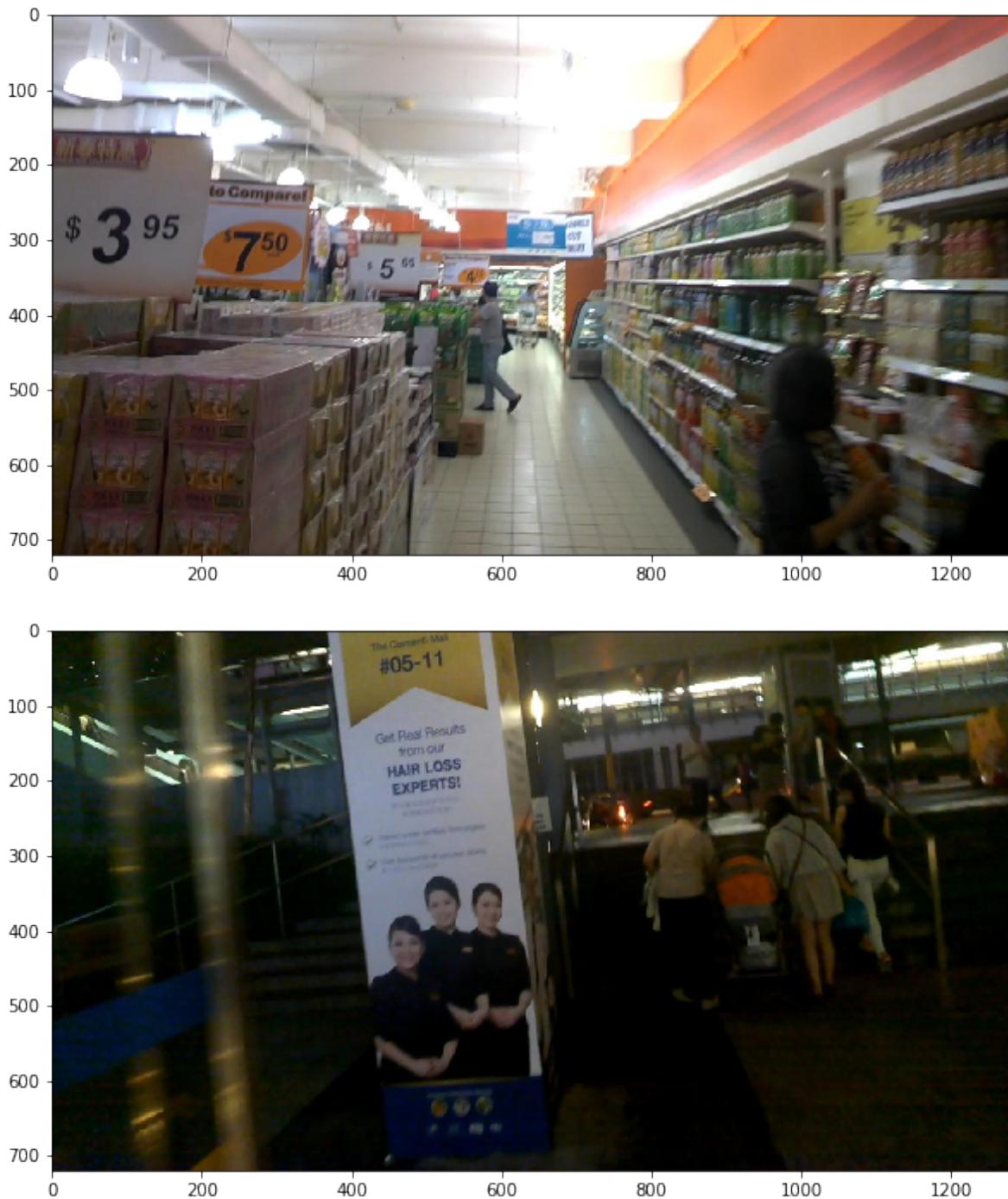
```

- The following 2 pictures are the images before getting through CopyPaste.

```

import cv2
import matplotlib.pyplot as plt
%matplotlib inline
img1 = cv2.imread(data1["img_path"])
img2 = cv2.imread(data1["ext_data"][0]["img_path"])
plt.figure(figsize=(10, 6))
plt.imshow(img1[:, :, ::-1])
plt.show()
plt.figure(figsize=(10, 6))
plt.imshow(img2[:, :, ::-1])
plt.show()

```



- Draw the updated label detection box, where the red boxes refer to the original labelled information, and the blue boxes are the added label boxes after CopyPaste.

```
import json
infos = copy_paste_demo.data_lines[idx]
infos = json.loads(infos.decode('utf-8').split("\t")[1])

img3 = data1["image"].copy()
plt.figure(figsize=(15,10))
```

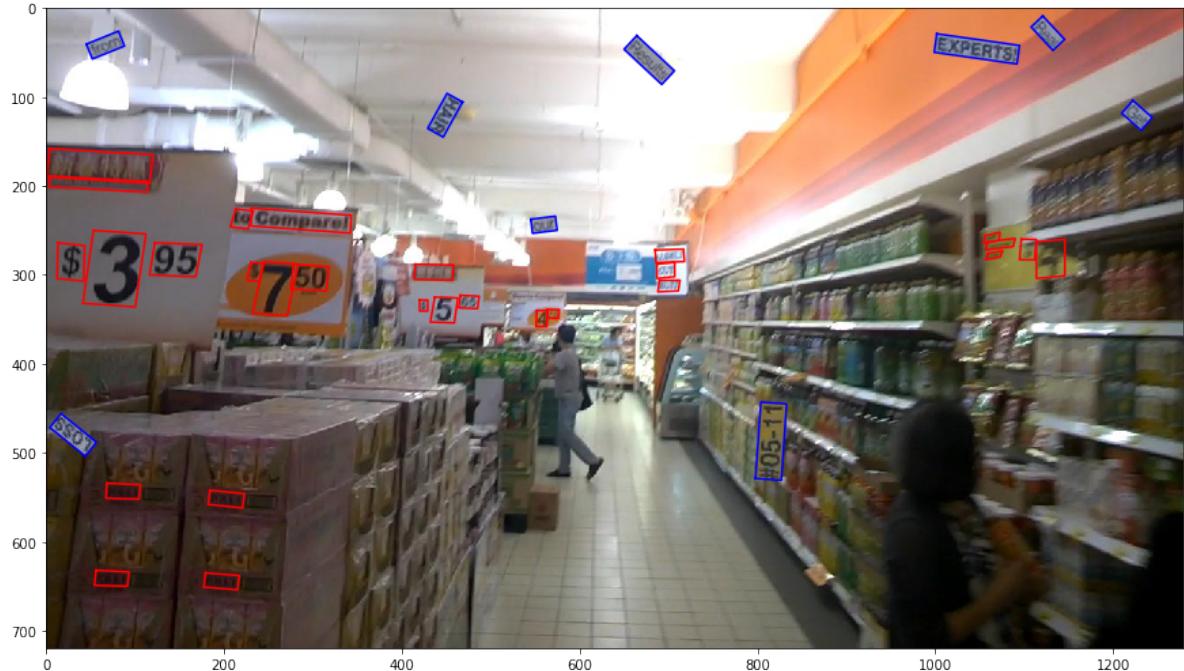
(continues on next page)

(continued from previous page)

```

plt.imshow(img3[:, :, ::-1])
# Original labelled information
for info in infos:
    xs, ys = zip(*info["points"])
    xs = list(xs)
    ys = list(ys)
    xs.append(xs[0])
    ys.append(ys[0])
    plt.plot(xs, ys, "r")
# Added labelled information
for poly_idx in range(len(infos), len(data1["polys"])):
    poly = data1["polys"][poly_idx]
    xs, ys = zip(*poly)
    xs = list(xs)
    ys = list(ys)
    xs.append(xs[0])
    ys.append(ys[0])
    plt.plot(xs, ys, "b")
plt.show()

```



Summary of Text Detection Optimization

In PP-OCRv2, the knowledge distillation scheme and the data augmentation strategy are adopted for the text detection model to increase its generalization performance. With the detection model size remaining the same, Hmean has increased from **0.759** to **0.795**. The ablation experiment is displayed below.

Strategy	Precision	Recall	Hmean	Model Size (M)	Inference Time (CPU, ms)
PP-OCR det	0.718	0.805	0.759	3.0	129
PP-OCR det + DML	0.743	0.815	0.777	3.0	129
PP-OCR det + CML	0.746	0.835	0.789	3.0	129
PP-OCR det + CML + CopyPaste	0.754	0.840	0.795	3.0	129

Table 2: Ablation study of CML and CopyPaste for text detection.

Figure 33: PP-OCRV2 detection model ablation experiment

The detection result of PP-OCRV2 is shown below.



Figure 34: The detection result of PP-OCRV2

6.3.2 Detailed Explanation of Text Recognition Model Optimization

PP-OCRv2 text recognition model is optimized through the backbone network optimization, UDM knowledge distillation strategy, and CTC loss improvement. Finally, the recognition accuracy increases from **66.7%** to **74.8%**. Its ablation experiments are as follows.

Strategy	Acc	Model Size (M)	Inference Time (CPU, ms)
PP-OCR rec (MV3)	0.667	5.0	7.7
PP-OCR rec (LCNet)	0.693	8.0	6.2
PP-OCR rec (LCNet) + U-DML	0.739	8.6	6.2
PP-OCR rec (LCNet) + U-DML + Enhanced CTC loss	0.748	8.6	6.2

Table 3: Ablation study of LCNet, U-DML, and Enhanced CTC loss for text recognition.

Figure 35: Ablation experiment of PP-OCRv2 recognition model

PP-LCNet Lightweight Backbone Network

Baidu has proposed a lightweight CPU network based on MKLDNN acceleration strategy, PP-LCNet, which can greatly improve the performance of lightweight models in image classification. In downstream tasks of computer vision, such as text recognition, target detection, and semantic segmentation, it performs well. It should be noted that PP-LCNet is customized for the **CPU+MKLDNN** scenario, and far better than other models on the speed and accuracy in the classification task. So if you need models in this scenario, PP-LCNet is recommended.

A PP-LCNet paper : [PP-LCNet: A Lightweight CPU Convolutional Neural Network](#)

PP-LCNet is improved from MobileNetV1, and its structure is shown below.

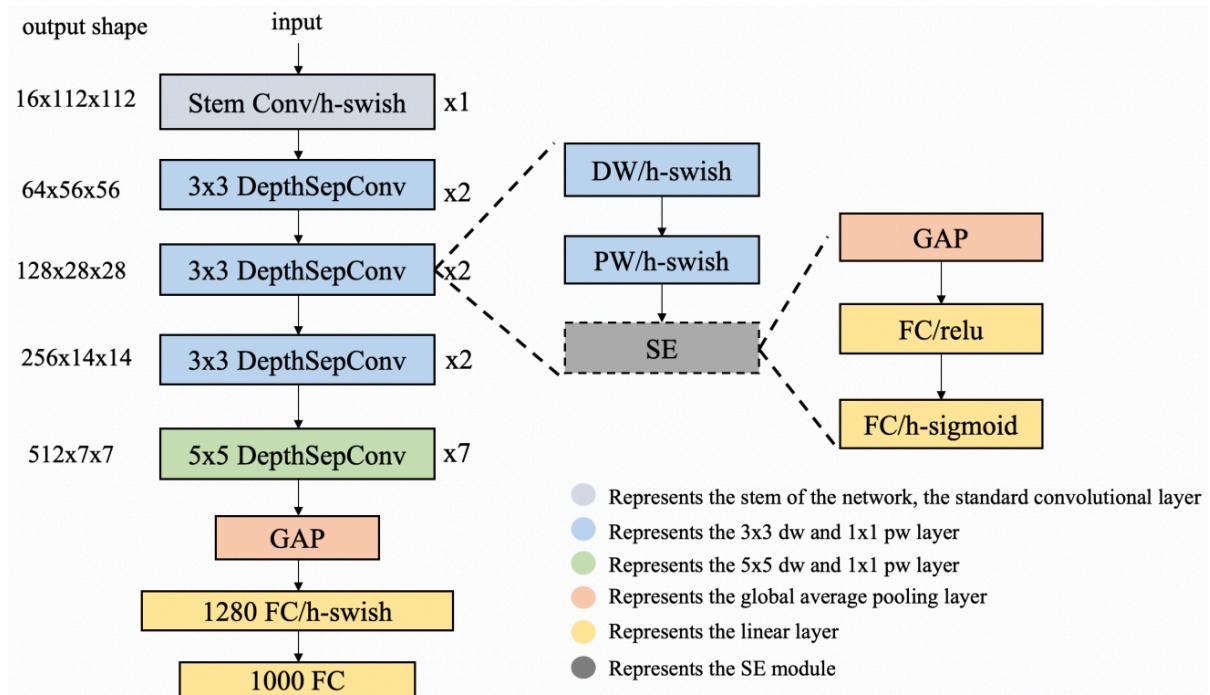


Figure 36: PP-LCNet structure

Compared with MobileNetV1, PP-LCNet integrates the activation function, head structure, SE modules and other strategy optimization techniques in the MobileNetV3 structure. At the same time, it analyzes the convolution kernel size of the convolution layer in the final stage. In the end, the model is able to guarantee the speed and outperform lightweight models such as MobileNet and GhostNet in accuracy.

PP-LCNet is optimized in four aspects:

- Except for SE modules, all relu activation functions in the network are replaced with h-swish, and the accuracy is improved by 1%-2%;
- In the fifth stage of PP-LCNet, the kernel size of DW turns into 5x5, and the accuracy is improved by 0.5%-1%;
- The last two DepthSepConv blocks of the fifth stage of PP-LCNet add SE modules, and the accuracy is increased by 0.5%-1%;
- A 1280-dimensional FC layer is added after the GAP layer to increase feature expression capability, and the accuracy is increased by 2%-3%.

On the ImageNet1k dataset, the comparison between PP-LCNet and other common lightweight classification models in Top1-Acc and the inference time are shown in the figure below. It can be seen that PP-LCNet excels in the two indicators.

Model	Top1-Acc (%)	Inference Time (ms)
MobileNetV1-0.75x	68.81	3.88
MobileNetV2-0.75x	69.83	4.56
MobileNetV3-small-1.0x	68.24	4.20
MobileNetV3-large-0.5x	69.24	4.54
GhostNet-0.5x	66.88	6.63
PP-LCNet-1.0x	71.32	3.16

Figure 37: Comparison of state-of-the-art light networks over classification accuracy

The PP-LCNet recognition model can be rapidly defined as follows.

```
# Reference Code
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/ppocr/modeling/backbones/
# rec_mv1_enhance.py
from ppocr.modeling.backbones.rec_mv1_enhance import MobileNetV1Enhance

x = paddle.rand([1, 3, 23, 320])

model = MobileNetV1Enhance(scale=0.5)

y = model(x)
print(y.shape)
```

[1, 512, 1, 80]

U-DML Knowledge Distillation Strategy

In the standard DML strategy, the distilled loss function only includes the supervision of the output layer. However, for two models with the identical structure, their intermediate features and output expectations should be the same. So in the supervision of the output layer, the supervision signal of the intermediate output feature map can be further added as a loss function, that is, the U-DML (Unified-Deep Mutual Learning) knowledge distillation method in PP-OCRv2.

The flow chart of U-DML knowledge distillation is shown below. The structure of the teacher network and the student network are exactly the same, and the initialization parameters are different. In addition, on the basis of the standard DML knowledge distillation, a new supervision mechanism for feature maps and a feature loss have been introduced.

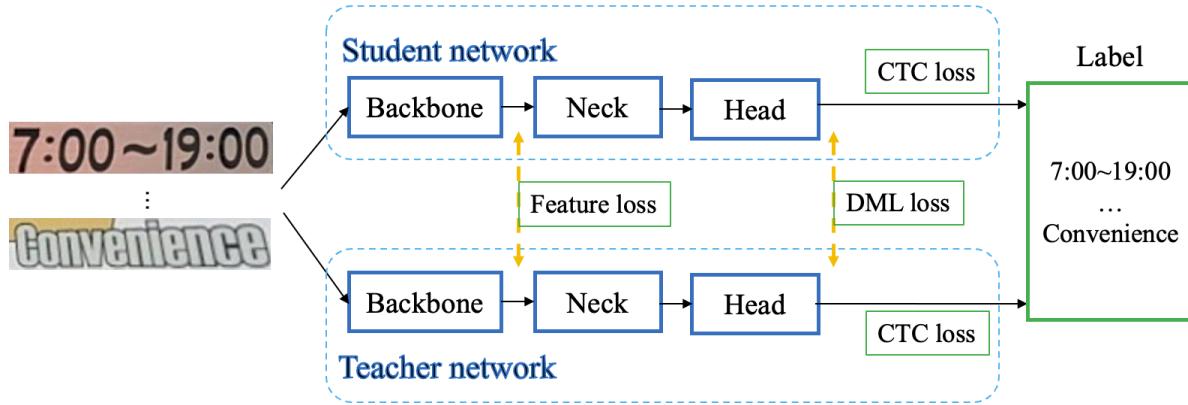


Figure 38: The flow chart of U-DML knowledge distillation

In the training process, there are three types of loss: GT loss, DML loss, Feature loss.

- GT loss

The model structure in the text recognition task is CRNN, so CTC loss is used as GT loss. The calculation of GT loss is as follows.

$$\text{Loss}_{ctc} = \text{CTC}(S_{hout}, gt) + \text{CTC}(T_{hout}, gt)$$

- DML loss

The calculation of DML loss is as follows. The teacher model and the student model calculate the KL divergence each other, and the DML loss is symmetrical.

$$\text{Loss}_{dml} = \frac{\text{KL}(S_{pout} || T_{pout}) + \text{KL}(T_{pout} || S_{pout})}{2}$$

- Feature loss

Feature loss uses L2 loss, and its calculation is shown below.

$$\text{Loss}_{feat} = \text{L2}(S_{bou}, T_{bou})$$

Finally, the calculation of loss function during training is as below.

$$\text{Loss}_{total} = \text{Loss}_{ctc} + \text{Loss}_{dml} + \text{Loss}_{feat}$$

In addition, during the training process, the model effect is improved by increasing the iteration number, adding tricks such as the FC layer to the head, balancing the feature encoding and decoding capabilities of the model.

The configuration file is `inch_PP-OCRv2_rec_distillation.yml`

```

Architecture:
  model_type: &model_type "rec" # Model category, like rec, det, etc., and each sub-
  ↪network of which is related to
  name: DistillationModel # Structure name, is DistillationModel in the distillation_
  ↪task, used to construct the corresponding structure
  algorithm: Distillation # Algorithm name
  Models: # Models, including the configuration information of subnets
    Teacher: # The name of the sub-network must contain `pretrained` and `freeze_`  

    ↪params, and the other parameters are the construction parameters of sub-networks
      pretrained: # Whether it is needed to load a pre-trained model in sub-networks
      freeze_params: false # Whether fixed parameters are required
      return_all_feats: true # The parameters of subnets indicate whether all_
      ↪features need to be returned. If it is False, only the final output is returned
      model_type: *model_type # Model category
      algorithm: CRNN # The algorithm names of sub-networks. The remaining_
      ↪participation of the sub-network is the construction parameter, which is consistent_
      ↪with the general model in training configuration
    Transform:
    Backbone:
      name: MobileNetV1Enhance
      scale: 0.5
    Neck:
      name: SequenceEncoder
      encoder_type: rnn
      hidden_size: 64
    Head:
      name: CTCHead
      mid_channels: 96 # Intersperse a layer in the head decoding process
      fc_decay: 0.00002
  Student: # Another sub-network, and here is a distillation example of DML. The_
  ↪two sub-networks share the same structure, and both need to learn parameters
    pretrained: # The following networking parameters are the same as above
    freeze_params: false
    return_all_feats: true
    model_type: *model_type
    algorithm: CRNN
  Transform:
  Backbone:
    name: MobileNetV1Enhance
    scale: 0.5
  Neck:
    name: SequenceEncoder
    encoder_type: rnn
    hidden_size: 64
  Head:
    name: CTCHead
    mid_channels: 96
    fc_decay: 0.00002

```

Of course, if you want to add more sub-networks for training, you can also add the corresponding fields in the configuration file according to the way of adding Student and Teacher. For example, if you want three models to supervise each other and train together, then Architecture can be written in the following way.

```

Architecture:
  model_type: &model_type "rec"
  name: DistillationModel

```

(continues on next page)

(continued from previous page)

```

algorithm: Distillation
Models:
  Teacher:
    pretrained:
    freeze_params: false
    return_all_feats: true
    model_type: *model_type
    algorithm: CRNN
  Transform:
    Backbone:
      name: MobileNetV1Enhance
      scale: 0.5
    Neck:
      name: SequenceEncoder
      encoder_type: rnn
      hidden_size: 64
    Head:
      name: CTCHead
      mid_channels: 96
      fc_decay: 0.00002
  Student:
    pretrained:
    freeze_params: false
    return_all_feats: true
    model_type: *model_type
    algorithm: CRNN
  Transform:
    Backbone:
      name: MobileNetV1Enhance
      scale: 0.5
    Neck:
      name: SequenceEncoder
      encoder_type: rnn
      hidden_size: 64
    Head:
      name: CTCHead
      mid_channels: 96
      fc_decay: 0.00002
  Student2: # Introduces new sub-networks in the knowledge_
  ↪distillation task, and the other parts remain the same as the above configuration
    pretrained:
    freeze_params: false
    return_all_feats: true
    model_type: *model_type
    algorithm: CRNN
  Transform:
    Backbone:
      name: MobileNetV1Enhance
      scale: 0.5
    Neck:
      name: SequenceEncoder
      encoder_type: rnn
      hidden_size: 64
    Head:
      name: CTCHead
      mid_channels: 96
      fc_decay: 0.00002

```

When the model is trained, it contains 3 sub-networks: Teacher, Student, and Student2.

For code example of the distillation model DistillationModel class, refer to [distillation_model.py](#).

The final output of the model forward is a dictionary, and the key is the names of all the sub-networks. For example, here are Student and Teacher, and the value is the output of the corresponding sub-network, which can be Tensor (only returns the last network layer) and dict (also returns the middle feature information).

In the recognition task, in order to add more loss functions and ensure the scalability of the distillation method, the output of each sub-network is saved as a dict, which contains the sub-module output. Take this recognition model as an example, the output result of each subnet is dict, the key includes backbone_out, neck_out, head_out, and value is the tensor of the corresponding module. Finally, based the above configuration file, DistillationModel is output as:

```
{  
    "Teacher": {  
        "backbone_out": tensor,  
        "neck_out": tensor,  
        "head_out": tensor,  
    },  
    "Student": {  
        "backbone_out": tensor,  
        "neck_out": tensor,  
        "head_out": tensor,  
    }  
}
```

In the knowledge distillation task, the loss function configuration is as follows.

```
Loss:  
    name: CombinedLoss # Loss function name, based on the name change, build a class  
    ↪for loss function  
    loss_config_list: # Loss function configuration file list, a necessary function for  
    ↪CombinedLoss  
    -DistillationCTCLoss: # CTC loss function based on distillation, inherited from  
    ↪standard CTC loss  
        weight: 1.0 # The weight of the loss function. In loss_config_list, the  
        ↪configuration of each loss function must include this field  
        model_name_list: ["Student", "Teacher"] # For the prediction results of the  
        ↪distillation model, extract the output of these two sub-networks and calculate the  
        ↪CTC loss with gt  
        key: head_out # Take the tensor corresponding to the key in the subnet output  
    ↪dict  
    -DistillationDMLLoss: # Distilled DML loss function, inherited from standard DMLLoss  
        weight: 1.0 # weight  
        act: "softmax" # Activation function, use activation function to process the  
        ↪input, it can be softmax, sigmoid or None, the default is None  
        model_name_pairs: # The subnet name pairs used to calculate DML loss. If you  
        ↪want to calculate the DML loss of other subnets, you can continue to fill in the  
        ↪list below  
        - ["Student", "Teacher"]  
        key: head_out # Take the tensor corresponding to the key in the subnet output  
    ↪dict  
    -DistillationDistanceLoss: # Distillation distance loss function  
        weight: 1.0 # weight  
        mode: "l2" # Distance calculation method, currently supports l1, l2, smooth_l1  
        model_name_pairs: # Subnet name pairs used to calculate distance loss  
        - ["Student", "Teacher"]
```

(continues on next page)

(continued from previous page)

```
key: backbone_out # Take the tensor corresponding to the key in the subnet_
→output dict
```

Among the above loss functions, all distillation loss functions are inherited from standard loss functions. They analyzes the output of the distillation model, find the tensor used for loss calculation, and then use standard loss functions class for calculation.

With the above configuration, the loss function in the distillation training contains three parts.

- The final output (head_out) of Student and Teacher and the CTC loss of gt, with a weight of 1. Here, because both sub-networks need to update parameters,they need to calculate the loss with gt.
- The DML loss between Student and Teacher's output (head_out), with a weight of 1.
- The l2 loss between Student and Teacher's backbone network output (backbone_out), with a weight of 1.

The CombinedLoss is implemented as follows.

```
class CombinedLoss(nn.Layer):
    """
    CombinedLoss:
        a combination of loss function
    """

    def __init__(self, loss_config_list=None):
        super().__init__()
        self.loss_func = []
        self.loss_weight = []
        assert isinstance(loss_config_list, list), (
            'operator config should be a list')
        for config in loss_config_list:
            assert isinstance(config,
                dict) and len(config) == 1, "yaml format error"
            name = list(config)[0]
            param = config[name]
            assert "weight" in param, "weight must be in param, but param just_"
            ↪contains {}".format(
                param.keys())
            self.loss_weight.append(param.pop("weight"))
            self.loss_func.append(eval(name)(**param))

    def forward(self, input, batch, **kargs):
        loss_dict = {}
        loss_all = 0.
        for idx, loss_func in enumerate(self.loss_func):
            loss = loss_func(input, batch, **kargs)
            if isinstance(loss, paddle.Tensor):
                loss = {"loss_{}": loss}.format(str(idx))
            weight = self.loss_weight[idx]
            loss = {key: loss[key] * weight for key in loss}
            if "loss" in loss:
                loss_all += loss["loss"]
            else:
                loss_all += paddle.add_n(list(loss.values()))
        loss_dict.update(loss)
        return loss_all, loss_dict
```

(continues on next page)

(continued from previous page)

```
loss_dict["loss"] = loss_all
return loss_dict
```

For implementation of CombinedLoss, please refer to: [combined_loss.py](#). For implementation of distillation loss functions such as DistillationCTCLoss, please refer to [distillation_loss.py](#).

In the distillation of the three models, the loss field also needs to be modified, and the loss between the three sub-networks also need to be considered.

```
Loss:
  name: CombinedLoss # Loss function name, build a class for the loss function based_
  ↪on the name change
  loss_config_list: # Loss function configuration file list, a necessary function for_
  ↪CombinedLoss.
  -DistillationCTCLoss: # CTC loss function based on distillation, inherited from_
  ↪standard CTC loss
    weight: 1.0 # The weight of the loss function. In loss_config_list, the_
    ↪configuration of each loss function must include this field
    model_name_list: ["Student", "Student2", "Teacher"] # For the prediction_
    ↪results of the distillation model, extract the output of these three sub-networks_
    ↪and calculate the CTC loss with gt
    key: head_out # Take the tensor corresponding to the key in the subnet output_
    ↪dict
  -DistillationDMLLoss: # Distilled DML loss function, inherited from standard DMLLoss
    weight: 1.0 # weight
    act: "softmax" # Activation function, it can be softmax, sigmoid or None, and_
    ↪the default is None
    model_name_pairs: # The subnet name pairs used to calculate DML loss. If you_
    ↪want to calculate the DML loss of other subnets, you can continue to fill in the_
    ↪list below
      -["Student", "Teacher"]
      -["Student2", "Teacher"]
      -["Student", "Student2"]
    key: head_out # Take the tensor corresponding to the key in the subnet output_
    ↪dict
  -DistillationDistanceLoss: # Distillation distance loss function
    weight: 1.0 # weight
    mode: "l2" # Distance calculation method, currently supports l1, l2, smooth_l1
    model_name_pairs: # Subnet name pairs used to calculate distance loss
      -["Student", "Teacher"]
      -["Student2", "Teacher"]
      -["Student", "Student2"]
    key: backbone_out # Take the tensor corresponding to the key in the subnet_
    ↪output dict
```

```
# Download data
!wget -nc https://paddleocr.bj.bcebos.com/dataset/rec_data_lesson_demo.tar && tar -xf_
  ↪rec_data_lesson_demo.tar && rm rec_data_lesson_demo.tar
# # Download the pre-trained model
!wget -nc https://paddleocr.bj.bcebos.com/PP-OCRV2/chinese/ch_PP-OCRV2_rec_train.tar &
  ↪& tar -xf ch_PP-OCRV2_rec_train.tar && rm ch_PP-OCRV2_rec_train.tar

!python tools/train.py -c configs/rec/ch_PP-OCRV2/ch_PP-OCRV2_rec_distillation.yml \
  -o Train.dataset.data_dir=". ./rec_data_lesson_demo/" \
  Train.dataset.label_file_list=[". ./rec_data_lesson_demo/train.txt"] \
```

(continues on next page)

(continued from previous page)

```

Train.loader.num_workers=0 \
Train.loader.batch_size_per_card=64 \
Eval.dataset.data_dir="./rec_data_lesson_demo/" \
Eval.dataset.label_file_list=[ "./rec_data_lesson_demo/val.txt"] \
Eval.loader.num_workers=0 \
Optimizer.lr.values=[0.0001,0.00001] \
Global.epoch_num=1 \
Global.pretrained_model="./ch_PP-OCRv2_rec_train/best_accuracy"

```

Enhanced CTC Loss

In the task of recognizing Chinese characters with OCR, there are too many similar characters, which are easy to be misrecognized. Learning from Metric Learning, OCR has introduced Center loss to enlarge the distance between classes. The core formula is shown below.

$$L = L_{ctc} + \lambda * L_{center}$$

$$L_{center} = \sum_{t=1}^T \|x_t - c_{y_t}\|_2^2$$

Here x_t represents the label at the time step t , and c_{y_t} represents the center corresponding to the label y_t .

In Enhanced CTC, the center initialization also has great influence on the result. In PP-OCRv2, the specific steps of center initialization are as follows.

1. Train a network based on the standard CTC loss;
2. Extract the correct image set from the training set and mark it as G;
3. Input the pictures in G into the network one by one, and extract the corresponding relationship between x_t and y_t of sequence features output by the head. The calculation of y_t is as follows:

$$y_t = argmax(W * x_t)$$

1. Aggregate x_t with y_t , and average them as the initial center.

First, train a basic network with [configs/rec/ch_PP-OCRv2/ch_PP-OCRv2_rec.yml](#)

For more training steps about center loss, please refer to: [Enhanced CTC Loss Usage Document](#)

Finally, use [configs/rec/ch_PP-OCRv2/ch_PP-OCRv2_rec_enhanced_ctc_loss.yml](#) for training, and the command is as follows.

```
python tools/train.py -c configs/rec/ch_PP-OCRv2/ch_PP-OCRv2_rec_enhanced_ctc_loss.yml
```

The Loss field is the focus. Compared with the standard CTCLoss, CenterLoss is added here. Then configure the number of categories, feature dimensions, and the center path.

```

Loss:
  name: CombinedLoss
  loss_config_list:
    - CTCLoss:
      use_focal_loss: false
      weight: 1.0
    - CenterLoss:
      weight: 0.05

```

(continues on next page)

(continued from previous page)

```

num_classes: 6625
feat_dim: 96
center_file_path: "./train_center.pkl"

```

Summary of Text Recognition Optimization

In the process of optimizing the PP-OCRv2 text recognition model, the model was improved in the backbone network, loss function and so on. And the knowledge distillation training method has been introduced, increasing the recognition accuracy from **66.7%** to **74.8%**. The ablation experiment is as below.

Strategy	Acc	Model Size (M)	Inference Time (CPU, ms)
PP-OCR rec (MV3)	0.667	5.0	7.7
PP-OCR rec (LCNet)	0.693	8.0	6.2
PP-OCR rec (LCNet) + U-DML	0.739	8.6	6.2
PP-OCR rec (LCNet) + U-DML + Enhanced CTC loss	0.748	8.6	6.2

Table 3: Ablation study of LCNet, U-DML, and Enhanced CTC loss for text recognition.

Figure 39: Ablation experiment of PP-OCRv2 recognition model

Based on the PP-OCRv2 text detection, the experiment result of the recognition model is as follows.



Figure 40: The text detection result of PP-OCRv2

6.4 Summary

This chapter mainly talks about the optimization strategies of PP-OCR and PP-OCRV2.

PP-OCR uses 19 strategies in the backbone network, learning rate strategy, data augmentation, and model tailoring and quantization, to optimize the model, creating a PP-OCR server system and a PP-OCR mobile system.

Compared with PP-OCR, PP-OCRV2 make more improvements in the three aspects— backbone network, data augmentation, and loss function— to tackle the poor end-to-side inference efficiency, complex background, and misrecognition of similar characters. At the same time, it introduces the knowledge distillation training strategy to further improve the accuracy of the model. Finally, PP-OCRV2 builds a text detection and recognition system outperforming PP-OCR on accuracy and speed.

6.5 Assignment

For details, refer to Optimization Strategy Objective Questions and Optimization Strategy Practical Questions in the compulsory assignment column.

INFERENCE AND DEPLOYMENT OF PP-OCRV2

7.1 Overview of Inference and Deployment

This chapter mainly introduces the high-performance inference, service deployment and mobile-device deployment of the PP-OCRV2 system. Through the study of this chapter, you can learn:

- How to choose appropriate inference deployment methods in different scenarios
- Inference methods of PP-OCRV2 models in various scenarios
- Inference deployment methods of Paddle Inference, Paddle Serving, and Paddle Lite

7.1.1 Introduction

In the previous chapters, a model has been trained by using methods of model training. When using it to predict, we need to define the model first, then load the trained model, and send preprocessed data to the network for inference and post-processing, and get the final result. This inference method is convenient for debugging, but low in efficiency.

For this, there are two offline inference solutions.

1. Inference based on the training engine uses the same engine as the training. It is convenient for debugging and can help us quickly locate problems and make verification. Its programming language is mostly Python.
2. Inference based on the inference engine transforms the trained model and removes the parts irrelevant to the inference. This method can speed up the inference. And its programming language is Python language or C++.

The differences between the two are shown below.

	Inference based on the training engine	Inference based on the inference engine
Features	1. Use the same engine as the training. 2. The network model needs to be defined in inference. 3. It is applicable to system integration	1. The model needs to be converted, with irrelevant parts removed. 2. * <i>There is no need to define a network model*</i> . 3. It is suitable for system integration
Programming language	Python	Python or C++
Inference steps	1. Define the network structure on the Python side. 2. Prepare the input data. 3. Load the training model. 4. Perform inference	1. Prepare the input data. 2. Load the model structure and model Parameters. 3. Perform inference

In the offline inference deployment, it is more appropriate to perform inference based on the inference engine.

PaddlePaddle provides the following inference deployment solutions for different application scenarios.

Deployment	Feature	Scene	Hardware
Paddle Inference	General, High performance	<ul style="list-style-type: none"> Complex model algorithm Hardware with high performance 	<ul style="list-style-type: none"> X86 CPU NVIDIA GPU Loongson/Feiteng and other domestic CPUs AI acceleration chips such as Kunlun/Ascend/Haiguang DCU
Paddle Lite	Lite-weight	<ul style="list-style-type: none"> Lite-weight model Miscellaneous hardware, limited resources, low power consumption 	<ul style="list-style-type: none"> Arm CPU Arm / Qualcomm / Apple GPU AI acceleration hardware such as Kunlun / Ascend / Kirin / Rockchip / Yingmai / Cambrian / Bitmain
Paddle Serving	High concurrency	<ul style="list-style-type: none"> Large traffic, high concurrency, low latency, large throughput Resource elasticity regulation to cope with changes in service traffic Model combination, encryption, hot update, etc. 	<ul style="list-style-type: none"> X86 / Arm CPU NVIDIA GPU Kunlun / Shengteng, etc.
Paddle.js	Inference on the browser	<ul style="list-style-type: none"> Chrome, Safari, Firefox, etc 	
Paddle2ONNX	Open and compatible	<ul style="list-style-type: none"> Third-party inference framework Support deployment in more AI chips 	<ul style="list-style-type: none"> Horizon X3 Corerain X3 Allwinner R329 Other AI chips

Figure 1: Different deployment options offered by PaddlePaddle

PaddleOCR provides three inference deployment solutions for different application scenarios.

- Offline Inference (Paddle Inference). It is mainly used in scenarios where the timeliness of the inference response is not high, especially in those requiring a large number of picture for inference, such as document digitization, and advertising information extraction. Although the inference request cannot be responded in real time, there is no network latency, making the calculation much more efficient and ensuring the data security.
- Service deployment (Paddle Serving). It is mainly used in scenarios requiring high timeliness of inference response, such as the APIs of commercial OCR, real-time photo translation, and snapping for answers. Although this method can make real-time response to inference, there will be high internet expenses, inefficient use of GPU, and data security risks.
- Mobile-device deployment (Paddle Lite). It aims to deploy the model to mobile devices such as mobile phones and robots out of convenience and data security. It is similar to ID card and bank card recognition in mobile APP, and instrument monitoring and recognition in industrial application scenarios. This method is more sensitive to the size of the OCR model. Although there is no network latency and low data security risks, its inference prediction efficiency is not high due to the limitation of computing power.

Based on PP-OCRV2, this chapter will talk about the process of text detection, recognition, and the end-to-end inference deployment process.

7.1.2 Environment Preparation

To participate in this chapter, you need to download the PaddleOCR code first and install related dependencies. The commands are as follows.

```
import os
os.chdir("/home/aistudio")
# Download code
!git clone https://gitee.com/paddlepaddle/PaddleOCR.git
```

(continues on next page)

(continued from previous page)

```
os.chdir("/home/aistudio/PaddleOCR")
# Install and run the required whl package
!pip install -U pip
!pip install -r requirements.txt
# This library is needed in VQA tasks
!pip install paddlenlp==2.2.1

# Import some libraries
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import os
```

7.2 Python Inference Based on Paddle Inference

7.2.1 Introduction

In a project, the inference performance of the model directly affects the cost, so it is expected that there is a trained model having a faster inference speed. If inference is directly performed on the training engine, the **efficiency is relatively low** for the model contains training-related operators. And it is needed to define the model, making it **difficult to decouple from the training code**. In this case, Paddle Inference came into being. It is a native inference library of PaddlePaddle, which acts on the server and the cloud to provide high-performance inference. Since its is based on the training operator of PaddlePaddle, Paddle Inference supports all models trained by PaddlePaddle.

Considering that every user's application scenarios vary, Paddle Inference has carried out in-depth adaptation for different platforms and different scenarios. It is high in throughput and low in latency, ensuring that trained PaddlePaddle models are ready-to-use on the server, and can be deployed fast.

This chapter will talk about introduces the inference of **PP-OCRv2** with Paddle Inference. To learn more about Paddle Inference, please refer to: [Paddle Inference Introduction](#).

In the model inference with Paddle Inference, there are several steps.

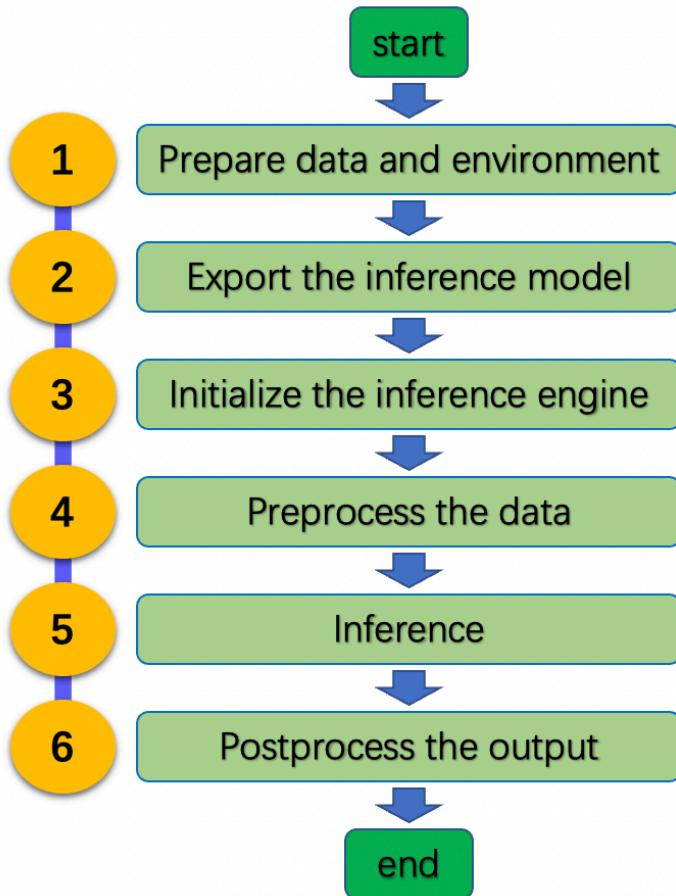


Figure 2: Inference process of Paddle Inference

The PP-OCRv2 system includes three models: text detection, direction classifier, and text recognition. The following will introduce the inference process of these three models based on Paddle Inference.

7.2.2 PP-OCRv2 Text Detection Model Inference

In PaddleOCR, when inference is based on the text detection model, it is needed to designate an image or the path of an image collection through the parameter `image_dir`, and the parameter `det_model_dir` specifies the path of the detected inference model.

In the following, we will perform the inference of the latest ultra-lightweight text detection model. For more models and using methods, please refer to [Text Detection Prediction Tutorial](#).

To learn more about the hyperparameters of other algorithms, please refer to [PaddleOCR Inference related parameters introduction](#).

Data and Environment Preparation

At the very beginning, Paddle and the corresponding dependencies are installed, and the environment is ready.

The test data is in the `doc/imgs` folder, and parts of the data are shown below.

```
# Switch the directory
os.chdir("/home/aistudio/PaddleOCR")

# View data
!ls doc/imgs/

# Choose 2 images for visualization
img1 = cv2.imread("doc/imgs/00006737.jpg")
img2 = cv2.imread("doc/imgs/00056221.jpg")
plt.figure(figsize=(15, 6))
plt.subplot(1, 2, 1)
plt.imshow(img1[:, :, ::-1])
plt.subplot(1, 2, 2)
plt.imshow(img2[:, :, ::-1])
plt.show()
```

00006737.jpg	00056221.jpg	00111002.jpg	1.jpg	japan_1.jpg
00009282.jpg	00057937.jpg	00207393.jpg	french_0.jpg	japan_2.jpg
00015504.jpg	00059985.jpg	11.jpg	ger_1.jpg	korean_1.jpg
00018069.jpg	00077949.jpg	12.jpg	ger_2.jpg	model_prod_flow_ch.png



Inference Model Preparation

Download the inference model, unzip it, and place it in the `inference`.

```
# Download model
!mkdir inference
!cd inference && wget https://paddleocr.bj.bcebos.com/PP-OCRv2/chinese/ch_PP-OCRv2_
->det_infer.tar -O ch_PP-OCRv2_det_infer.tar && tar -xf ch_PP-OCRv2_det_infer.tar
!tree -h inference/ch_PP-OCRv2_det_infer
```

```
--2021-12-25 14:55:13-- https://paddleocr.bj.bcebos.com/PP-OCRv2/chinese/ch_PP-  
↳OCRv2_det_infer.tar  
Resolving paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com) ... 100.67.200.6  
Connecting to paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com)|100.67.200.6|:443..  
↳. connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 3190272 (3.0M) [application/x-tar]  
Saving to: 'ch_PP-OCRv2_det_infer.tar'  
  
ch_PP-OCRv2_det_inf 100%[=====] 3.04M --.-KB/s in 0.07s  
  
2021-12-25 14:55:13 (42.2 MB/s) - 'ch_PP-OCRv2_det_infer.tar' saved [3190272/  
↳3190272]  
  
inference/ch_PP-OCRv2_det_infer  
├── [2.2M] inference.pdiparams  
├── [ 23K] inference.pdiparams.info  
└── [845K] inference.pdmodel  
  
0 directories, 3 files
```

- If you want to export the model you have trained and deploy it with Paddle Inference, you can use the following commands to convert the pre-trained model into an inference model through transforming dynamic diagrams into static diagrams.

```
# Reference Code  
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/tools/export_model.py  
# Download the pre-trained model  
!wget https://paddleocr.bj.bcebos.com/PP-OCRv2/chinese/ch_PP-OCRv2_det_distill_train.  
↳tar && tar -xf ch_PP-OCRv2_det_distill_train.tar && rm ch_PP-OCRv2_det_distill_  
↳train.tar  
# Export inference model  
!python tools/export_model.py -c configs/det/ch_PP-OCRv2/ch_PP-OCRv2_det_cml.yml \  
    -o Global.pretrained_model="ch_PP-OCRv2_det_distill_train/best_accuracy" \  
    Global.save_inference_dir=".my_model"  
# The PP-OCRv2 detection model contains three sub-networks: teacher, student, and  
↳student 2. Therefore, there are three sub-files in export, but only one student  
↳network is needed in inference.  
!tree -h my_model
```

```
--2021-12-25 14:55:21-- https://paddleocr.bj.bcebos.com/PP-OCRv2/chinese/ch_PP-  
↳OCRv2_det_distill_train.tar  
Resolving paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com) ... 100.67.200.6  
Connecting to paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com)|100.67.200.6|:443..  
↳. connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 63830016 (61M) [application/x-tar]  
Saving to: 'ch_PP-OCRv2_det_distill_train.tar'  
  
ch_PP-OCRv2_det_dis 100%[=====] 60.87M 81.4MB/s in 0.7s  
  
2021-12-25 14:55:22 (81.4 MB/s) - 'ch_PP-OCRv2_det_distill_train.tar' saved  
↳[63830016/63830016]  
  
W1225 14:55:24.746377 1078 device_context.cc:447] Please NOTE: device: 0, GPU  
↳Compute Capability: 7.0, Driver API Version: 10.1, Runtime API Version: 10.1
```

(continues on next page)

(continued from previous page)

```

W1225 14:55:24.749907 1078 device_context.cc:465] device: 0, cuDNN Version: 7.6.
[2021/12/25 14:55:30] root INFO: load pretrain successful from ch_PP-OCRv2_det_
↳ distill_train/best_accuracy
[2021/12/25 14:55:31] root INFO: inference model is saved to ./my_model/Teacher/
↳ inference
[2021/12/25 14:55:33] root INFO: inference model is saved to ./my_model/Student/
↳ inference
[2021/12/25 14:55:34] root INFO: inference model is saved to ./my_model/Student2/
↳ inference
my_model
└── [4.0K] Student
    ├── [2.2M] inference.pdiparams
    ├── [ 23K] inference.pdiparams.info
    └── [961K] inference.pdmodel
└── [4.0K] Student2
    ├── [2.2M] inference.pdiparams
    ├── [ 23K] inference.pdiparams.info
    └── [962K] inference.pdmodel
└── [4.0K] Teacher
    ├── [ 47M] inference.pdiparams
    ├── [ 12K] inference.pdiparams.info
    └── [568K] inference.pdmodel

3 directories, 9 files

```

A Preliminary Study of Text Detection Functions

Let's first take a look at the results after loading the inference model.

```

# Reference Code
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/tools/infer/predict_det.
↳ py
# Inference
!python tools/infer/predict_det.py --image_dir=".doc/imgs/00018069.jpg" --det_model_
↳ dir=".inference/ch_PP-OCRv2_det_infer" --use_gpu=False

# Read the image and display it, and show the result
plt.figure(figsize=(20, 8))
img_ori = cv2.imread("./doc/imgs/00018069.jpg")
img_out = cv2.imread("./inference_results/det_res_00018069.jpg")
plt.subplot(1, 2, 1)
plt.imshow(img_ori[:, :, ::-1])
plt.subplot(1, 2, 2)
plt.imshow(img_out[:, :, ::-1])
plt.show()

```

The following information will be printed.

```
[2021/12/25 20:48:47] root INFO: 00018069.jpg      [[[378, 249], ...]
```

The following figure will be displayed.

代号	项目	结果	参考值	单位
ALT	谷丙转氨酶	25.6	0--40	U/L
TBIL	总胆红素	11.2	<20	umol/L
DBIL	直接胆红素	3.3	0--7	umol/L
IBIL	间接胆红素	7.9	1.5--15	umol/L
TP	总蛋白	58.9 ↓	60--80	g/L
ALB	白蛋白	35.1	33--55	g/L
GLO	球蛋白	23.8	20--30	g/L
A/G	白球比	1.5	1.5--2.5	
ALP	碱性磷酸酶	93	15--112	IU/L
GGT	谷氨酰转肽酶	14.3	<50	U/L
AST	谷草转氨酶	16.3	8--40	U/L
LDH	乳酸脱氢酶	167	114--240	U/L
ADA	腺苷脱氨酶	12.6	4--24	U/L

代号	项目	结果	参考值	单位
ALT	谷丙转氨酶	25.6	0--40	U/L
TBIL	总胆红素	11.2	<20	umol/L
DBIL	直接胆红素	3.3	0--7	umol/L
IBIL	间接胆红素	7.9	1.5--15	umol/L
TP	总蛋白	58.9 ↓	60--80	g/L
ALB	白蛋白	35.1	33--55	g/L
GLO	球蛋白	23.8	20--30	g/L
A/G	白球比	1.5	1.5--2.5	
ALP	碱性磷酸酶	93	15--112	IU/L
GGT	谷氨酰转肽酶	14.3	<50	U/L
AST	谷草转氨酶	16.3	8--40	U/L
LDH	乳酸脱氢酶	167	114--240	U/L
ADA	腺苷脱氨酶	12.6	4--24	U/L

So how does it work? The following is a detailed explanation of the loading of the inference model and inference code of PP-OCRv2.

First, you need to set the parameters as shown below. To learn more the parameters, please refer to: Introduction about parameters in the PaddleOCR inference process.

```
# Reference Code
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/tools/infer/utility.py
import argparse
import os
import sys
import cv2
import numpy as np
import paddle
from PIL import Image, ImageDraw, ImageFont
import math
from paddle import inference
import time
from ppocr.utils.logging import get_logger

def str2bool(v):
    return v.lower() in ("true", "t", "1")

def init_args():
    parser = argparse.ArgumentParser()
    # params for prediction engine
    parser.add_argument("--use_gpu", type=str2bool, default=True)
    parser.add_argument("--ir_optim", type=str2bool, default=True)
    parser.add_argument("--use_tensorrt", type=str2bool, default=False)
    parser.add_argument("--min_subgraph_size", type=int, default=15)
    parser.add_argument("--precision", type=str, default="fp32")
    parser.add_argument("--gpu_mem", type=int, default=500)

    # params for text detector
    parser.add_argument("--image_dir", type=str)
    parser.add_argument("--det_algorithm", type=str, default='DB')
    parser.add_argument("--det_model_dir", type=str)
    parser.add_argument("--det_limit_side_len", type=float, default=960)
    parser.add_argument("--det_limit_type", type=str, default='max')

    # DB parmas
    parser.add_argument("--det_db_thresh", type=float, default=0.3)
```

(continues on next page)

(continued from previous page)

```

parser.add_argument("--det_db_box_thresh", type=float, default=0.6)
parser.add_argument("--det_db_unclip_ratio", type=float, default=1.5)
parser.add_argument("--max_batch_size", type=int, default=10)
parser.add_argument("--use_dilation", type=str2bool, default=False)
parser.add_argument("--det_db_score_mode", type=str, default="fast")
# EAST parmas
parser.add_argument("--det_east_score_thresh", type=float, default=0.8)
parser.add_argument("--det_east_cover_thresh", type=float, default=0.1)
parser.add_argument("--det_east_nms_thresh", type=float, default=0.2)

# SAST parmas
parser.add_argument("--det_sast_score_thresh", type=float, default=0.5)
parser.add_argument("--det_sast_nms_thresh", type=float, default=0.2)
parser.add_argument("--det_sast_polygon", type=str2bool, default=False)

# PSE parmas
parser.add_argument("--det_pse_thresh", type=float, default=0)
parser.add_argument("--det_pse_box_thresh", type=float, default=0.85)
parser.add_argument("--det_pse_min_area", type=float, default=16)
parser.add_argument("--det_pse_box_type", type=str, default='box')
parser.add_argument("--det_pse_scale", type=int, default=1)

# params for text recognizer
parser.add_argument("--rec_algorithm", type=str, default='CRNN')
parser.add_argument("--rec_model_dir", type=str)
parser.add_argument("--rec_image_shape", type=str, default="3, 32, 320")
parser.add_argument("--rec_batch_num", type=int, default=6)
parser.add_argument("--max_text_length", type=int, default=25)
parser.add_argument(
    "--rec_char_dict_path",
    type=str,
    default="./ppocr/utils/ppocr_keys_v1.txt")
parser.add_argument("--use_space_char", type=str2bool, default=True)
parser.add_argument(
    "--vis_font_path", type=str, default="./doc/fonts/simfang.ttf")
parser.add_argument("--drop_score", type=float, default=0.5)

# params for e2e
parser.add_argument("--e2e_algorithm", type=str, default='PGNet')
parser.add_argument("--e2e_model_dir", type=str)
parser.add_argument("--e2e_limit_side_len", type=float, default=768)
parser.add_argument("--e2e_limit_type", type=str, default='max')

# PGNet parmas
parser.add_argument("--e2e_pgnet_score_thresh", type=float, default=0.5)
parser.add_argument(
    "--e2e_char_dict_path", type=str, default="./ppocr/utils/ic15_dict.txt")
parser.add_argument("--e2e_pgnet_valid_set", type=str, default='totaltext')
parser.add_argument("--e2e_pgnet_mode", type=str, default='fast')

# params for text classifier
parser.add_argument("--use_angle_cls", type=str2bool, default=False)
parser.add_argument("--cls_model_dir", type=str)
parser.add_argument("--cls_image_shape", type=str, default="3, 48, 192")
parser.add_argument("--label_list", type=list, default=['0', '180'])
parser.add_argument("--cls_batch_num", type=int, default=6)

```

(continues on next page)

(continued from previous page)

```

parser.add_argument("--cls_thresh", type=float, default=0.9)

parser.add_argument("--enable_mkldnn", type=str2bool, default=False)
parser.add_argument("--cpu_threads", type=int, default=10)
parser.add_argument("--use_pderving", type=str2bool, default=False)
parser.add_argument("--warmup", type=str2bool, default=False)

#
parser.add_argument(
    "--draw_img_save_dir", type=str, default="./inference_results")
parser.add_argument("--save_crop_res", type=str2bool, default=False)
parser.add_argument("--crop_res_save_dir", type=str, default="./output")

# multi-process
parser.add_argument("--use_mp", type=str2bool, default=False)
parser.add_argument("--total_process_num", type=int, default=1)
parser.add_argument("--process_id", type=int, default=0)

parser.add_argument("--benchmark", type=str2bool, default=False)
parser.add_argument("--save_log_path", type=str, default="./log_output/")

parser.add_argument("--show_log", type=str2bool, default=True)
parser.add_argument("--use_onnx", type=str2bool, default=False)
# It should be noted here that this is added because if you parse directly in the
# notebook, sys.argv will add the following content at the back, causing the parsing
# to fail
# '-f', '/home/aistudio/.local/share/jupyter/runtime/kernel-e1221262-c656-4129-
# 896f-1b197b6b782c.json'
parser.add_argument("-f", type=str, default=None)
return parser

def parse_args():
    parser = init_args()
    return parser.parse_args()

```

Let's look at the code for text detection.

```

# Reference Code
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/tools/infer/predict_det.
→py
import os
import sys
import cv2
import numpy as np
import time

import tools.infer.utility as utility
from ppocr.utils.logging import get_logger
from ppocr.utils.utility import get_image_file_list, check_and_read_gif
from ppocr.data import create_operators, transform
from ppocr.postprocess import build_post_process
import json
logger = get_logger()

# Text detection

```

(continues on next page)

(continued from previous page)

```

class TextDetector(object):
    def __init__(self, args):
        self.args = args
        self.det_algorithm = args.det_algorithm
        pre_process_list = [
            {
                'DetResizeForTest': {
                    'limit_side_len': args.det_limit_side_len,
                    'limit_type': args.det_limit_type,
                }
            },
            {
                'NormalizeImage': {
                    'std': [0.229, 0.224, 0.225],
                    'mean': [0.485, 0.456, 0.406],
                    'scale': '1./255.',
                    'order': 'hwc'
                }
            },
            {
                'ToCHWImage': None
            },
            {
                'KeepKeys': {
                    'keep_keys': ['image', 'shape']
                }
            }
        ]
        postprocess_params = {}
        if self.det_algorithm == "DB":
            postprocess_params['name'] = 'DBPostProcess'
            postprocess_params["thresh"] = args.det_db_thresh
            postprocess_params["box_thresh"] = args.det_db_box_thresh
            postprocess_params["max_candidates"] = 1000
            postprocess_params["unclip_ratio"] = args.det_db_unclip_ratio
            postprocess_params["use_dilation"] = args.use_dilation
            postprocess_params["score_mode"] = args.det_db_score_mode
        else:
            logger.info("unknown det_algorithm:{}".format(self.det_algorithm))
            sys.exit(0)
        # Initialize the inference engine
        self.predictor, self.input_tensor, self.output_tensors, self.config = utility.
        create_predictor(args, 'det', logger)
        # Build the preprocessing operator
        self.preprocess_op = create_operators(pre_process_list)
        # Build the postprocessing operator
        self.postprocess_op = build_post_process(postprocess_params)

    def order_points_clockwise(self, pts):
        """
        Refer to: https://github.com/jrosebr1/imutils/blob/master/imutils/perspective.py
        Sort out the detected points clockwise
        """
        xSorted = pts[np.argsort(pts[:, 0]), :]
        leftMost = xSorted[:2, :]
        rightMost = xSorted[2:, :]

```

(continues on next page)

(continued from previous page)

```

leftMost = leftMost[np.argsort(leftMost[:, 1]), :]
(tl, bl) = leftMost

rightMost = rightMost[np.argsort(rightMost[:, 1]), :]
(tr, br) = rightMost

rect = np.array([tl, tr, br, bl], dtype="float32")
return rect

def clip_det_res(self, points, img_height, img_width):
    # Limit the detection results according to the width and height to prevent them from exceeding the image boundaries
    for pno in range(points.shape[0]):
        points[pno, 0] = int(min(max(points[pno, 0], 0), img_width - 1))
        points[pno, 1] = int(min(max(points[pno, 1], 0), img_height - 1))
    return points

def filter_tag_det_res(self, dt_boxes, image_shape):
    # Remove test results smaller than a certain size
    img_height, img_width = image_shape[0:2]
    dt_boxes_new = []
    for box in dt_boxes:
        box = self.order_points_clockwise(box)
        box = self.clip_det_res(box, img_height, img_width)
        rect_width = int(np.linalg.norm(box[0] - box[1]))
        rect_height = int(np.linalg.norm(box[0] - box[3]))
        if rect_width <= 3 or rect_height <= 3:
            continue
        dt_boxes_new.append(box)
    dt_boxes = np.array(dt_boxes_new)
    return dt_boxes

def filter_tag_det_res_only_clip(self, dt_boxes, image_shape):
    # Limit the boundary of the detection result
    img_height, img_width = image_shape[0:2]
    dt_boxes_new = []
    for box in dt_boxes:
        box = self.clip_det_res(box, img_height, img_width)
        dt_boxes_new.append(box)
    dt_boxes = np.array(dt_boxes_new)
    return dt_boxes

def __call__(self, img):
    ori_im = img.copy()
    data = {'image': img}

    st = time.time()

    # Data preprocessing
    data = transform(data, self.preprocess_op)
    img, shape_list = data
    if img is None:
        return None, 0
    # Extended bs dimension: CHW -> NCHW
    img = np.expand_dims(img, axis=0)
    shape_list = np.expand_dims(shape_list, axis=0)

```

(continues on next page)

(continued from previous page)

```

    img = img.copy()
    # Copy the data to the inference engine
    self.input_tensor.copy_from_cpu(img)
    # Automatic inference
    self.predictor.run()
    outputs = []
    # Copy the returned result from the inference engine to the CPU
    for output_tensor in self.output_tensors:
        output = output_tensor.copy_to_cpu()
        outputs.append(output)

    preds = {}
    if self.det_algorithm in ['DB', 'PSE']:
        preds['maps'] = outputs[0]
    else:
        raise NotImplementedError

    # Post-processing
    post_result = self.postprocess_op(preds, shape_list)
    dt_boxes = post_result[0]['points']
    dt_boxes = self.filter_tag_det_res(dt_boxes, ori_im.shape)

    et = time.time()
    return dt_boxes, et - st

# Set parameters
args = parse_args()
args.det_model_dir = "./inference/ch_PP-OCRv2_det_infer"
args.image_dir = "./doc/imgs/00018069.jpg"

# Get the picture list
image_file_list = get_image_file_list(args.image_dir)
# Create a text detector
text_detector = TextDetector(args)

count = 0
total_time = 0
draw_img_save = "./inference_results"

if not os.path.exists(draw_img_save):
    os.makedirs(draw_img_save)
save_results = []
for image_file in image_file_list:
    img = cv2.imread(image_file)
    if img is None:
        logger.info("error in loading image:{}".format(image_file))
        continue
    st = time.time()
    dt_boxes, _ = text_detector(img)
    elapse = time.time() - st
    if count > 0:
        total_time += elapse
    count += 1
    save_pred = os.path.basename(image_file) + "\t" + str(
        json.dumps(np.array(dt_boxes).astype(np.int32).tolist())) + "\n"
    save_results.append(save_pred)

```

(continues on next page)

(continued from previous page)

```

logger.info(save_pred)
logger.info("The predict time of {}: {}".format(image_file, elapse))
src_im = utility.draw_text_det_res(dt_boxes, image_file)
img_name_pure = os.path.split(image_file)[-1]
img_path = os.path.join(draw_img_save,
                        "det_res_{}".format(img_name_pure))
cv2.imwrite(img_path, src_im)
logger.info("The visualized image saved in {}".format(img_path))

break

with open(os.path.join(draw_img_save, "det_results.txt"), 'w') as f:
    f.writelines(save_results)
    f.close()

plt.figure(figsize=(10, 10))
plt.imshow(src_im[:, :, ::-1])
plt.show()

```

The following information will be printed.

```
[2021/12/25 20:48:47] root INFO: 00018069.jpg      [[[378, 249], ....
```

The following figure will be displayed.

代号	项目	结果	参考值	单位
ALT	谷丙转氨酶	25.6	0--40	U/L
TBIL	总胆红素	11.2	<20	umol/L
DBIL	直接胆红素	3.3	0--7	umol/L
IBIL	间接胆红素	7.9	1.5--15	umol/L
TP	总蛋白	58.9 ↓	60--80	g/L
ALB	白蛋白	35.1	33--55	g/L
GLO	球蛋白	23.8	20--30	g/L
A/G	白球比	1.5	1.5--2.5	
ALP	碱性磷酸酶	93	15--112	IU/L
GGT	谷氨酰转肽酶	14.3	<50	U/L
AST	谷草转氨酶	16.3	8--40	U/L
LDH	乳酸脱氢酶	167	114--240	U/L
ADA	腺苷脱氨酶	12.6	4--24	U/L

This is the whole text detection process.

7.2.3 Inference of PP-OCRv2 Direction Classifier Model

Similarly, we can also use the following commands to quickly experience the direction classifier model.

```
# Download model
!cd inference && wget https://paddleocr.bj.bcebos.com/dygraph_v2.0/ch/ch_ppocr_mobile_
→v2.0_cls_infer.tar -O ch_ppocr_mobile_v2.0_cls_infer.tar && tar -xf ch_ppocr_mobile_
→v2.0_cls_infer.tar
# Inference
!python tools/infer/predict_cls.py \
    --image_dir="../doc/imgs_words/ch/word_1.jpg" \
    --cls_model_dir="../inference/ch_ppocr_mobile_v2.0_cls_infer" \
    --use_gpu=False
# Draw the image
img = cv2.imread("./doc/imgs_words/ch/word_1.jpg")
plt.imshow(img[:, :, ::-1])
plt.show()
```

```
--2021-12-25 15:40:13-- https://paddleocr.bj.bcebos.com/dygraph_v2.0/ch/ch_ppocr_
→mobile_v2.0_cls_infer.tar
Resolving paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com) ... 100.67.200.6
Connecting to paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com) |100.67.200.6|:443...
→. connected.
HTTP request sent, awaiting response... 200 OK
Length: 1454080 (1.4M) [application/x-tar]
Saving to: 'ch_ppocr_mobile_v2.0_cls_infer.tar'

ch_ppocr_mobile_v2. 100%[=====] 1.39M --.-KB/s in 0.04s

2021-12-25 15:40:13 (33.6 MB/s) - 'ch_ppocr_mobile_v2.0_cls_infer.tar' saved
→[1454080/1454080]

[2021/12/25 15:40:15] root INFO: Predicts of ./doc/imgs_words/ch/word_1.jpg:[0,_
→0.9998784]
```



The picture is horizontal and its text is not reversed, and the prediction result is correct.

The implementation code of the direction classifier is as follows.

```
# Reference Code
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/tools/infer/predict_cls.
→py
import copy

# Implementation of the direction classifier
class TextClassifier(object):
```

(continues on next page)

(continued from previous page)

```

def __init__(self, args):
    self.cls_image_shape = [int(v) for v in args.cls_image_shape.split(",")]
    self.cls_batch_num = args.cls_batch_num
    self.cls_thresh = args.cls_thresh
    postprocess_params = {
        'name': 'ClsPostProcess',
        "label_list": args.label_list,
    }
    # Post-processing operator
    self.postprocess_op = build_post_process(postprocess_params)
    # Initialize the inference engine
    self.predictor, self.input_tensor, self.output_tensors, _ = \
        utility.create_predictor(args, 'cls', logger)

# Resize and normalize the image
def resize_norm_img(self, img):
    imgC, imgH, imgW = self.cls_image_shape
    h = img.shape[0]
    w = img.shape[1]
    ratio = w / float(h)
    if math.ceil(imgH * ratio) > imgW:
        resized_w = imgW
    else:
        resized_w = int(math.ceil(imgH * ratio))
    resized_image = cv2.resize(img, (resized_w, imgH))
    resized_image = resized_image.astype('float32')
    if self.cls_image_shape[0] == 1:
        resized_image = resized_image / 255
        resized_image = resized_image[np.newaxis, :]
    else:
        resized_image = resized_image.transpose((2, 0, 1)) / 255
    resized_image -= 0.5
    resized_image /= 0.5
    padding_im = np.zeros((imgC, imgH, imgW), dtype=np.float32)
    padding_im[:, :, 0:resized_w] = resized_image
    return padding_im

def __call__(self, img_list):
    img_list = copy.deepcopy(img_list)
    img_num = len(img_list)
    # Record the aspect ratio
    width_list = []
    for img in img_list:
        width_list.append(img.shape[1] / float(img.shape[0]))
    # Sort and speed up the subsequent pre-processing
    indices = np.argsort(np.array(width_list))

    cls_res = [['', 0.0]] * img_num
    batch_num = self.cls_batch_num
    elapse = 0
    for beg_img_no in range(0, img_num, batch_num):
        end_img_no = min(img_num, beg_img_no + batch_num)
        norm_img_batch = []
        max_wh_ratio = 0
        starttime = time.time()
        # Preprocess data, and group batches

```

(continues on next page)

(continued from previous page)

```

for ino in range(beg_img_no, end_img_no):
    h, w = img_list[indices[ino]].shape[0:2]
    wh_ratio = w * 1.0 / h
    max_wh_ratio = max(max_wh_ratio, wh_ratio)
for ino in range(beg_img_no, end_img_no):
    norm_img = self.resize_norm_img(img_list[indices[ino]])
    norm_img = norm_img[np.newaxis, :]
    norm_img_batch.append(norm_img)
norm_img_batch = np.concatenate(norm_img_batch)
norm_img_batch = norm_img_batch.copy()
# Copy the data to the inference engine
self.input_tensor.copy_from_cpu(norm_img_batch)
# Automatically perform the inference
self.predictor.run()
# Copy the data back to the CPU
prob_out = self.output_tensors[0].copy_to_cpu()
# Post-process
cls_result = self.postprocess_op(prob_out)
elapse += time.time() - starttime
for rno in range(len(cls_result)):
    label, score = cls_result[rno]
    cls_res[indices[beg_img_no + rno]] = [label, score]
    if '180' in label and score > self.cls_thresh:
        img_list[indices[beg_img_no + rno]] = cv2.rotate(
            img_list[indices[beg_img_no + rno]], 1)
return img_list, cls_res, elapse

args = parse_args()
args.cls_model_dir = "./inference/ch_ppocr_mobile_v2.0_cls_infer"
args.image_dir = "./doc/imgs_words/ch/word_4.jpg"

image_file_list = get_image_file_list(args.image_dir)
text_classifier = TextClassifier(args)
valid_image_file_list = []
img_list = []
for image_file in image_file_list:
    img = cv2.imread(image_file)
    # Rotate the image 180 degrees before inference
    # img = cv2.rotate(img, cv2.ROTATE_180)
    if img is None:
        logger.info("error in loading image:{}".format(image_file))
        continue
    valid_image_file_list.append(image_file)
    img_list.append(img)
img_list, cls_res, predict_time = text_classifier(img_list)
for ino in range(len(img_list)):
    logger.info("Predicts of {}:{}".format(valid_image_file_list[ino],
                                             cls_res[ino]))

plt.imshow(img[:, :, ::-1])
plt.show()

```

[2021/12/25 15:43:28] root INFO: Predicts of ./doc/imgs_words/ch/word_4.jpg:[‘0’, ↴0.9999982]



Here we can also rotate the image 180 degrees to see the classification result of the direction classifier.

This is the inference process of the direction classifier.

7.2.4 Inference of PP-OCRV2 Text Recognition Model

It is feasible to use the following commands to quickly experience the functions of the text recognition model.

```
# Download the model
!cd inference && wget https://paddleocr.bj.bcebos.com/PP-OCRV2/chinese/ch_PP-OCRV2_
->rec_infer.tar -O ch_PP-OCRV2_rec_infer.tar && tar -xf ch_PP-OCRV2_rec_infer.tar
# Inference
!python tools/infer/predict_rec.py \
--image_dir="../doc/imgs_words/ch/word_4.jpg" \
--rec_model_dir="../inference/ch_PP-OCRV2_rec_infer" \
--use_gpu=False

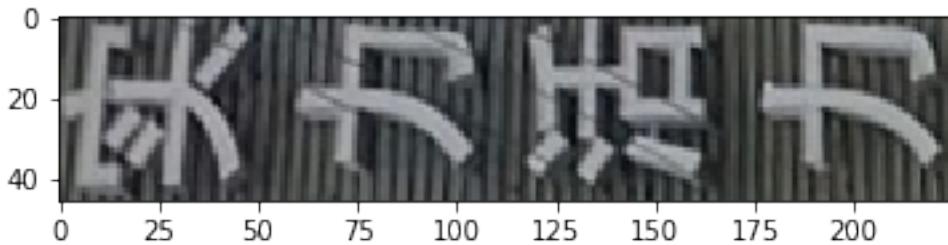
# Read the image and display it
img = cv2.imread("./doc/imgs_words/ch/word_4.jpg")
plt.imshow(img[:, :, ::-1])
plt.show()
```

```
--2021-12-25 15:43:40-- https://paddleocr.bj.bcebos.com/PP-OCRV2/chinese/ch_PP-
->OCRV2_rec_infer.tar
Resolving paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com)... 100.67.200.6
Connecting to paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com)|100.67.200.6|:443..
->. connected.
HTTP request sent, awaiting response... 200 OK
Length: 8875520 (8.5M) [application/x-tar]
Saving to: 'ch_PP-OCRV2_rec_infer.tar'

ch_PP-OCRV2_rec_inf 100%[=====] 8.46M --.-KB/s in 0.1s

2021-12-25 15:43:40 (64.5 MB/s) - 'ch_PP-OCRV2_rec_infer.tar' saved [8875520/
->8875520]

[2021/12/25 15:43:42] root INFO: Predicts of ./doc/imgs_words/ch/word_4.jpg: ('家', 0.9409585)
```



The code for the text recognition is shown below.

```
# Reference Code
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/tools/infer/predict_rec.py
class TextRecognizer(object):
    def __init__(self, args):
        self.rec_image_shape = [int(v) for v in args.rec_image_shape.split(",")]
        self.rec_batch_num = args.rec_batch_num
        self.rec_algorithm = args.rec_algorithm
        postprocess_params = {
            'name': 'CTCLabelDecode',
            "character_dict_path": args.rec_char_dict_path,
            "use_space_char": args.use_space_char
        }
        # Initialize the inference engine
        self.predictor, self.input_tensor, self.output_tensors, self.config = \
            utility.create_predictor(args, 'rec', logger)
        # Build Post-processing
        self.postprocess_op = build_post_process(postprocess_params)

    # Pre-process the core logic
    def resize_norm_img(self, img, max_wh_ratio):
        imgC, imgH, imgW = self.rec_image_shape
        assert imgC == img.shape[2]
        imgW = int((32 * max_wh_ratio))
        h, w = img.shape[:2]
        ratio = w / float(h)
        if math.ceil(imgH * ratio) > imgW:
            resized_w = imgW
        else:
            resized_w = int(math.ceil(imgH * ratio))
        resized_image = cv2.resize(img, (resized_w, imgH))
        resized_image = resized_image.astype('float32')
        # [0, 255] -> [0, 1]
        resized_image = resized_image.transpose((2, 0, 1)) / 255
        # [0, 1] -> [-0.5, 0.5]
        resized_image -= 0.5
        # [-0.5, 0.5] -> [-1, 1]
        resized_image /= 0.5
        padding_im = np.zeros((imgC, imgH, imgW), dtype=np.float32)
        padding_im[:, :, 0:resized_w] = resized_image
        return padding_im

    # Process the image list
    def __call__(self, img_list):
        img_num = len(img_list)
        # Record aspect ratio
```

(continues on next page)

(continued from previous page)

```

width_list = []
for img in img_list:
    width_list.append(img.shape[1] / float(img.shape[0]))
# Sort and speed up the process
indices = np.argsort(np.array(width_list))
rec_res = [['', 0.0]] * img_num
batch_num = self.rec_batch_num
st = time.time()
for beg_img_no in range(0, img_num, batch_num):
    end_img_no = min(img_num, beg_img_no + batch_num)
    norm_img_batch = []
    max_wh_ratio = 0
    for ino in range(beg_img_no, end_img_no):
        h, w = img_list[indices[ino]].shape[0:2]
        wh_ratio = w * 1.0 / h
        max_wh_ratio = max(max_wh_ratio, wh_ratio)
    # Call preprocessing method and group batch
    for ino in range(beg_img_no, end_img_no):
        norm_img = self.resize_norm_img(img_list[indices[ino]],
                                         max_wh_ratio)
        norm_img = norm_img[np.newaxis, :]
        norm_img_batch.append(norm_img)
    norm_img_batch = np.concatenate(norm_img_batch)
    norm_img_batch = norm_img_batch.copy()

    # Copy the data to the prediction engine
    self.input_tensor.copy_from_cpu(norm_img_batch)
    # Automated inference process
    self.predictor.run()
    outputs = []
    # Copy data to CPU
    for output_tensor in self.output_tensors:
        output = output_tensor.copy_to_cpu()
        outputs.append(output)
    if len(outputs) != 1:
        preds = outputs
    else:
        preds = outputs[0]
    # Post-process
    rec_result = self.postprocess_op(preds)
    for rno in range(len(rec_result)):
        rec_res[indices[beg_img_no + rno]] = rec_result[rno]
return rec_res, time.time() - st

# Define parameters
args = parse_args()
args.rec_model_dir = "./inference/ch_PP-OCRv2_rec_infer"
args.image_dir = "./doc/imgs_words/ch/word_4.jpg"
img_list = []

image_file_list = get_image_file_list(args.image_dir)
text_recognizer = TextRecognizer(args)
valid_image_file_list = []
for image_file in image_file_list:
    img = cv2.imread(image_file)

```

(continues on next page)

(continued from previous page)

```

if img is None:
    logger.info("error in loading image:{}".format(image_file))
    continue
valid_image_file_list.append(image_file)
img_list.append(img)
rec_res, _ = text_recognizer(img_list)
for ino in range(len(img_list)):
    logger.info("Predicts of {}:{}".format(valid_image_file_list[ino],
                                             rec_res[ino]))

```

[2021/12/25 15:51:51] root INFO: Predicts of ./doc/imgs_words/ch/word_4.jpg:('四', 0.9409561)

7.2.5 End-to-end Inference of PP-OCRV2 System

The previous parts have introduced the separate inference processes of the detection model, the direction classifier, and the recognition model in the PP-OCRV2 system. To provider more convenience for end-users, we have connected these three modules to form the PP-OCRV2 system, and provided the inference script.

When performing system inference of PP-OCRV2, you need to designate the path of a single image or an image collection through the parameter `image_dir`, and designate the `inference_model` path of the detection, the direction classifier, and the recognition through the parameters `det_model_dir`, `cls_model_dir` and `rec_model_dir` respectively. The parameter `use_angle_cls` is used to control whether to enable the direction classifier model. `use_mp` indicates whether to use multiple processes. `total_process_num` indicates the number of processes.

Take the image file `./doc/imgs/00018069.jpg` as an example, and the inferred original image is as follows.

代号	项目	结果	参考值	单位
ALT	谷丙转氨酶	25.6	0--40	U/L
TBIL	总胆红素	11.2	<20	umol/L
DBIL	直接胆红素	3.3	0--7	umol/L
IBIL	间接胆红素	7.9	1.5--15	umol/L
TP	总蛋白	58.9↓	60--80	g/L
ALB	白蛋白	35.1	33--55	g/L
GLO	球蛋白	23.8	20--30	g/L
A/G	白球比	1.5	1.5--2.5	
ALP	碱性磷酸酶	93	15--112	IU/L
GGT	谷氨酰转肽酶	14.3	<50	U/L
AST	谷草转氨酶	16.3	8--40	U/L
LDH	乳酸脱氢酶	167	114--240	U/L
ADA	腺苷脱氨酶	12.6	4--24	U/L

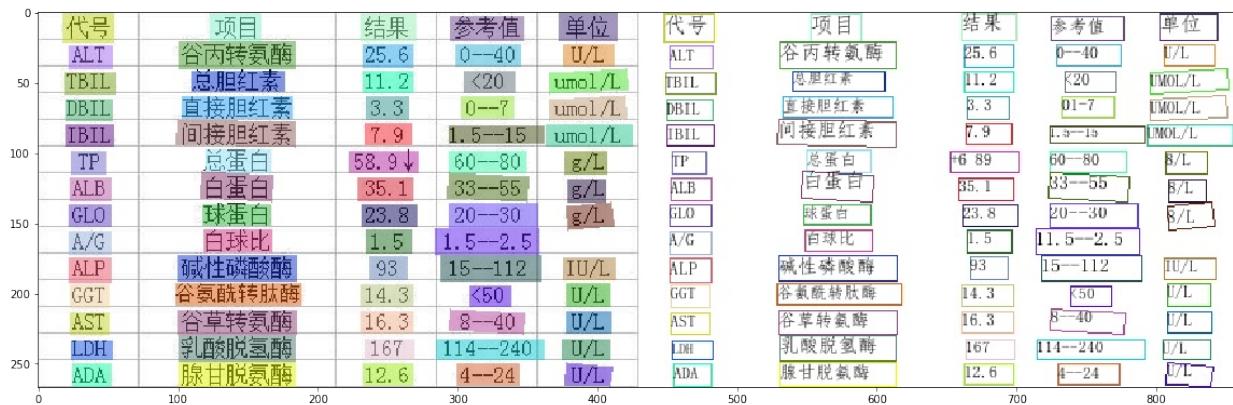
Figure 3: The original image

If you use a direction classifier for end-to-end inference, you can use the following command to make inference.

```
# Use the direction classifier to run the PP-OCRv2 system
!python tools/infer/predict_system.py \
--image_dir="../doc/imgs/00018069.jpg" \
--det_model_dir="../inference/ch_PP-OCRv2_det_infer/" \
--cls_model_dir="../inference/ch_ppocr_mobile_v2.0_cls_infer/" \
--rec_model_dir="../inference/ch_PP-OCRv2_rec_infer/" \
--use_angle_cls=True

# Visualize
img = cv2.imread("./inference_results/00018069.jpg")
plt.figure(figsize=(20, 8))
plt.imshow(img[..., ::-1])
plt.show()
```

The visualization result is saved in the `./inference_results` folder by default, which is as shown below.



The detection frame and the recognition result are visualized in the image, and the specific recognized file and the file reading the path information are also printed in the above notebook.

If you want to save the cropped recognition results, set the `save_crop_res` parameter to True, and the final result is saved in the `output` directory. Some of the cropped images are shown below. The saved results can be used for the coming labeling and training of recognition models.

```
# Tailor the result image of text detection and save it
!python tools/infer/predict_system.py \
--image_dir="../doc/imgs/00018069.jpg" \
--det_model_dir="../inference/ch_PP-OCRv2_det_infer/" \
--cls_model_dir="../inference/ch_ppocr_mobile_v2.0_cls_infer/" \
--rec_model_dir="../inference/ch_PP-OCRv2_rec_infer/" \
--use_angle_cls=True \
--save_crop_res=True

!ls output

plt.figure(figsize=(8, 8))
plt.imshow(cv2.imread("./doc/imgs/00018069.jpg")[:, :, ::-1])
plt.show()

plt.figure(figsize=(14, 4))
plt.subplot(1, 3, 1)
plt.imshow(cv2.imread("output/mg_crop_0.jpg")[:, :, ::-1])
plt.subplot(1, 3, 2)
```

(continues on next page)

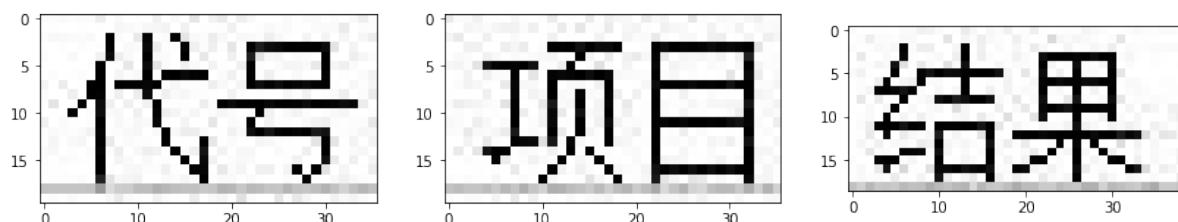
(continued from previous page)

```
plt.imshow(cv2.imread("output/mg_crop_1.jpg")[:, :, ::-1])
plt.subplot(1, 3, 3)
plt.imshow(cv2.imread("output/mg_crop_2.jpg")[:, :, ::-1])
plt.show()
```

The original image is as follows.

代号	项目	结果	参考值	单位
ALT	谷丙转氨酶	25.6	0--40	U/L
TBIL	总胆红素	11.2	<20	umol/L
DBIL	直接胆红素	3.3	0--7	umol/L
IBIL	间接胆红素	7.9	1.5--15	umol/L
TP	总蛋白	58.9↓	60--80	g/L
ALB	白蛋白	35.1	33--55	g/L
GLO	球蛋白	23.8	20--30	g/L
A/G	白球比	1.5	1.5--2.5	
ALP	碱性磷酸酶	93	15--112	IU/L
GGT	谷氨酰转肽酶	14.3	<50	U/L
AST	谷草转氨酶	16.3	8--40	U/L
LDH	乳酸脱氢酶	167	114--240	U/L
ADA	腺苷脱氨酶	12.6	4--24	U/L

Images after crop are as follows.



The end-to-end inference is implemented by TextSystem, and the process and definition of the function are as follows.

```
# Reference Code@https://github.com/PaddlePaddle/PaddleOCR/blob/release%2F2.4/tools/
→infer/predict_system.py
from tools.infer.utility import draw_ocr_box_txt, get_rotate_crop_image
from ppocr.utils.utility import get_image_file_list

class TextSystem(object):
    # Initialize the function
    def __init__(self, args):
        self.args = args
        # If you do not want to display log, you can set show_log to False
```

(continues on next page)

(continued from previous page)

```

if not args.show_log:
    logger.setLevel(logging.INFO)
# Define the text recognition model inference engine
self.text_detector = TextDetector(args)
# Define the text recognition model inference engine
self.text_recognizer = TextRecognizer(args)
# Use the direction classifier or not
self.use_angle_cls = args.use_angle_cls
# Score the threshold, and determine whether the detection and recognition
→results need to be visualized or returned
self.drop_score = args.drop_score
# Define the direction classifier inference engine
if self.use_angle_cls:
    self.text_classifier = TextClassifier(args)

# Save the result image in the text detection
def draw_crop_rec_res(self, output_dir, img_crop_list, rec_res):
    os.makedirs(output_dir, exist_ok=True)
    bbox_num = len(img_crop_list)
    for bno in range(bbox_num):
        cv2.imwrite(
            os.path.join(output_dir,
                         f"mg_crop_{bno+self.crop_image_res_index}.jpg"),
            img_crop_list[bno])
        logger.debug(f"{bno}, {rec_res[bno]}")
    self.crop_image_res_index += bbox_num

# Core inference function
def __call__(self, img, cls=True):
    ori_im = img.copy()
    # Get the detection result of the detected text
    dt_boxes, elapse = self.text_detector(img)
    logger.debug("dt_boxes num : {}, elapse : {}".format(
        len(dt_boxes), elapse))
    if dt_boxes is None:
        return None, None
    img_crop_list = []
    # Sort the detection boxes, and the order is: from top to bottom, and from
    →left to right
    dt_boxes = sorted_boxes(dt_boxes)
    # Perform perspective transformation and correction on the detection result
    for bno in range(len(dt_boxes)):
        tmp_box = copy.deepcopy(dt_boxes[bno])
        img_crop = get_rotate_crop_image(ori_im, tmp_box)
        img_crop_list.append(img_crop)
    # Use the direction classifier to correct the detection result
    if self.use_angle_cls and cls:
        img_crop_list, angle_list, elapse = self.text_classifier(
            img_crop_list)
        logger.debug("cls num : {}, elapse : {}".format(
            len(img_crop_list), elapse))
    # Get the text recognition result
    rec_res, elapse = self.text_recognizer(img_crop_list)
    logger.debug("rec_res num : {}, elapse : {}".format(
        len(rec_res), elapse))
    # Save the corrected images in the text detection

```

(continues on next page)

(continued from previous page)

```

if self.args.save_crop_res:
    self.draw_crop_rec_res(self.args.crop_res_save_dir, img_crop_list,
                           rec_res)
filter_boxes, filter_rec_res = [], []
#Filter the results according to the threshold of the recognition score, and
→if the score is less than the threshold, filter it out
for box, rec_reuslt in zip(dt_boxes, rec_res):
    text, score = rec_reuslt
    if score >= self.drop_score:
        filter_boxes.append(box)
        filter_rec_res.append(rec_reuslt)
return filter_boxes, filter_rec_res

def sorted_boxes(dt_boxes):
    # Sort the detection boxes: first from top to bottom, then from left to right
    num_boxes = dt_boxes.shape[0]
    sorted_boxes = sorted(dt_boxes, key=lambda x: (x[0][1], x[0][0]))
    _boxes = list(sorted_boxes)

    for i in range(num_boxes - 1):
        if abs(_boxes[i + 1][0][1] - _boxes[i][0][1]) < 10 and \
           (_boxes[i + 1][0][0] < _boxes[i][0][0]):
            tmp = _boxes[i]
            _boxes[i] = _boxes[i + 1]
            _boxes[i + 1] = tmp
    return _boxes

args = parse_args()
args.cls_model_dir = "./inference/ch_ppocr_mobile_v2.0_cls_infer"
args.det_model_dir = "./inference/ch_PP-OCRv2_det_infer/"
args.rec_model_dir = "./inference/ch_PP-OCRv2_rec_infer/"
args.image_dir = "./doc/imgs/00018069.jpg"
args.use_angle_cls=True
args.use_gpu=True

image_file_list = get_image_file_list(args.image_dir)
image_file_list = image_file_list[args.process_id::args.total_process_num]
text_sys = TextSystem(args)
is_visualize = True
font_path = args.vis_font_path
drop_score = args.drop_score

total_time = 0
cpu_mem, gpu_mem, gpu_util = 0, 0, 0
_st = time.time()
count = 0
for idx, image_file in enumerate(image_file_list):
    img = cv2.imread(image_file)
    if img is None:
        logger.debug("error in loading image:{}\n".format(image_file))
        continue
    starttime = time.time()
    dt_boxes, rec_res = text_sys(img)
    elapse = time.time() - starttime
    total_time += elapse

```

(continues on next page)

(continued from previous page)

```

logger.debug(
    str(idx) + " Predict time of %s: %.3fs" % (image_file, elapse))
for text, score in rec_res:
    logger.debug("{}， {:.3f}".format(text, score))

if is_visualize:
    image = Image.fromarray(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    boxes = dt_boxes
    txts = [rec_res[i][0] for i in range(len(rec_res))]
    scores = [rec_res[i][1] for i in range(len(rec_res))]

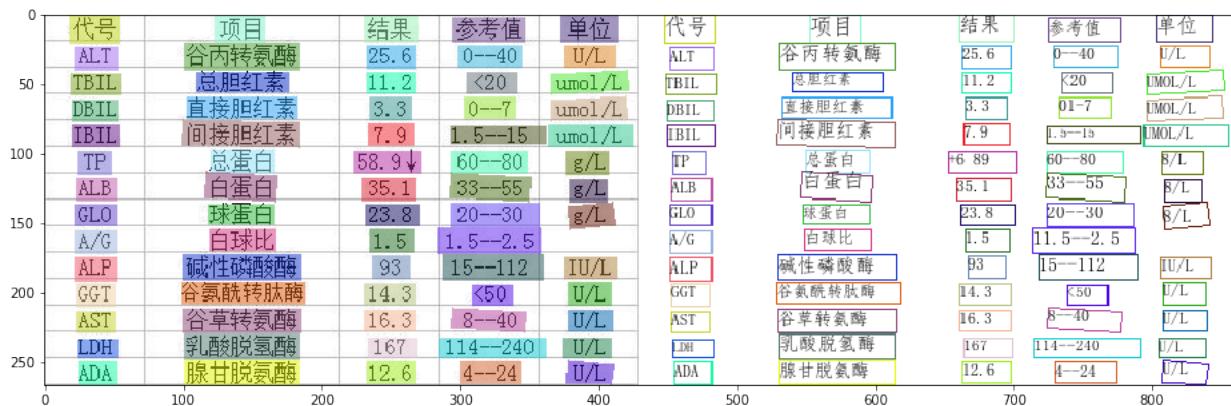
    draw_img = draw_ocr_box_txt(
        image,
        boxes,
        txts,
        scores,
        drop_score=drop_score,
        font_path=font_path)
    draw_img_save_dir = args.draw_img_save_dir
    os.makedirs(draw_img_save_dir, exist_ok=True)
    cv2.imwrite(
        os.path.join(draw_img_save_dir, os.path.basename(image_file)),
        draw_img[:, :, ::-1])
    logger.debug("The visualized image saved in {}".format(
        os.path.join(draw_img_save_dir, os.path.basename(image_file)))))

logger.info("The predict total time is {}".format(time.time() - _st))

plt.figure(figsize=(8, 8))
plt.imshow(image)
plt.show()
plt.figure(figsize=(16, 8))
plt.imshow(draw_img)
plt.show()

```

The inference result is as follows.



7.2.6 inference using WHL Package in PP-OCRV2

To get started with the OCR text detection and recognition model more conveniently, PaddleOCR provides a whl package based on the Paddle Inference inference engine. You can experience PaddleOCR after one-click installation.

Installing whl Package

Use pip to install the whl package of PaddleOCR, and the command is as follows.

```
!pip install "paddleocr==2.3.0.2"

# If you want to get the latest features, you can install it based on compiling of
# the source code
#     python3 setup.py bdist_wheel
#     pip3 install dist/paddleocr-x.x.x-py3-none-any.whl # x.x.x Is the version
# number of paddleocr
```

Using whl Package for Inference

The PaddleOCR whl package will automatically download the PP-OCRV2 ultra-lightweight model as the default model. It also supports custom model paths, inference configurations and other parameters. The parameter names are the same as those in the Python inference in Paddle Inference.

- Perform Testing Separately

Run the following code to quickly experience the inference of the text detection model.

```
from paddleocr import PaddleOCR, draw_ocr

ocr = PaddleOCR(use_gpu=False) # need to run only once to download and load model
# into memory
img_path = '/home/aistudio/PaddleOCR/doc/imgs/11.jpg'
result = ocr.ocr(img_path, rec=False)
for line in result:
    print(line)

# Display the result
from PIL import Image

image = Image.open(img_path).convert('RGB')
im_show = draw_ocr(image, result, txts=None, scores=None, font_path='/home/aistudio/
#PaddleOCR/doc/fonts/simfang.ttf')
plt.figure(figsize=(15, 8))
plt.imshow(im_show)
plt.show()
```

The output is as follows.

```
...
[[27.0, 459.0], [135.0, 459.0], [135.0, 479.0], [27.0, 479.0]]
[[29.0, 431.0], [369.0, 431.0], [369.0, 444.0], [29.0, 444.0]]
[[26.0, 397.0], [361.0, 397.0], [361.0, 414.0], [26.0, 414.0]]
...
```



- Perform identification separately

You can specify `det=False` to run only a single recognition module.

```
from paddleocr import PaddleOCR

ocr = PaddleOCR(use_gpu=False) # need to run only once to download and load model into memory
img_path = '/home/aistudio/PaddleOCR/doc/imgs_words/ch/word_1.jpg'
result = ocr.ocr(img_path, det=False)
for line in result:
    print(line)

# expected output: ('纯臻', 0.9967349)
```

- Separately execute the direction classifier

You can designate `det=False, rec=False, cls=True` and only run the direction classifier.

```
from paddleocr import PaddleOCR

ocr = PaddleOCR(use_angle_cls=True, use_gpu=False) # need to run only once to download and load model into memory
img_path = '/home/aistudio/PaddleOCR/doc/imgs_words/ch/word_1.jpg'
```

(continues on next page)

(continued from previous page)

```

result = ocr.ocr(img_path, det=False, rec=False, cls=True)
for line in result:
    print(line)

img = cv2.imread(img_path)
plt.imshow(img[...,:-1])
plt.show()

# expected output: ['0', 0.9998784]

```

- Experience the whole process of *Detection + direction classifier + recognition

```

from paddleocr import PaddleOCR, draw_ocr
import matplotlib.pyplot as plt
%matplotlib inline

# PaddleOCR currently supports many languages, including Chinese, English, French, ↵
# German, Korean, Japanese, and you can switch the mode by modifying the lang ↵
# parameter
# Their parameters are `ch`, `en`, `french`, `german`, `korean`, `japan`.
ocr = PaddleOCR(use_angle_cls=True, lang="ch", use_gpu=False) # need to run only ↵
# once to download and load model into memory
img_path = '/home/aistudio/PaddleOCR/doc/imgs/11.jpg'
result = ocr.ocr(img_path, cls=True)
for line in result:
    print(line)

# Display the result
from PIL import Image

image = Image.open(img_path).convert('RGB')
boxes = [line[0] for line in result]
txts = [line[1][0] for line in result]
scores = [line[1][1] for line in result]
im_show = draw_ocr(image, boxes, txts, scores, font_path='/home/aistudio/PaddleOCR/ ↵
#doc/fonts/simfang.ttf')
plt.figure(figsize=(15, 8))
plt.imshow(im_show)
plt.show()

'''

Expected output:
...
[[[28.0, 113.0], [330.0, 113.0], [330.0, 133.0], [28.0, 133.0]], ('0450/000010000000', ↵
# 0.90023524)]
[[[26.0, 143.0], [281.0, 144.0], [281.0, 164.0], [26.0, 163.0]], ('022001000000', 0. ↵
#9793598)]
...
'''
```

The result is a list, within which each item contains a text box, a text and recognition confidence.

7.3 C++ Inference Based on Paddle Inference

In the inference deployment, the performance of C++ is better than Python. Therefore, in many scenarios, C++ is chosen as the development language for inference.

The Paddle Inference introduced in the previous section also supports the C++ inference. This section mainly talks about the C++ PP-OCRv2 inference.

In the C++ inference on the PP-OCRv2 system based on Paddle Inference, there are several steps:

- (1) Preparing the model
- (2) Compiling the opencv library
- (3) Getting Paddle Inference prediction library
- (4) Compiling PaddleOCR C++ inference code
- (5) Running the PP-OCRv2 system

Limited by the version on AiStudio, the process will not be demonstrated but only explained here. It is recommended that you experience the C++ inference process of PP-OCRv2 locally.

For more details about this section, please refer to: [PP-OCRv2 C++ Inference Tutorial](#).

7.3.1 Preparing Model

Use the following command to prepare the inference model of PP-OCRv2.

```
cd deploy/cpp_infer
wget https://paddleocr.bj.bcebos.com/PP-OCRv2/chinese/ch_PP-OCRv2_det_infer.tar -O ch_
PP-OCRv2_det_infer.tar && tar -xf ch_PP-OCRv2_det_infer.tar
wget https://paddleocr.bj.bcebos.com/PP-OCRv2/chinese/ch_PP-OCRv2_rec_infer.tar -O ch_
PP-OCRv2_rec_infer.tar && tar -xf ch_PP-OCRv2_rec_infer.tar
wget https://paddleocr.bj.bcebos.com/dygraph_v2.0/ch/ch_ppocr_mobile_v2.0_cls_infer.
tar -O ch_ppocr_mobile_v2.0_cls_infer.tar && tar -xf ch_ppocr_mobile_v2.0_cls_infer.
tar
```

7.3.2 Compiling OpenCV Library

- First, download the package compiled in the Linux environment from the opencv's official website. Take opencv 3.4.7 as an example, and its download command is as follows.

```
wget https://paddleocr.bj.bcebos.com/libs/opencv/opencv-3.4.7.tar.gz
tar -xf opencv-3.4.7.tar.gz
```

Finally, you can find the folder `opencv-3.4.7/` in the current directory.

- Compile OpenCV, set `OpenCV root_path` and `install_path`. Enter into the path of the opencv source code and compile it in the following way.

```
root_path="your_opencv_root_path"
install_path=${root_path}/opencv3
build_dir=${root_path}/build

rm -rf ${build_dir}
```

(continues on next page)

(continued from previous page)

```

mkdir ${build_dir}
cd ${build_dir}

cmake .. \
-DCMAKE_INSTALL_PREFIX=${install_path} \
-DCMAKE_BUILD_TYPE=Release \
-DBUILD_SHARED_LIBS=OFF \
-DWITH_IPP=OFF \
-DBUILD_IPP_IW=OFF \
-DWITH_LAPACK=OFF \
-DWITH_EIGEN=OFF \
-DCMAKE_INSTALL_LIBDIR=lib64 \
-DWITH_ZLIB=ON \
-DBUILD_ZLIB=ON \
-DWITH_JPEG=ON \
-DBUILD_JPEG=ON \
-DWITH_PNG=ON \
-DBUILD_PNG=ON \
-DWITH_TIFF=ON \
-DBUILD_TIFF=ON

make -j
make install

```

You can also modify the content of `tools/build_opencv.sh`, and then run the following command for compiling.

```
sh tools/build_opencv.sh
```

`root_path` is the source code path of the downloaded opencv, and `install_path` is the opencv's installation path. After `make install`, header files and library files of opencv will be generated in this folder for the coming OCR code compilation.

The file structure in the installation path is shown below.

```

opencv3/
|-- bin
|-- include
|-- lib
|-- lib64
|-- share

```

7.3.3 Downloading the Inference Library of Paddle Inference

- The Linux prediction libraries with different cuda versions are available on [Paddle Inference library official website](#). You can choose the version on the official website according to your own environment.
- After downloading, use the following method for decompression.

```

wget https://paddle-inference-lib.bj.bcebos.com/2.2.1/cxx_c/Linux/GPU/x86-64_gcc8.2_
`avx_mkl_cuda10.2_cudnn8.1.1_trt7.2.3.4/paddle_inference.tgz -O paddle_inference.tgz
tar -xf paddle_inference.tgz

```

A subfolder of `paddle_inference/` will be generated in the current folder.

7.3.4 Compiling the Inference Code of PaddleOCR

The compilation command is as follows, and the addresses of Paddle C++ inference library opencv, and other dependent libraries need to be replaced with the real addresses on your own computer.

```
sh tools/build.sh
```

- You need to modify the environment path in tools/build.sh:

```
OPENCV_DIR=your_opencv_dir  
LIB_DIR=your_paddle_inference_dir  
CUDA_LIB_DIR=your_cuda_lib_dir  
CUDNN_LIB_DIR=/your_cudnn_lib_dir
```

OPENCV_DIR is the address where opencv is compiled and installed; LIB_DIR is the address where the Paddle inference library is downloaded (the paddle_inference folder) or compiled (build/paddle_inference_install_dir folder); CUDA_LIB_DIR is the address of the cuda library file, and it is /usr/local/cuda/lib64 in the docker; UDNN_LIB_DIR is the address of the cudnn library file, and it is /usr/lib/x86_64-linux-gnu/ in the docker. **Note: The above paths should as absolute paths instead of relative paths.**

- After the compilation, an executable file named ppocr will be generated in the build folder.

7.3.5 Running the PP-OCRV2 System

Running mode

```
./build/ppocr <mode> [--param1] [--param2] [...]
```

mode is a required parameter, which represented the selection function, and the value range is ['det','rec','system'], which represent calling detection, recognition, and end-to-end detection and recognition (including the direction classifier) respectively. The commands are as follows:

- Run the text detection model only

```
./build/ppocr det \  
--det_model_dir=./ch_PP-OCRV2_det_infer/ \  
--image_dir=../../doc/imgs/12.jpg
```

- Run the text detection model only

```
./build/ppocr rec \  
--rec_model_dir=./ch_PP-OCRV2_rec_infer/ \  
--image_dir=../../doc/imgs_words/ch/
```

- Run the PP-OCRV2 system

```
# Do not use the direction classifier  
./build/ppocr system \  
--det_model_dir=./ch_PP-OCRV2_det_infer/ \  
--rec_model_dir=./ch_PP-OCRV2_rec_infer/ \  
--image_dir=../../doc/imgs/12.jpg  
  
# Use the direction classifier
```

(continues on next page)

(continued from previous page)

```
./build/ppoocr system \
    --det_model_dir=./ch_PP-OCRv2_det_infer/ \
    --rec_model_dir=./ch_PP-OCRv2_rec_infer/ \
    --use_angle_cls=true \
    --cls_model_dir=./ch_ppocr_mobile_v2.0_cls_infer \
    --image_dir=../../../../doc/imgs/12.jpg
```

7.4 Service Deployment using Paddle Serving

In Sections 2 and 3, we have introduced the inference of the PP-OCRv2 system inference based on Paddle Inference, which is an offline inference where the code deployed on a specific machine can only be used on this machine and cannot be visited on other machines. Therefore, the demand for model service deployment is generated.

In Service deployment, the model is deployed as a service, and other devices can access the service by sending a request to obtain the inference result of the model service. Its schematic diagram is shown below.



Figure 4: Schematic diagram of service deployment

After the model is deployed, different users can get the inference service by sending requests as clients.

Paddle Serving is a tool created by PaddlePaddle to help developers perform service deployment. This section is mainly about the service deployment of the PP-OCRv2 system with Paddle Serving.

7.4.1 Introduction to Paddle Serving

Paddle Serving is an open-source service deployment framework of PaddlePaddle. Its long-term goal is to provide more and more professional, reliable, and easy-to-use services to deal with the last mile of AI's landing. Paddle Serving now provides two frameworks: C++ Serving and Python Pipeline. The Python Pipeline framework tends to facilitate the secondary development, and the C++ Serving framework tends to pursue high performance.

When the PP-OCRv2 model is deployed as a service with Paddle Serving, the process is as follows.

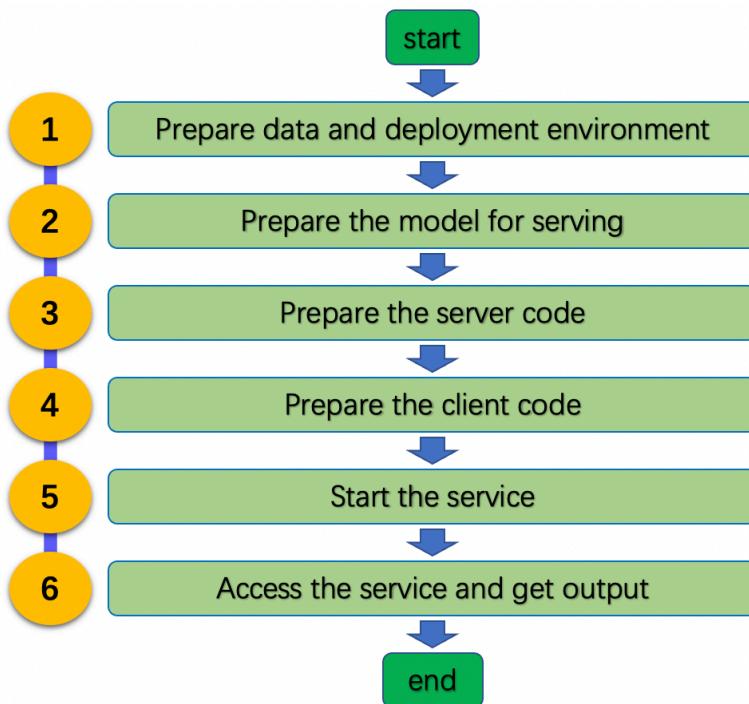


Figure 5: Flow chart of PP-OCRv2 system deployment based on Paddle Serving

7.4.2 Preparing Inference Data and Deployment Environment

The data should be in consistent with those used for the model inference.

Before running Paddle Serving, three packages of Paddle Serving need to be installed: paddle-serving-server, paddle-serving-client and paddle-serving-app. The commands are as follows.

```

!wget https://paddle-serving.bj.bcebos.com/test-dev/wheel/paddle_serving_server_gpu-0.7.0-
˓→.post102-py3-none-any.whl
!pip install paddle_serving_server_gpu-0.7.0.post102-py3-none-any.whl

!wget https://paddle-serving.bj.bcebos.com/test-dev/wheel/paddle_serving_client-0.7.0-
˓→.cp37-none-any.whl
!pip install paddle_serving_client-0.7.0-cp37-none-any.whl

!wget https://paddle-serving.bj.bcebos.com/test-dev/wheel/paddle_serving_app-0.7.0-py3-
˓→.none-any.whl
!pip install paddle_serving_app-0.7.0-py3-none-any.whl

!rm ./*.whl
  
```

7.4.3 Preparing for model deployment

Before the model service deployment, first convert the inference model into a model for user service deployment.

First run the following command to download the inference model.

```
os.chdir("/home/aistudio/PaddleOCR/deploy/pdserving/")

# Download and unzip the OCR text detection model
!wget https://paddleocr.bj.bcebos.com/PP-OCRV2/chinese/ch_PP-OCRV2_det_infer.tar -O_
˓→ch_PP-OCRV2_det_infer.tar && tar -xf ch_PP-OCRV2_det_infer.tar && rm ch_PP-OCRV2_
˓→det_infer.tar
# Download and unzip the OCR text detection model
!wget https://paddleocr.bj.bcebos.com/PP-OCRV2/chinese/ch_PP-OCRV2_rec_infer.tar -O_
˓→ch_PP-OCRV2_rec_infer.tar && tar -xf ch_PP-OCRV2_rec_infer.tar && rm ch_PP-OCRV2_
˓→rec_infer.tar

--2021-12-25 16:25:32-- https://paddleocr.bj.bcebos.com/PP-OCRV2/chinese/ch_PP-
˓→OCRV2_det_infer.tar
Resolving paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com) ... 100.67.200.6
Connecting to paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com) |100.67.200.6|:443...
˓→. connected.
HTTP request sent, awaiting response... 200 OK
Length: 3190272 (3.0M) [application/x-tar]
Saving to: 'ch_PP-OCRV2_det_infer.tar'

ch_PP-OCRV2_det_inf 100%[=====] 3.04M --.-KB/s in 0.09s

2021-12-25 16:25:32 (35.0 MB/s) - 'ch_PP-OCRV2_det_infer.tar' saved [3190272/
˓→3190272]

--2021-12-25 16:25:33-- https://paddleocr.bj.bcebos.com/PP-OCRV2/chinese/ch_PP-
˓→OCRV2_rec_infer.tar
Resolving paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com) ... 100.67.200.6
Connecting to paddleocr.bj.bcebos.com (paddleocr.bj.bcebos.com) |100.67.200.6|:443...
˓→. connected.
HTTP request sent, awaiting response... 200 OK
Length: 8875520 (8.5M) [application/x-tar]
Saving to: 'ch_PP-OCRV2_rec_infer.tar'

ch_PP-OCRV2_rec_inf 100%[=====] 8.46M --.-KB/s in 0.1s

2021-12-25 16:25:33 (69.0 MB/s) - 'ch_PP-OCRV2_rec_infer.tar' saved [8875520/
˓→8875520]
```

Run the command to convert the model.

```
# Convert the detection model
!python -m paddle_serving_client.convert --dirname ./ch_PP-OCRV2_det_infer/ \
--model_filename inference.pdmodel \
--params_filename inference.pdiparams \
--serving_server ./ppocrv2_det_serving/ \
--serving_client ./ppocrv2_det_client/ \
\\
\\

# Convert the recognition model
!python -m paddle_serving_client.convert --dirname ./ch_PP-OCRV2_rec_infer/ \
--model_filename inference.pdmodel \
\\
```

(continues on next page)

(continued from previous page)

```
--params_filename inference.pdiparams      \
--serving_server ./ppocrv2_rec_serving/   \
--serving_client ./ppocrv2_rec_client/
```

```
# View the folder
!tree -h *_client *_serving
```

```
ppocrv2_det_client
├── [ 296]  serving_client_conf.prototxt
└── [ 98]  serving_client_conf.stream.prototxt
ppocrv2_rec_client
├── [ 284]  serving_client_conf.prototxt
└── [ 93]  serving_client_conf.stream.prototxt
ppocrv2_det_serving
├── [2.2M]  inference.pdiparams
├── [842K]  inference.pdmodel
├── [ 296]  serving_server_conf.prototxt
└── [ 98]  serving_server_conf.stream.prototxt
ppocrv2_rec_serving
├── [7.9M]  inference.pdiparams
├── [527K]  inference.pdmodel
├── [ 284]  serving_server_conf.prototxt
└── [ 93]  serving_server_conf.stream.prototxt

0 directories, 12 files
```

After the detection model is converted, there will be additional folders of ppocrv2_det_mobile_serving and ppocrv2_det_mobile_client in the current folder, with the following format:

```
| - ppocrv2_det_mobile_serving/
|   | - __model__
|   | - __params__
|   | - serving_server_conf.prototxt
|   | - serving_server_conf.stream.prototxt
|
| - ppocrv2_det_mobile_client
|   | - serving_client_conf.prototxt
|   | - serving_client_conf.stream.prototxt
```

The recognition model is the same.

7.4.4 Deploying Paddle Serving pipeline

Note: Modify the two `model_config` fields in the `PaddleOCR/deploy/pdserving/config.yml` file to `ppocrv2_det_mobile_serving` and `ppocrv2_rec_mobile_serving` respectively, to correspond to the folder of model conversion.

```
The pdserving directory contains the code to start the pipeline service and send prediction requests, including:
```
__init__.py
config.yml # Start the configuration file of the service
```

(continues on next page)

(continued from previous page)

```
ocr_reader.py # The code for OCR model pre-processing and post-processing
pipeline_http_client.py # Script of sending the inference request
web_service.py # Script of starting the server
```
```

Starting the service

Run the following command to start the service:

Open a new terminal and run the following command to start the service

```
```
Start the service and save the running log in web_serving_log.txt
cd PaddleOCR/deploy/pdserving/
nohup python web_service.py &>web_serving_log.txt &
````
```

After the service starts, a log similar to the following will be printed in web_serving_log.txt

```
--- Running analysis [inference_op_replace_pass]
--- Running analysis [memory_optimize_pass]
I0308 09:47:57.704764 65137 memory_optimize_pass.cc:200] Cluster name : conv2d_89.tmp_0 size: 153600
I0308 09:47:57.704782 65137 memory_optimize_pass.cc:200] Cluster name : elementwise_add_7 size: 358400
I0308 09:47:57.704787 65137 memory_optimize_pass.cc:200] Cluster name : conv2d_90.tmp_0 size: 614400
I0308 09:47:57.704792 65137 memory_optimize_pass.cc:200] Cluster name : batch_norm_48.tmp_2 size: 9830400
I0308 09:47:57.704795 65137 memory_optimize_pass.cc:200] Cluster name : relu_2.tmp_0 size: 13107200
I0308 09:47:57.704799 65137 memory_optimize_pass.cc:200] Cluster name : conv2d_96.tmp_0 size: 2457600
I0308 09:47:57.704803 65137 memory_optimize_pass.cc:200] Cluster name : conv2d_92.tmp_0 size: 9830400
I0308 09:47:57.704807 65137 memory_optimize_pass.cc:200] Cluster name : tmp_1 size: 2457600
I0308 09:47:57.704811 65137 memory_optimize_pass.cc:200] Cluster name : x size: 4915200
--- Running analysis [ir_graph_to_program_pass]
I0308 09:47:57.706780 65160 memory_optimize_pass.cc:200] Cluster name : conv2d_89.tmp_0 size: 153600
I0308 09:47:57.706801 65160 memory_optimize_pass.cc:200] Cluster name : elementwise_add_7 size: 358400
I0308 09:47:57.706807 65160 memory_optimize_pass.cc:200] Cluster name : conv2d_90.tmp_0 size: 614400
I0308 09:47:57.706813 65160 memory_optimize_pass.cc:200] Cluster name : batch_norm_48.tmp_2 size: 9830400
I0308 09:47:57.706817 65160 memory_optimize_pass.cc:200] Cluster name : relu_2.tmp_0 size: 13107200
I0308 09:47:57.706825 65160 memory_optimize_pass.cc:200] Cluster name : conv2d_96.tmp_0 size: 2457600
I0308 09:47:57.706831 65160 memory_optimize_pass.cc:200] Cluster name : conv2d_92.tmp_0 size: 9830400
I0308 09:47:57.706836 65160 memory_optimize_pass.cc:200] Cluster name : tmp_1 size: 2457600
I0308 09:47:57.706841 65160 memory_optimize_pass.cc:200] Cluster name : x size: 4915200
I0308 09:47:57.708473 65173 analysis_predictor.cc:548] ===== optimize end =====
I0308 09:47:57.708534 65173 naive_executor.cc:107] --- skip [feed], feed -> x
--- Running analysis [ir_graph_to_program_pass]
I0308 09:47:57.710592 65173 naive_executor.cc:107] --- skip [save_infer_model/scale_0.tmp_0], fetch -> fetch
I0308 09:47:57.743366 65137 analysis_predictor.cc:548] ===== optimize end =====
I0308 09:47:57.743443 65137 naive_executor.cc:107] --- skip [feed], feed -> x
I0308 09:47:57.745395 65137 naive_executor.cc:107] --- skip [relu_2.tmp_0], fetch -> fetch
I0308 09:47:57.752557 65160 analysis_predictor.cc:548] ===== optimize end =====
I0308 09:47:57.752624 65160 naive_executor.cc:107] --- skip [feed], feed -> x
I0308 09:47:57.754582 65160 naive_executor.cc:107] --- skip [relu_2.tmp_0], fetch -> fetch
[]
```

Figure 6: Server log

Sending the service request

```
!python pipeline_http_client.py
```

The output is as follows.

You can adjust the number of concurrency in config.yml. Here only the running effect is demonstrated here, and it is set to 1 by default.

```
det:  
    #Concurrency, when is_thread_op=True, it is thread concurrency; otherwise, it  
    ↪is process concurrency  
    concurrency: 1  
    ...  
rec:  
    #Concurrency, when is_thread_op=True, it is thread concurrency; otherwise, it  
    ↪is process concurrency  
    concurrency: 1  
    ...
```

The inference performance data will be automatically written into the PipelineServingLogs/pipeline.tracer file.

7.4.5 FAQ

Q1: After sending the request, there is no result returned or the output decoding error is prompted

A1: Do not set the proxy when you start the service and send the request. You can close the proxy before. The command to close the proxy is:

```
unset https_proxy  
unset http_proxy
```

7.5 End-to-side inference based on Paddle Lite

As the mobile Internet is more and more popular, there are increasing mobile phones and embedded devices. At the same time, out of data security and economy of model operation, more and more models are directly run in end-side devices.

Paddle Lite is a lightweight inference engine of PaddlePaddle. It provides efficient inference capabilities for mobile phones and IOT terminals, and extensively integrates cross-platform hardware to offer lightweight deployment solutions for end-side deployment and the landing of applications.

This section will introduce the steps to deploy the ultra-lightweight Chinese detection and recognition model of PaddleOCR on the mobile terminal based on [Paddle Lite](#).

The following will show you the demonstration of the PP-OCRv2 series model running on Android.

Android demo link

Since it cannot be demonstrated, here will explain how to develop the PP-OCRv2 system running program based on Paddle Lite.

If you want practice, you can refer to [PaddleOCR deployment document based on Paddle Lite](#).

7.5.1 Environment Preparation

You need to prepare the cross-compilation environment and the Paddle Lite inference library. The cross-compilation environment is used to generate executable files that can be used on end-side devices. It is recommended to use docker as the cross-compilation environment.

7.5.2 Preparing model

In the model inference with Paddle Lite, it is necessary to first convert the inference model into an optimized model for Paddle Lite inference (the suffix is usually nb). A variety of strategies are adopted to automatically optimize the original model here, including quantization, subgraph fusion, hybrid scheduling, and kernel optimization. The optimized model is lighter and faster.

7.5.3 Compiling

Run `make -j` for compilation and get the executable file. In the first execution, of this command will download dependent libraries such as opencv. After the download is complete, run `make -j` again.

7.5.4 Uploading to mobile terminals such as mobile phones

Use tools such as adb to transfer executable files, model files, and configuration files to mobile devices such as mobile phones.

7.5.5 Running

Run the executable file on the mobile terminal to get the result, and the output example.

```
The detection visualized image saved in ./vis.jpg
0      纯臻营养护发素  0.993604
1      产品信息/参数  0.992728
2      (45元 / 每公斤, 100公斤起订)  0.97417
3      每瓶22元, 1000瓶起订)  0.993976
4      【品牌】: 代加工方式/OEMODM  0.985133
5      【品名】: 纯臻营养护发素  0.995007
6      【产品编号】: YM-X-3011 0.96899
7      【净含量】: 220ml  0.996577
8      【适用人群】: 适合所有肤质  0.995842
9      【主要成分】: 鲸蜡硬脂醇、燕麦B-葡聚  0.961928
10     糖、椰油酰胺丙基甜菜碱、泛醌  0.925898
11     (成品包材)  0.972573
12     【主要功能】: 可紧致头发鳞层, 从而达到  0.994448
13     即时持久改善头发光泽的效果, 给干燥的头  0.990198
14     发足够的滋养  0.997668
花费了0.457335秒
```

Figure 7: Output results on the mobile terminal

7.5.6 FAQ

Q1: What if I want to change the model, do I need to go through the whole process again?

A1: If you have gone through the above steps, you only need to replace the .nb model file. Also, update the dictionary at the same time.

Q2: How to test with another picture?

A2: Replace the .jpg test image under debug with the image you want to test, and just push adb to the phone.

Q3: How to package the demo and send it to the mobile APP?

A3: This demo aims to provide the core algorithm part for running OCR on mobile phones. You can refer to PadleOCR/deploy/android_demo. It is an example of packaging this demo into an application on the mobile phone.

7.6 Homework

Please refer to Inference Deployment Objective Questions and Inference Deployment Practice Questions part.

DOCUMENT ANALYSIS TECHNOLOGY

8.1 Introduction to Document Analysis Technology

This chapter mainly introduces the theoretical knowledge of document analysis technology, including its background, algorithm categories and ideas behind algorithms.

In this chapter, you can learn:

1. Categories and ideas of layout analysis
2. Categories and ideas of table recognition
3. Categories and ideas of information extraction

Documents are the carrier of information. Its different layout fits different kinds of information, such as the list and the ID card. Document analysis is of automatic information reading, interpretation, and extraction. It often includes the following research directions:

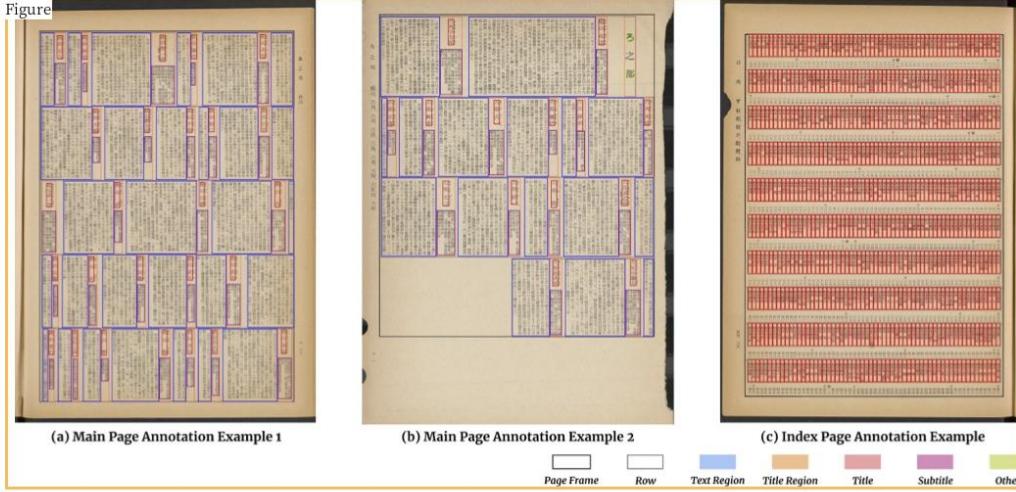
1. Layout analysis module: It divides each document page into different content regions. This module can be used not only to divide relevant and irrelevant regions, but also to classify the content it recognizes.
2. Optical Character Recognition (OCR) module: It locates and recognizes all texts in the document.
3. Table recognition module: It recognizes and converts table in the document into an excel file.
4. Information extraction module: Use OCR results and image information to understand and identify the specific information expressed in the document or the relationship between the information.

Since the OCR module has been detailed in the previous chapters, the other three modules will be introduced one by one here. In each part, the classic or common methods and datasets of the module will be introduced.

8.1.1 Layout Analysis

Background Introduction

Layout analysis is mainly used for document retrieval, key information extraction, content classification, etc. It aims to classify document images. The categories include plain texts, titles, tables, pictures, and lists. However, many factors make layout analysis still a challenging task, involving the diversity and complexity of document layouts and formats, poor quality of document images, and the lack of large-scale annotated datasets. The visualization of the layout analysis task is shown in the figure below:



TextFigure 7: **Annotation Examples in HJDataset.** (a) and (b) show two examples for the labeling of main pages. The boxes are colored differently to reflect the layout element categories. Illustrated in (c), the items in each index page row are categorized as title blocks, and the annotations are denser.

Textover union (IOU) level [0.50:0.95]², on the test data. In general, the high mAP values indicate accurate detection of the layout elements. The Faster R-CNN and Mask R-CNN achieve comparable results, better than RetinaNet. Noticeably, the detections for small blocks like title are less precise, and the accuracy drops sharply for the title category. In Figure 8, (a) and (b) illustrate the accurate prediction results of the Faster R-CNN model.

2.2. Pre-training for other datasets

TextWe also examine how our dataset can help with a real-world document digitization application. When digitizing new publications, researchers usually do not generate large scale ground truth data to train their layout analysis models. If they are able to adapt our dataset, or models trained on our dataset, to develop models on their data, they can build their pipelines more efficiently and develop more accurate models. To this end, we conduct two experiments. First we examine how layout analysis models trained on the main pages can be used for understanding index pages. Moreover, we study how the pre-trained models perform on other historical Japanese documents.

TextTable 4 compares the performance of five Faster R-CNN models that are trained differently on index pages. If the model loads pre-trained weights from HJDataset, it includes information learned from main pages. Models trained over

TextThis is a core metric developed for the COCO competition [12] for evaluating the object detection quality.

Textthe training data can be viewed as the benchmarks, while training with few samples (five in this case) are considered to mimic real-world scenarios. Given different training data, models pre-trained on HJDataset perform significantly better than those initialized with COCO weights. Intuitively, models trained on more data perform better than those with fewer samples. We also directly use the model trained on main to predict index pages without fine-tuning. The low zero-shot prediction accuracy indicates the dissimilarity between index and main pages. The large increase in mAP from 0.344 to 0.471 after the model is

TextTable 3: Detection mAP @ IOU [0.50:0.95] of different models for each category on the test set. All values are given as percentages.

| Category | Faster R-CNN | Mask R-CNN ^a | RetinaNet |
|--------------|--------------|-------------------------|-----------|
| Page Frame | 99.046 | 99.097 | 99.038 |
| Row | 98.831 | 98.482 | 95.067 |
| Title Region | 87.571 | 89.483 | 69.593 |
| Text Region | 94.463 | 86.798 | 89.531 |
| Title | 65.908 | 71.517 | 72.566 |
| Subtitle | 84.093 | 84.174 | 85.865 |
| Other | 44.023 | 39.849 | 14.371 |
| mAP | 81.991 | 81.343 | 75.223 |

Texttraining Mask R-CNN, the segmentation masks are the quadrilateral regions for each block. Compared to the rectangular bounding boxes, they delineate the text region more accurately.

Figure 1: Diagram of the layout analysis

The existing solutions are generally based on object detection or semantic segmentation, which detect or segment different

patterns in the document as different objects.

Some representative papers are divided into the above two categories, as shown in the following table:

| Category | Main papers |
|---------------------------------------|--|
| Method based on object detection | Visual Detection with Context, Object Detection, VSR |
| Method based on semantic segmentation | Semantic Segmentation |

Method Based on Object Detection

Soto Carlos [1] learns from the object detection algorithm Faster R-CNN, uses contextual information and the inherent location information of the document content to improve the performance of the region detection. Li Kai [2] et al. also propose a document analysis method based on object detection, which solves the cross-domain problem by introducing a feature pyramid alignment module, a region alignment module, and a rendering layer alignment module. These three modules complement each other, adjust the domain from a general image perspective and a specific document image perspective, thus solving the inconsistency between large label training datasets and the target domain. The following figure is a flow chart of layout analysis based on the object detection algorithm Faster R-CNN.

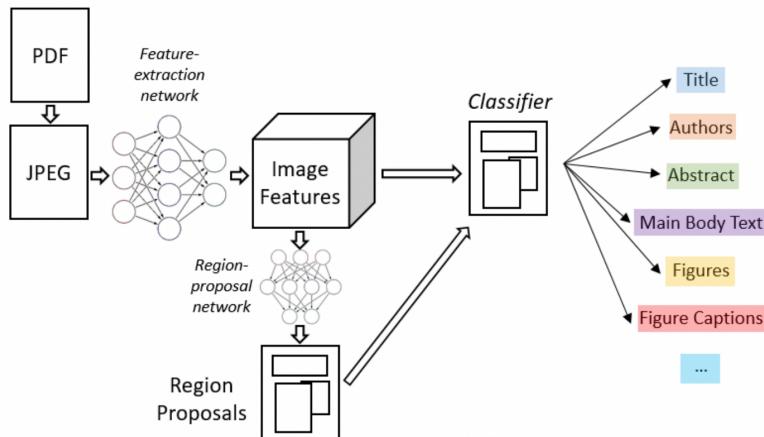


Figure 2: Flow chart of layout analysis based on Faster R-CNN

Methods Based on Semantic Segmentation

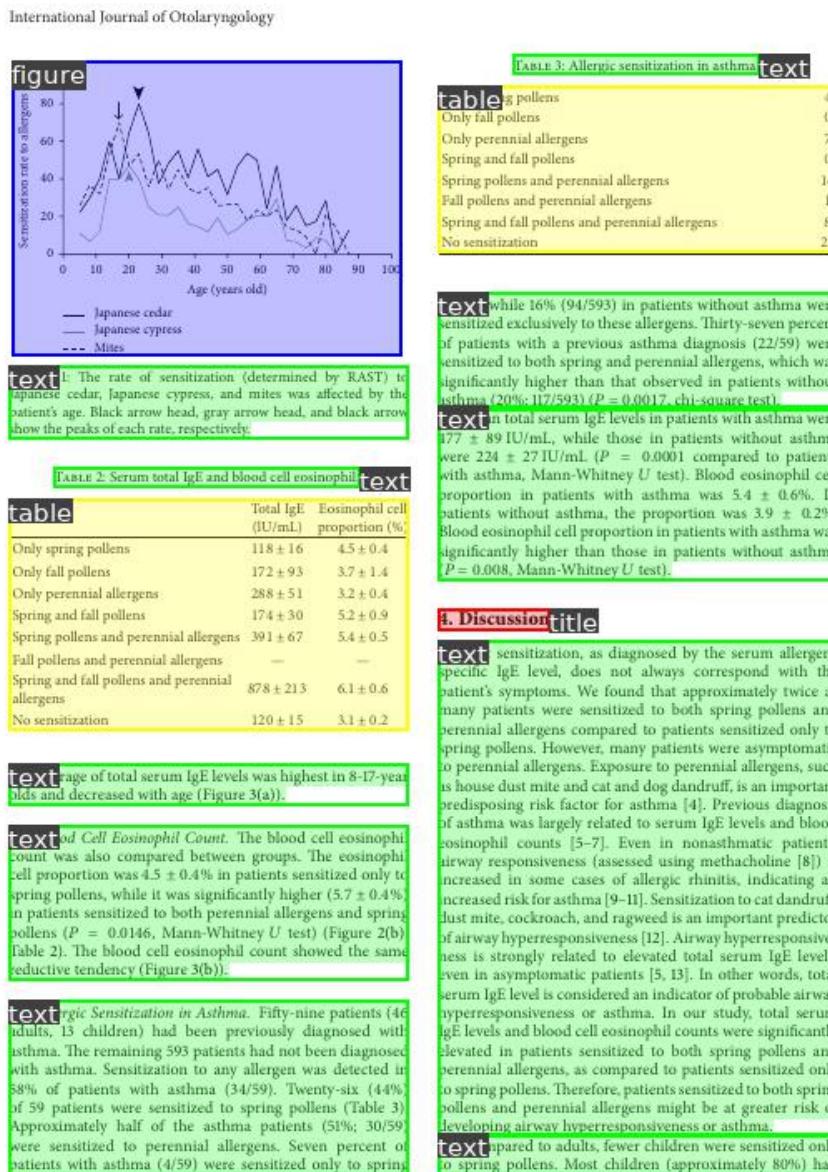
Sarkar Mausoom [3] et al. propose a priori-based segmentation mechanism to train a document segmentation model on high-resolution images, which solves the problem that structures of dense regions cannot be distinguished and merged due to the excessive shrinkage of the original image. Zhang Peng [4] et al. propose a unified framework VSR (Vision, Semantics and Relations) for document layout analysis. The framework uses a two-stream network to extract the visual and semantic features, and adaptively fuses these features through the adaptive aggregation module, going beyond the low efficiency of fusing different modules and lack of relationship modeling between layout components in the existing CV-based methods.

Datasets

Although the existing methods can solve the layout analysis task to some extent, these methods rely on a large amount of labeled training data. Recently, many datasets have been proposed to be used in document analysis.

1. PubLayNet[5]: The dataset contains 500,000 document images, of which 400,000 are used for training, 50,000 are used for verification, and 50,000 are used for testing. Tables, texts, pictures, titles, and lists are labelled in the dataset.
2. HJDataset[6]: The dataset contains 2271 document images. Besides bounding boxes and masks of the content, it also involves the hierarchical structure and reading order of layout elements.

Some samples of the PubLayNet dataset are shown in the figure below:



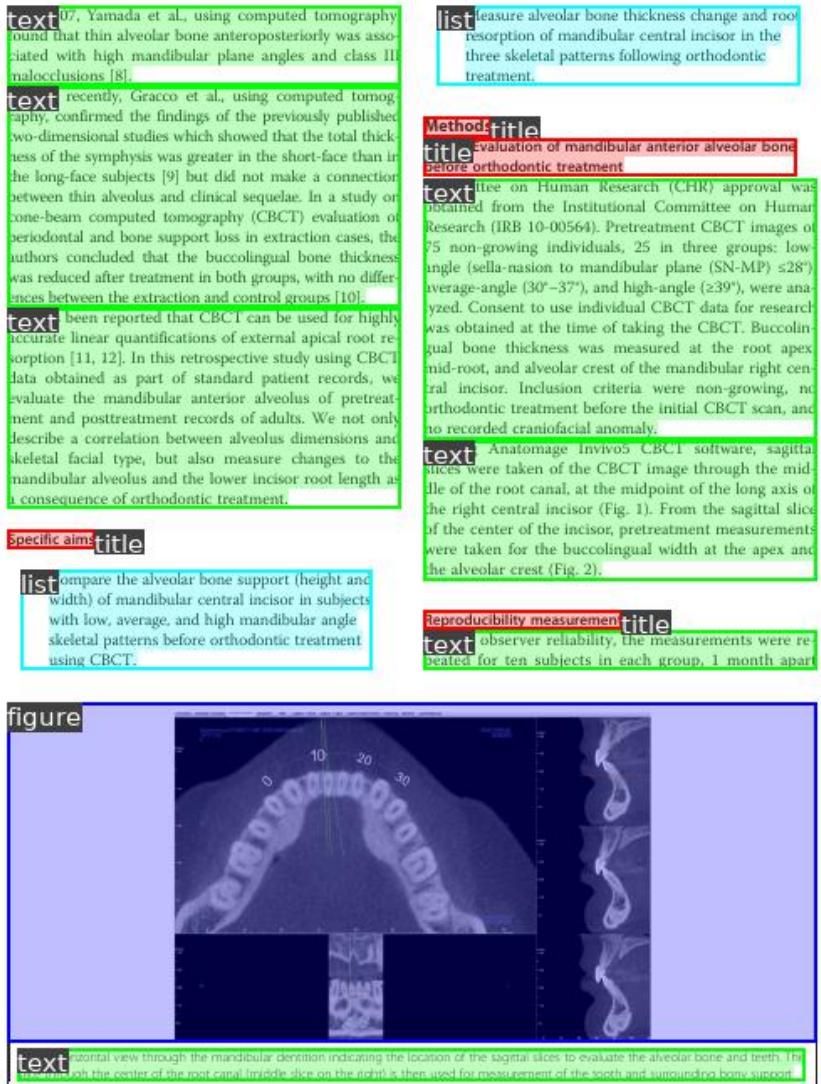


Figure 3: PubLayNet example

Reference

- [1] Soto C, Yoo S. Visual detection with context for document layout analysis[C]//Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019: 3464-3470.
- [2] Li K, Wigington C, Tensmeyer C, et al. Cross-domain document object detection: Benchmark suite and method[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 12915-12924.
- [3] Sarkar M, Aggarwal M, Jain A, et al. Document Structure Extraction using Prior based High Resolution Hierarchical

Semantic Segmentation[C]//European Conference on Computer Vision. Springer, Cham, 2020: 649-666.

[4] Zhang P, Li C, Qiao L, et al. VSR: A Unified Framework for Document Layout Analysis combining Vision, Semantics and Relations[J]. arXiv preprint arXiv:2105.06220, 2021.

[5] Zhong X, Tang J, Yepes A J. Publaynet: largest dataset ever for document layout analysis[C]//2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2019: 1015-1022.

[6] Li M, Xu Y, Cui L, et al. DocBank: A benchmark dataset for document layout analysis[J]. arXiv preprint arXiv:2006.01038, 2020.

[7] Shen Z, Zhang K, Dell M. A large dataset of historical Japanese documents with complex layouts[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020: 548-549.

8.1.2 Table Recognition

Background Introduction

Tables are common page elements in various types of documents. With the explosive growth of various documents, it becomes urgent to learn to efficiently find tables from documents and obtain their contents and structures, which is called Table Recognition. The difficulties of table recognition are summarized as follows:

1. The types and styles of tables are complex and diverse, such as *different rows and columns combined, various content types*, etc.
2. The styles of the document itself is various.
3. The lighting environment during shooting is complex, etc.

The task of table recognition is to convert table information of a document to an excel file. The task visualization is as follows:

| 代号 | 项目 | 结果 | 参考值 | 单位 |
|------|--------|--------|---------|--------|
| ALT | 谷丙转氨酶 | 25.6 | 0~40 | U/L |
| TBIL | 总胆红素 | 11.2 | <20 | umol/L |
| DBIL | 直接胆红素 | 3.3 | 0~7 | umol/L |
| IBIL | 间接胆红素 | 7.9 | 1.5~15 | umol/L |
| TP | 总蛋白 | 58.9 ↓ | 60~80 | g/L |
| ALB | 白蛋白 | 35.1 | 33~55 | g/L |
| GLO | 球蛋白 | 23.8 | 20~30 | g/L |
| A/G | 白球比 | 1.5 | 1.5~2.5 | |
| ALP | 碱性磷酸酶 | 93 | 15~112 | IU/L |
| GGT | 谷氨酰转肽酶 | 14.3 | <50 | U/L |
| AST | 谷草转氨酶 | 16.3 | 8~40 | U/L |
| LDH | 乳酸脱氢酶 | 167 | 114~240 | U/L |
| ADA | 腺苷脱氨酶 | 12.6 | 4~24 | U/L |

| 代号 | 项目 | 结果 | 参考值 | 单位 |
|------|--------|-------|---------|--------|
| ALT | 谷丙转氨酶 | 25.6 | 0~40 | U/A |
| TBIL | 总胆红素 | 11.2 | <20 | umol/ |
| DBIL | 直接胆红素 | 3.3 | 0~7 | umol/L |
| IBIL | 间接胆红素 | 7.9 | 1.5~15 | umol/L |
| TP | 总蛋白 | 58.94 | 60~80 | g/L |
| ALB | 白蛋白 | 35.1 | 33~55 | g/L |
| GLO | 球蛋白 | 23.8 | 20~30 | g/L |
| A/G | 白球比 | 1.5 | 1.5~2.5 | |
| ALP | 碱性磷酸酶 | 93 | 15~112 | IUA |
| GGT | 谷氨酰转肽酶 | 14.3 | <50 | E50 |
| AST | 谷草转氨酶 | 16.3 | 8~40 | A |
| LDH | 乳酸脱氢酶 | 167 | 114~240 | U/A |
| ADA | 腺苷脱氨酶 | 12. | 4~24 | U/L |

| Variables | Prevalence | | | |
|--------------|-------------|---------------------|------|--------------------|
| | % | (95%CI) | n | OR (95%CI) |
| Sex | male | 73.4
(69.8~77.0) | 572 | 1.06 (0.83~1.36) |
| | female | 27.2
(68.9~75.5) | 220 | 1 |
| Birthplace | Catalonia | 65.5
(62.4~68.4) | 890 | 1 |
| | Other place | 89.7
(86.6~92.8) | 378 | 4.57 (3.15~6.66) 1 |
| Habitat | urban | 71.9
(69.2~74.6) | 1070 | 1.75 (1.11~2.76) 2 |
| | rural | 77.0
(71.0~83.0) | 222 | 1.31 (0.93~1.84) |
| Social class | I-III | 66.7
(62.5~70.9) | 492 | 1 |
| | IV-V | 77.5
(74.2~80.8) | 618 | 1.72 (1.31~2.27) 1 |

| Variables | Prevalence | | | |
|--------------|-------------|---------------------|------|--------------------|
| | % | (95%CI) | n | OR (95%CI) |
| Sex | male | 53.4
(69.8~77.0) | 572 | 1.06 (0.83~1.36) |
| | female | 47.2
(68.9~75.5) | 220 | 1 |
| Birthplace | Catalonia | 65.5
(62.4~68.6) | 890 | 1 |
| | Other place | 59.7
(86.6~92.8) | 378 | 4.57 (3.15~6.66) 1 |
| Habitat | urban | 51.9
(69.2~74.6) | 1070 | 1.75 (1.11~2.76) 2 |
| | rural | 57.0
(71.0~83.0) | 222 | 1.31 (0.93~1.84) |
| Social class | I-III | 66.7
(62.5~70.9) | 492 | 1 |
| | IV-V | 77.5
(74.2~80.8) | 618 | 1.72 (1.31~2.27) 1 |

Figure 4: Example image of table recognition. The left is the original image, and the right is the result after table recognition, presented in

Existing table recognition algorithms can be divided into the following four categories according to the principle of table structure reconstruction:

1. Method based on heuristic rules
2. CNN-based method
3. GCN-based method
4. Method based on End to End

Some representative papers are divided into the above four categories, as shown in the following table:

| Category | Idea | Main papers |
|---------------------------------|---|--|
| Method based on heuristic rules | Artificially designed rules, connected domain detection analysis and processing | T-Rect, pdf2table |
| CNN-based method | target detection, semantic segmentation | CascadeTabNet, Multi-Type-TD-TSR, LGPMA, tabstruct-net, CDeC-Net, TableNet, TableSense, Deepdesrt, Deeptabstr, GTE, Cycle-CenterNet, FCN |
| GCN-based method | Consider the table recognition as a graph reconstruction problem on the basis of graph neural network | GNN, TGRNet, GraphTSR |
| Method based on End to End | Use attention mechanism | Table-Master |

Traditional Algorithm Based on Heuristic Rules

Early research on table recognition was mainly based on heuristic rules. For example, the T-Rect system proposed by Kieninger [1] et al. analyze the connected domain of document images bottom-up, merge them according to defined rules to obtain logical texts. Then, pdf2table proposed by Yildiz[2] et al. is the first method of table recognition on PDF documents, which utilizes some unique information in PDF files (such as texts, drawing paths and other information that are difficult to get in image documents) to assist with table recognition. Recently, Koci[3] et al. present the layout region in the page as a graph, and then used the Remove and Conquer (RAC) algorithm to identify the table as a subgraph.

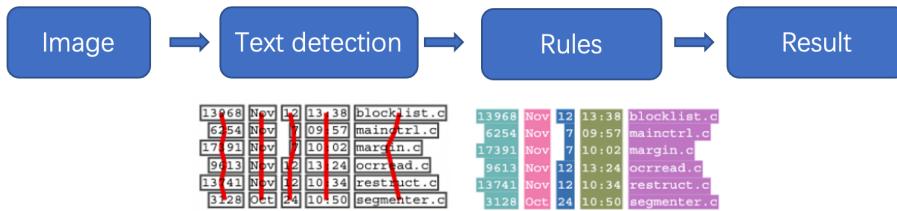


Figure 5: Schematic of heuristic algorithm

Method Based on Deep Learning CNN

With the rapid development of deep learning technology in computer vision, natural language processing, speech processing and so on, researchers have applied deep learning technology to the field of table recognition and achieved good results.

In the DeepTabStR algorithm, Siddiqui Shoaib Ahmed [12] et al. describe the table structure recognition problem as a object detection problem, and use deformable convolution to better detect table cells. Raja Sachin[6] et al. propose that TabStruct-Net combines cell detection and structure recognition visually to perform table structure recognition, which solves the problem of recognition errors due to large changes in the table layout. However, this method cannot deal with many empty cells in rows and columns.

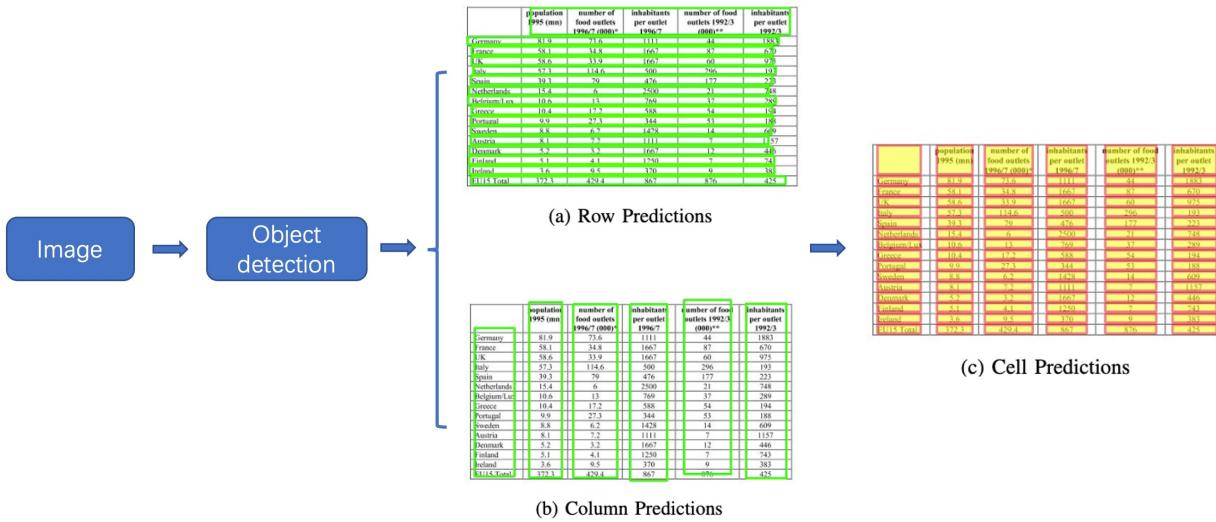


Figure 6: Schematic of algorithm based on deep learning CNN

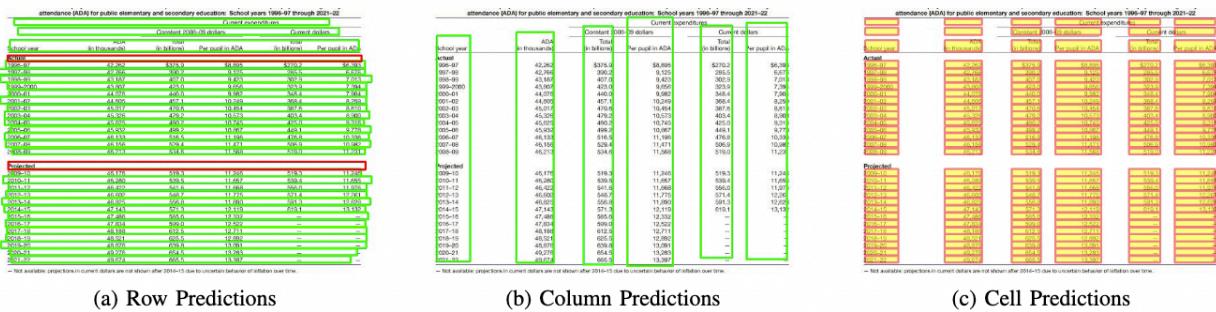


Figure 7: Example of algorithm errors based on deep learning CNN

The previous table structure recognition methods start from the elements of different granularities (row/column and text region), and it is easy to ignore the problem of merging empty cells. Qiao Liang [10] et al. propose a new framework LGPMA, which makes full use of the information from local and global features through mask re-scoring strategy, and then can obtain more reliable aligned cell region, and finally introduces table structure restoration pipelines including cell matching, empty cell searching and merging to handle the problem of table structure recognition.

In addition to the above algorithm for table recognition alone, there are also some methods that detect and recognize tables in one model. Schreiber Sebastian [11] et al. propose DeepDeSRT, which uses Faster RCNN for table detection and FCN semantic segmentation model for detecting structures, and rows and columns of tables. But it uses two independent models to finish the two tasks. Prasad Devashish [4] et al. propose an end-to-end deep learning method CascadeTabNet, which uses the Cascade Mask R-CNN HRNet model to detect tables and recognize their structures simultaneously, which goes beyond the limitations of using two independent methods for table recognition. Paliwal Shubham [8] et al. propose a novel end-to-end deep multi-task architecture TableNet, which is used for table detection and structure recognition. At the same time, additional spatial semantic features are added to TableNet during training to further improve the performance of the model. Zheng Xinyi [13] et al. propose a systematic framework GTE for table recognition, using a cell detection network to guide the training of the table detection network. Also, they put forward a hierarchical network and a new cluster-based cell structure recognition algorithm. This framework can be connected to the back of any object detection model to facilitate the training of different table recognition algorithms. Previous research mainly focus on analyzing table images with simple layouts and good alignment in scanned PDF documents. However, the tables in real scenes maybe complex and seriously distorted, curved, or covered. Therefore, Long Ruijiao [14] et al. also construct a table recognition dataset WTW in real complex scenarios, and come up with a Cycle-CenterNet method, which uses the cyclic pairing

module optimization and the proposed new pairing loss to accurately group discrete units into structured tables, and the performance of table recognition is improved.

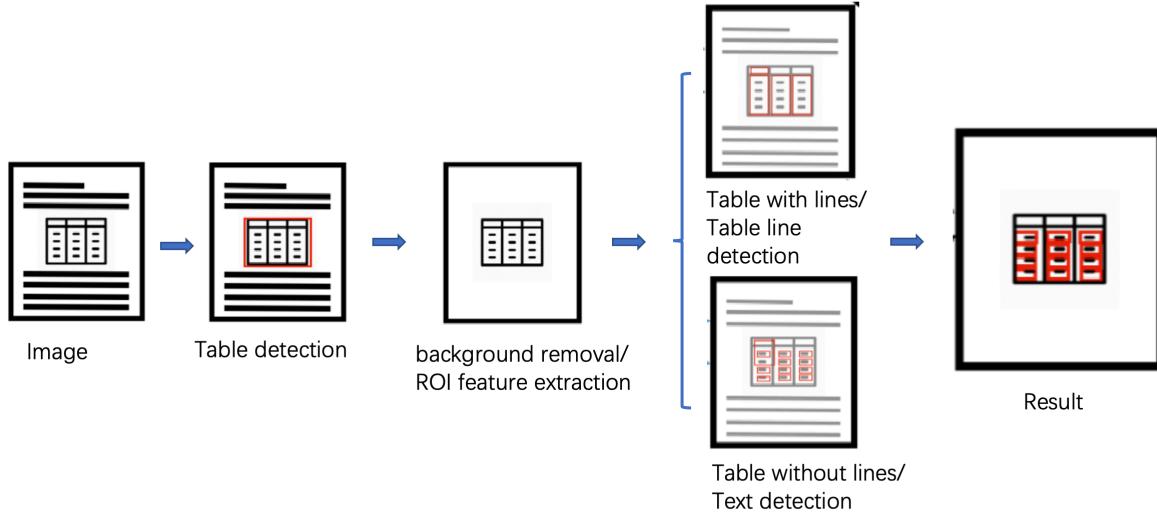


Figure 8: Schematic of end-to-end algorithm

The CNN-based method is not good at handling the tables crossing columns and rows, so there are two research methods to tackle them.

Method Based on Deep Learning GCN

In recent years, with the rise of the Graph Convolutional Network (GCN), some researchers have tried to apply GCN to the problem of table structure recognition. Qasim Shah Rukh [20] et al. consider the table structure recognition problem as a problem with compatibility of graph neural networks, and design a novel differentiable architecture that can make use of the advantage of convolutional neural networks on feature extraction and the effective interaction between the vertices of the graph neural network. But this method only uses the location features of cells, and does not use their semantic features. Chi Zewen [19] et al. propose another graph neural network, GraphTSR, for table structure recognition in PDF files. It takes table cells as input, and then uses the connection between edges and nodes of the graph to predict the relationship between cells to identify the table structure, helping to solve the recognition of cell across rows or columns. Xue Wenyuan [21] et al. reformulate the table structure recognition as table reconstruction, and propose an end-to-end method, TGRNet, which includes the cell detection branch and the cell logic location branch. These two branches predict the spatial and logical positions of different cells, tackling the problem that the previous method did not pay attention to the logical position of cells.

Diagram of GraphTSR table recognition algorithm:

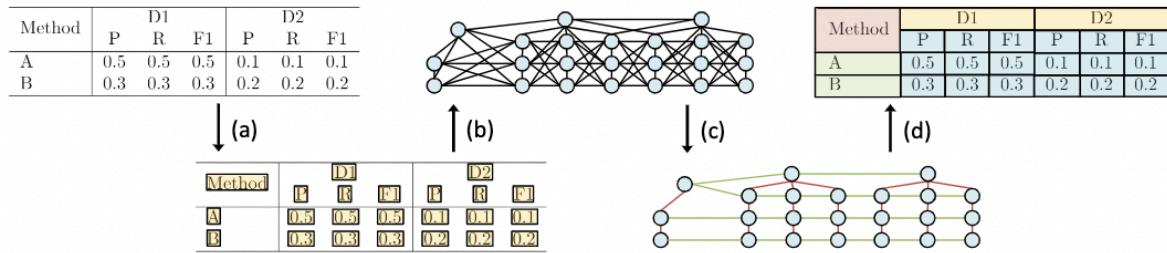
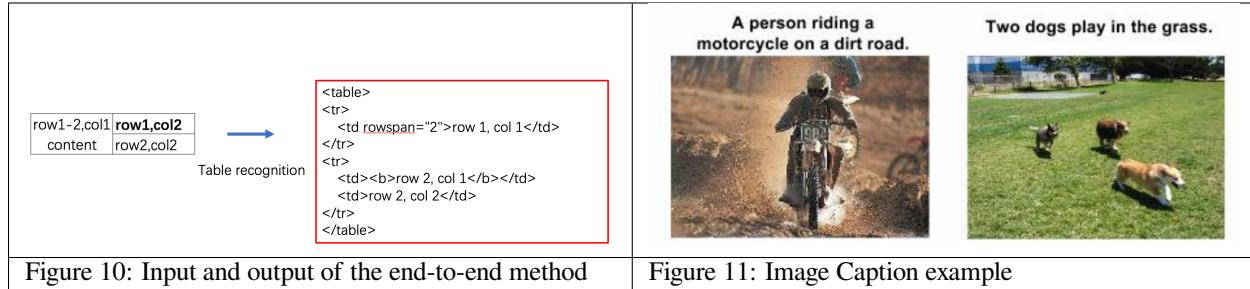


Figure 2: Overview of our method. Given a table in PDF as input, our method recognize its structure by the following four steps: (a) Pre-processing: obtaining cell contents and their corresponding bounding box from PDF; (b) Graph construction: building an undirected graph on these cells; (c) Relation prediction: predicting adjacent relations by our proposed GraphTSR; (d) Post-processing: recovering table structure from the labeled graph.

Figure 9: Diagram of GraphTSR

Method Based on End-to-End

Different from other methods using post-processing to reconstruct the table structure, the method based on the end-to-end use the network to complete the HTML representation output of the table structure



The method mainly uses Seq2Seq of Image Caption to predict the table structure, such as methods based on attention or transformer.

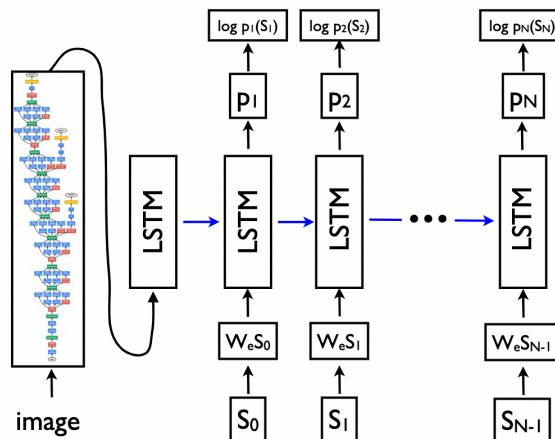


Figure 12: Schematic of Seq2Seq

Ye Jiaquan [22] gets the table structure model in the output in TableMaster by improving the Master algorithm based on Transformer. In addition, a branch is added for the coordinate regression of the box. The author did not split the model into two branches in the last layer, but decoupled the sequence prediction and the box regression after the first Transformer decoding layer. The comparison between its network structure and the original Master network is shown in the figure below:

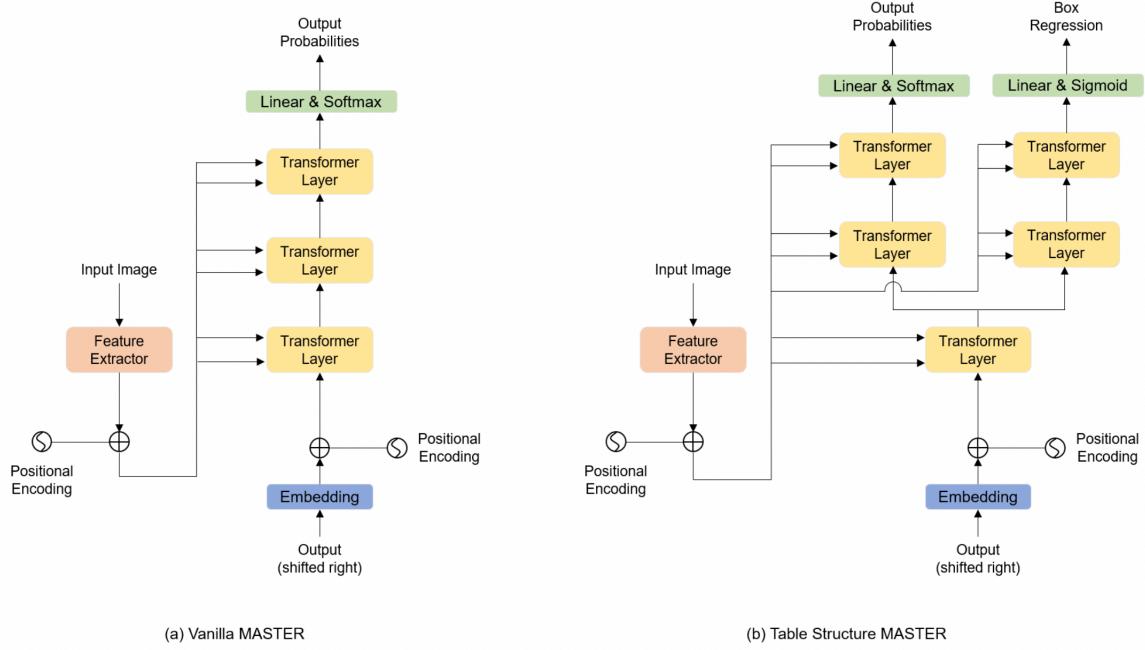


Figure 1: (a) Architecture of vanilla MASTER; (b) Architecture of table structure MASTER

Figure 13: Left: Master network, right: TableMaster network

Datasets

Since the deep learning method is data-driven, a large amount of labeled data is required to train the model. The small size of the existing datasets is also an important constraint, so some datasets have also been proposed.

1. PubTabNet[16]: It contains 568k table images and structured HTML representations.
2. PubMed Tables (PubTables-1M) [17]: A table structure recognition dataset contains highly-detailed structural annotations. 460,589 pdf images are used for table detection tasks, and 947,642 table images are used for table recognition tasks.
3. TableBank[18]: It refers to a table detection and recognition dataset using Word and Latex documents on the Internet to construct table data containing 417K high-quality annotations.
4. SciTSR[19]: It refers to a table structure recognition dataset where most of the images are converted from the paper. And those images contain 15,000 tables from PDF files and corresponding structure tags.
5. TabStructDB[12]: It includes 1081 table regions, which are marked with dense row and column information.
6. WTW[14]: It refers to large-scale dataset of scene table detection and recognition, which contains 14,581 images in total whose tables are distorted, curved, or occluded.

Data set example

Table Detection

Table 1: A cross-sectional analysis of factors associated with incidence of abdominal pain in children. Data from 1998-2000.

| Variable | Definition | Mean | SD | Range | n | Missing |
|---|---|------|-----|----------|-----|---------|
| Gender | Gender | 1.00 | .43 | .00-2.00 | 100 | 0 |
| Age (months) | Age (months) | 10.7 | 4.1 | 0-24 | 100 | 0 |
| Sex ratio | Sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Number of children | Number of children | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size | Family size | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family income | Family income | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income | Family size × income | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × gender | Family size × income × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio | Family size × income × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × number of children | Family size × income × number of children | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × family size | Family size × income × family size | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × gender | Family size × income × sex ratio × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children | Family size × income × sex ratio × number of children | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × family size | Family size × income × sex ratio × family size | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × sex ratio | Family size × income × sex ratio × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × gender | Family size × income × sex ratio × number of children × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × sex ratio | Family size × income × sex ratio × number of children × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × family size | Family size × income × sex ratio × number of children × family size | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × sex ratio × gender | Family size × income × sex ratio × number of children × sex ratio × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × sex ratio × sex ratio | Family size × income × sex ratio × number of children × sex ratio × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × family size × gender | Family size × income × sex ratio × number of children × family size × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × family size × sex ratio | Family size × income × sex ratio × number of children × family size × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × family size × sex ratio × gender | Family size × income × sex ratio × number of children × family size × sex ratio × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × family size × sex ratio × sex ratio | Family size × income × sex ratio × number of children × family size × sex ratio × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × family size × sex ratio × sex ratio × gender | Family size × income × sex ratio × number of children × family size × sex ratio × sex ratio × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × family size × sex ratio × sex ratio × sex ratio | Family size × income × sex ratio × number of children × family size × sex ratio × sex ratio × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |

Table Structure Recognition

Table 1: A cross-sectional analysis of factors associated with incidence of abdominal pain in children. Data from 1998-2000.

| Variable | Definition | Mean | SD | Range | n | Missing |
|---|---|------|-----|----------|-----|---------|
| Gender | Gender | 1.00 | .43 | .00-2.00 | 100 | 0 |
| Age (months) | Age (months) | 10.7 | 4.1 | 0-24 | 100 | 0 |
| Sex ratio | Sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Number of children | Number of children | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size | Family size | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family income | Family income | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income | Family size × income | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × gender | Family size × income × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio | Family size × income × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × number of children | Family size × income × number of children | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × gender | Family size × income × sex ratio × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children | Family size × income × sex ratio × number of children | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × family size | Family size × income × sex ratio × family size | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × sex ratio | Family size × income × sex ratio × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × gender | Family size × income × sex ratio × number of children × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × sex ratio | Family size × income × sex ratio × number of children × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × family size | Family size × income × sex ratio × number of children × family size | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × sex ratio × gender | Family size × income × sex ratio × number of children × sex ratio × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × sex ratio × sex ratio | Family size × income × sex ratio × number of children × sex ratio × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × family size × gender | Family size × income × sex ratio × number of children × family size × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × family size × sex ratio | Family size × income × sex ratio × number of children × family size × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × family size × sex ratio × gender | Family size × income × sex ratio × number of children × family size × sex ratio × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × family size × sex ratio × sex ratio | Family size × income × sex ratio × number of children × family size × sex ratio × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |

Table Functional Analysis

Table 1: A cross-sectional analysis of factors associated with incidence of abdominal pain in children. Data from 1998-2000.

| Variable | Definition | Mean | SD | Range | n | Missing |
|---|---|------|-----|----------|-----|---------|
| Gender | Gender | 1.00 | .43 | .00-2.00 | 100 | 0 |
| Age (months) | Age (months) | 10.7 | 4.1 | 0-24 | 100 | 0 |
| Sex ratio | Sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Number of children | Number of children | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size | Family size | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family income | Family income | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income | Family size × income | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × gender | Family size × income × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio | Family size × income × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × number of children | Family size × income × number of children | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × gender | Family size × income × sex ratio × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children | Family size × income × sex ratio × number of children | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × family size | Family size × income × sex ratio × family size | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × sex ratio | Family size × income × sex ratio × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × gender | Family size × income × sex ratio × number of children × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × sex ratio | Family size × income × sex ratio × number of children × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × family size | Family size × income × sex ratio × number of children × family size | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × sex ratio × gender | Family size × income × sex ratio × number of children × sex ratio × gender | 1.00 | .00 | .00-1.00 | 100 | 0 |
| Family size × income × sex ratio × number of children × sex ratio × sex ratio | Family size × income × sex ratio × number of children × sex ratio × sex ratio | 1.00 | .00 | .00-1.00 | 100 | 0 |

Figure 2: Illustration of the three subtasks of table extraction addressed by the PubTables-1M dataset.

Figure 14: Sample of PubTables-1M dataset



Figure 15: Sample of WTW data set

Reference

- [1] Kieninger T, Dengel A. A paper-to-HTML table converting system[C]//Proceedings of document analysis systems (DAS). 1998, 98: 356-365.
- [2] Yildiz B, Kaiser K, Miksch S. pdf2table: A method to extract table information from pdf files[C]//IICAI. 2005: 1773-1785.

- [3] Koci E, Thiele M, Lehner W, et al. Table recognition in spreadsheets via a graph representation[C]//2018 13th IAPR International Workshop on Document Analysis Systems (DAS). IEEE, 2018: 139-144.
- [4] Prasad D, Gadpal A, Kapadni K, et al. CascadeTabNet: An approach for end to end table detection and structure recognition from image-based documents[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020: 572-573.
- [5] Fischer P, Smajic A, Abrami G, et al. Multi-Type-TD-TSR—Extracting Tables from Document Images Using a Multi-stage Pipeline for Table Detection and Table Structure Recognition: From OCR to Structured Table Representations[C]//German Conference on Artificial Intelligence (Künstliche Intelligenz). Springer, Cham, 2021: 95-108.
- [6] Raja S, Mondal A, Jawahar C V. Table structure recognition using top-down and bottom-up cues[C]//European Conference on Computer Vision. Springer, Cham, 2020: 70-86.
- [7] Agarwal M, Mondal A, Jawahar C V. Cdec-net: Composite deformable cascade network for table detection in document images[C]//2020 25th International Conference on Pattern Recognition (ICPR). IEEE, 2021: 9491-9498.
- [8] Paliwal S S, Vishwanath D, Rahul R, et al. Tablenet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images[C]//2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2019: 128-133.
- [9] Dong H, Liu S, Han S, et al. Tablesense: Spreadsheet table detection with convolutional neural networks[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2019, 33(01): 69-76.
- [10] Qiao L, Li Z, Cheng Z, et al. LGPMA: Complicated Table Structure Recognition with Local and Global Pyramid Mask Alignment[J]. arXiv preprint arXiv:2105.06224, 2021.
- [11] Schreiber S, Agne S, Wolf I, et al. Deepdesrt: Deep learning for detection and structure recognition of tables in document images[C]//2017 14th IAPR international conference on document analysis and recognition (ICDAR). IEEE, 2017, 1: 1162-1167.
- [12] Siddiqui S A, Fateh I A, Rizvi S T R, et al. Deeptabstr: Deep learning based table structure recognition[C]//2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2019: 1403-1409.
- [13] Zheng X, Burdick D, Popa L, et al. Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context[C]//Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2021: 697-706.
- [14] Long R, Wang W, Xue N, et al. Parsing Table Structures in the Wild[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 944-952.
- [15] Siddiqui S A, Khan P I, Dengel A, et al. Rethinking semantic segmentation for table structure recognition in documents[C]//2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2019: 1397-1402.
- [16] Zhong X, ShafieiBavani E, Jimeno Yepes A. Image-based table recognition: data, model, and evaluation[C]//Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16. Springer International Publishing, 2020: 564-580.
- [17] Smock B, Pesala R, Abraham R. PubTables-1M: Towards a universal dataset and metrics for training and evaluating table extraction models[J]. arXiv preprint arXiv:2110.00061, 2021.
- [18] Li M, Cui L, Huang S, et al. Tablebank: Table benchmark for image-based table detection and recognition[C]//Proceedings of the 12th Language Resources and Evaluation Conference. 2020: 1918-1925.
- [19] Chi Z, Huang H, Xu H D, et al. Complicated table structure recognition[J]. arXiv preprint arXiv:1908.04729, 2019.
- [20] Qasim S R, Mahmood H, Shafait F. Rethinking table recognition using graph neural networks[C]//2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2019: 142-147.
- [21] Xue W, Yu B, Wang W, et al. TGRNet: A Table Graph Reconstruction Network for Table Structure Recognition[J]. arXiv preprint arXiv:2106.10598, 2021.

[22] Ye J, Qi X, He Y, et al. PingAn-VCGroup's Solution for ICDAR 2021 Competition on Scientific Literature Parsing Task B: Table Recognition to HTML[J]. arXiv preprint arXiv:2105.01848, 2021.

8.1.3 Document VQA

The boss sends a task: to develop an ID card recognition system



How to choose a plan:

1. Use rules to extract information after text detection
 2. Use scale types to extract information after text detection
 3. Resort to outsourcing

Background Introduction

In the VQA (Visual Question Answering) task, questions and answers mainly focus on the content of the image. But since the core information of text images is the text, VQA can be divided into two types: Text-VQA for natural scenes and DocVQA for scanned texts. The relationship between the three is shown in the figure below.

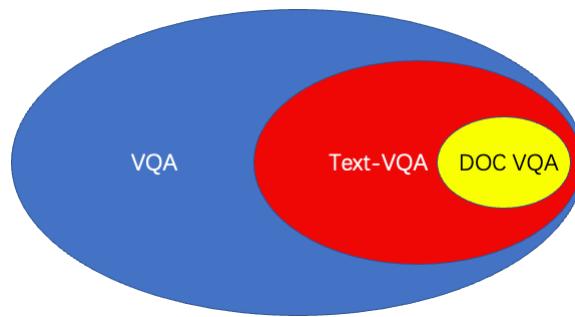
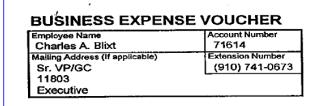


Figure 16: The hierarchy of VQA

The sample pictures of VQA, Text-VQA and DocVQA are shown in the figure below.

| Task type | VQA | Text-VQA | DocVQA |
|------------------|---|---|--|
| Task description | Ask questions regarding picture content | Ask questions regarding texts on pictures | Ask questions regarding texts of document images |
| Sample picture |  <p>Question: What can the red object on the ground be used for ?
 Answer: Firefighting
 Support Fact: Fire hydrant can be used for fighting fires.</p> |  <p>TextVQA: Q: what is the name of the store?
 A: tellus mater inc.</p> |  <p>Q: What is the Extension Number as per the voucher?
 GT: (910) 741-0673</p> |

Because DocVQA is closer to actual application scenarios, it is more widely used in academic research and industrial scenarios. Usually, the questions asked in DocVQA are fixed. For example, the questions in the ID card scenario are generally:

1. What is the ID number?
2. What is your name?
3. What ethnic group are you from ?



Figure 17: Example of an ID card

Based on this prior knowledge, research on DocVQA has begun to focus on the Key Information Extraction (KIE) task. Here we mainly discuss the KIE-related research. The KIE mainly extracts the key information needed from the image, such as the name and the ID number in the ID card.

KIE is usually divided into two sub-tasks for research

1. SER: It refers to Semantic Entity Recognition, which is to classify each detected text, such as mark it as the name of the ID card . The example is shown in the picture on the left.
2. RE: It refers to Relation Extraction, which classifies each detected text, such as into questions and answers. Then it finds the corresponding answer to each question. As shown in the figure below, the red and black boxes represent the question and the answer, respectively, and the yellow line represents the relation between the question and the answer.



Figure 18: Examples of SER and RE

The general KIE method is researched based on Named Entity Recognition (NER) [4], but this kind of methods only use the text information in the image and lack visual and structural information, so they are not so accurate. So recently, visual and structural information have begun to be integrated together. According to their principles used in the fusion of multimodal information, these methods can be divided into the following three types:

1. Grid-based method
2. Token-based method
3. GCN-based method
4. Based on the End to End method

Some representative papers are included in the above three categories:

| Category | Ideas | Main Papers |
|----------------------------|--|----------------------------------|
| Grid-based method | Fusing multi-modal information on images (texts, layouts, images) | Chargrid |
| Token-based method | Using methods such as Bert for multi-modal information fusion | LayoutLM, LayoutLMv2, StrucText, |
| GCN-based method | Using the graph network structure for multi-modal information fusion | GCN, PICK, SDMG-R, SERA |
| Method based on End-to-End | Unifying OCR and key information extraction into one network | Trie |

Grid-Based Method

The Grid-based method performs multimodal information fusion in images. Chargrid[5] firstly detects and recognizes the characters in images, constructs the network input by filling the one-hot code into the corresponding character regions (the non-black part in the right image below), and make the input pass through the CNN network of the encoder-decoder structure to perform coordinate detection and classification of key information.

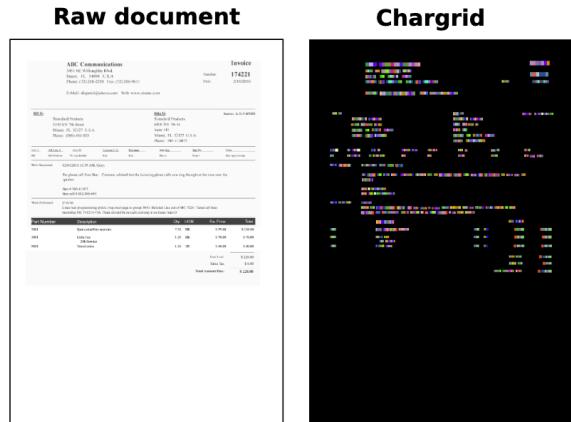


Figure 1: Example for a document page (left) and corresponding chargrid representation g (right).

Figure 19: Example of Chargrid data

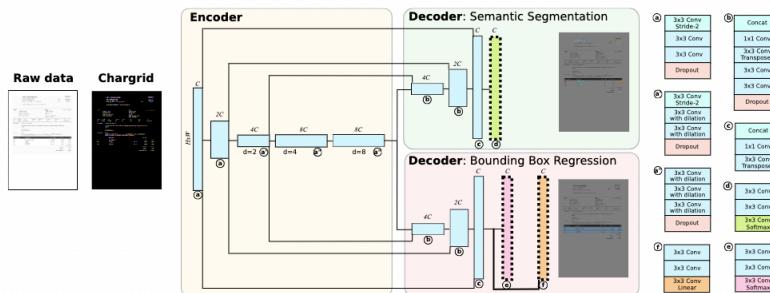


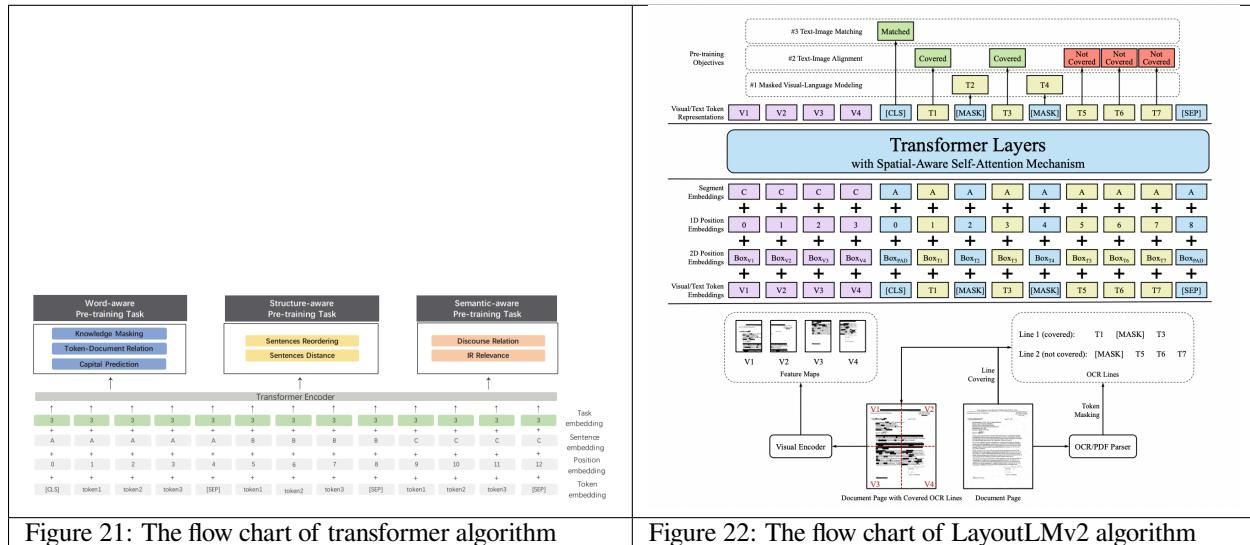
Figure 2: Network architecture for document understanding, the *chargrid-net*. Each convolutional block in the network is represented as a box. The height of a box is a proxy for feature map resolution while the width is a proxy for the number of output channels. C corresponds to the number of ‘base’ channels, which in turns corresponds to the number of output channels in the first encoder block. d denotes dilation rate.

Figure 20: Chargrid network

Compared with traditional methods based only on text, this method can use both text information and structural information, so its accuracy can be higher. But it doesn't organically combine the two.

Token-Based Method

LayoutLM[6] encodes 2D position information and text information together into the BERT model, and draws on the pre-training of Bert in NLP to pre-train large-scale datasets. In downstream tasks, LayoutLM also introduces image information to improve the performance of the model. Although LayoutLM combines text, location and image information, the image information is fused in the training of downstream tasks, which makes the multi-modal combination not so sufficient. Based on LayoutLM, LayoutLMv2 [7] integrates image information with text and layout information in the pre-training stage through transformers, and also adds a spatial perception self-attention mechanism to the Transformer to assist the model in the integration of visual and text features. Although LayoutLMv2 fuses text, location and image information in the pre-training stage, the visual features learned by the model are not fine enough due to the limitation of the pre-training task. So StrucText [8] proposes two new tasks, Sentence Length Prediction (SLP) and Paired Boxes Direction (PBD) in the pre-training to help the network learn fine visual features. The SLP task makes the model learn the length of the text segment, and the PBD task allows the model to learn the matching relationship between box directions. In this way, the deep cross-modal fusion between text, visual and layout information can be accelerated.



GCN-Based Method

Although the existing GCN-based methods [10] make use of text and structure information, but do not make good use of image information. PICK [11] adds image information to the GCN network and proposes a graph learning module to automatically learn edge types. SDMG-R [12] encodes the image as a bimodal graph. The nodes of the graph are visual and textual information of the text region. The edges represent the direct spatial relationship between adjacent texts. By iteratively spreading information along the edges and inferring graph node categories, SDMG -R solves the problem that existing methods fail to handle novel templates.

The PICK flow chart is shown in the figure below:

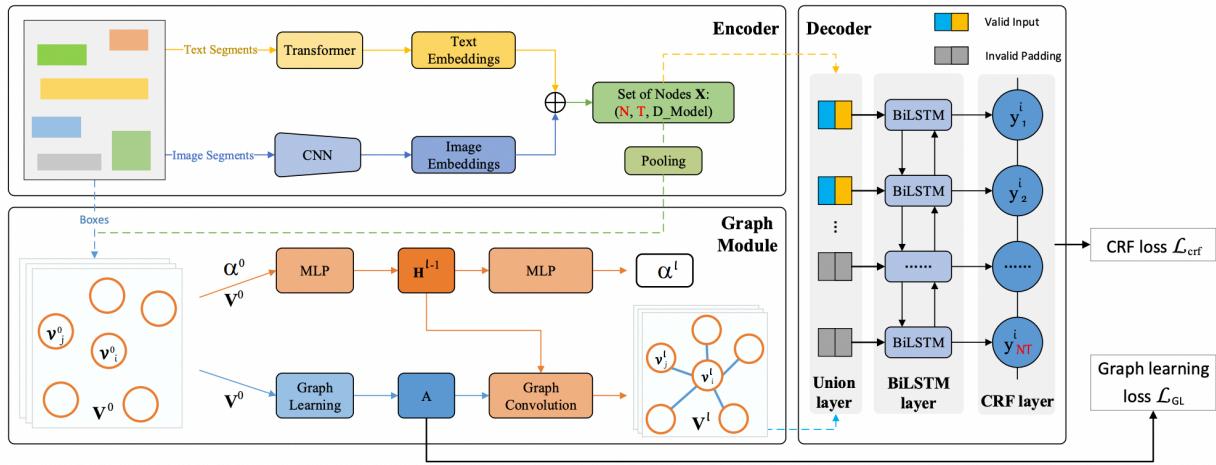


Figure 23: The flow chart of PICK algorithm

SERA[10] introduces the biaffine parser in dependency syntax analysis to the document relation extraction, and GCN is used to fuse text and visual information.

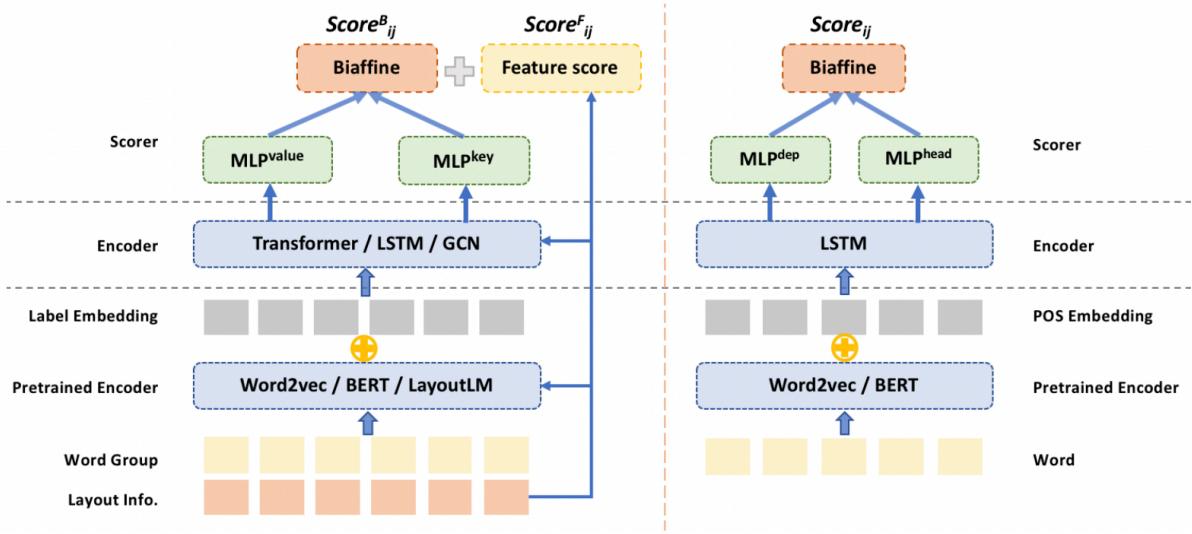


Figure 24: The flow chart of SERA algorithm

Method Based on End-to-End

Existing methods divide KIE into two independent tasks: text reading and information extraction. However, they just focus on improving the task of information extraction, ignoring that text reading and information extraction are interrelated. Therefore, Trie [9] proposes a unified end-to-end network where the two tasks can be learned at the same time and reinforce each other in the process.

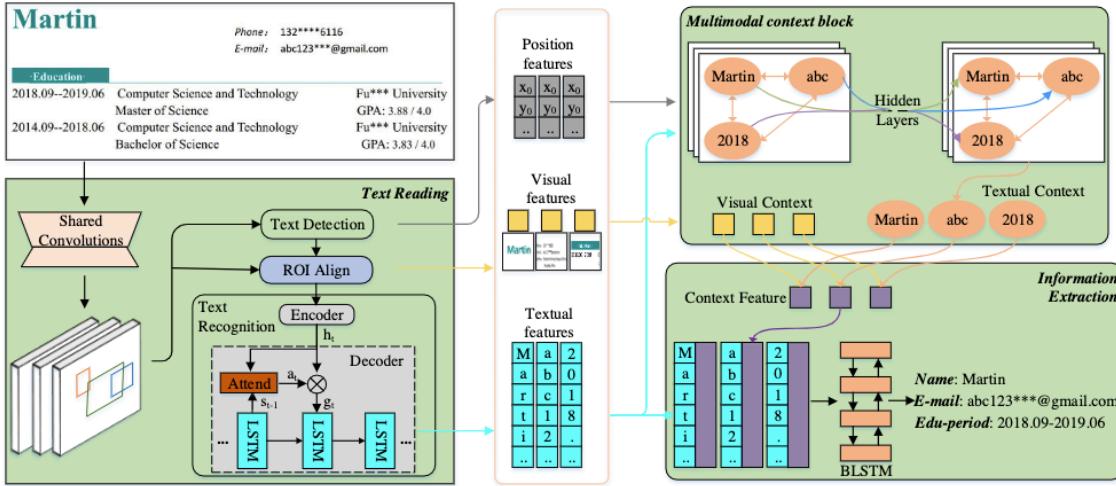


Figure 3: Overall architecture. The network predicts text regions, text content and extract entities of interest in a single forward pass.

Figure 25: The flow chart of Trie algorithm

Datasets

The datasets used for KIE mainly include:

1. SROIE: Its task3 [2] aims to extract four pieces of predefined information from the scanned receipt: company, date, address or the total number. There are 626 samples in the datasets for training and 347 samples for testing.
2. FUNSD: FUNSD dataset [3] is used to grasp table information from scanned documents. It contains 199 marked forms from real scenarios. 149 of the 199 samples are used for training and 50 for testing. The FUNSD dataset assigns a semantic tag to each word: question, answer, title or other.
3. XFUN: The XFUN dataset is a multilingual dataset proposed by Microsoft. It contains 7 languages and each language has 149 training sets and 50 test sets.

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---------------|------|-----------|------|------------------|--|----------|--------|-------|--------|-------------------|---------------|---|-------------|-----------|----------|-----|---------|-----|------|--|-------------|-----|---------|-----|-----|--|------|-----|--------|-----|-----|--|-------|-----|-------|-----|-------|-----|-----------|-----|-----------|--|------------|-----|-----------|-----|
| <p>STARBUCKS Store #10208
11302 Euclid Avenue
Cleveland, OH (216) 229-0749</p> <p>CHK 664290
12/07/2014 06:43 PM
1912003 Drawer: 2. Reg: 2</p> <table border="1"> <tbody> <tr><td>Vt Pep Mocha</td><td>4.95</td></tr> <tr><td>Sbux Card</td><td>4.95</td></tr> <tr><td>XXXXXXXXXXXX3228</td><td></td></tr> <tr><td>Subtotal</td><td>\$4.95</td></tr> <tr><td>Total</td><td>\$4.95</td></tr> <tr><td>Change Due</td><td>\$0.00</td></tr> </tbody> </table> <p>Check Closed
12/07/2014 06:43 PM</p> <p>SBUX Card x3228 New Balance: 37.45
Card is registered.</p> | Vt Pep Mocha | 4.95 | Sbux Card | 4.95 | XXXXXXXXXXXX3228 | | Subtotal | \$4.95 | Total | \$4.95 | Change Due | \$0.00 | <p>MR 1999 (3-69) 100
BROWN & WILLIAMS TOBACCO CORPORATION
FILTER SCORES</p> <p>Brand: RALSIGH (BELAIR portion not tested) Project #: 1969-105
Commercial: LAKE - NEW PACK :40 (with BELAIR Nadine ton 120) Sample: 316
Code #: BM-R2-69-98 PMS Base: (234)
Supplier: AUDIENCE STUDIES</p> <p>TEST DATA</p> <table border="1"> <tbody> <tr><td>L. Angeles:</td><td>8/5 and 6</td></tr> <tr><td>Chicago:</td><td>8/8</td></tr> </tbody> </table> <p>PMS SCORES</p> <table border="1"> <tbody> <tr><td>Overall</td><td>1.7</td></tr> <tr><td>CITY</td><td></td></tr> <tr><td>Los Angeles</td><td>0.0</td></tr> <tr><td>Chicago</td><td>3.3</td></tr> <tr><td>SEX</td><td></td></tr> <tr><td>Male</td><td>0.0</td></tr> <tr><td>Female</td><td>3.3</td></tr> <tr><td>AGE</td><td></td></tr> <tr><td>18-25</td><td>0.0</td></tr> <tr><td>26-35</td><td>0.0</td></tr> <tr><td>36-45</td><td>0.0</td></tr> <tr><td>46 & Over</td><td>9.3</td></tr> <tr><td>EDUCATION</td><td></td></tr> <tr><td>35 & Under</td><td>0.0</td></tr> <tr><td>36 & Over</td><td>5.0</td></tr> </tbody> </table> <p>This commercial was tested in color.</p> <p>465607116 P</p> | L. Angeles: | 8/5 and 6 | Chicago: | 8/8 | Overall | 1.7 | CITY | | Los Angeles | 0.0 | Chicago | 3.3 | SEX | | Male | 0.0 | Female | 3.3 | AGE | | 18-25 | 0.0 | 26-35 | 0.0 | 36-45 | 0.0 | 46 & Over | 9.3 | EDUCATION | | 35 & Under | 0.0 | 36 & Over | 5.0 |
| Vt Pep Mocha | 4.95 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Sbux Card | 4.95 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| XXXXXXXXXXXX3228 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Subtotal | \$4.95 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Total | \$4.95 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Change Due | \$0.00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| L. Angeles: | 8/5 and 6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Chicago: | 8/8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Overall | 1.7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CITY | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Los Angeles | 0.0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Chicago | 3.3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SEX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Male | 0.0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Female | 3.3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AGE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 18-25 | 0.0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 26-35 | 0.0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 36-45 | 0.0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 46 & Over | 9.3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EDUCATION | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 35 & Under | 0.0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 36 & Over | 5.0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Reference

- [1] Mathew M, Karatzas D, Jawahar C V. Docvqa: A dataset for vqa on document images[C]//Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2021: 2200-2209.
- [2] Huang Z, Chen K, He J, et al. Icdar2019 competition on scanned receipt ocr and information extraction[C]//2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2019: 1516-1520.
- [3] Jaume G, Ekenel H K, Thiran J P. Funsd: A dataset for form understanding in noisy scanned documents[C]//2019 International Conference on Document Analysis and Recognition Workshops (ICDARW). IEEE, 2019, 2: 1-6.
- [4] Lample G, Ballesteros M, Subramanian S, et al. Neural architectures for named entity recognition[J]. arXiv preprint arXiv:1603.01360, 2016.
- [5] Katti A R, Reisswig C, Guder C, et al. Chargrid: Towards understanding 2d documents[J]. arXiv preprint arXiv:1809.08799, 2018.
- [6] Xu Y, Li M, Cui L, et al. Layoutlm: Pre-training of text and layout for document image understanding[C]//Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020: 1192-1200.
- [7] Xu Y, Xu Y, Lv T, et al. LayoutLMv2: Multi-modal pre-training for visually-rich document understanding[J]. arXiv preprint arXiv:2012.14740, 2020.
- [8] Li Y, Qian Y, Yu Y, et al. StrucTextT: Structured Text Understanding with Multi-Modal Transformers[C]//Proceedings of the 29th ACM International Conference on Multimedia. 2021: 1912-1920.
- [9] Zhang P, Xu Y, Cheng Z, et al. Trie: End-to-end text reading and information extraction for document understanding[C]//Proceedings of the 28th ACM International Conference on Multimedia. 2020: 1413-1422.
- [10] Liu X, Gao F, Zhang Q, et al. Graph convolution for multimodal information extraction from visually rich documents[J]. arXiv preprint arXiv:1903.11279, 2019.

[11] Yu W, Lu N, Qi X, et al. Pick: Processing key information extraction from documents using improved graph learning-convolutional networks[C]//2020 25th International Conference on Pattern Recognition (ICPR). IEEE, 2021: 4363-4370.

[12] Sun H, Kuang Z, Yue X, et al. Spatial Dual-Modality Graph Reasoning for Key Information Extraction[J]. arXiv preprint arXiv:2103.14470, 2021.

8.2 OCR Table Recognition Practice

This section will introduce how to use PaddleOCR to train and run the table recognition algorithm, including:

1. Understanding the principle of the table recognition algorithm
2. Mastering the training and prediction of PaddleOCR table recognition code

8.2.1 Quick Start

To quickly demonstrate the PP-Structure prediction, first download the PaddleOCR code and install dependency packages.

```
# clone PaddleOCR code
! git clone -b release/2.4 https://gitee.com/paddlepaddle/PaddleOCR

# Install dependencies
! pip install -U https://paddleocr.bj.bcebos.com/whl/layoutparser-0.0.0-py3-none-any.whl
! pip install -r PaddleOCR/requirements.txt
! pip install pandas
```

After the installation, quickly complete the table recognition through the following command:

```
# Switch to working directory
import os
os.chdir('PaddleOCR/ppstructure')

# Download the model
! mkdir inference && cd inference
# Download the detection model of the ultra-lightweight table English OCR model and unzip it
! wget -P ./inference/ https://paddleocr.bj.bcebos.com/PP-OCRV2/chinese/ch_PP-OCRV2_det_infer.tar && cd inference && tar xf ch_PP-OCRV2_det_infer.tar && cd ..
# Download the recognition model of the ultra-lightweight table English OCR model and unzip it
! wget -P ./inference/ https://paddleocr.bj.bcebos.com/PP-OCRV2/chinese/ch_PP-OCRV2_rec_infer.tar && cd ..
# Download the ultra-lightweight English table inch model and unzip it
! wget -P ./inference/ https://paddleocr.bj.bcebos.com/dygraph_v2.0/table/en_ppocr_mobile_v2.0_table_structure_infer.tar && cd inference && tar xf en_ppocr_mobile_v2.0_table_structure_infer.tar && cd ..

# Read the image and display it
import cv2
from matplotlib import pyplot as plt
%matplotlib inline
```

(continues on next page)

(continued from previous page)

```
img = cv2.imread('../doc/table/table.jpg')
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f87db2672d0>
```

| Methods | R | P | F | FPS |
|-----------------------|--------------|--------------|--------------|-------|
| SegLink [26] | 70.0 | 86.0 | 77.0 | 8.9 |
| PixelLink [4] | 73.2 | 83.0 | 77.8 | - |
| TextSnake [18] | 73.9 | 83.2 | 78.3 | 1.1 |
| TextField [37] | 75.9 | 87.4 | 81.3 | 5.2 |
| MSR[38] | 76.7 | 87.4 | 81.7 | - |
| FTSN [3] | 77.1 | 87.6 | 82.0 | - |
| LSE[30] | 81.7 | 84.2 | 82.9 | - |
| CRAFT [2] | 78.2 | 88.2 | 82.9 | 8.6 |
| MCN [16] | 79 | 88 | 83 | - |
| ATRR[35] | 82.1 | 85.2 | 83.6 | - |
| PAN [34] | 83.8 | 84.4 | 84.1 | 30.2 |
| DB[12] | 79.2 | 91.5 | 84.9 | 32.0 |
| DRRG [41] | 82.30 | 88.05 | 85.08 | - |
| Ours (SynText) | 80.68 | 85.40 | 82.97 | 12.68 |
| Ours (MLT-17) | 84.54 | 86.62 | 85.57 | 12.31 |

```
# https://github.com/PaddlePaddle/PaddleOCR/blob/dygraph/ppstructure/table/predict_


```

(continues on next page)

(continued from previous page)

```
from IPython.core.display import display, HTML
display(HTML(pred_html))
```

```
[2022/03/17 21:27:08] root DEBUG: dt_boxes num : 78, elapse : 0.5478317737579346
[2022/03/17 21:27:10] root DEBUG: rec_res num : 78, elapse : 1.2926230430603027
```

```
<IPython.core.display.HTML object>
```

```
# Read the excel and display it
import pandas as pd
df = pd.read_excel('1.xlsx').fillna('')
print(df)
```

| | Methods | R | P | F | FPS |
|----|----------------|-------|-------|-------|-------|
| 0 | SegLink I26] | 70.00 | 86.00 | 77.00 | 8.9 |
| 1 | PixelLink [4j | 73.20 | 83.00 | 77.80 | : |
| 2 | TextSnake [18J | 73.90 | 83.20 | 78.30 | 1.1 |
| 3 | TextField [37] | 75.90 | 87.40 | 81.30 | 5.2 |
| 4 | MSR[38] | 76.70 | 87.40 | 81.70 | 1 |
| 5 | FTSN 13j | 77.10 | 87.60 | 82.00 | : |
| 6 | LSEI30] | 81.70 | 84.20 | 82.90 | |
| 7 | CRAFT I2] | 78.20 | 88.20 | 82.90 | 8.6 |
| 8 | MCN [16] | 79.00 | 88.00 | 83.00 | : |
| 9 | ATRRRI35] | 82.10 | 85.20 | 83.60 | |
| 10 | PAN [34] | 83.80 | 84.40 | 84.10 | 30.2 |
| 11 | DBI2J | 79.20 | 91.50 | 84.90 | 32.0 |
| 12 | DRRG141] | 82.30 | 88.05 | 85.08 | : |
| 13 | OURS (SYNTexT) | 80.68 | 85.40 | 82.97 | 12.68 |
| 14 | Ours (MLT-17) | 84.54 | 86.62 | 85.57 | 12.31 |

8.2.2 Explanation of Prediction Principle:

Introduction to the Overall Pipeline

PP-Structure's table recognition model algorithm is an end-to-end algorithm.

The table recognition algorithm consists of three models:

1. Text detection model: It is used to detect the single line text in the table.
2. Text recognition model: It is used to recognize the detected text.
3. Table structure and cell prediction model: It is used to predict HTML information of the table structure and cell coordinates.

The pipeline of table recognition algorithm is shown in the figure below:

The process is:

1. Use the text detection model to detect the single line text in the table;
2. Use the text recognition model to recognize the detected text. At this step, we get the text box and text information;
3. Use table structure and cell prediction model to predict HTML information of the table structure and cell coordinates;

4. Aggregate the text box in 1 and the cell coordinates in 3, as shown in the figure below. Determine if aggregation is needed according to the IOU between the red text detection boxes and the blue cell coordinate detection boxes.
5. After the text box aggregation, sort the text boxes from top to bottom and from left to right. You can get the corresponding text information with the index of the sorted text boxes, and then make string splicing to get the text content of the cells.

Introduction to Table Structure Inference Model

Table recognition requires three models: text detection, text recognition and table structure recognition models. The text detection and recognition models have been introduced, and here the table structure inference model will be elaborated.

The table structure inference model predicts the table structure and detects table cell coordinates. The structure model is modified from the RARE algorithm, and improvements have been made mainly in the following aspects:

Input Data

For the text recognition model, each character marked in the dataset is independent, but in the table structure inference model, the category to be predicted is not a single character. The following is a dictionary comparison between RARE and the table structure inference model:

| model | dictionary |
|-----------------------|--|
| RARE | '<', 's', 'u', 'p', '>', '<', '/', 's', 'u', 'b', '>', '<', 'b', '>', '<', 'b', '>', '<', 'i', '>', '<', '/', 'i', '>' |
| Table structure model | 'sos', '<thead>', '<tr>', '<td>', '</td>', '</tr>', '</thead>', '<tbody>', '</tbody>', '<td', ' colspan="5"', '>', ' colspan="2"', ' colspan="3"', ' rowspan="2"', ' colspan="4"', ' colspan="6"', ' rowspan="3"', ' colspan="9"', ' colspan="10"', ' colspan="7"', ' rowspan="4"', ' colspan="8"', ' rowspan="5"', ' colspan="6"', ' rowspan="7"', ' colspan="10"', 'eos' |

The table structure inference model treats a string like <thead> as one character to recognition.

Model

The comparison chart of the table structure recognition model and RARE is as follows:

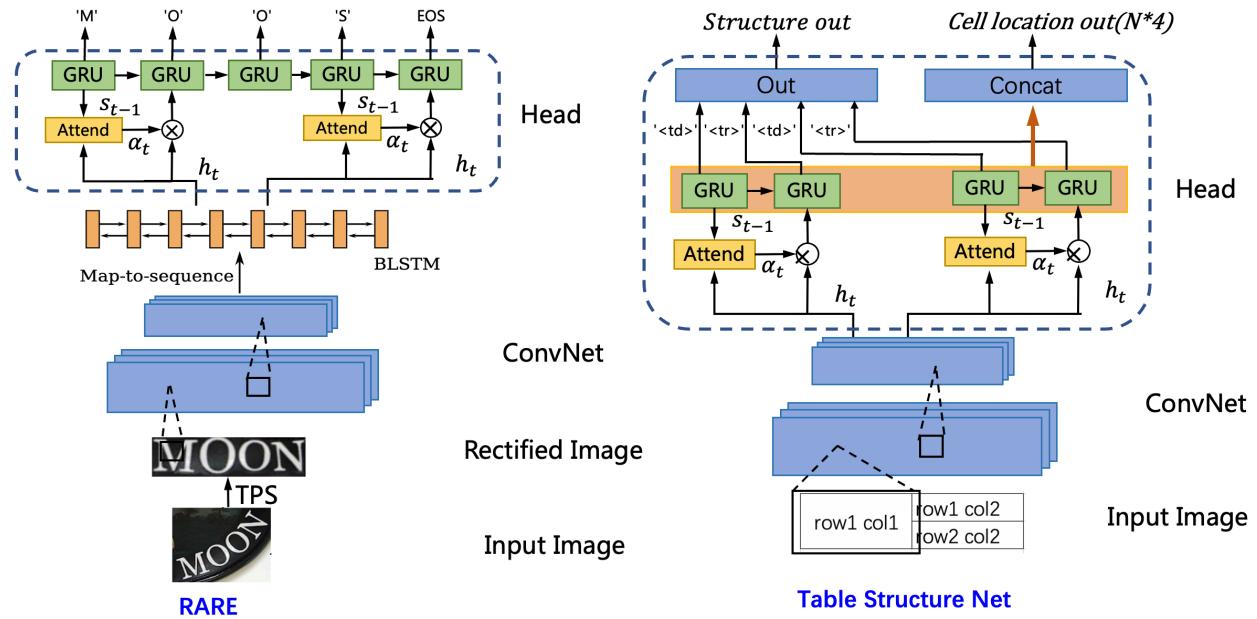


Figure 3: Table structure recognition model

The RARE model is composed of TPS+CNN+RNN+AttentionHead, and functions of each part are as follows:

1. TPS: It corrects the curved texts in the images.
2. CNN: It extracts features from the images.
3. RNN: It further enhances the extracted features and extracts semantic features.
4. AttentionHead: It performs the output.

In the table structure recognition model, the input image is a complete image, so the TPS module is removed. In addition, it is proved through experiments that RNN has little effect on the result. Therefore, the RNN module can be removed. The structure of the table structure recognition model is CNN+AttentionHead.

In order to output the cell coordinates, we have tried to detect the cell coordinates in the detection model. On the basis of the DB model, schemes 2 and 3 have been put forward.

| Solution | Result | | | | | | | | | | | | | | | | | | | | | |
|---|--|---------------------|-------------------|-----------------------------------|--------------------------------------|-----------------------|---------------------------------|-----------------------|--------------------|---------------------|---------------------|----|-----------------------------------|--------------------------------------|------------------|--------------------|---------------------|---------------------|----|-------------------------------|-------------------------------|---------------|
| 1. Single-line text detection | <table border="1"> <thead> <tr> <th>Tissue</th><th>Experion® (ng/µL)</th><th>NanoDrop (ng/µL)</th><th>Total RNA (ng)</th><th>Laser fires per cap</th><th>Cell Count per cap^b</th><th>Cell Count per sample</th></tr> </thead> <tbody> <tr> <td>MA (<i>n</i>=27)</td><td>7.9±0.8 (2.1, 16.7)</td><td>8.6±0.8 (2.3, 17.8)</td><td>86</td><td>3,277.9±141.0
(778.0; 8,212.0)</td><td>6,202.2±223.2
(1,182.6; 12,511.5)</td><td>24,808.7±1,270.2</td></tr> <tr> <td>MG (<i>n</i>=27)</td><td>4.5±0.4 (1.8, 10.4)</td><td>4.9±0.5 (1.8, 14.5)</td><td>49</td><td>785.6 33.0
(61.0; 2,190.0)</td><td>994.2±42.5
(69.5; 2,198.2)</td><td>8,192.2±728.3</td></tr> </tbody> </table> | Tissue | Experion® (ng/µL) | NanoDrop (ng/µL) | Total RNA (ng) | Laser fires per cap | Cell Count per cap ^b | Cell Count per sample | MA (<i>n</i> =27) | 7.9±0.8 (2.1, 16.7) | 8.6±0.8 (2.3, 17.8) | 86 | 3,277.9±141.0
(778.0; 8,212.0) | 6,202.2±223.2
(1,182.6; 12,511.5) | 24,808.7±1,270.2 | MG (<i>n</i> =27) | 4.5±0.4 (1.8, 10.4) | 4.9±0.5 (1.8, 14.5) | 49 | 785.6 33.0
(61.0; 2,190.0) | 994.2±42.5
(69.5; 2,198.2) | 8,192.2±728.3 |
| Tissue | Experion® (ng/µL) | NanoDrop (ng/µL) | Total RNA (ng) | Laser fires per cap | Cell Count per cap ^b | Cell Count per sample | | | | | | | | | | | | | | | | |
| MA (<i>n</i> =27) | 7.9±0.8 (2.1, 16.7) | 8.6±0.8 (2.3, 17.8) | 86 | 3,277.9±141.0
(778.0; 8,212.0) | 6,202.2±223.2
(1,182.6; 12,511.5) | 24,808.7±1,270.2 | | | | | | | | | | | | | | | | |
| MG (<i>n</i> =27) | 4.5±0.4 (1.8, 10.4) | 4.9±0.5 (1.8, 14.5) | 49 | 785.6 33.0
(61.0; 2,190.0) | 994.2±42.5
(69.5; 2,198.2) | 8,192.2±728.3 | | | | | | | | | | | | | | | | |
| 2. Use one model to detect texts and cells | <table border="1"> <thead> <tr> <th>Tissue</th><th>Experion® (ng/µL)</th><th>NanoDrop (ng/µL)</th><th>Total RNA (ng)</th><th>Laser fires per cap</th><th>Cell Count per cap^b</th><th>Cell Count per sample</th></tr> </thead> <tbody> <tr> <td>MA (<i>n</i>=27)</td><td>7.9±0.8 (2.1, 16.7)</td><td>8.6±0.8 (2.3, 17.8)</td><td>86</td><td>3,277.9±141.0
(778.0; 8,212.0)</td><td>6,202.2±223.2
(1,182.6; 12,511.5)</td><td>24,808.7±1,270.2</td></tr> <tr> <td>MG (<i>n</i>=27)</td><td>4.5±0.4 (1.8, 10.4)</td><td>4.9±0.5 (1.8, 14.5)</td><td>49</td><td>785.6 33.0
(61.0; 2,190.0)</td><td>994.2±42.5
(69.5; 2,198.2)</td><td>8,192.2±728.3</td></tr> </tbody> </table> | Tissue | Experion® (ng/µL) | NanoDrop (ng/µL) | Total RNA (ng) | Laser fires per cap | Cell Count per cap ^b | Cell Count per sample | MA (<i>n</i> =27) | 7.9±0.8 (2.1, 16.7) | 8.6±0.8 (2.3, 17.8) | 86 | 3,277.9±141.0
(778.0; 8,212.0) | 6,202.2±223.2
(1,182.6; 12,511.5) | 24,808.7±1,270.2 | MG (<i>n</i> =27) | 4.5±0.4 (1.8, 10.4) | 4.9±0.5 (1.8, 14.5) | 49 | 785.6 33.0
(61.0; 2,190.0) | 994.2±42.5
(69.5; 2,198.2) | 8,192.2±728.3 |
| Tissue | Experion® (ng/µL) | NanoDrop (ng/µL) | Total RNA (ng) | Laser fires per cap | Cell Count per cap ^b | Cell Count per sample | | | | | | | | | | | | | | | | |
| MA (<i>n</i> =27) | 7.9±0.8 (2.1, 16.7) | 8.6±0.8 (2.3, 17.8) | 86 | 3,277.9±141.0
(778.0; 8,212.0) | 6,202.2±223.2
(1,182.6; 12,511.5) | 24,808.7±1,270.2 | | | | | | | | | | | | | | | | |
| MG (<i>n</i> =27) | 4.5±0.4 (1.8, 10.4) | 4.9±0.5 (1.8, 14.5) | 49 | 785.6 33.0
(61.0; 2,190.0) | 994.2±42.5
(69.5; 2,198.2) | 8,192.2±728.3 | | | | | | | | | | | | | | | | |
| 3. Use two models to detect texts and cells | <table border="1"> <thead> <tr> <th>Tissue</th><th>Experion® (ng/µL)</th><th>NanoDrop (ng/µL)</th><th>Total RNA (ng)</th><th>Laser fires per cap</th><th>Cell Count per cap^b</th><th>Cell Count per sample</th></tr> </thead> <tbody> <tr> <td>MA (<i>n</i>=27)</td><td>7.9±0.8 (2.1, 16.7)</td><td>8.6±0.8 (2.3, 17.8)</td><td>86</td><td>3,277.9±141.0
(778.0; 8,212.0)</td><td>6,202.2±223.2
(1,182.6; 12,511.5)</td><td>24,808.7±1,270.2</td></tr> <tr> <td>MG (<i>n</i>=27)</td><td>4.5±0.4 (1.8, 10.4)</td><td>4.9±0.5 (1.8, 14.5)</td><td>49</td><td>785.6 33.0
(61.0; 2,190.0)</td><td>994.2±42.5
(69.5; 2,198.2)</td><td>8,192.2±728.3</td></tr> </tbody> </table> | Tissue | Experion® (ng/µL) | NanoDrop (ng/µL) | Total RNA (ng) | Laser fires per cap | Cell Count per cap ^b | Cell Count per sample | MA (<i>n</i> =27) | 7.9±0.8 (2.1, 16.7) | 8.6±0.8 (2.3, 17.8) | 86 | 3,277.9±141.0
(778.0; 8,212.0) | 6,202.2±223.2
(1,182.6; 12,511.5) | 24,808.7±1,270.2 | MG (<i>n</i> =27) | 4.5±0.4 (1.8, 10.4) | 4.9±0.5 (1.8, 14.5) | 49 | 785.6 33.0
(61.0; 2,190.0) | 994.2±42.5
(69.5; 2,198.2) | 8,192.2±728.3 |
| Tissue | Experion® (ng/µL) | NanoDrop (ng/µL) | Total RNA (ng) | Laser fires per cap | Cell Count per cap ^b | Cell Count per sample | | | | | | | | | | | | | | | | |
| MA (<i>n</i> =27) | 7.9±0.8 (2.1, 16.7) | 8.6±0.8 (2.3, 17.8) | 86 | 3,277.9±141.0
(778.0; 8,212.0) | 6,202.2±223.2
(1,182.6; 12,511.5) | 24,808.7±1,270.2 | | | | | | | | | | | | | | | | |
| MG (<i>n</i> =27) | 4.5±0.4 (1.8, 10.4) | 4.9±0.5 (1.8, 14.5) | 49 | 785.6 33.0
(61.0; 2,190.0) | 994.2±42.5
(69.5; 2,198.2) | 8,192.2±728.3 | | | | | | | | | | | | | | | | |

It can be seen that detecting texts and cells in the segmentation model will lead to the peculiarities of GT: Is the GT of the background between each line in the cell text or background?

Among the three models of the entire table recognition pipeline, only the text detection and table structure recognition models can obtain the information of the entire image. Therefore, an additional regression-based branch is added to the AttentionHead of the table structure recognition model to complete the cell coordinate (x0, y0, x1, y1) detection.

Forward Analysis of Table Structure Inference Model

Forward analysis of the model analyzes the output shape changes in each module of the image input from preprocessing to network output to better understand table cell inference and table structure inference models. The modules involved are as follows:

| Type | Module Name |
|-----------------|--------------------|
| Data Processing | ResizeTableImage |
| Data Processing | PaddingTableImage |
| Backbone | MobileNetV3 |
| Head | TableAttentionHead |

Input Data Processing

In this example, the input image and the output of the data processing module are visualized as follows:

```
# Switch to the PaddleOCR directory
os.chdir('../')
from ppocr.data import create_operators, transform
plt.figure(figsize=(24, 8))

# Read the input image
img = cv2.imread('doc/table/table.jpg')

# Display the input image
plt.subplot(1, 3, 1)
```

(continues on next page)

(continued from previous page)

```
plt.title('src, shape:{}'.format(img.shape))
plt.imshow(img)

# Implement ResizeTableImage
# https://github.com/PaddlePaddle/PaddleOCR/blob/dygraph/ppocr/data/imaug/gen_table_
#mask.py#L182

pre_process_list = [ {'ResizeTableImage': {'max_len': args.table_max_len }}] # Scale_
#the long side of the picture to the specified length, and scale the short side in_
#equal proportions
preprocess_op = create_operators(pre_process_list)
data = {'image': img}
data = transform(data, preprocess_op)

# Display the image after ResizeTableImage
plt.subplot(1,3,2)
plt.title('ResizeTableImage, shape:{}'.format(data['image'].shape))
plt.imshow(data['image'])

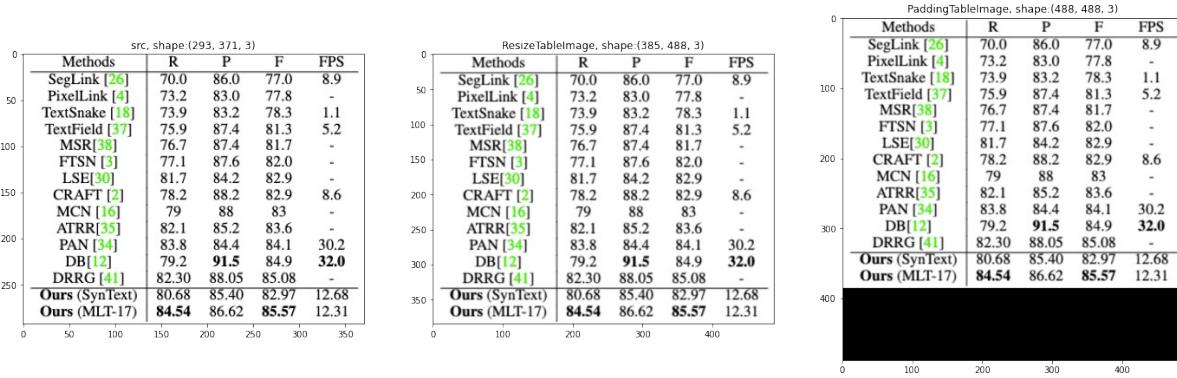
# Implement PaddingTableImage
# https://github.com/PaddlePaddle/PaddleOCR/blob/dygraph/ppocr/data/imaug/gen_table_
#mask.py#L232

pre_process_list = [ {'PaddingTableImage': None}]
preprocess_op = create_operators(pre_process_list)

data = transform(data, preprocess_op)

# Image after displaying PaddingTableImage
plt.subplot(1,3,3)
plt.title('PaddingTableImage, shape:{}'.format(data['image'].shape))
plt.imshow(data['image']/255)
plt.show()

# Define a list of processing ops
pre_process_list = [
    {'ResizeTableImage': {'max_len': args.table_max_len }},
    {'NormalizeImage':{'scale':1./255., 'mean': [0.485, 0.456, 0.406], 'std': [0.229,_
#0.224, 0.225], 'order': 'hwc'}},
    {'PaddingTableImage': None},
    {'ToCHWImage': None}
]
# Create op list
preprocess_op = create_operators(pre_process_list)
# Execute op list
data = {'image': img}
data = transform(data, preprocess_op)
```



```
# Download the pre-trained model
! wget -P ./pretrained_models/ https://paddleocr.bj.bcebos.com/dygraph_v2.1/table/en_
→ppocr_mobile_v2.0_table_structure_train.tar && cd pretrained_models && tar xf en_
→ppocr_mobile_v2.0_table_structure_train.tar && cd ..

# Downloaded the pre-trained model
import paddle

# Read pre-training parameters and divide them into backbone parameters and head_
→parameters
pretrain_params = paddle.load('pretrained_models/en_ppocr_mobile_v2.0_table_structure_
→train/best_accuracy.pdparams')
def filter_params(pretrain_params, prefix):
    new_dict = {}
    for k,v in pretrain_params.items():
        if k.startswith(prefix):
            new_dict[k.replace(prefix+'.', '')] = v
    return new_dict
# Extract parameters
backbone_dict = filter_params(pretrain_params, 'backbone')
head_dict = filter_params(pretrain_params, 'head')
```

Backbone

The backbone is the same as the detected backbone, and both output four feature maps with sizes of 1/4, 1/8, 1/16 and 1/32 of the input image. Relevant backbones have been introduced in the text detection chapter, and will not be repeated here.

```
# https://github.com/PaddlePaddle/PaddleOCR/blob/dygraph/ppocr/modeling/backbones/det_
→mobilenet_v3.py

from ppocr.modeling.backbones import build_backbone
# Initialize the backbone
backbone = build_backbone(dict(name='MobileNetV3', scale=1.0, model_name='large'),model_
→type='table')
backbone.eval()
# Load backbone parameters
backbone.set_state_dict(backbone_dict)
```

```
import numpy as np
x = np.expand_dims(data['image'], axis=0)
```

(continues on next page)

(continued from previous page)

```
x = paddle.to_tensor(x)
backbone_out = backbone(x)
for item in backbone_out:
    print(item.shape)
```

```
[1, 24, 122, 122]
[1, 40, 61, 61]
[1, 112, 31, 31]
[1, 960, 16, 16]
```

Head

The input of the head is the four feature maps output by the backbone, and the output is the inference result of the table structure and cell coordinates

The meanings of the input parameters are:

| Parameter | Meaning |
|-----------------|---|
| in_channels | The number of channels of the input feature map |
| hidden_size | The hidden layer unit of the RNN module in Attention |
| max_elem_length | Maximum number of inferred characters |
| in_max_len | The size of the input image |
| loc_type | Input of the output cell coordinate branch1: Only the hidden layer after Attention is used 2: Fuse CNN + Attention features |

The code is as follows:

```
# https://github.com/PaddlePaddle/PaddleOCR/blob/dygraph/ppocr/modeling/heads/table_
→att_head.py

from paddle import nn
import paddle.nn.functional as F
from ppocr.modeling.heads.table_att_head import AttentionGRUCell

class TableAttentionHead(nn.Layer):
    def __init__(self,
                 in_channels,
                 hidden_size,
                 loc_type=2,
                 in_max_len=488, # The size of the input image
                 max_elem_length=800, # Maximum number of output labels
                 **kwargs):
        super(TableAttentionHead, self).__init__()
        self.input_size = in_channels[-1]
        self.hidden_size = hidden_size
        self.elem_num = 30
        self.max_elem_length = max_elem_length

        self.structure_attention_cell = AttentionGRUCell(
            self.input_size, hidden_size, self.elem_num, use_gru=False)
        self.structure_generator = nn.Linear(hidden_size, self.elem_num)
        self.loc_type = loc_type
```

(continues on next page)

(continued from previous page)

```

self.in_max_len = in_max_len

# Coordinate box regression branch
if self.loc_type == 1:
    self.loc_generator = nn.Linear(hidden_size, 4)
else:
    if self.in_max_len == 640:
        # 640 The size of the last feature map passed through the backbone is ↵20*20, so the size of the input feature map here is 400
        self.loc_fea_trans = nn.Linear(400, self.max_elem_length + 1)
    elif self.in_max_len == 800:
        # 800 The size of the last feature map after the backbone is 23*25, ↵so the size of the input feature map here is 625
        self.loc_fea_trans = nn.Linear(625, self.max_elem_length + 1)
    elif self.in_max_len == 488:
        # 800 The size of the last feature map passed through the backbone is ↵16*16, so the size of the input feature map here is 256
        self.loc_fea_trans = nn.Linear(256, self.max_elem_length + 1)
    self.loc_generator = nn.Linear(self.input_size + hidden_size, 4)

def _char_to_onehot(self, input_char, onehot_dim):
    input_ont_hot = F.one_hot(input_char, onehot_dim)
    return input_ont_hot

def forward(self, inputs, targets=None):
    # Take out the smallest map output by the backbone
    fea = inputs[-1]
    if len(fea.shape) == 3:
        pass
    else:
        # Reshape B,C,H,W as B,C,H*W
        last_shape = int(np.prod(fea.shape[2:]))
        fea = paddle.reshape(fea, [fea.shape[0], fea.shape[1], last_shape])
        # Change B,C,W into B,W,C
        fea = fea.transpose([0, 2, 1])
    batch_size = fea.shape[0]

    hidden = paddle.zeros((batch_size, self.hidden_size))
    output_hiddens = []
    if self.training and targets is not None:
        structure = targets[0]
        for i in range(self.max_elem_length + 1):
            elem_onehots = self._char_to_onehot(
                structure[:, i], onehot_dim=self.elem_num)
            (outputs, hidden), alpha = self.structure_attention_cell(
                hidden, fea, elem_onehots)
            output_hiddens.append(paddle.unsqueeze(outputs, axis=1))
        output = paddle.concat(output_hiddens, axis=1)
        structure_probs = self.structure_generator(output)
        if self.loc_type == 1:
            loc_preds = self.loc_generator(output)
            loc_preds = F.sigmoid(loc_preds)
        else:
            loc_fea = fea.transpose([0, 2, 1])
            loc_fea = self.loc_fea_trans(loc_fea)
            loc_fea = loc_fea.transpose([0, 2, 1])

```

(continues on next page)

(continued from previous page)

```

loc_concat = paddle.concat([output, loc_fea], axis=2)
loc_preds = self.loc_generator(loc_concat)
loc_preds = F.sigmoid(loc_preds)

else:
    temp_elem = paddle.zeros(shape=[batch_size], dtype="int32")
    structure_probs = None
    loc_preds = None
    elem_onehots = None
    outputs = None
    alpha = None
    max_elem_length = paddle.to_tensor(self.max_elem_length)
    i = 0
    # Attention forward
    while i < max_elem_length + 1:
        elem_onehots = self._char_to_onehot(
            temp_elem, onehot_dim=self.elem_num)
        (outputs, hidden), alpha = self.structure_attention_cell(
            hidden, fea, elem_onehots)
        output_hiddens.append(paddle.unsqueeze(outputs, axis=1))
        structure_probs_step = self.structure_generator(outputs)
        temp_elem = structure_probs_step.argmax(axis=1, dtype="int32")
        i += 1

    output = paddle.concat(output_hiddens, axis=1)
    print('Attention output shape', output.shape)
    # Table structure branch
    structure_probs = self.structure_generator(output)
    structure_probs = F.softmax(structure_probs)

    # Cell coordinate branch
    if self.loc_type == 1:
        loc_preds = self.loc_generator(output)
        loc_preds = F.sigmoid(loc_preds)
    else:
        # Change B,W,C into B,C,W
        locfea = fea.transpose([0, 2, 1])

        locfea = self.loc_fea_trans(locfea)
        locfea = locfea.transpose([0, 2, 1])
        loc_concat = paddle.concat([output, locfea], axis=2)
        loc_preds = self.loc_generator(loc_concat)
        loc_preds = F.sigmoid(loc_preds)

    return {'structure_probs': structure_probs, 'loc_preds': loc_preds}

```

```

# Initialize the head
head = TableAttentionHead(in_channels=backbone.out_channels, hidden_size=256, loc_
    _type=2)
head.eval()
# Load the head parameter
head.set_state_dict(head_dict)

# Execute the head
print('*'*10, 'head forward shape', '*'*10)
head_out = head(backbone_out)
print('*'*10, 'head out shape', '*'*10)

```

(continues on next page)

(continued from previous page)

```
# Print the head output and the corresponding shape
for key in head_out:
    print(key, head_out[key].shape)
```

```
***** head forward shape *****
Attention output shape [1, 801, 256]
***** head out shape *****
structure_probs [1, 801, 30]
loc_preds [1, 801, 4]
```

Post-processing

The dictionary file for post-processing is ppocr/utils/dict/table_structure_dict.txt

The idea of post-processing decoding:

1. Perform CTC decoding on structure_probs: remove background characters sos and eos, and take only one of the consecutively repeated characters.
2. Normalize the output coordinates to 0-1, multiply the coordinates by the width and height of the image, and decode them to the image space

```
# https://github.com/PaddlePaddle/PaddleOCR/blob/dygraph/ppocr/postprocess/rec_
→postprocess.py#L441

from ppocr.postprocess.rec_postprocess import TableLabelDecode

def post_process(out):
    character_dict_path = 'ppocr/utils/dict/table_structure_dict.txt'
    # Initialize the post-processing of op
    post_op = TableLabelDecode(character_dict_path)

    post_result = post_op(out)

    structure_str_list = post_result['structure_str_list']

    # The normalized coordinates are restored to the size of the original image size
    res_loc = post_result['res_loc']
    img_h, img_w = img.shape[0:2]
    res_loc_final = []
    for rno in range(len(res_loc[0])):
        x0, y0, x1, y1 = res_loc[0][rno]
        left = max(int(img_w * x0), 0)
        top = max(int(img_h * y0), 0)
        right = min(int(img_w * x1), img_w - 1)
        bottom = min(int(img_h * y1), img_h - 1)
        res_loc_final.append([left, top, right, bottom])

    # Process the structure information
    structure_str_list = structure_str_list[0]
    structure_str_list = ['<html>', '<body>', '<table>'] + structure_str_list + ['</
→table>', '</body>', '</html>']
    return structure_str_list, res_loc_final
structure_str_list, res_loc_final = post_process(head_out)
```

(continues on next page)

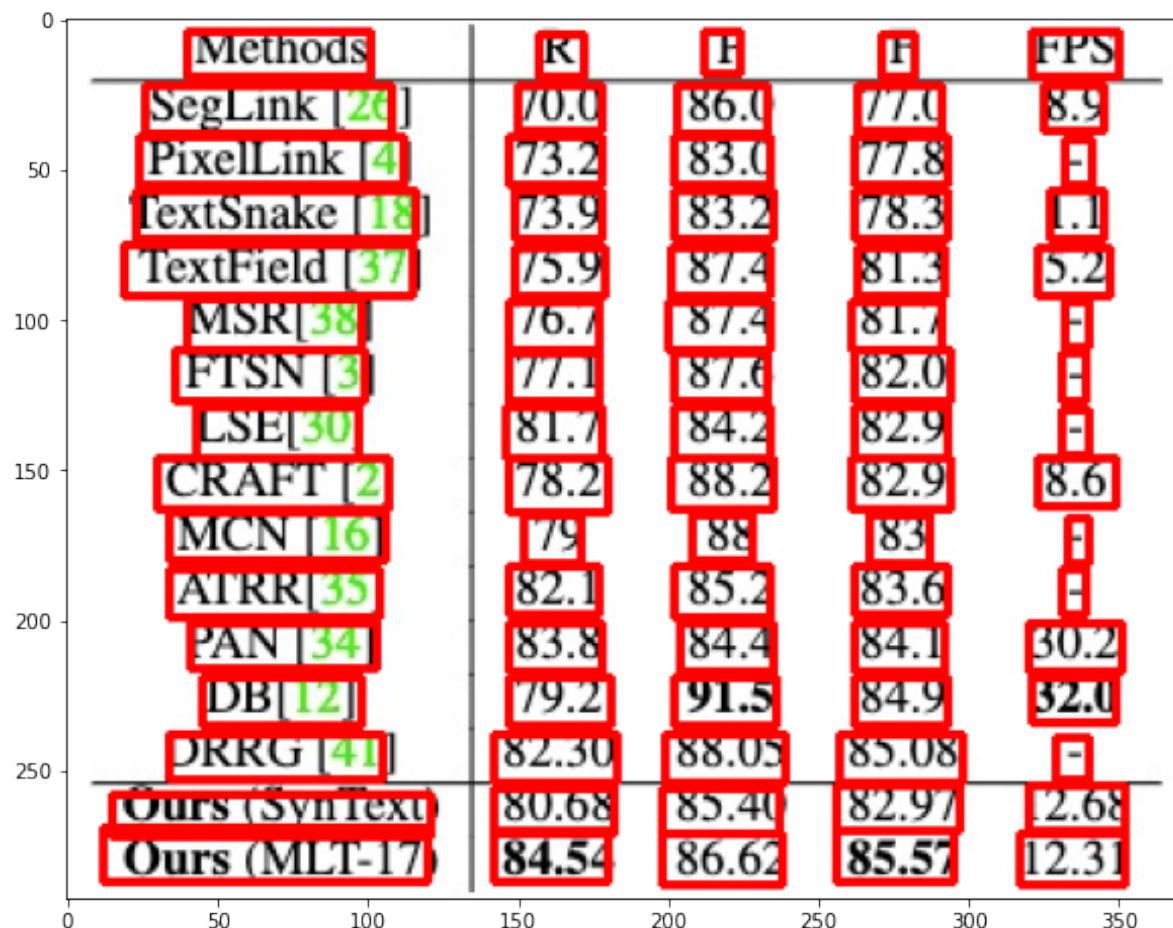
(continued from previous page)

```
# print the first 10 outputs
print(structure_str_list[:10])
print(res_loc_final[:10])

# Visualize the inference box
plt.figure(figsize=(24,8))
img_show = img.copy()
for box in res_loc_final:
    cv2.rectangle(img_show, (box[0], box[1]), (box[2], box[3]), (255, 0, 0), 2)
plt.imshow(img_show)

['<html>', '<body>', '<table>', '<thead>', '<tr>', '<td>', '</td>', '<td>', '</td>', '<td>'],
[[40, 4, 101, 19], [157, 5, 172, 19], [212, 4, 224, 18], [271, 5, 282, 19], [321, 4, 350, 19], [26, 22, 108, 38], [150, 22, 178, 37], [203, 22, 233, 38], [263, 22, 291, 37], [325, 22, 345, 37]]]

<matplotlib.image.AxesImage at 0x7f87e84392d0>
```



8.2.3 Training

In the training of table recognition, three models need to be trained, including text detection, text recognition, and table structure models. For the training of text detection and recognition models, you can refer to the previous sections. Only the training of the table structure model is introduced here.

This section uses the pubtabnet dataset and MobileNetV3 as the table structure model of the backbone network to introduce how to train, evaluate, and test the table structure model.

Data Preparation

This experiment selects the PubTabNet dataset for demonstrate. The sample diagram of the PubTabNet dataset is shown in the following figure:

Part of the PubTabNet dataset has been downloaded in the project and stored in /home/aistudio/data/data119702. You can run the following command to unzip the the dataset, or download it from <https://github.com/ibm-aur-nlp/PubTabNet>

```
# Unzip the dataset
! cd /home/aistudio/data/data119702 && tar -xf pubtabnet_val.tar && cd -
! ls /home/aistudio/data/data119702
```

```
/home/aistudio/PaddleOCR
PubTabNet_2.0.0_val.jsonl  pubtabnet_val.tar  val
```

After running the above command /home/aistudio/data/data119702 there are a folder and a file

```
/home/aistudio/data/data119702
  └── val/                               Folder for image storage
      └── PubTabNet_2.0.0_val.jsonl/        Label information
```

The label format of this dataset is

```
{
  'filename': 'PMC5755158_010_01.png',                                # Image name
  'split': 'train',                                                     # Does the
  ↵image belong to the training set or the validation set?
  'imgid': 0,                                                          #
  ↵          # The index of the image
  'html': {
    'structure': {'tokens': ['<thead>', '<tr>', '<td>', ...]},   #
    ↵          # HTML string of the table
    'cell': [
      {
        'tokens': ['P', 'a', 'd', 'd', 'l', 'e', 'P', 'a', 'd', 'd', 'l', 'e'], #
        ↵          # Single text in the table
        'bbox': [x0, y0, x1, y1]                                              # The coordinate of a
      ↵          # single text in the table
      }
    ]
  }
}
```

Data Preprocessing

There are requirements for the format and size of the input pictures in the training. Therefore, before the data are input into the model, they need to be preprocessed to make the pictures and labels meet the needs of network training and inference.

The data preprocessing of the table structure model mainly includes:

- DecodeImage, which converts the image into the format of Numpy;
- ResizeTableImage, which resizes the picture and the long sides to a specified size, and equally scales the short sides;
- TableLabelEncode, which parses the label information in the label file and save it in a unified format;
- NormalizeImage, which changes the input value distribution of any neuron in each layer of the neural network to a standard normal distribution with a mean value of 0 and a variance of 1, by normalization, so that the optimization of the optimal solution will obviously become smooth and the training is easier to converge;
- PaddingTableImage, which pads the short sides of the image to the size same as the long sides
- ToCHWImage, where the image data format is [H, W, C] (height, width, channel number), and the training data format used by the neural network is [C, H, W], so the image data needs to be rearranged such as changing [224, 224, 3] into [3, 224, 224];
- KeepKeys, which filters the dict

TableLabelEncode

To analyze the label information in the label file, first load the label data and take out a label

```
# Load a piece of data in the dataset
import json
from pprint import pprint
with open('/home/aistudio/data/data119702/PubTabNet_2.0.0_val.jsonl', "rb") as f:
    data_lines = f.readlines()
    for line in data_lines:
        data_line = line.decode('utf-8').strip("\n")
        info = json.loads(data_line)
        break
```

Run the following code to observe the comparison before and after the TableLabelEncode label encoding.

```
from ppocr.data.imaug import TableLabelEncode
# Initialize the label encoder
label_eocoder_op = TableLabelEncode(max_text_length=100, # Unused
                                     max_elem_length=50, # How many cels can be
                                     predicted at most for each picture
                                     max_cell_num=500, # unused
                                     character_dict_path='ppocr/utils/dict/table_
                                     structure_dict.txt')
# Construct input data
cells = info['html']['cells']
structure = info['html']['structure']
# 2. Print the label before decoding
print("The cells and structure before decode")
print("cells: ", cells)
print("structure: ", structure)

image = cv2.imread(os.path.join('/home/aistudio/data/data119702/val', info['filename
                                     '']))
```

(continues on next page)

(continued from previous page)

```

data = {'image':image, 'cells': cells, 'structure':structure}
# Execute label encoder
data = label_eocoder_op(data)
# Print the encoded information
print("The bbox_list and structure after decode")
print("bbox_list:",data['bbox_list'].tolist())
print("structure:", data['structure'].tolist())

```

will get the following output

```

The cells and structure before decode
cells: [
{'tokens': []},
{'tokens': ['<b>', 'W', 'e', 'a', 'n', 'i', 'n', 'g', '</b>'], 'bbox': [66, 4, 96, 13]},
{'tokens': ['<b>', 'W', 'e', 'e', 'k', ' ', '1', '5', '</b>'], 'bbox': [131, 4, 160, 13]},
...
}
structure: {'tokens': ['<thead>', '<tr>', '<td>', ..., '</td>', '</tr>', '</tbody>']}
The bbox_list and structure after decode
bbox_list: [
[0.0, 0.0, 0.0, 0.0],
[0.0, 0.0, 0.0, 0.0],
[0.0, 0.0, 0.0, 0.0],
[0.0, 0.0, 0.0, 0.0],
[0.0, 0.0, 0.0, 0.0],
[0.27731093764305115, 0.06779661029577255, 0.40336135029792786, 0.22033898532390594],
...
[0.0, 0.0, 0.0, 0.0]
]
structure: [0, 1, 2, ..., 5, 8, 29, 0, 0, 0, 0, 0]

```

Loss Function

The loss of the model is divided into two parts:

1. Structure loss: structure loss uses CrossEntropyLoss
2. Loc loss: loc loss uses MSELoss

The two losses are fused by weighting, and the total losses of structure_weight and loc_weight are 100 and 10000 respectively

```
total_loss = structure_loss * structure_weight + loc_loss * loc_weight
```

Model Training

Once the data are processed and the loss function are defined, the model training can start.

The training is based on the PaddleOCR training engine, in the form of parameter configuration, refer to the configuration file of the table recognition model https://github.com/PaddlePaddle/PaddleOCR/blob/dygraph/configs/table/table_mv3.yml, the network structure parameters are as follows

```
Architecture:  
  model_type: table  
  algorithm: TableAttn  
  Backbone:  
    name: MobileNetV3  
    scale: 1.0  
    model_name: large  
  Head:  
    name: TableAttentionHead  
    hidden_size: 256  
    loc_type: 2  
    max_text_length: 100  
    max_elem_length: 800  
    max_cell_num: 500
```

The loss function parameters are as follows:

```
Loss:  
  name: TableAttentionLoss  
  structure_weight: 100.0  
  loc_weight: 10000.0
```

After the configuration is complete, the training can be started by running the following command

```
# Configurate the dataset  
# !mkdir -p train_data/table/pubtabnet  
!cd train_data/table/pubtabnet && ln -s /home/aistudio/data/data119702/PubTabNet_2.0.  
→0_val.jsonl PubTabNet_2.0.0_train.jsonl \  
&& ln -s /home/aistudio/data/data119702/PubTabNet_2.0.0_val.jsonl PubTabNet_2.0.0_val.  
→jsonl \  
&& ln -s /home/aistudio/data/data119702/val train \  
&& ln -s /home/aistudio/data/data119702/val val
```

```
ln: failed to create symbolic link 'PubTabNet_2.0.0_train.jsonl': File exists
```

```
! python tools/train.py -c configs/table/table_mv3.yml -o Global.use_gpu=False Global.  
→print_batch_step=1 Train.loader.batch_size_per_card=1 Eval.loader.batch_size_per_  
→card=1
```

During the training process, the following log will be output

```
[2021/12/26 19:57:29] root INFO: train with paddle 2.2.1 and device CPUPlace  
[2021/12/26 19:57:29] root INFO: Initialize indexs of datasets:train_data/table/  
→pubtabnet/PubTabNet_2.0.0_train.jsonl  
[2021/12/26 19:57:29] root INFO: Initialize indexs of datasets:train_data/table/  
→pubtabnet/PubTabNet_2.0.0_val.jsonl  
[2021/12/26 19:57:29] root INFO: train from scratch
```

(continues on next page)

(continued from previous page)

```
[2021/12/26 19:57:29] root INFO: train dataloader has 9115 iters
[2021/12/26 19:57:29] root INFO: valid dataloader has 9115 iters
[2021/12/26 19:57:29] root INFO: During the training process, after the 0th iteration,
    ↪ an evaluation is run every 400 iterations
[2021/12/26 19:57:29] root INFO: Initialize indexs of datasets:train_data/table/
    ↪pubtabnet/PubTabNet_2.0.0_train.jsonl
[2021/12/26 19:57:47] root INFO: epoch: [1/400], iter: 1, lr: 0.001000, loss: 358.
    ↪711182, structure_loss: 277.904785, loc_loss: 80.806374, acc: 0.000000, reader_
    ↪cost: 0.05254 s, batch_cost: 17.39120 s, samples: 2, ips: 0.11500
[2021/12/26 19:57:55] root INFO: epoch: [1/400], iter: 2, lr: 0.001000, loss: 353.
    ↪381165, structure_loss: 208.200623, loc_loss: 137.825607, acc: 0.000000, reader_
    ↪cost: 0.00041 s, batch_cost: 8.65134 s, samples: 1, ips: 0.11559
...
...
```

Model Evaluation

During the training process, two models are saved by default: one is the latest trained model named `latest`, and the other is the model named `best_accuracy` with the highest accuracy. Next, use the saved model parameters to evaluate the accuracy of the test set:

The accuracy evaluation code of the table structure model is located in [PaddleOCR/ppocr/metrics/table_metric.py](#). Call `tools/eval.py` to evaluate the accuracy of the trained model.

```
!python tools/eval.py -c configs/table/table_mv3.yml -o Global.checkpoints=/home/
    ↪aistudio/PaddleOCR/pre_train/en_ppocr_mobile_v2.0_table_structure_train/best_
    ↪accuracy Global.use_gpu=False Eval.loader.batch_size_per_card=1
```

The following log will be output during the evaluation process, the data set is too large, here we manually terminate the process.

```
[2021/12/26 20:00:08] root INFO: train with paddle 2.2.1 and device CPUPlace
[2021/12/26 20:00:08] root INFO: Initialize indexs of datasets:train_data/table/
    ↪pubtabnet/PubTabNet_2.0.0_val.jsonl
[2021/12/26 20:00:08] root INFO: resume from /home/aistudio/PaddleOCR/pre_train/en_
    ↪ppocr_mobile_v2.0_table_structure_train/best_accuracy
[2021/12/26 20:00:08] root INFO: metric in ckpt ****
[2021/12/26 20:00:08] root INFO: acc:0.7380142622051563
[2021/12/26 20:00:08] root INFO: fps:8.360272547972942
[2021/12/26 20:00:08] root INFO: best_epoch:7
[2021/12/26 20:00:08] root INFO: start_epoch:8
eval model::: 0% | 2/9115 [00:07<8:55:26, 3.53s/it]^C
```

Model Inference

After training the model, you can also use the saved model to perform model inference on a single picture or an image in a folder, and observe the inference result.

```
! python tools/infer_table.py -c configs/table/table_mv3.yml -o Global.
    ↪checkpoints=pretrained_models/en_ppocr_mobile_v2.0_table_structure_train/best_
    ↪accuracy Global.infer_img=doc/table/table.jpg Global.use_gpu=False
```

The structure information and cell box information of the table are output during the inference process.

8.2.4 Summary

This section introduces the principle of PP-Structure table recognition algorithm in PaddleOCR, and the process of the table structure model from data processing to the end of training.

8.2.5 Assignment

<https://aistudio.baidu.com/aistudio/education/objective/28711>

8.3 DOC-VQA SER Practice

This section will introduce how to use PaddleOCR to train and run the DOC-VQA SER algorithm, including:

1. Understanding the principle of DOC-VQA SER algorithm
 2. Mastering the training process of DOC-VQA SER code in PaddleOCR

8.3.1 Quick Start

Prepare the code and the environment

```
# Clone PaddleOCR code
! git clone -b release/2.4 https://gitee.com/paddlepaddle/PaddleOCR

# Install dependencies
! pip install -r PaddleOCR/requirements.txt
! pip install paddleocr

# Install dependencies
! pip install yacs qnureadline paddlenlp==2.2.1
```

```
# Change to the vqa directory
import os
os.chdir('PaddleOCR/ppstructure/vqa')
```

```
# Download the model
! mkdir pretrained_models
# Download SER model and unzip it
! wget -P ./pretrained_models/ https://paddleocr.bj.bcebos.com/pplayout/PP-Layout_v1.
˓→0_ser_pretrained.tar && cd pretrained_models && tar xf PP-Layout_v1.0_ser_
˓→pretrained.tar && cd ..
```

```
# Perform SER inference
# https://github.com/PaddlePaddle/PaddleOCR/blob/release%2F2.4/ppstructure/vqa/infer_
˓→ser_e2e.py

! python infer_ser_e2e.py \
--model_name_or_path "./pretrained_models/PP-Layout_v1.0_ser_pretrained/" \
--max_seq_length 512 \
--output_dir "output/res_e2e/" \
--infer_imgs "images/input/zh_val_42.jpg"

import cv2
from matplotlib import pyplot as plt
%matplotlib inline

img = cv2.imread('output/res_e2e/zh_val_42_ser.jpg')
plt.figure(figsize=(48,24))
plt.imshow(img)
```

```
process: [0/1], save result to output/res_e2e/zh_val_42_ser.jpg
Corrupt JPEG data: premature end of data segment
```

```
<matplotlib.image.AxesImage at 0x7fa228317cd0>
```

| | | | | | | | | | | |
|--|------------------|---------------------------|-----------------|-----------------|-----------------|-------------------|-------------|-------------------------|---------|--|
| 个人信息登记表 | | | | | | | | | | |
| 申报学院(部门) : | | | | | | | | | | |
| 一寸报名照
粘贴处 | 姓名
(拼音) | (岳欣欣) | | 性别 | 女 | 出生日期 | 1997年12月17日 | 政治面貌 | 群众 | |
| | 国籍 | 中国 | | 民族 | 汉 | 婚育状况 | 未婚 | 生源地 / 培养形式
(应届毕业生填写) | 辽宁省西丰县 | |
| | 现工作(学习)
单位 | 大连海事大学 | | 单位地址 | 辽宁省大连市甘井子区凌海路3号 | | | 邮编 | 100085 | |
| | 现专业技术职
务及评定时间 | | | 专技等级 | | | | 现任行政职务 | | |
| | 学科专业 | 船只制造 | | 主要学术兼职 | 无 | | | | | |
| | 最高学历 | 研究生 | | 毕业单位 | 大连海事大学 | | | 毕业时间 | 2019年6月 | |
| | 最高学位 | 硕士学位 | | 获得单位 | 大连海事大学 | | | 获得时间 | 2019年6月 | |
| | 证件类型 | 学位证书 | 证件号码 | D47854648486893 | | | 户籍详址 | 辽宁省西丰县安民镇永淳村二组62号 | | |
| | 档案所在
单位及地址 | 大连海事大学
辽宁省大连市甘井子区凌海路3号 | | | 联系电话 | 13585662395 | | 邮编 | 100085 | |
| | 有否境外永居
权 | 否 | | 有无上海市居住证 | 无 | | | 居住证有效期 | | |
| 联系
方式 | 本人联系电话 | 13585662395 | | | 电子邮箱 | 2895695897@qq.com | | | | |
| | 通讯地址 | 辽宁省大连市甘井子区凌海路3号 | | | 单位及地址 | 辽宁省大连市甘井子区凌海路3号 | | | | |
| 个人简历
(从大学顺序填起) | 起始年月 | 终止年月 | 学校或工作单位 | 卒业 | 专业 | 学习或任职情况 | 备注 | | | |
| | 2016.09 | 2019.06 | 大连海事大学 | 本科 | 轮机工程 | 轮机长 | 2019年6月 | | | |
| | | | 联系方式 | 方式 | | | | | | |
| | | | 备注 | 起始年月 | 终止年月 | 学校或工作单位 | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| * 个人简历时间上要连续。本栏“学习或任职情况”对大专、本硕博(后)或任职岗位、访问学者等实际情况。
* 研究生、博士、访问学者、联合培养等经历需在“备注”内写明。
* 兼职、派遣等经历,请在“备注”中注明;无就业或学习经历的阶段需填写“待业”或注明原因。 | | | | | | | | | | |
| 语言
能力 | 证书名称
(等级或分数) | 口语水平
(等级或分数) | 写作水平
(等级或分数) | 其他
资格
证书 | 等级
或分数 | 取得时间 | 等级
或分数 | 取得时间 | | |
| | 语种 | 证书名称
(等级或分数) | 口语水平
(等级或分数) | | 等级
或分数 | 取得时间 | 等级
或分数 | 取得时间 | | |
| | 语种 | 证书名称
(等级或分数) | 口语水平
(等级或分数) | | 等级
或分数 | 取得时间 | 等级
或分数 | 取得时间 | | |
| | 语种 | 证书名称
(等级或分数) | 口语水平
(等级或分数) | | 等级
或分数 | 取得时间 | 等级
或分数 | 取得时间 | | |
| * 证书水平分为:精通、熟练、一般。
* 语种水平分为:精读、口译、写作水平分为:精通、熟练、一般。 | | | | | | | | | | |

8.3.2 Explanation of The Principle

algorithms of the DOC-VQA series in PaddleOCR are currently implemented based on the LayoutXML paper, providing two tasks: SER and RE

LayoutXML is a multi-language version of LayoutLMV2. The schematic of LayoutLMV2 is as follows:

Compared with Bert in NLP, LayoutXML adds the layout information of the text in the image and image features to the input of the model. LayoutXML has been implemented in PaddleNLP, so here we introduce the data and network from the perspective of the model forward.

Input Data Processing

First, perform OCR or pdf analysis on the image, obtain texts and bbox information, and build the three inputs of the model on this basis:

1. Text Embedding

First, use WordPiece to segment the text recognized by OCR, then add [CLS] and [SEP] tags, and use [PAD] to fill in the length to get the input sequence of the text:

$$S = \{[CLS], w_1, w_2, \dots, [SEP], [PAD], [PAD], \dots\}, |S| = L$$

Then add the word vector, the one-dimensional position vector, and the segmented vector to get the text vector, the formula is as follows:

$$t_i = TokEmb(w_i) + PosEmb1D(i) + SegEmb(s_i), 0 \leq i < L$$

One-dimensional position vector: the index of the word

Segmented vector: A

```
# Text Embedding demo

from paddlenlp.transformers import LayoutXLMTOKENIZER

TOKENIZER = LayoutXLMTOKENIZER.from_pretrained('pretrained_models/PP-Layout_v1.0_ser_'
                                               'pretrained')
# Tokenization
print('测试', TOKENIZER.tokenize('测试'))
# The result of tokenization
print('测试', TOKENIZER.encode('测试'))
```

```
['_E', 'E', 'E']
[0, 13129, 84072, 1801, 2, 0, 0]
```

1. Image Embedding

We use the ResNeXt-FPN network as the image encoder. First, extract the feature map of the original document image, then average it into a fixed size ($B * 256 * 7 * 7$), and expand the feature map in row ($B * 256 * 49$) to get the characteristic sequence corresponding to the image after linear projection ($B * 49 * 256$). Corresponding to the composition of the text vector, the image vector is also supplemented with one-dimensional relative position and segmentation information. Finally, add the feature vector, the one-dimensional position vector, and the segmented vector to get the image vector:

$$v_i = Proj(VisTokEmb(I)_i) + PosEmb1D(i) + SegEmb([C]), 0 \leq i < WH$$

Segmented vector[C]

2. Layout Embedding

For the coordinate range of each word or image region on the page, a bounding box parallel to the coordinate axis is used to represent the layout information, and each bounding box is represented by 4 boundary coordinate values, width, and height. The layout vector is obtained by concatenating the vectors corresponding to the 6 features:

$$I_i = \text{Concat}(\text{PosEmb2D}_x(x_0, x_1, w), \text{PosEmb2D}_y(y_0, y_1, h)), 0 \leq i < WH + L$$

The following demonstrates the process of constructing a network input from an input image. The whole process mainly includes the following steps:

1. Performing OCR recognition on the image
2. Preprocessing the image, including scaling to a particular size and normalization
3. Tokenizing the recognized text and converting it into the index
4. Normalizing the text box and keeping its value between 0-1000
5. Padding the result after processing 3 and 4 to facilitate batch grouping

```
# Construct the input of inference
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/ppstructure/vqa/vqa_
→utils.py

import cv2
import numpy as np
import paddle
from copy import deepcopy
from paddleocr import PaddleOCR
from paddlenlp.transformers import LayoutXLMTokenizer

from infer_ser_e2e import trans_poly_to_bbox, pad_sentences, split_page

def parse_ocr_info_for_ser(ocr_result):
    # The OCR result is converted into the format of the dictionary, and the text box_
    →is converted to a bounding rectangle
    ocr_info = []
    for res in ocr_result:
        ocr_info.append({
            "text": res[1][0],
            "bbox": trans_poly_to_bbox(res[0]),
            "poly": res[0],
        })
    return ocr_info

def preprocess(
    tokenizer,
    ori_img,
    ocr_info,
    img_size=(224, 224),
    pad_token_label_id=-100,
    max_seq_len=512,
    add_special_ids=False,
    return_attention_mask=True,
):
    ocr_info = deepcopy(ocr_info)
    height = ori_img.shape[0]
    width = ori_img.shape[1]
```

(continues on next page)

(continued from previous page)

```

# Resize the image to a particular shape
img = cv2.resize(ori_img, img_size).transpose([2, 0, 1]).astype(np.float32)

segment_offset_id = [] # Store the ending position of each text in input_ids
bbox_list = [] # Store the box normalized to 0-1000
input_ids_list = [] # Store the index of the tokenized text segment in the vocabulary
token_type_ids_list = [] # Store the category information of the text segment

for info in ocr_info:
    # Normalize the box to 0-1000
    # x1, y1, x2, y2
    bbox = info["bbox"]
    bbox[0] = int(bbox[0] * 1000.0 / width)
    bbox[2] = int(bbox[2] * 1000.0 / width)
    bbox[1] = int(bbox[1] * 1000.0 / height)
    bbox[3] = int(bbox[3] * 1000.0 / height)

    # Tokenizer the text information, including tokenization and conversion to the index in the vocabulary
    text = info["text"]
    encode_res = tokenizer.encode(
        text, pad_to_max_seq_len=False, return_attention_mask=True)

    # Decide whether to delete special characters according to the parameters
    if not add_special_ids:
        # TODO: use tok.all_special_ids to remove
        encode_res["input_ids"] = encode_res["input_ids"][1:-1]
        encode_res["token_type_ids"] = encode_res["token_type_ids"][1:-1]
        encode_res["attention_mask"] = encode_res["attention_mask"][1:-1]

    input_ids_list.extend(encode_res["input_ids"])
    token_type_ids_list.extend(encode_res["token_type_ids"])
    bbox_list.extend([bbox] * len(encode_res["input_ids"]))
    segment_offset_id.append(len(input_ids_list))

encoded_inputs = {
    "input_ids": input_ids_list,
    "token_type_ids": token_type_ids_list,
    "bbox": bbox_list,
    "attention_mask": [1] * len(input_ids_list),
}
# Pad the val to a particular length, and use 0 to supplement the length if it is not enough
encoded_inputs = pad_sentences(
    tokenizer,
    encoded_inputs,
    max_seq_len=max_seq_len,
    return_attention_mask=return_attention_mask)

# input_ids> 512, divide into 2 batches
ncoded_inputs = split_page(encoded_inputs)

fake_bs = encoded_inputs["input_ids"].shape[0]

encoded_inputs["image"] = paddle.to_tensor(img).unsqueeze(0).expand(

```

(continues on next page)

(continued from previous page)

```

[fake_bs] + list(img.shape))

encoded_inputs["segment_offset_id"] = segment_offset_id

return encoded_inputs

img = cv2.imread('images/input/zh_val_42.jpg')

ocr_engine = PaddleOCR(use_angle_cls=False, show_log=False)
# Perform OCR recognition
ocr_result = ocr_engine.ocr(img, cls=False)
# The OCR result is converted into the format of the dictionary, and the text box is
# converted to a bounding rectangle
ocr_info = parse_ocr_info_for_ser(ocr_result)

tokenizer = LayoutXLMTOKENIZER.from_pretrained('pretrained_models/PP-Layout_v1.0_ser_
    -pretrained')
# Resize the image,
# Tokenize the text, and convert it to the dictionary index and so on,
# Normalize the box
max_seq_length = 512
inputs = preprocess(tokenizer=tokenizer, ori_img=img, ocr_info=ocr_info, max_seq_len=max_
    -seq_length, img_size=(224, 224))

print(inputs.keys())
print(inputs['image'].shape)

```

Corrupt JPEG data: premature end of data segment

```

dict_keys(['input_ids', 'token_type_ids', 'bbox', 'attention_mask', 'image',
    'segment_offset_id'])
[2, 3, 224, 224]

```

The processed data are a dictionary which contains the following fields:

| Field | Meaning |
|-----------------------------|---|
| image | Image resized in 224*224 |
| bbox | Box normalized to 0-1000 |
| input_ids | The index of the tokenized text segment in the vocabulary |
| to- ken_type_ids | Category information of the text segment |
| atten- tion_mask | It masks the text segment, marks the corresponding position of the special character as 0, and that of the text segment as 1. |
| seg- ment_offset_id | Record the ending position of each text in input_ids |

SER Network

SER refers to Semantic Entity Recognition, which can recognize and classify texts in images. A fully connected classification head is added to the output of the SER network LayoutXLMMModel, and the code is as follows:

```
# https://github.com/PaddlePaddle/PaddleNLP/blob/develop/paddlenlp/transformers/
→layoutxlm/modeling.py#L846

from paddlenlp.transformers import LayoutXLMPretrainedModel
from paddle import nn
class LayoutXLMForTokenClassification(LayoutXLMPretrainedModel):
    def __init__(self, layoutxlm, num_classes=2, dropout=None):
        super(LayoutXLMForTokenClassification, self).__init__()
        self.num_classes = num_classes
        if isinstance(layoutxlm, dict):
            self.layoutxlm = LayoutXLMMModel(**layoutxlm)
        else:
            self.layoutxlm = layoutxlm
        self.dropout = nn.Dropout(dropout if dropout is not None else self.layoutxlm.
→config["hidden_dropout_prob"])
        self.classifier = nn.Linear(self.layoutxlm.config["hidden_size"], num_classes)
        self.classifier.apply(self.init_weights)

    def get_input_embeddings(self):
        return self.layoutxlm.embeddings.word_embeddings

    def forward(self, input_ids=None, bbox=None, image=None, attention_mask=None,_
→token_type_ids=None, position_ids=None, head_mask=None, labels=None):
        # Calculate backbone
        outputs = self.layoutxlm(input_ids=input_ids, bbox=bbox, image=image,_
→attention_mask=attention_mask, token_type_ids=token_type_ids, position_ids=position_\
→ids, head_mask=head_mask)
        seq_length = input_ids.shape[1]
        # Calculate head
        sequence_output, image_output = outputs[0][:, :seq_length], outputs[0][:, seq_
→length:]
        sequence_output = self.dropout(sequence_output)
        logits = self.classifier(sequence_output)

        outputs = logits,
        # Calculate loss
        if labels is not None:
            loss_fct = nn.CrossEntropyLoss()

            if attention_mask is not None:
                active_loss = attention_mask.reshape([-1, ]) == 1
                active_logits = logits.reshape([-1, self.num_classes])[active_loss]
                active_labels = labels.reshape([-1, ])[active_loss]
                loss = loss_fct(active_logits, active_labels)
            else:
                loss = loss_fct(logits.reshape([-1, self.num_classes]), labels.
→reshape([-1, ]))
                outputs = (loss, ) + outputs
        return outputs
```

```
# Initialize the network
net = LayoutXMLForTokenClassification.from_pretrained('pretrained_models/PP-Layout_v1.
    →0_ser_pretrained')
net.eval()
# Perform network forward
outputs = net(input_ids=inputs["input_ids"],
               bbox=inputs["bbox"],
               image=inputs["image"],
               token_type_ids=inputs["token_type_ids"],
               attention_mask=inputs["attention_mask"])
print(outputs[0].shape)
```

```
[2, 512, 7]
```

Post-Processing

Post-processing mainly matches inferred results of the text segment output by the model and the text, and combines the inferred result with the OCR result. It mainly includes the following steps:

1. For each text, count the inferred labels of all text segments under the text;
2. Select the label with the most inferences of all text segments as the label of the text.

```
# https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.4/ppstructure/vqa/vqa_
→utils.py

import paddle
import numpy as np

from infer_ser_e2e import get_bio_label_maps

label2id_map, id2label_map = get_bio_label_maps('labels/labels_ser.txt')

def postprocess(attention_mask, preds, id2label_map):
    if isinstance(preds, paddle.Tensor):
        preds = preds.numpy()
    preds = np.argmax(preds, axis=2)

    preds_list = [[] for _ in range(preds.shape[0])]

    # keep batch info
    for i in range(preds.shape[0]):
        for j in range(preds.shape[1]):
            if attention_mask[i][j] == 1:
                preds_list[i].append(id2label_map[preds[i][j]])

    return preds_list

def merge_preds_list_with_ocr_info(ocr_info, segment_offset_id, preds_list,
                                   label2id_map_for_draw):
    # list flatten
    preds = [p for pred in preds_list for p in pred]

    # Convert the dictionary of label2idx to the field of idx2label, and remove the
    →prefixes of B- and I-
```

(continues on next page)

(continued from previous page)

```

id2label_map = dict()
for key in label2id_map_for_draw:
    val = label2id_map_for_draw[key]
    if key == "O":
        id2label_map[val] = key
    if key.startswith("B-") or key.startswith("I-"):
        id2label_map[val] = key[2:]
    else:
        id2label_map[val] = key
print("id2label_map:", id2label_map)

# Count the inferred label of each text
for idx in range(len(segment_offset_id)):
    if idx == 0:
        start_id = 0
    else:
        start_id = segment_offset_id[idx - 1]

    end_id = segment_offset_id[idx]
    # Take out the range of text in the output
    curr_pred = preds[start_id:end_id]
    # Take out all the inference results of the text in the output
    curr_pred = [label2id_map_for_draw[p] for p in curr_pred]

    if len(curr_pred) <= 0:
        pred_id = 0
    else:
        # print("pred label:", curr_pred)
        # Count the label
        counts = np.bincount(curr_pred)
        # print("counts:", counts)
        pred_id = np.argmax(counts)
    ocr_info[idx]["pred_id"] = int(pred_id)
    ocr_info[idx]["pred"] = id2label_map[int(pred_id)]
    # print("pred label:", id2label_map[int(pred_id)])
return ocr_info

preds = postprocess(inputs["attention_mask"], outputs[0], id2label_map)

# Replace the value label whose initial is I with that beginning with B
label2id_map_for_draw = dict()
for key in label2id_map:
    if key.startswith("I-"):
        label2id_map_for_draw[key] = label2id_map["B" + key[1:]]
    else:
        label2id_map_for_draw[key] = label2id_map[key]
print("label2id_map:", label2id_map)
print("label2id_map_for_draw:", label2id_map_for_draw)
# Combine the forecast information and the OCR information
ocr_info_with_ser = merge_preds_list_with_ocr_info(ocr_info, inputs["segment_offset_id"]
    ↪, preds, label2id_map_for_draw)
print(ocr_info_with_ser)

```

Part of the output information is as follows

```

label2id_map: {'O': 0, 'B-QUESTION': 1, 'I-QUESTION': 2, 'B-ANSWER': 3, 'I-ANSWER': 4,
    ↵ 'B-HEADER': 5, 'I-HEADER': 6}
label2id_map_for_draw: {'O': 0, 'B-QUESTION': 1, 'I-QUESTION': 1, 'B-ANSWER': 3, 'I-
    ↵ANSWER': 3, 'B-HEADER': 5, 'I-HEADER': 5}
id2label_map: {0: 'O', 1: 'QUESTION', 3: 'ANSWER', 5: 'HEADER'}
result: [
    {'text': '问题1', 'bbox': [1026.0, 292.0, 1495.0, 377.0], 'poly': [[1027.0, 292.0], ↵
        ↵[1495.0, 300.0], [1494.0, 377.0], [1026.0, 369.0]], 'pred_id': 5, 'pred': 'HEADER'},
    {'text': '问题2', 'bbox': [207.0, 424.0, 587.0, 475.0], 'poly': [[207.0, 424.0], ↵
        ↵[587.0, 424.0], [587.0, 475.0], [207.0, 475.0]], 'pred_id': 1, 'pred': 'QUESTION'},
    {'text': '答案1', 'bbox': [1144.0, 526.0, 1218.0, 566.0], 'poly': [[1144.0, 526.0], ↵
        ↵[1218.0, 526.0], [1218.0, 566.0], [1144.0, 566.0]], 'pred_id': 1, 'pred': 'QUESTION'
    ↵}
    ...
]

```

8.3.3 Training

This section takes the XFUN Chinese dataset as an example to introduce how to train, evaluate, and test the SER model.

Data Preparation

Here, the **XFUN** dataset is used as the experimental dataset. It is a multilingual dataset proposed by Microsoft for KIE tasks. It contains seven datasets, each of which has 149 training sets and 50 validation sets.

- ZH (Chinese)
- JA (Japanese)
- ES (Spain)
- FR (French)
- IT (Italy)
- DE (German)
- PT (Portuguese)

This experiment uses the Chinese dataset for demonstration and the French dataset for the practical course. Samples of the datasets are shown in the figure below

You can run the following command to download and unzip the Chinese dataset, or download it from <https://github.com/doc-analysis/XFUND>.

```

! wget https://paddleocr.bj.bcebos.com/dataset/XFUND.tar
! tar -xf XFUND.tar

# Use the following code to convert other datasets of XFUN
# https://github.com/PaddlePaddle/PaddleOCR/blob/release%2F2.4/ppstructure/vqa/helper/
→trans_xfun_data.py

```

File ‘XFUND.tar’ already there; not retrieving.

After that, there are 2 folders in the /home/aistudio/PaddleOCR/ppstructure/vqa/XFUND, and the directory structure is as follows:

```
/home/aistudio/PaddleOCR/ppstructure/vqa/XFUND
└ zh_train/ training set
    ├── image/ image storage folder
    └── xfun_normalize_train.json label information
└ zh_val/ verification set
    ├── image/ image storage folder
    └── xfun_normalize_val.json label information
```

The label format of this dataset is:

```
{
    "height": 3508, # Image height
    "width": 2480, # Image width
    "ocr_info": [
        {
            "text": "\u2022\u2022\u2022", # Text content
            "label": "question", # The category of the text
            "bbox": [261, 802, 483, 859], # Text box
            "id": 54, # Text Index
            "linking": [[54, 60]], # The relationship between the current text and other texts [question, answer]
            "words": []
        },
        {
            "text": "\u2022\u2022\u2022\u2022\u2022\u2022\u2022\u2022", # Text content
            "label": "answer",
            "bbox": [487, 810, 862, 859],
            "id": 60,
            "linking": [[54, 60]],
            "words": []
        }
    ]
}
```

Loss Function

Since it is a classification problem, choose CrossEntropyLoss as the loss function

Model Training

After processing the data processing and defining the loss function, you can start to train the model.

The commands are as follows:

```
! python train_ser.py \
--model_name_or_path "layoutxlm-base-uncased" \
--ser_model_type "LayoutXLM" \
--train_data_dir "XFUND/zh_train/image" \
--train_label_path "XFUND/zh_train/xfun_normalize_train.json" \
--eval_data_dir "XFUND/zh_val/image" \
--eval_label_path "XFUND/zh_val/xfun_normalize_val.json" \
```

(continues on next page)

(continued from previous page)

```
--per_gpu_train_batch_size 8 \
--per_gpu_eval_batch_size 8 \
--num_train_epochs 200 \
--eval_steps 10 \
--output_dir "./output/ser/" \
--learning_rate 5e-5 \
--warmup_steps 50 \
--evaluate_during_training \
--num_workers 0 \
--seed 2048
```

During the training process, the following log will be output

```
[2021/12/26 20:12:07] root INFO: ----- Configuration Arguments -----
[2021/12/26 20:12:07] root INFO: adam_epsilon: 1e-08
[2021/12/26 20:12:07] root INFO: det_model_dir: None
[2021/12/26 20:12:07] root INFO: eval_data_dir: XFUND/zh_val/image
[2021/12/26 20:12:07] root INFO: eval_label_path: XFUND/zh_val/xfun_normalize_val.json
[2021/12/26 20:12:07] root INFO: eval_steps: 10
[2021/12/26 20:12:07] root INFO: evaluate_during_training: True
[2021/12/26 20:12:07] root INFO: infer_imgs: None
[2021/12/26 20:12:07] root INFO: label_map_path: ./labels/labels_ser.txt
[2021/12/26 20:12:07] root INFO: learning_rate: 5e-05
[2021/12/26 20:12:07] root INFO: max_grad_norm: 1.0
[2021/12/26 20:12:07] root INFO: max_seq_length: 512
[2021/12/26 20:12:07] root INFO: model_name_or_path: layoutxlm-base-uncased
[2021/12/26 20:12:07] root INFO: num_train_epochs: 200
[2021/12/26 20:12:07] root INFO: num_workers: 0
[2021/12/26 20:12:07] root INFO: ocr_json_path: None
[2021/12/26 20:12:07] root INFO: output_dir: ./output/ser/
[2021/12/26 20:12:07] root INFO: per_gpu_eval_batch_size: 8
[2021/12/26 20:12:07] root INFO: per_gpu_train_batch_size: 8
[2021/12/26 20:12:07] root INFO: re_model_name_or_path: None
[2021/12/26 20:12:07] root INFO: rec_model_dir: None
[2021/12/26 20:12:07] root INFO: resume: False
[2021/12/26 20:12:07] root INFO: seed: 2048
[2021/12/26 20:12:07] root INFO: ser_model_type: LayoutXLM
[2021/12/26 20:12:07] root INFO: train_data_dir: XFUND/zh_train/image
[2021/12/26 20:12:07] root INFO: train_label_path: XFUND/zh_train/xfun_normalize_
➥train.json
[2021/12/26 20:12:07] root INFO: warmup_steps: 50
[2021/12/26 20:12:07] root INFO: weight_decay: 0.0
[2021/12/26 20:12:07] root INFO: -----
[2021-12-26 20:12:07,259] [      INFO] - Already cached /home/aistudio/.paddlenlp/
➥models/layoutxlm-base-uncased/sentencepiece.bpe.model
[2021-12-26 20:12:07,928] [      INFO] - Already cached /home/aistudio/.paddlenlp/
➥models/layoutxlm-base-uncased/model_state.pdparams
W1226 20:12:07.929606 1085 device_context.cc:447] Please NOTE: device: 0, GPU_
➥Compute Capability: 7.0, Driver API Version: 10.1, Runtime API Version: 10.1
W1226 20:12:07.933472 1085 device_context.cc:465] device: 0, cuDNN Version: 7.6.
[2021/12/26 20:12:18] root INFO: train from scratch
[2021/12/26 20:12:18] root INFO: ***** Running training *****
[2021/12/26 20:12:18] root INFO: Num examples = 149
[2021/12/26 20:12:18] root INFO: Num Epochs = 200
[2021/12/26 20:12:18] root INFO: Instantaneous batch size per GPU = 8
[2021/12/26 20:12:18] root INFO: Total train batch size (w. parallel, distributed) =
➥= 8
```

(continues on next page)

(continued from previous page)

```
[2021/12/26 20:12:18] root INFO: Total optimization steps = 3800
[2021/12/26 20:12:20] root INFO: epoch: [0/200], iter: [0/19], global_step:1, train_
↳ loss: 1.983819, lr: 0.000001, avg_reader_cost: 1.32728 sec, avg_batch_cost: 1.49863_
↳ sec, avg_samples: 8.00000, ips: 5.33822 images/sec
[2021/12/26 20:12:21] root INFO: epoch: [0/200], iter: [1/19], global_step:2, train_
↳ loss: 1.935008, lr: 0.000002, avg_reader_cost: 0.61179 sec, avg_batch_cost: 0.72010_
↳ sec, avg_samples: 8.00000, ips: 11.10955 images/sec
[2021/12/26 20:12:23] root INFO: epoch: [0/200], iter: [2/19], global_step:3, train_
↳ loss: 1.957709, lr: 0.000003, avg_reader_cost: 0.75516 sec, avg_batch_cost: 0.85305_
↳ sec, avg_samples: 8.00000, ips: 9.37815 images/sec
Corrupt JPEG data: 18 extraneous bytes before marker 0xc4
[2021/12/26 20:12:24] root INFO: epoch: [0/200], iter: [3/19], global_step:4, train_
↳ loss: 1.842568, lr: 0.000004, avg_reader_cost: 0.76927 sec, avg_batch_cost: 0.86650_
↳ sec, avg_samples: 8.00000, ips: 9.23258 images/sec
[2021/12/26 20:12:25] root INFO: epoch: [0/200], iter: [4/19], global_step:5, train_
↳ loss: 1.941558, lr: 0.000005, avg_reader_cost: 0.67992 sec, avg_batch_cost: 0.77854_
↳ sec, avg_samples: 8.00000, ips: 10.27559 images/sec
[2021/12/26 20:12:26] root INFO: epoch: [0/200], iter: [5/19], global_step:6, train_
↳ loss: 1.879326, lr: 0.000006, avg_reader_cost: 0.62112 sec, avg_batch_cost: 0.71867_
↳ sec, avg_samples: 8.00000, ips: 11.13167 images/sec
[2021/12/26 20:12:27] root INFO: epoch: [0/200], iter: [6/19], global_step:7, train_
↳ loss: 1.833748, lr: 0.000007, avg_reader_cost: 0.79442 sec, avg_batch_cost: 0.89132_
↳ sec, avg_samples: 8.00000, ips: 8.97544 images/sec
[2021/12/26 20:12:29] root INFO: epoch: [0/200], iter: [7/19], global_step:8, train_
↳ loss: 1.747398, lr: 0.000008, avg_reader_cost: 0.74634 sec, avg_batch_cost: 0.84421_
↳ sec, avg_samples: 8.00000, ips: 9.47633 images/sec
[2021/12/26 20:12:30] root INFO: epoch: [0/200], iter: [8/19], global_step:9, train_
↳ loss: 1.603032, lr: 0.000009, avg_reader_cost: 0.79887 sec, avg_batch_cost: 0.89827_
↳ sec, avg_samples: 8.00000, ips: 8.90600 images/sec
[2021/12/26 20:12:31] root INFO: epoch: [0/200], iter: [9/19], global_step:10, train_
↳ loss: 1.678029, lr: 0.000010, avg_reader_cost: 0.78243 sec, avg_batch_cost: 0.88950_
↳ sec, avg_samples: 8.00000, ips: 8.99385 images/sec
[2021/12/26 20:12:33] root INFO: [Eval]process: 0/7, loss: 1.41839
[2021/12/26 20:12:34] root INFO: [Eval]process: 1/7, loss: 1.60403
[2021/12/26 20:12:35] root INFO: [Eval]process: 2/7, loss: 1.70345
[2021/12/26 20:12:36] root INFO: [Eval]process: 3/7, loss: 1.60751
[2021/12/26 20:12:38] root INFO: [Eval]process: 4/7, loss: 1.49639
Corrupt JPEG data: premature end of data segment
[2021/12/26 20:12:39] root INFO: [Eval]process: 5/7, loss: 1.66062
[2021/12/26 20:12:39] root INFO: [Eval]process: 6/7, loss: 1.56035
[2021/12/26 20:12:40] root INFO:
    precision    recall   f1-score   support
ANSWER      0.01      0.01      0.01     1514
HEADER      0.00      0.00      0.00       58
QUESTION     0.03      0.02      0.02     1155
micro avg   0.02      0.01      0.01     2727
macro avg   0.01      0.01      0.01     2727
weighted avg 0.02      0.01      0.01     2727
[2021/12/26 20:12:40] root INFO: ***** Eval results *****
[2021/12/26 20:12:40] root INFO: f1 = 0.013078227173649792
[2021/12/26 20:12:40] root INFO: loss = 1.5786780970437186
[2021/12/26 20:12:40] root INFO: precision = 0.01925820256776034
[2021/12/26 20:12:40] root INFO: recall = 0.009900990099009901
```

(continues on next page)

(continued from previous page)

```
[2021/12/26 20:12:44] root INFO: Saving model checkpoint to ./output/ser/best_model
[2021/12/26 20:12:44] root INFO: [epoch 0/200][iter: 9/19] results: {'loss': 1.
↪5786780970437186, 'precision': 0.01925820256776034, 'recall': 0.009900990099009901,
↪'f1': 0.013078227173649792}
[2021/12/26 20:12:44] root INFO: best metrics: {'loss': 1.5786780970437186, 'precision
↪': 0.01925820256776034, 'recall': 0.009900990099009901, 'f1': 0.013078227173649792}
...
...
```

Model Evaluation

During the training process, two models are saved by default, one is the latest trained model named latest, and the other is the most accurate model named best. The folder structure for saving the model is as follows:

```
output/ser/
└── best_model
    ├── model_config.json # Model configuration
    ├── model_state.pdparams # Model parameters
    ├── sentencepiece.bpe.model # Parameters of tokenizer
    ├── tokenizer_config.json # tokenizer configuration
    └── training_args.bin # Parameters for starting training
    └── infer_results.txt
    └── latest_model
        ├── model_config.json
        ├── model_state.pdparams
        ├── sentencepiece.bpe.model
        ├── tokenizer_config.json
        └── training_args.bin
    └── test_gt.txt
    └── test_pred.txt
    └── train.log # Training log
```

Next, use the saved model parameters to evaluate the accuracy on the test set:

```
! python eval_ser.py \
--model_name_or_path "output/ser/best_model" \
--ser_model_type "LayoutXLM" \
--eval_data_dir "XFUND/zh_val/image" \
--eval_label_path "XFUND/zh_val/xfun_normalize_val.json" \
--per_gpu_eval_batch_size 8 \
--num_workers 8 \
--output_dir "output/ser/" \
--seed 2048
```

The following log will be output during the evaluation process

```
[2021/12/26 20:13:05] root INFO: ----- Configuration Arguments -----
[2021/12/26 20:13:05] root INFO: adam_epsilon: 1e-08
[2021/12/26 20:13:05] root INFO: det_model_dir: None
[2021/12/26 20:13:05] root INFO: eval_data_dir: XFUND/zh_val/image
[2021/12/26 20:13:05] root INFO: eval_label_path: XFUND/zh_val/xfun_normalize_val.json
[2021/12/26 20:13:05] root INFO: eval_steps: 10
[2021/12/26 20:13:05] root INFO: evaluate_during_training: False
[2021/12/26 20:13:05] root INFO: infer_imgs: None
```

(continues on next page)

(continued from previous page)

```
[2021/12/26 20:13:05] root INFO: label_map_path: ./labels/labels_ser.txt
[2021/12/26 20:13:05] root INFO: learning_rate: 5e-05
[2021/12/26 20:13:05] root INFO: max_grad_norm: 1.0
[2021/12/26 20:13:05] root INFO: max_seq_length: 512
[2021/12/26 20:13:05] root INFO: model_name_or_path: output/ser/best_model
[2021/12/26 20:13:05] root INFO: num_train_epochs: 3
[2021/12/26 20:13:05] root INFO: num_workers: 8
[2021/12/26 20:13:05] root INFO: ocr_json_path: None
[2021/12/26 20:13:05] root INFO: output_dir: output/ser/
[2021/12/26 20:13:05] root INFO: per_gpu_eval_batch_size: 8
[2021/12/26 20:13:05] root INFO: per_gpu_train_batch_size: 8
[2021/12/26 20:13:05] root INFO: re_model_name_or_path: None
[2021/12/26 20:13:05] root INFO: rec_model_dir: None
[2021/12/26 20:13:05] root INFO: resume: False
[2021/12/26 20:13:05] root INFO: seed: 2048
[2021/12/26 20:13:05] root INFO: ser_model_type: LayoutXLM
[2021/12/26 20:13:05] root INFO: train_data_dir: None
[2021/12/26 20:13:05] root INFO: train_label_path: None
[2021/12/26 20:13:05] root INFO: warmup_steps: 0
[2021/12/26 20:13:05] root INFO: weight_decay: 0.0
[2021/12/26 20:13:05] root INFO: -----
W1226 20:13:05.816488 1230 device_context.cc:447] Please NOTE: device: 0, GPU_ ↵Compute Capability: 7.0, Driver API Version: 10.1, Runtime API Version: 10.1
W1226 20:13:05.820412 1230 device_context.cc:465] device: 0, cuDNN Version: 7.6.
Corrupt JPEG data: premature end of data segment
[2021/12/26 20:13:18] root INFO: [Eval]process: 0/7, loss: 1.41839
[2021/12/26 20:13:18] root INFO: [Eval]process: 1/7, loss: 1.60403
[2021/12/26 20:13:19] root INFO: [Eval]process: 2/7, loss: 1.70345
[2021/12/26 20:13:19] root INFO: [Eval]process: 3/7, loss: 1.60751
[2021/12/26 20:13:19] root INFO: [Eval]process: 4/7, loss: 1.49639
[2021/12/26 20:13:19] root INFO: [Eval]process: 5/7, loss: 1.66062
[2021/12/26 20:13:19] root INFO: [Eval]process: 6/7, loss: 1.56035
[2021/12/26 20:13:20] root INFO:
    precision      recall   f1-score   support
ANSWER       0.01      0.01      0.01     1514
HEADER       0.00      0.00      0.00      58
QUESTION      0.03      0.02      0.02    1155
micro avg    0.02      0.01      0.01    2727
macro avg    0.01      0.01      0.01    2727
weighted avg  0.02      0.01      0.01    2727

[2021/12/26 20:13:20] root INFO: ***** Eval results *****
[2021/12/26 20:13:20] root INFO:   f1 = 0.013078227173649792
[2021/12/26 20:13:20] root INFO:   loss = 1.5786780970437186
[2021/12/26 20:13:20] root INFO:   precision = 0.01925820256776034
[2021/12/26 20:13:20] root INFO:   recall = 0.009900990099009901
[2021/12/26 20:13:20] root INFO: {'loss': 1.5786780970437186, 'precision': 0.01925820256776034, 'recall': 0.009900990099009901, 'f1': 0.013078227173649792}
```

Model Inference

After training the model, you can also use the saved model to perform model inference on a single picture or an image in a folder, and observe the inference result of the model.

```
! python infer_ser_e2e.py \
    --model_name_or_path "./pretrained_models/PP-Layout_v1.0_ser_pretrained/" \
    --ser_model_type "LayoutXLM" \
    --max_seq_length 512 \
    --output_dir "output/ser_e2e/" \
    --infer_imgs "images/input/zh_val_42.jpg"
```

During the inference process, the following log will be output

```
process: [0/1], save result to output/ser_e2e/zh_val_42_ser.jpg
```

8.3.4 Assignment

Experiment

<https://aistudio.baidu.com/aistudio/projectdetail/3281385>

END-TO-END ALGORITHM

9.1 Background

OCR algorithms mainly consists of two algorithms: two-stage algorithms and end-to-end algorithms. Two-stage OCR algorithms include two tasks: text detection and recognition. The text detection algorithm is to locate text regions of the input image, and the text recognition algorithm is to recognize the text content in the image. Basically, end-to-end OCR algorithms aim to integrate detection and recognition in a unified framework. The two parts share the same backbone network but have specialized modules for detection and recognition, so that they can be trained at the same time. The end-to-end algorithm simplifies the process, therefore the model is smaller and the processing speed is faster.

In this section, We introduce some end-to-end text recognition methods based on deep learning in recent years. These approaches can be broadly classified into two categories^[1]

1) End-to-end regular text recognition.

2) End-to-end arbitrary-shaped text recognition.

End-to-end regular text recognition algorithms mainly address the detection and recognition of horizontal or multi-directional texts. However, there are a large number of curved and distorted text in natural scenes, such as seals. In detecting and recognizing these texts, end-to-end arbitrary-shaped text recognition algorithms are needed. At the same time, these algorithms can also fit regular texts.

This section has filtered out some representative end-to-end recognition methods from 2017 to 2021, and their classification according to the above two categories is shown in the table below.

Table 1 End-to-end recognition methods

| Category | Main papers |
|--|---|
| End-to-end regular text recognition methods | FOTS, TextSpotter |
| End-to-end arbitrary-shaped text recognition methods | Mask TextSpotterv1, Mask TextSpotter2, Mask TextSpotterv3, TextDragon, CharNet, TUTS, ABCNet, ABCNetV2, Text Perceptron, PGNet, PAN++ |

Note: If there is any omission in the table, and if you have any questions, please contact us at [link](#).

9.2 Algorithms

9.2.1 End-to-end Regular Text Recognition Algorithms

Xuebo Liu and Ding Liang et al. (2018) [1] propose a unified end-to-end trainable network FOTS (Fast Oriented Text Spotting) for simultaneous detection and recognition. Its network structure is shown in Figure 1. It consists of four parts: shared convolutions, the text detection branch, ROI Rotate operation and the text recognition branch.

1 Use shared convolutions to extract feature maps. By sharing convolutional features the network can detect and recognize text simultaneously with little computation overhead, which leads to real-time speed;

2 Construct a text detection branch based on the fully convolutional network (FCN) after the feature maps are extracted, in order to predict the detection bounding boxes;

3 The ROI Rotate operator extracts the oriented text regions from convolutional feature maps. This operation unifies text detection and recognition into an end-to-end pipeline.

4 Finally, the text proposal features are fed into Recurrent Neural Network (RNN) encoder and Connectionist Temporal Classification (CTC) decoder for text recognition.

Since all the modules in the network are differentiable, the whole system can be trained end-to-end, which does not require complicated post-processing and hyperparameter tuning.

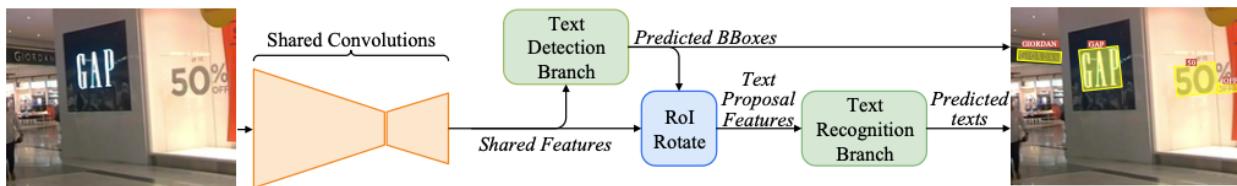


Fig. 1 Overall structure of the FOTS model

To validate the FOTS, the authors have conducted experiments on ICDAR 2015, ICDAR 2017 MLT and ICDAR 2013 datasets and the visualised results are shown in Figure 2.

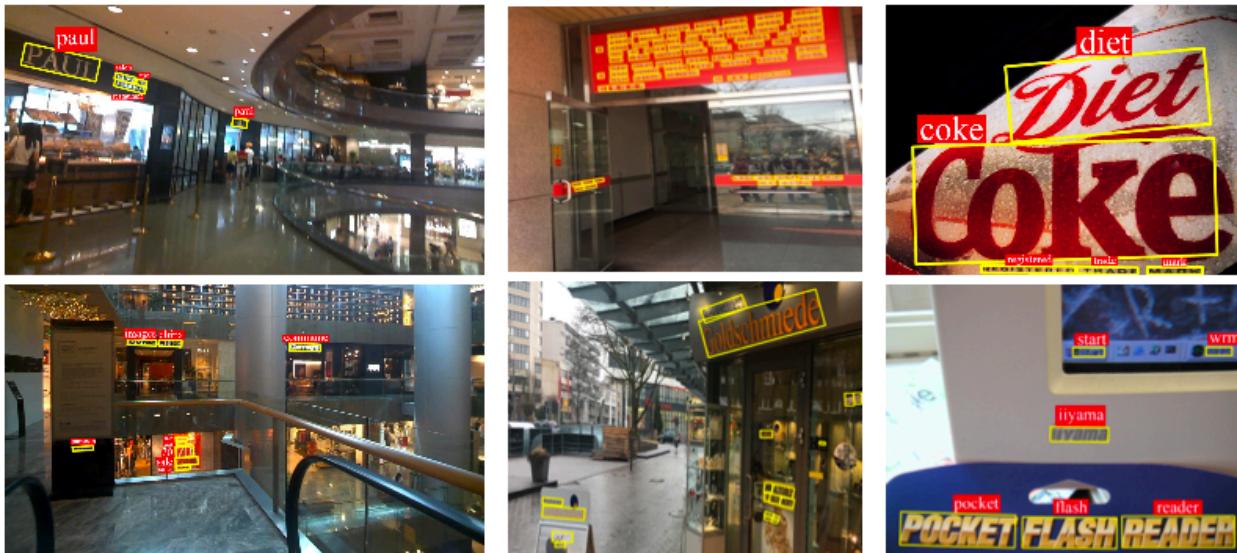


Fig. 2 The result of FOTS

Note: As ICDAR 2017 MLT does not involve the text recognition, only the text detection result is shown.

Tong He et al.(2018)[2] propose a simple but effective model, TextSpotter[2] the overall architecture is presented in **Figure 3**. The detection model is a fully convolution architecture built on the PVANet framework and introduce a new recurrent branch for word recognition. TextSpotter can process the detection and recognition in one shot. The RNN branch is composed of a new text-alignment layer and a LSTM-based recurrent module with a novel character attention embedding mechanism. This method develops a text-alignment layer by introducing a grid sampling scheme instead of conventional ROI pooling. It computes fixed-length convolutional features that precisely align to a detected text region of arbitrary orientation. Also, the character attention mechanism by using character spatial information as an additional supervision is introduced and make the RNN focus on current attentional features. Finally, TextSpotter adopts a learning strategy, this allows text detection and recognition to work collaboratively by sharing convolutional features.

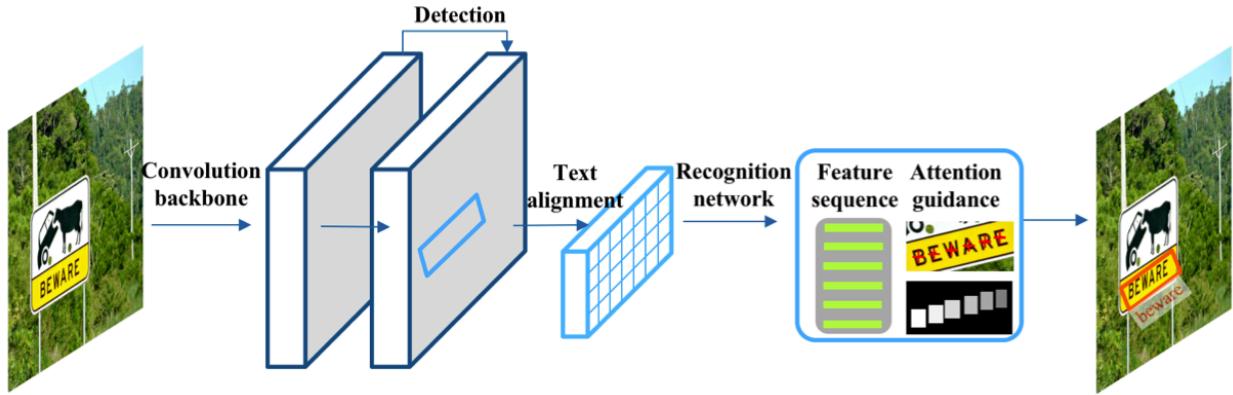


Fig. 3 The overall structure of the TextSpotter model

To validate the TextSpotter[2] authors have experimented on ICDAR2013 and ICDAR2015 datasets, and its visualized result is as follows in **Figure 4**.



Fig. 4 The result of TextSpotter

9.2.2 End-to-end Arbitrary-shaped Text Recognition Algorithms

Pengyuan Lyu and Minghui Liao et al. (2018) [3] propose Mask TextSpotter, an end-to-end trainable neural network model for scene text spotting. It can detect and recognize text instances (the right in Figure 5) in arbitrary shapes (horizontal, oriented, or curved) unlike some methods that can only detect and recognize horizontal texts (the left in Figure 5) or oriented texts (the middle in Figure 5).

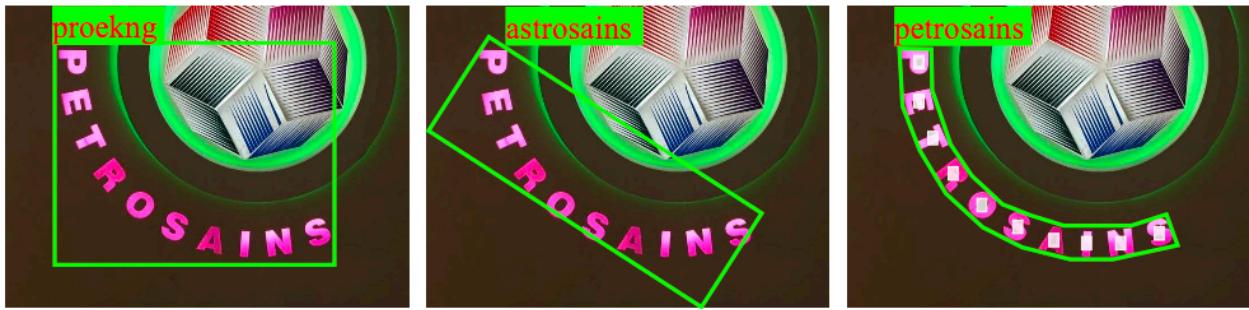


Fig. 5 The comparison of available text types in recognition

Inspired by Mask R-CNN, Mask TextSpotter can detect text by segmenting the instance text regions. Thus it is able to detect text of arbitrary shapes. The network structure of Mask TextSpotter is shown in Figure 6.

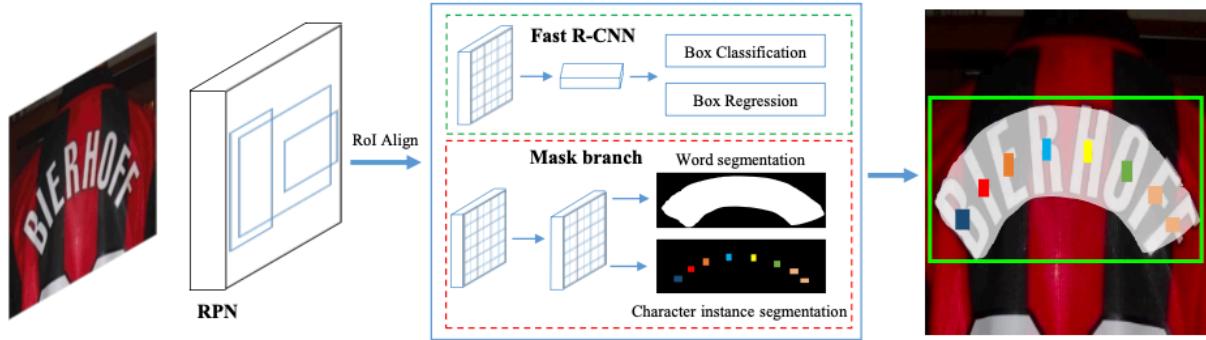


Fig. 6 Overall structure of the Mask TextSpotter model

To verify the effectiveness of the Mask TextSpotter, the authors have conducted experiments on ICDAR2013, ICDAR2015 and Total-Text datasets, and the visualized results are as shown in **Figure 7**.



Fig. 7 The results of Mask TextSpotter

Minghui Liao et al. (2019) [4] propose Mask TextSpotter V2 based on Mask TextSpotter. Mask TextSpotter recognizes single characters, requires positions of characters during training, and post-processing algorithm is required to yield text sequence. To overcome these limitations, Mask TextSpotter V2 introduces a spatial attention module (SAM). This method applies a Spatial Attention Module (SAM) for the recognition part, which can globally predict the label sequence of each word with a spatial attention mechanism. SAM only requires the word-level annotations for training, significantly reducing the need of character- level annotations for training. The network structure of Mask TextSpotter V2 is shown in **Figure 8**.

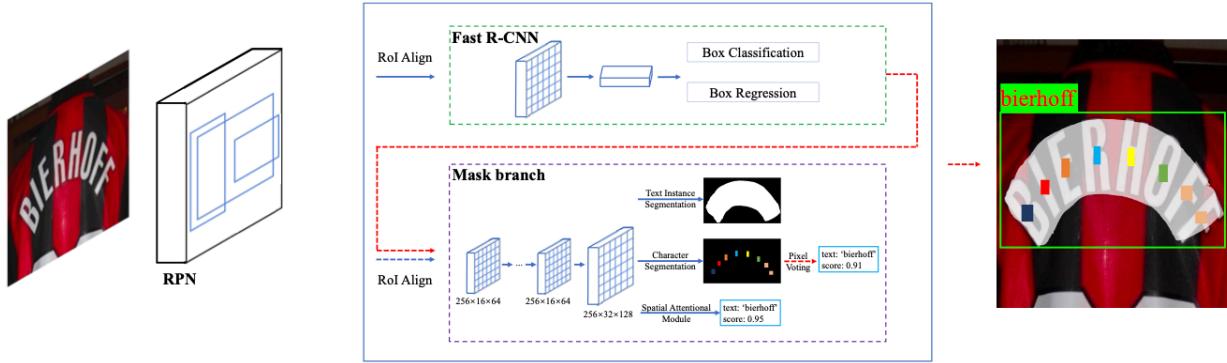


Figure 8 Overall structure of the Mask TextSpotter V2 model

To verify the effectiveness of the Mask TextSpotter V2, the authors have conducted experiments on five datasets, ICDAR 2013, ICDAR 2015, COCO-Text, Total-Text, and MIT. The visualized results on the ICDAR 2013 (first column), ICDAR 2015 (second column) and Total-Text (last two columns) datasets are shown in **Figure 9**.

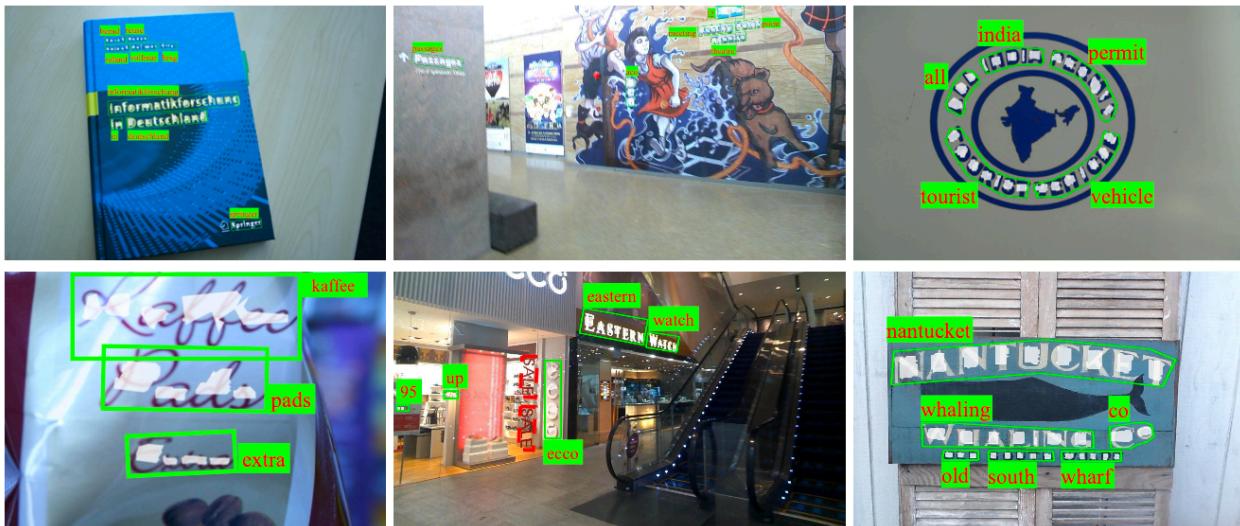


Figure 9 The result of Mask TextSpotter V2

Minghui Liao et al. (2020) [5] also propose Mask TextSpotter V3 based on Mask TextSpotter V1 and V2. In the previous versions, text detection module is based on Mask R-CNN, and fails to detect long text lines due to the limitation of RPN. Instead of using RPN, the authors use Segmentation Proposal Network (SPN) in V3. It is therefore superior to RPN in detecting text instances of extreme aspect ratios or irregular shapes. The authors also propose hard ROI masking, which can effectively suppress neighboring text instances or background noise. The network structure of TextSpotter V3 is shown in **Figure 10**.

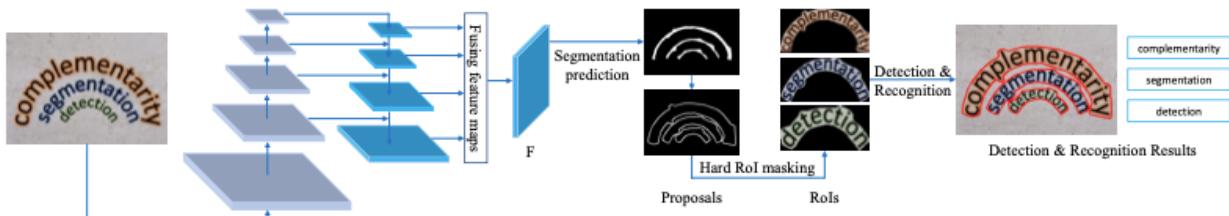


Fig. 10 Overall structure of the Mask TextSpotter V3 model

To verify the effectiveness of the Mask TextSpotter V3, the authors have conducted experiments on the Rotated ICDAR 2013, Total-Text, and MSRA-TD500 datasets. The visualized results on the Total-Text dataset are shown in **Figure 11**.



Fig. 11 The results of Mask TextSpotter V3

Wei Feng et al. (2019) [6] propose a novel text spotting framework, TextDragon, which only uses word/line-level annotations for training, and its network structure is shown in **Figure 12**. In TextDragon, a text detector is designed to describe the shape of text with a series of quadrangles, which can handle text of arbitrary shapes. In order to extract arbitrary text regions from feature maps, TextDragon has designed a new differentiable operator named RoISlide, which is the key to connect arbitrary shaped text detection and recognition. Also, Based on the extracted features through RoISlide, a CNN and CTC based text recognizer is introduced to make the framework free from labeling the location of characters. The model becomes more useful for the framework only needs word/line level annotations instead of positions of annotated characters.

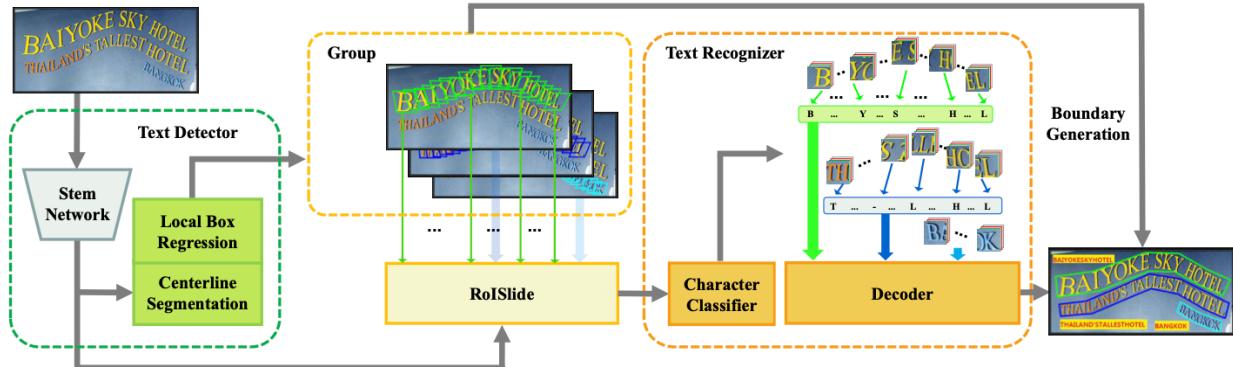


Fig. 12 Overall structure of the TextDragon model

To validate the effectiveness of the TextDragon, the authors have conducted experiments on two curved text benchmark datasets, CTW1500 and Total-Text, as well as ICDAR 2015 dataset. The visualized results are shown in **Figure 13**.



Fig. 13 The results of TextDragon

Linjie Xing et al. (2019) [7] propose Convolutional Character Networks (CharNet) for end-to-end text detection and recognition, and the network structure is shown in **Figure 14**. CharNet introduces a new branch of direct character detection and recognition, which can be integrated into existing text detection framework. The authors also explore characters as basic unit, which overcome the limitations of the RoI pooling and the RNN recognition module, both of which are major limitations of current two-stage framework. In addition, an iterative character detection method is proposed in the paper, which allows CharNet to transform the character detection capabilities learned from synthetic data to real world images. This makes it possible to train CharNet on real world images without additional annotations of character-level bounding boxes.

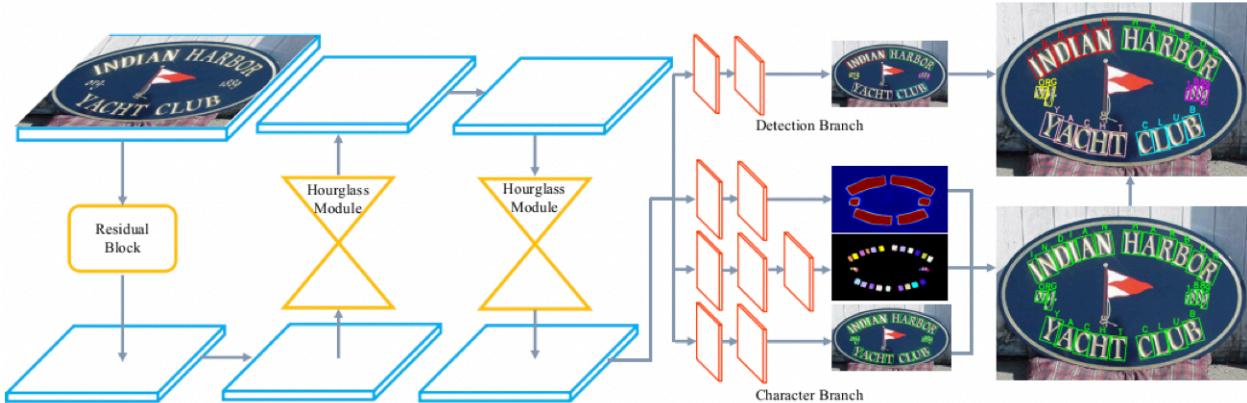


Fig. 14 Overall structure of the CharNet model

To verify the effectiveness of the CharNet method, the authors have conducted experiments on the benchmark datasets ICDAR 2015, Total-Text and ICDAR MLT 2017 and the visualized results are shown in **Figure 15**.



Fig. 15 The results of CharNet

Siyang Qin et al. (2019) [8] propose an end-to-end trainable model TUTS and its network structure is shown in **Figure 16**. TUTS is a simple and flexible OCR model based on a Mask R-CNN detector and a sequence-to-sequence(seq2seq) attention decoder. And this method can detecte and recognize text of arbitrary shape, and proposes a simple and effective ROI masking step, aiming to obtain useful irregularly shaped text instance features from image scale features. In addition, partially labeled data is automatically labeled by an existing multi-stage OCR engine, which can greatly optimize the model detection and recognition results.

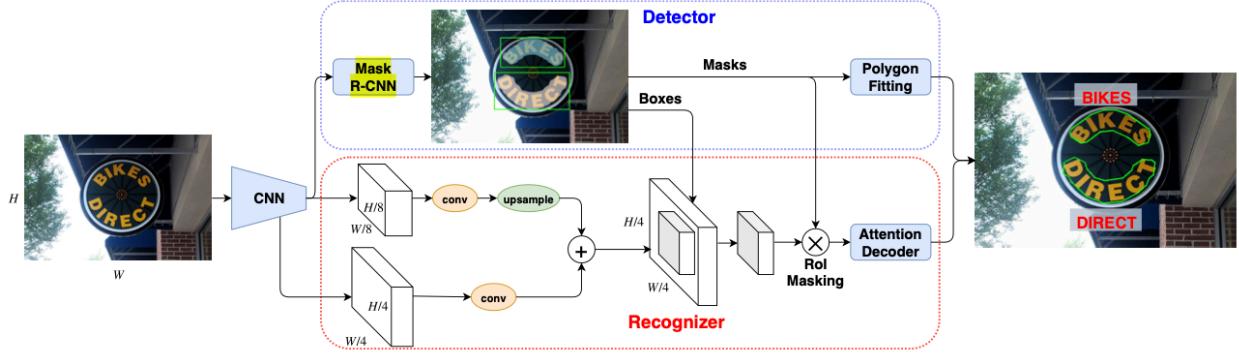


Fig. 16 Overall structure of the TUTS model

In order to verify the effectiveness of the TUTS method, the authors have conducted experiments on the benchmark dataset ICDAR2015 and Total-Text, and the visualized results are illustrated in **Figure 17**, where the two left columns are ICDAR2015 results and two right columns are results of Total-Text .



Fig. 17 The results of TUTS

The blue parts in the bottom right image indicates the errors of the inference.

Yuliang Liu et al. (2020) [9] propose an end-to-end trainable model Adaptive Bezier-Curve Network (ABCNet) for arbitrary-shaped texts, and its network structure is shown in **Figure 18**. For the first time, the authors design a new concise parametric representation of curved scene text using Bezier curves. Also, they design a novel BezierAlign layer for extracting accurate convolutional features of a text instance with arbitrary shapes. The computation overhead of the proposed Bezier curve detection method is negligible compared with previous standard detection methods. And finally, by sharing backbone features, the recognition branch can be designed as a lightweight structure. The ABCNet consists of two parts: 1) Bezier curve detection; 2) BezierAlign and recognition branch, and the network structure has advantages in efficiency and accuracy.

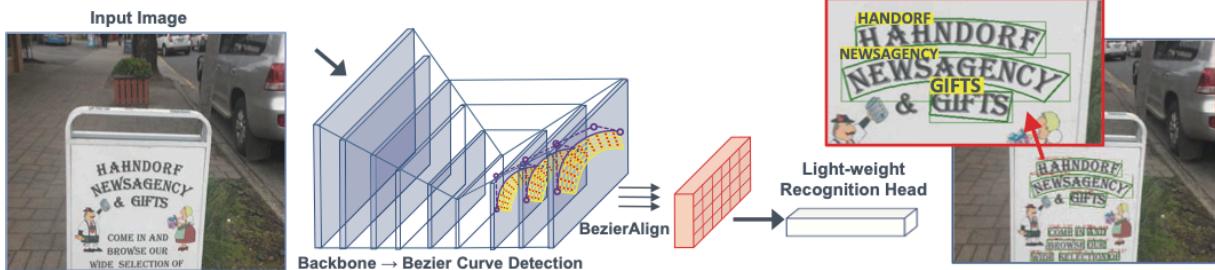


Fig. 18 Overall structure of the ABCNet model

To verify the effectiveness of the ABCNet method, the authors have conducted experiments on the arbitrary-shaped benchmark datasets Total-Text and CTW1500, and the visualized results on Total-Text dataset are shown in **Figure 19**.



Fig. 19 The results of ABCNet

Yuliang Liu et al. (2021) [10] propose ABCNet v2 based on ABCNet with improvements in four aspects: the feature extractor, detection branch, recognition branch and the end-to-end training, which can achieve state-of-the-art performance while maintaining very high efficiency. The detection model of ABCNet v2 is more general for processing multi-scale text instances by considering bidirectional multi-scale pyramidal global text features. Observing that the feature alignment in the detection branch is essential for the subsequent text recognition, the authors adopt a coordinate encoding approach with negligible computation overhead to explicitly encode the position in the convolutional filters, leading to considerable improvement in accuracy. In the recognition branch, a character attention module is integrated which can recursively predict the characters of each word without using character-level annotations. Finally, to achieve effective end-to-end training, the authors further propose an Adaptive end-to-end training (AET) strategy to match the detection for end-to-end training. The network structure of ABCNet v2 is shown in **Figure 20**.

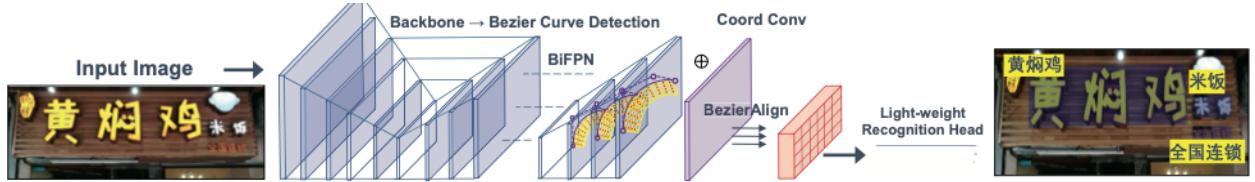


Fig. 20 Overall structure of the ABCNet v2 model

In order to validate the ABCNet v2 method, the authors have worked on ICDAR 2015, MSRA-TD500, ReCTS, TotalText, and SCUT-CTW1500, and some visualized results are shown in **Figure 21**.



Fig. 21 The results of ABCNet V2

Liang Qiao et al. (2020) [11] propose an end-to-end trainable arbitrary-shaped text recognition method, Text Perceptron, whose overall architecture is shown in **Figure 22**. This model consists of three parts. 1) An efficient segmentation-based detection module that uses ResNet and FPN as the backbone network, which describes a text region as four subregions: the central region, head, tail, top&bottom boundary regions. The boundary information not only helps separate text regions that are very close to each other, but also contributes to capture latent reading-orders. 2) A novel Shape Transform Module (STM) module is designed to transform the detected feature regions into morphologies without extra parameters, and integrates irregular text detection and recognition into an end-to-end trainable model. 3) A sequence-based recognition module for generating final character sequences.

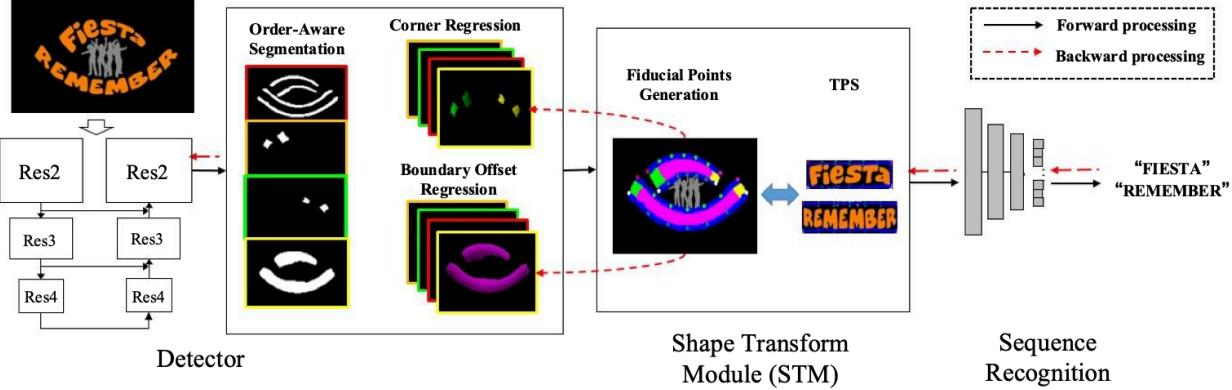


Fig. 22 Overall structure of the Text Perceptron model

To validate the Text Perceptron method, the authors have conducted experiments on SynthText 800k , ICDAR2013, ICDAR2015, Total-Text and CTW1500. The visualized results on the Total-Text and CTW1500 datasets are shown in **Figure 23**.



Fig. 23 The results of Text Perceptron

Pengfei Wang et al. (2021) [12] propose a novel fully convolutional Point Gathering Network (PGNet), whose overall architecture is shown in **Figure 24**. The input image of the PGNet model is integrated into four branches after feature extraction: TBO (Text Border Offset), TCL (Text Center Line), TDO (Text Direction Offset), and TCC map(Text Character Classification). The input of TBO and TCL can generate the text detection result after post-processing, and TCL, TDO, and TCC are for text recognition. The PGNet algorithm has exclusive advantages, including: designing PGNet loss to guide training without character-level annotations, speeding up inference without NMS and ROI operations, proposing a module to infer the reading order within text lines, and putting forward the graph refinement module (GRM) to improve the model recognition. This algorithm is more accurate and faster in inference.

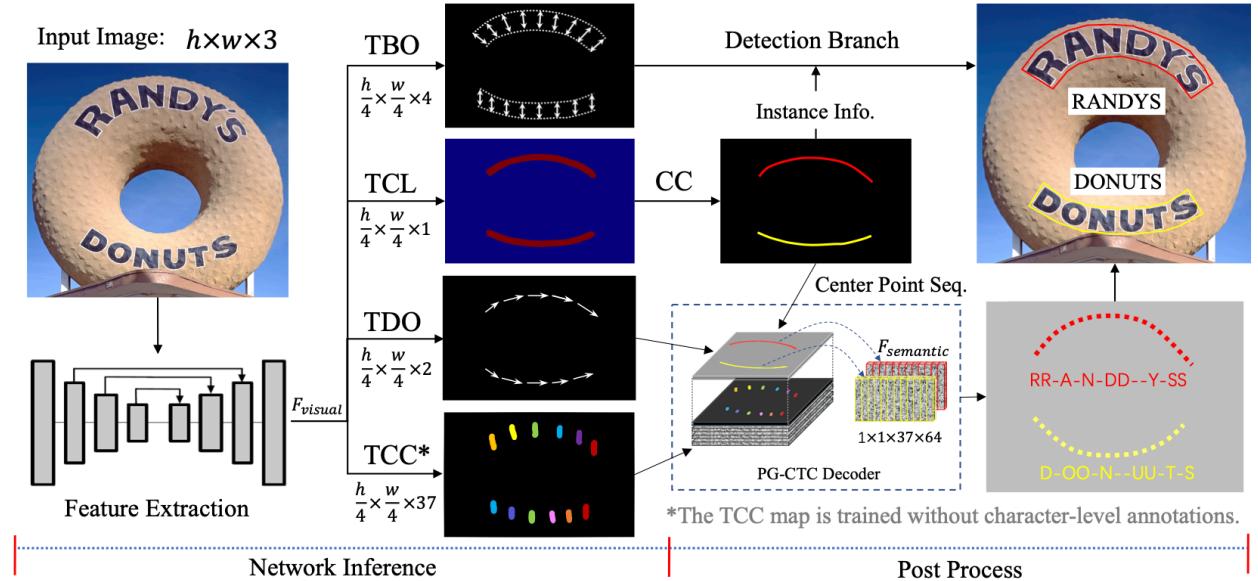


Fig. 24 Overall structure of the PGNet model

To verify the effectiveness of the PGNet method, the authors have conducted experiments on ICDAR2015 and Total-Text datasets and the visualized results are as shown in **Figure 25**, where the left two columns are the results of Total-Text and the right two columns are the results of ICDAR2015.



Fig. 25 The results of PGNet

Wenhai Wang et al. (2021) [13] propose an end-to-end text recognition algorithm, PAN++, which can effectively detect and recognize arbitrary-shaped texts in natural scenes, whose overall architecture is shown in **Figure 26**. PAN++ is based on the kernel representation that reformulates a text line as a text kernel (central region) surrounded by peripheral pixels, and can well distinguish adjacent texts. Furthermore, as a pixel-based representation, the kernel representation can be predicted by a single fully convolutional network, which is very friendly to real-time applications. Taking the advantage of the kernel representation, the authors have designed a series of components: 1) a efficient feature enhancement network composed of stacked Feature Pyramid Enhancement Modules (FPEMs); 2) a lightweight detection head cooperating with Pixel Aggregation(PA); 3) an efficient attention-based recognition head with Masked RoI.

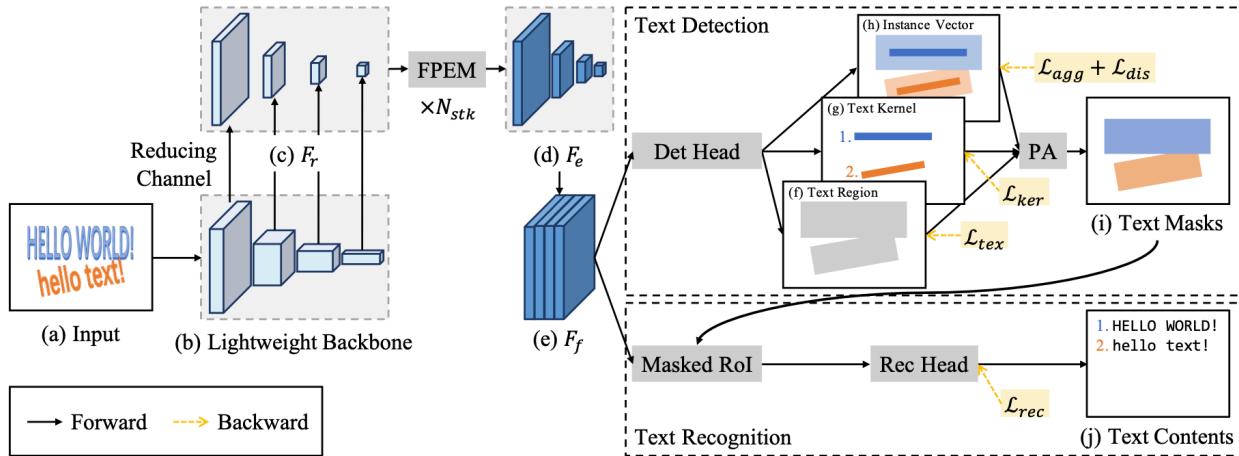


Fig. Overall structure of the PAN++ model

To validate the PAN++ method, the authors have experimented on Total-Text, CTW1500, ICDAR2015, MSRA-TD500, and RCTW-17 datasets, and the visualized results are shown in **Figure 27**.



Figure 27 The results of PAN++

9.3 Summary

OCR text detection and recognition is a basic task of many applications such as text retrieval and office automation. Most of existing work takes the text detection and recognition as two separate tasks. There is no shared feature between the two tasks and training two models is time-consuming. In recent years, some methods are committed to end-to-end text recognition methods, which aims to detect and recognize text simultaneously in one network. We summarize some end-to-end text recognition methods based on deep learning, which can be roughly divided into two categories: 1) end-to-end regular text recognition and 2) end-to-end arbitrary shape text recognition. The common algorithms of the first category include FOTS and TextSpotter, which are mainly used for regular text detection and recognition. Those common algorithms of the latter category include Mask TextSpotterV1, Mask TextSpotterV2, Mask TextSpotterV3, TextDragon, CharNet, TUTS, ABCNet, ABCNetV2, PGNet, PAN++ and so on, which are mainly used for arbitrary shape text detection and recognition.

9.4 References

- [1] Liu X, Liang D, Yan S, et al. Fots: Fast oriented text spotting with a unified network[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 5676-5685.
- [2] He T, Tian Z, Huang W, et al. An end-to-end textspotter with explicit alignment and attention[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 5020-5029.
- [3] Lyu P, Liao M, Yao C, et al. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 67-83.
- [4] Liao M, Lyu P, He M, et al. Mask TextSpotter: An End-to-End Trainable Neural Network for Spotting Text with Arbitrary Shapes[J]. arXiv preprint arXiv:1908.08207, 2019.
- [5] Liao M, Pang G, Huang J, et al. Mask textspotter v3: Segmentation proposal network for robust scene text spotting[C]//Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16. Springer International Publishing, 2020: 706-722.
- [6] Feng W, He W, Yin F, et al. TextDragon: An end-to-end framework for arbitrary shaped text spotting[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 9076-9085.
- [7] Xing L, Tian Z, Huang W, et al. Convolutional character networks[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 9126-9136.
- [8] Qin S, Bissacco A, Raptis M, et al. Towards unconstrained end-to-end text spotting[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 4704-4714.
- [9] Liu Y, Chen H, Shen C, et al. Abcnet: Real-time scene text spotting with adaptive bezier-curve network[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 9809-9818.
- [10] Liu Y, Shen C, Jin L, et al. ABCNet v2: Adaptive Bezier-Curve Network for Real-time End-to-end Text Spotting[J]. arXiv preprint arXiv:2105.03620, 2021.
- [11] Qiao L, Tang S, Cheng Z, et al. Text perceptron: Towards end-to-end arbitrary-shaped text spotting[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(07): 11899-11907.
- [12] Wang P, Zhang C, Qi F, et al. PGNet: Real-time Arbitrarily-Shaped Text Spotting with Point Gathering Network[J]. arXiv preprint arXiv:2104.05458, 2021.
- [13] Wang W, Xie E, Li X, et al. PAN++: Towards Efficient and Accurate End-to-End Spotting of Arbitrarily-Shaped Text[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.

PRE-PROCESSING ALGORITHM

10.1 Background

In OCR text detection and recognition, the quality of images directly affects the performance of detection and recognition. Low quality images often have problems such as tilt, fold, blur, virtual scene and so on. Therefor, image preprocessing is an important part for OCR. This section will focus on the common algorithms of data augmentation, image binarization and denoising in image preprocessing. Data augmentation is a commonly-used techniques in deep learning, which increases the number and diversity of samples by transforming training data to strengthen the generalization ability of the model. Image binarisation transforms the image into a black-white one, which separates the text from the background and is conducive to text detection and recognition. Denoising can remove noise interference (such as salt-and-pepper noise, Gaussian noise, etc.) in the image. It is necessary to recognize the type of noise, and then choose a suitable denoising method by considering the noise characteristics. Binarisation and denoising are preprocessing algorithms commonly used in traditional OCR, and have performed well on dealing with printed and scanned documents. They can also get clearer pictures after preprocessing algorithm, and then use the detection and recognition method based on deep learning.

In this section, some common data augmentation, binarisation, and denoising methods are sceened, as shown in the following three tables.

Table 1 Data augmentation methods

| Data Augmentation | Methods |
|----------------------------|---|
| Standard data augmentation | Rotation, perspective transformation, blurring, Gaussian noise, random cropping and so on |
| Image transformation | AutoAugment, RandAugment, TimmAutoAugment |
| Image cropping | CutOut, RandErasing, HideAndSeek, GridMask |
| Image mixture | Mixup, Cutmix |

Table 2 Binarization methods

| Binarization | Methods |
|--------------------------------|---|
| Global thresholding | Fixed thresholding, Otus |
| Local thresholding | Adaptive thresholding, NiBlack, Sauvola, Bernsen |
| Methods based on deep learning | U-Net, Grid LSTM, Full Convolutional Neural Networks etc. |

Table 3 Denoising methods

| Denoising | Methods |
|----------------------------|---|
| Spatial domain filtering | Mean filtering, Gaussian filtering, median filtering, bilateral filtering, non-local means algorithm(NLM) |
| Transform domain filtering | Fourier transform, wavelet transform |
| BM3D | BM3D |
| Deep learning | DnCNNs, FFDNet, MPRNet |

Note: If there is any omission in the table, and if you have any questions, please contact us at [link](#).

10.2 Data Augmentation

OCR handwritten and scene text detection and recognition face many problems, such as different forms, complex background, fuzzy text and so on. Therefore, training a robust recognition model requires a large amount of data to cover as many scenes as possible. Compared to data collection and annotation, data augmentation is a less costly way to improve the robustness of the model. This section focuses on some standard data augmentation methods, such as colour space transformation, blur, and noise. There are also many improved data augmentation strategies and some augmentation methods that new operations are inserted into different stages. We roughly divide these operations into four categories:

- Common data augmentation methods: rotation, perspective transformation, blur, Gaussian noise, randCrop, etc.;
- Image transformation: transform images after RandCrop. It mainly contains AutoAugment and RandAugment;
- Image cropping techniques: crop images after transposition. It mainly contains CutOut, RandErasing, HideAndSeek and GridMask;
- Image mixing techniques: mix the data after batch processing. It mainly contains Mixu and Cutmix.

First, import the modules and packages needed in the experiment.

```
import numpy as np
import cv2
import random
# When using matplotlib.pyplot in a notebook for drawing, you need to add this
# command for display
%matplotlib inline
import matplotlib.pyplot as plt
```

We read an image as a sample for the experiment, output the shape of the image, and display the image.

```
img = cv2.imread('test.jpg')
print('image shape:{}'.format(img.shape))
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img_rgb)
plt.show()
```

image shape:(90, 314, 3)



Fig.1 Example of test picture

Most images generated by data augmentation methods are somewhat random. In order to facilitate the observation of the effect of data augmentation, a drawing function `show_img` is defined below. This function displays the input image `img` and the image `new_img` after data augmentation.

```
def show_img(img, new_img):
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    new_img_rgb = cv2.cvtColor(new_img, cv2.COLOR_BGR2RGB)

    plt.subplot(1, 2, 1)
    plt.imshow(img_rgb)
    plt.title('origin')

    plt.subplot(1, 2, 2)
    plt.imshow(new_img_rgb)
    plt.title('transform')

    plt.show()
```

10.2.1 Standard Data Augmentation

Colour space transformation (`cvtColor`): it transforms an image from one colour space to another.

```
def flag():
    """
    flag
    """
    return 1 if random.random() > 0.5000001 else -1

def cvtColor(img):
    """
    cvtColor
    """
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    delta = 0.001 * random.random() * flag()
    hsv[:, :, 2] = hsv[:, :, 2] * (1 + delta)
    new_img = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
```

(continues on next page)

(continued from previous page)

```

    return new_img

cvtColor_img = cvtColor(img)
show_img(img, cvtcolor_img)

```

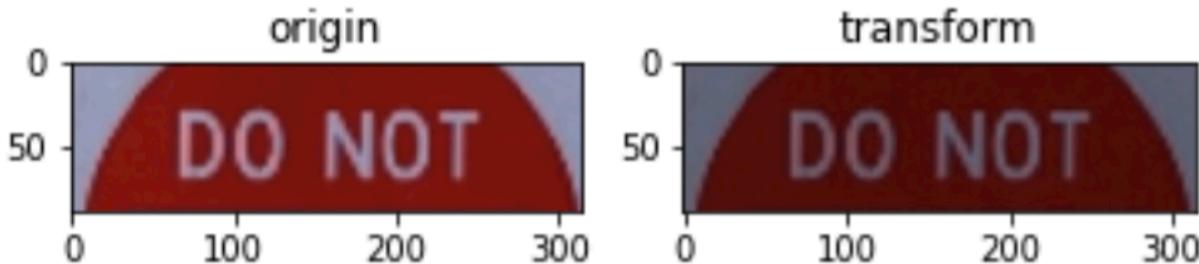


Fig.2 Color space conversion contrast diagram

2. Blur: The blurring effect is achieved by reducing the difference in pixel values of various points in the image. The Gaussian blur method can be used through `cv2.GaussianBlur`. And the parameters respectively represent the image array, the width and height of the kernel, and the standard deviation of the Gaussian kernel in the x-direction. The larger the value of the width and height of the kernel is, the blurrier the image will be.

```

def blur(img):
    """
    blur
    """
    h, w, _ = img.shape
    if h > 10 and w > 10:
        return cv2.GaussianBlur(img, (5, 5), 1)
    else:
        return img
blur_img = blur(img)
show_img(img, blur_img)

```

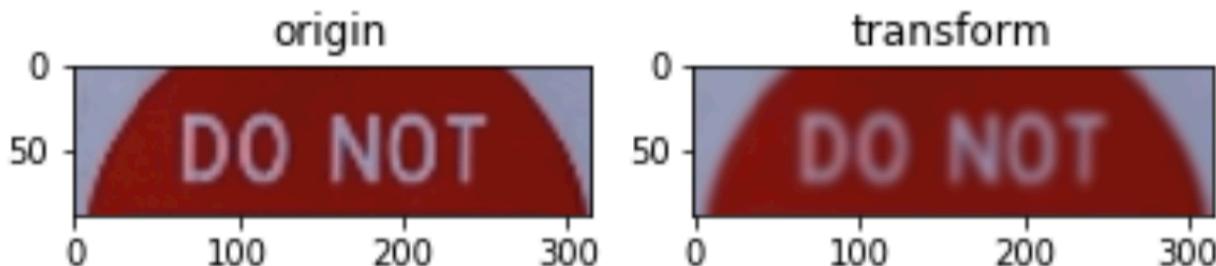


Fig.3 blur process

3. Jitter: The effect of jittering is achieved by randomly changing pixel values of the image.

```

def jitter(img):
    """
    jitter
    """
    w, h, _ = img.shape

```

(continues on next page)

(continued from previous page)

```

if h > 10 and w > 10:
    thres = min(w, h)
    s = int(random.random() * thres * 0.01)
    src_img = img.copy()
    for i in range(s):
        img[i:, i:, :] = src_img[:w - i, :h - i, :]
    return img
else:
    return img
jitter_img = jitter(img)
show_img(img, jitter_img)

```

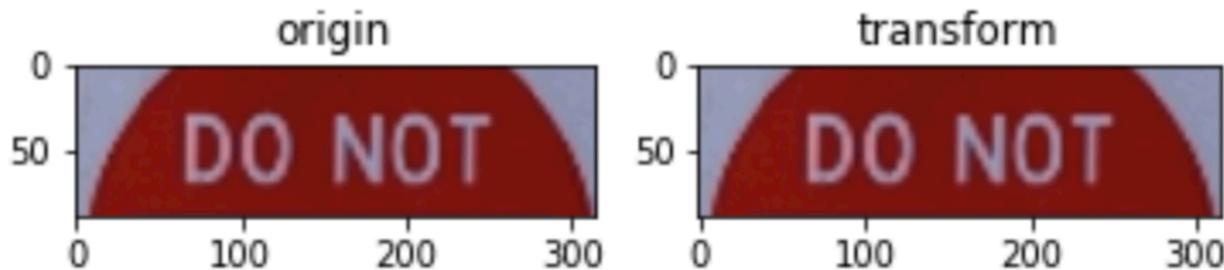


Fig.4 Jitter process

4 Noise: Noise is an unpredictable factor in real images, so adding noise to real data simulation is a simple and effective data augmentation method. The common methods include Gaussian noise, pretzel noise, etc. Here, Gaussian noise is taken as an example to show the process of adding noise to an image. mean represents the mean value and var represents the variance. The larger the variance is, the greater the noise will be.

```

def add_gasuss_noise(image, mean=0, var=0.1):
    """
    Gasuss noise
    """

    noise = np.random.normal(mean, var**0.5, image.shape)
    out = image + 0.5 * noise
    out = np.clip(out, 0, 255)
    out = np.uint8(out)
    return out
noise_img = add_gasuss_noise(img)
show_img(img, noise_img)

```

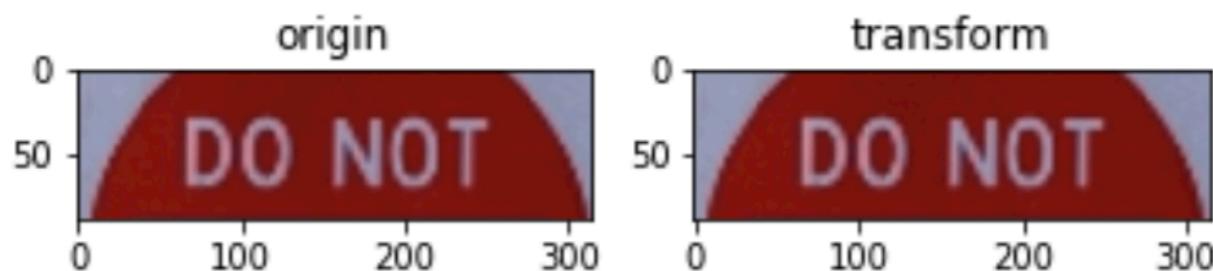


Fig.5 Noise process

(5) Random Crop: This method randomly selects a region from the image and crop it out to get a new sample. Considering that the height of the image is small in the text recognition, we set top_min and top_max to limit the cropping size.

```
def get_crop(image):
    """
    random crop
    """
    h, w, _ = image.shape
    top_min = 1
    top_max = 8
    top_crop = int(random.randint(top_min, top_max))
    top_crop = min(top_crop, h - 1)
    crop_img = image.copy()
    ratio = random.randint(0, 1)
    if ratio:
        crop_img = crop_img[top_crop:h, :, :]
    else:
        crop_img = crop_img[0:h - top_crop, :, :]
    return crop_img
crop_img = get_crop(img)
show_img(crop_img, img)
```

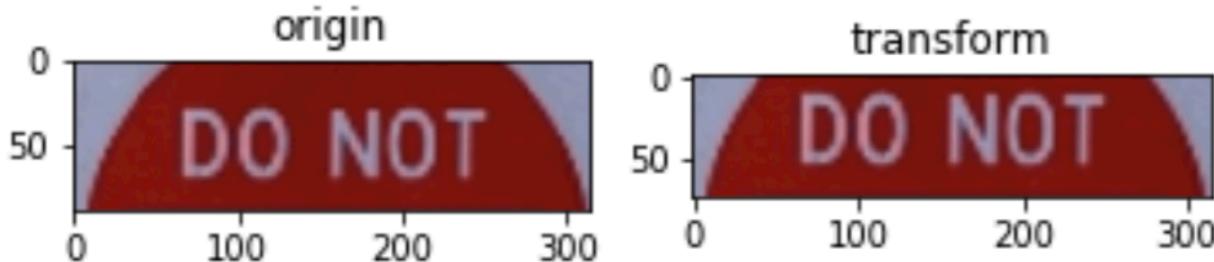


Fig.6 Crop process

(6) Perspective transformation: It refers to the projection of an image to a new plane with a projection matrix.

```
from warp_mls import WarpMLS

def tia_perspective(src):
    img_h, img_w = src.shape[:2]

    thresh = img_h // 2

    src_pts = list()
    dst_pts = list()

    src_pts.append([0, 0])
    src_pts.append([img_w, 0])
    src_pts.append([img_w, img_h])
    src_pts.append([0, img_h])

    dst_pts.append([0, np.random.randint(thresh)])
    dst_pts.append([img_w, np.random.randint(thresh)])
    dst_pts.append([img_w, img_h - np.random.randint(thresh)])
    dst_pts.append([0, img_h - np.random.randint(thresh)])

    trans = WarpMLS(src, src_pts, dst_pts, img_w, img_h)
    dst = trans.generate()
```

(continues on next page)

(continued from previous page)

```

return dst
perspective_img = tia_perspective(img)
show_img(img, perspective_img)

```

(7) Colour Reverse: The method inverts the image colours by subtracting the original pixel value from the maximum value of grayscale level in the image. After the inversion, bright regions become darker and dark regions become brighter.

```

def reverse(img):
    new_img = 255 - img
    return new_img
reverse_img = reverse(img)
show_img(img, reverse_img)

```

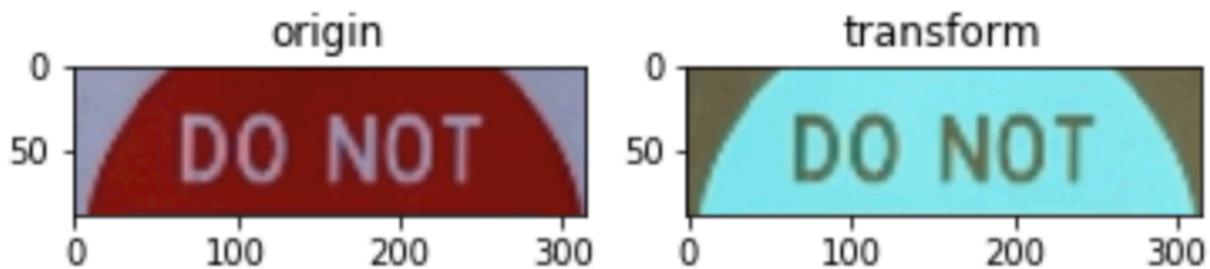


Fig.8 Color Reverse Process

1. TIA [1] is another effective data augmentation method, which first initializes a set of datum points in the image, and then randomly shifts these points through geometric transformation to generate a new image.

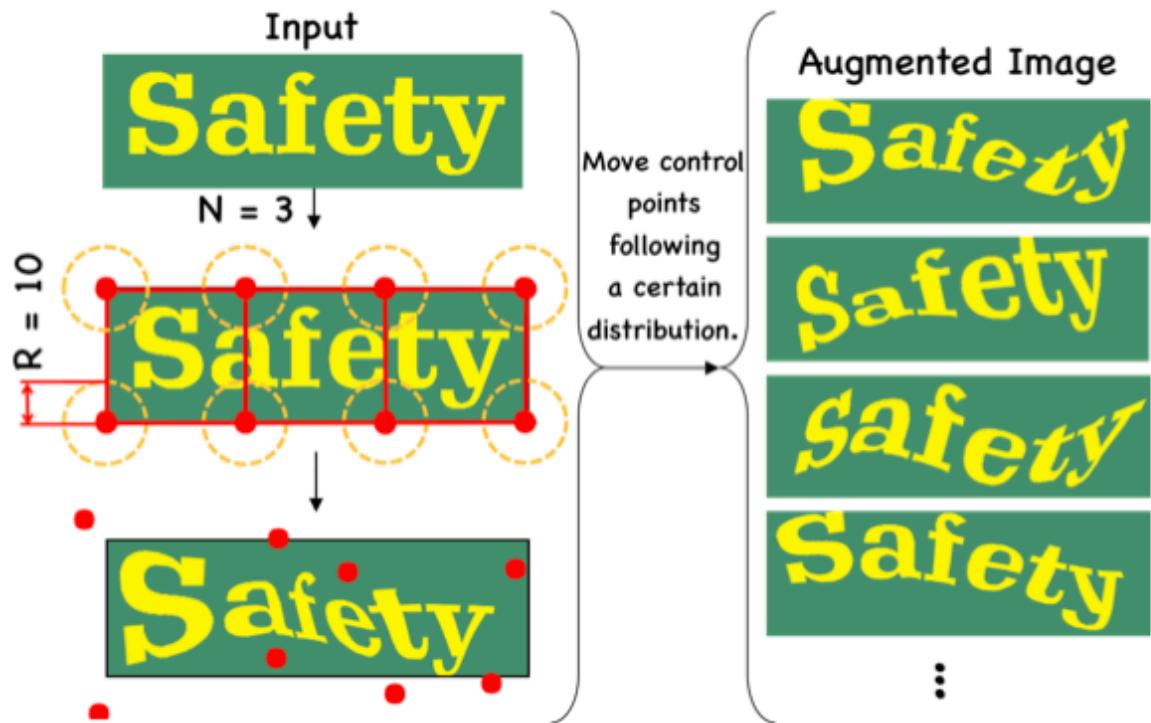


Fig.9 TIA data augmentation

```

def tia_distort(src, segment=4):
    img_h, img_w = src.shape[:2]

    cut = img_w // segment
    thresh = cut // 3

    src_pts = list()
    dst_pts = list()

    src_pts.append([0, 0])
    src_pts.append([img_w, 0])
    src_pts.append([img_w, img_h])
    src_pts.append([0, img_h])

    dst_pts.append([np.random.randint(thresh), np.random.randint(thresh)])
    dst_pts.append(
        [img_w - np.random.randint(thresh), np.random.randint(thresh)])
    dst_pts.append(
        [img_w - np.random.randint(thresh), img_h - np.random.randint(thresh)])
    dst_pts.append(
        [np.random.randint(thresh), img_h - np.random.randint(thresh)])

    half_thresh = thresh * 0.5

    for cut_idx in np.arange(1, segment, 1):
        src_pts.append([cut * cut_idx, 0])
        src_pts.append([cut * cut_idx, img_h])
        dst_pts.append([
            cut * cut_idx + np.random.randint(thresh) - half_thresh,
            np.random.randint(thresh) - half_thresh
        ])
        dst_pts.append([
            cut * cut_idx + np.random.randint(thresh) - half_thresh,
            img_h + np.random.randint(thresh) - half_thresh
        ])

    trans = WarpMLS(src, src_pts, dst_pts, img_w, img_h)
    dst = trans.generate()

return dst

distort_img = tia_distort(img)
show_img(img, distort_img)

```

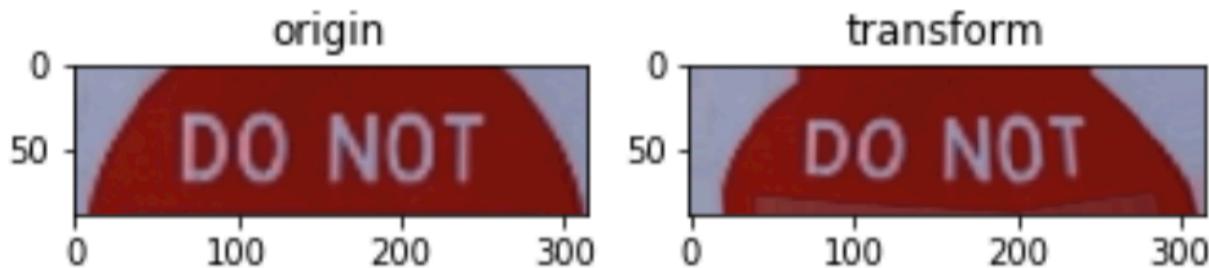


Fig.10 TIA distort

```

def tia_stretch(src, segment=4):
    img_h, img_w = src.shape[:2]

    cut = img_w // segment
    thresh = cut * 4 // 5

    src_pts = list()
    dst_pts = list()

    src_pts.append([0, 0])
    src_pts.append([img_w, 0])
    src_pts.append([img_w, img_h])
    src_pts.append([0, img_h])

    dst_pts.append([0, 0])
    dst_pts.append([img_w, 0])
    dst_pts.append([img_w, img_h])
    dst_pts.append([0, img_h])

    half_thresh = thresh * 0.5

    for cut_idx in np.arange(1, segment, 1):
        move = np.random.randint(thresh) - half_thresh
        src_pts.append([cut * cut_idx, 0])
        src_pts.append([cut * cut_idx, img_h])
        dst_pts.append([cut * cut_idx + move, 0])
        dst_pts.append([cut * cut_idx + move, img_h])

    trans = WarpMLS(src, src_pts, dst_pts, img_w, img_h)
    dst = trans.generate()

    return dst

stretch_img = tia_stretch(img)
show_img(img, stretch_img)

```

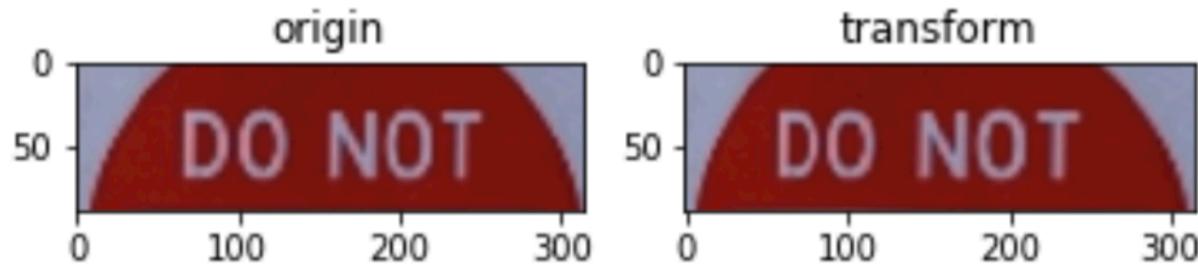


Fig.11 TIA stretch

10.2.2 Image Transformation Techniques

Transformation means performing some transformations on the image after RandCrop. It mainly contains²

- AutoAugment
- RandAugment
- TimmAutoAugment

Unlike conventional artificially designed image augmentation methods, AutoAugment[2] is an image augmentation solution suitable for a specific data set found by certain search algorithm in the search space of a series of image augmentation sub-strategies. For the ImageNet dataset, the final data augmentation solution contains 25 sub-strategy combinations. Each sub-strategy contains two transformations. For each image, a sub-strategy combination is randomly selected and then determined with a certain probability Perform each transformation in the sub-strategy. The ten images in Figure 12 are used as test images to observe the data transformation.



Fig. 12 Original test image

The images after AutoAugment are as follows.



Fig. 13 Diagram of AutoAugment data augmentation

The search method of AutoAugment[3] is relatively violent. Searching for the optimal strategy for this data set directly on the data set requires a lot of computation. In RandAugment, the author found that on the one hand, for larger models

and larger datasets, the gains generated by the augmentation method searched using AutoAugment are smaller. On the other hand, the searched strategy is limited to certain dataset, which has poor generalization performance and not suitable for other datasets.

In RandAugment, the author proposes a random augmentation method. Instead of using a specific probability to determine whether to use a certain sub-strategy, all sub-strategies are selected with the same probability. The experiments in the paper also show that this method performs well even for large models.

The images after RandAugment are as follows.

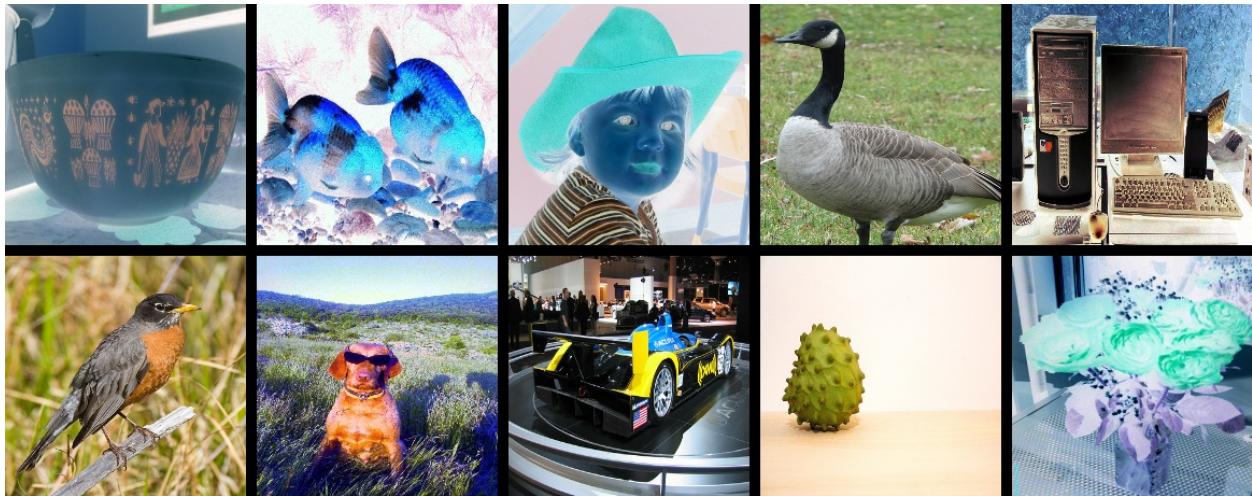


Fig. 14 Schematic of RandAugmentation data augmentation

TimmAutoAugment is an improvement of AutoAugment and RandAugment by open source authors. Facts have proved that it has better performance on many visual tasks. At present, most VisionTransformer models are implemented based on TimmAutoAugment.

10.2.3 Image Cropping Techniques

Cropping means performing some transformations on the image after Transpose, setting pixels of the cropped area as certain constant. It mainly contains:

- CutOut
- RandErasing
- HideAndSeek
- GridMask

Image cropping methods can be operated before or after normalization. The difference is that if we crop the image before normalization and fill the areas with 0, the cropped areas' pixel values will not be 0 after normalization, which will cause grayscale distribution change of the data. The above-mentioned cropping transformation ideas are the similar, all to solve the problem of poor generalization ability of the trained model on occlusion images, the difference lies in that their cropping details.

Cutout[4] is a kind of dropout, but occludes input image rather than feature map. It is more robust to noise than noise. Cutout has two advantages:

- (1) Using Cutout, we can simulate the situation when the subject is partially occluded.
- (2) It can promote the model to make full use of more content in the image for classification, and prevent the network from focusing only on the saliency area, thereby causing overfitting. The images after Cutout are as follows.



Fig. 15 Schematic of CutOut data augmentation

RandomErasing[5] is similar to the Cutout. It is also to solve the problem of poor generalization ability of the trained model on images with occlusion. The author also pointed out in the paper that the way of random cropping is complementary to random horizontal flipping. The author also verified the effectiveness of the method on pedestrian re-identification (REID). Unlike Cutout, in , RandomErasing is operated on the image with a certain probability, size and aspect ratio of the generated mask are also randomly generated according to pre-defined hyperparameters.The images after RandomErasing are as follows.

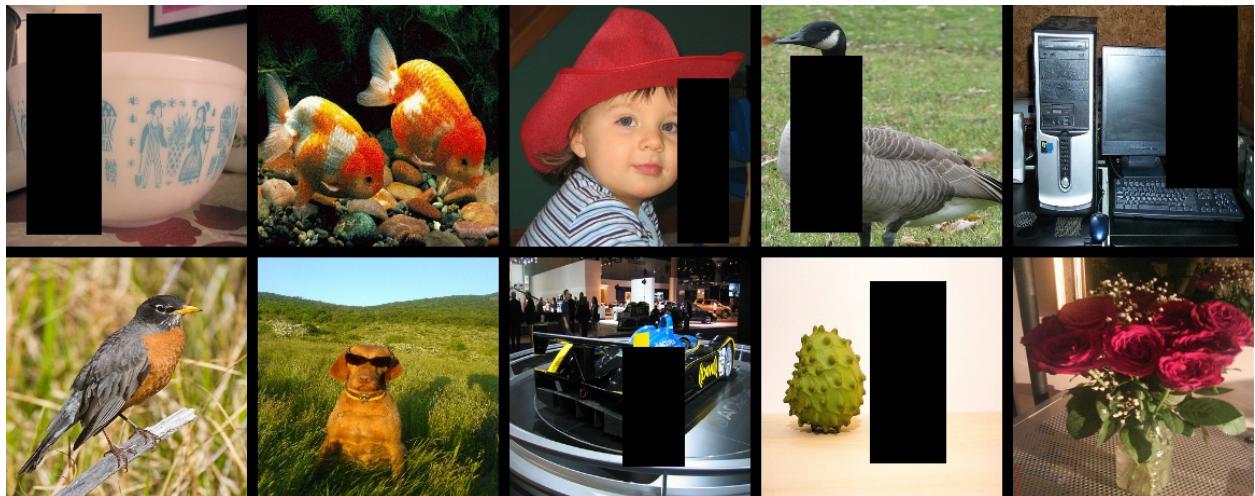


Fig. 16 Schematic of RandomErasing data augmentation

Images are divided into some patches for HideAndSeek[6] and masks are generated with certain probability for each patch. The meaning of the masks in different areas is shown in the figure below.

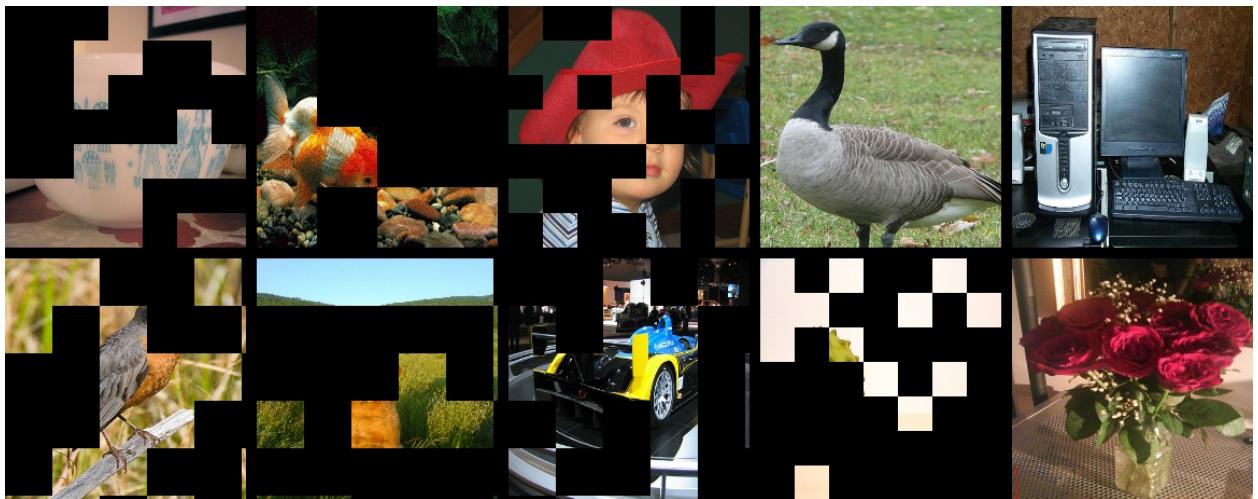


Fig. 17 Diagram of HideAndSeek data augmentation

GridMask[7] is to generate a mask with the same resolution as the original image and multiply it with the original image. The mask grid and size are adjusted by the [hyperparameters](#). In the training process, there are two methods to use:

- Set a probability p and use the GridMask to augment the image with probability p from the beginning of training.
- Initially set the augmentation probability to 0, and the probability is increased with number of iterations from 0 to p .

It shows that the second method is better. The images after GridMask are as follows.



Fig. 18 Schematic of GridMask data augmentation

10.2.4 Image Mixing Techniques

Mix means performing some transformations on the image after Batch, which contains:

- Mixup
- Cutmix

Data augmentation methods introduced before are based on single image while mixing is carried on a certain batch to generate a new batch.

Mixup[8] is the first solution for image aliasing, it is easy to realize and performs well not only on image classification but also on object detection. Mixup is usually carried out in a batch for simplification, so as Cutmix. The images after Mixup are as follows.

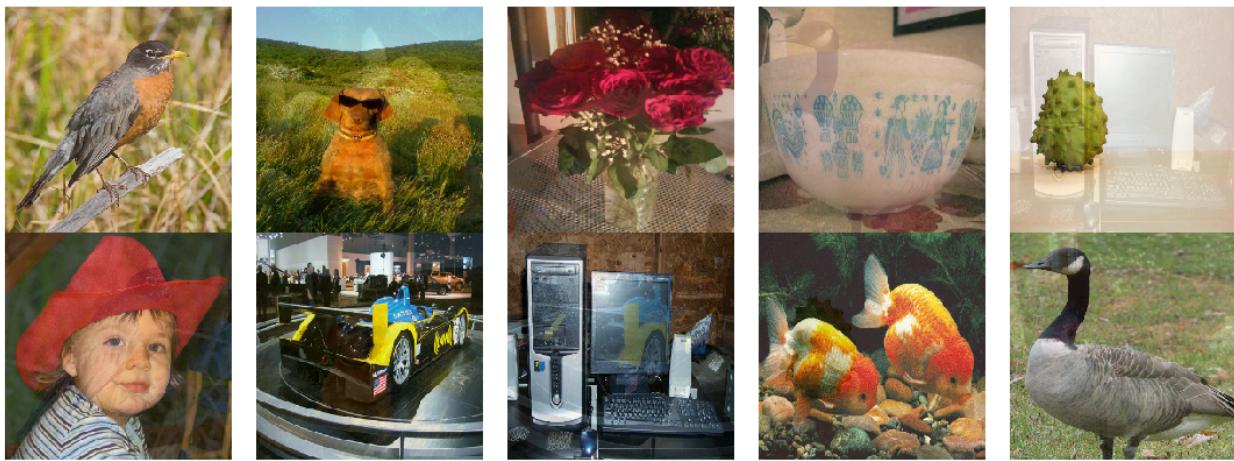


Fig. 19 Schematic of Mixup data augmentation

Unlike Mixup which directly adds two images, for Cutmix[9], an ROI is cut out from one image and Cutmix randomly cuts out an ROI from one image, and then covered onto the corresponding area in the another image. The images after Cutmix are as follows.

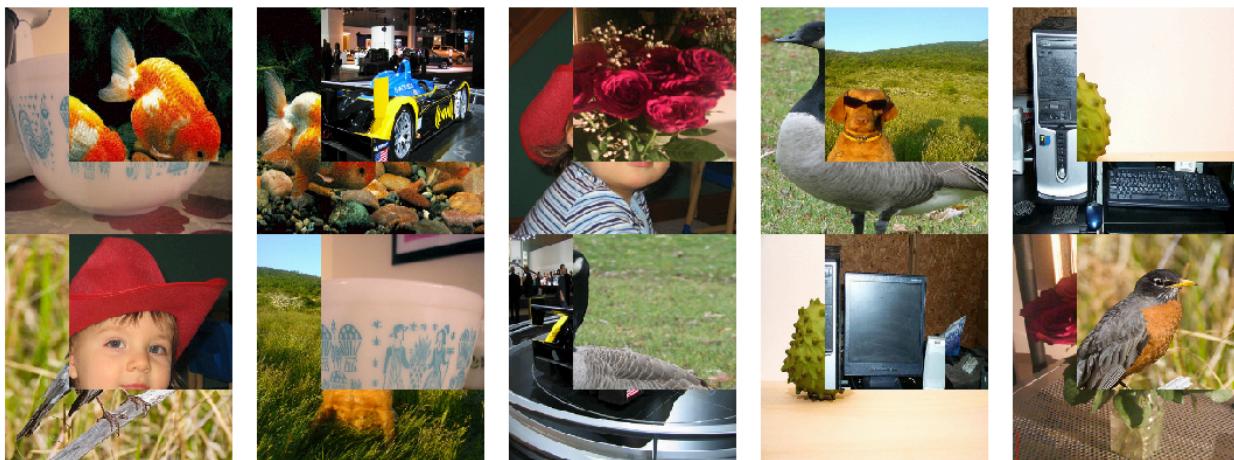


Fig. 20 Schematic of Cutmix data augmentation

10.3 Image Binarization

Image binarization refers to setting greyscale values of pixels to 0 or 255, so that the whole image presents an obvious black and white effect. Each pixel in a binary image has only two values: 0 and 255, 0 represents black and 255 represents white. Image binarization can reduce the interference caused by noise, eliminate background interference and thus highlight the contours of the object. The OCR text recognition performance can be improved if the binarization can distinguish between the foreground and the background. We summarize the commonly used image binarization methods, which are mainly classified into global binarization, local binarization, and deep learning methods.

10.3.1 Global Thresholding

The global thresholding refers to the global processing of all pixel in an image with the same threshold value. The methods include fixed thresholding, Otsu, etc.

The fixed threshold method uses a fixed value for all pixels as the global threshold T . If the pixel value of the current pixel point is greater than or equal to the threshold T , then the point is assigned a value of 255, otherwise it is assigned a value of 0. It is usually necessary to set different thresholds for experimental observation of the binarization effect, and it is difficult to determine the optimal threshold for different images. The different binarization effects of different thresholds are shown in the **Figure 21**. To overcome this problem, a Japanese scholar Nobuyuki Otsu proposed an adaptive thresholding approach, Otsu[10], in 1979. Otsu divides the image into foreground and background. The larger the variance between pixels is, the lower the correlation is, and the foreground and the background are more distinct from each other. Therefore, the variance of pixels between the gray value of each pixel of the image is calculated to find the gray value with the maximum variance, which is the binarization threshold.



Fig.21 Effects of binarization with different thresholds

10.3.2 Local Thresholding

If the image has some problems such as uneven illumination, the binarization methods based on global threshold will not achieve the expected binarization effect, so in this case the binarization method based on local threshold is more suitable. Common local thresholding methods for image binarization include adaptive thresholding algorithm [11], NiBlack [12], Sauvola [13] and so on.

In the adaptive thresholding algorithm, there is a sliding window with the size of $s * s$ centred on a pixel, and it sweeps across the whole image. In each slide, the pixels in the current window will be averaged and the average value will be used as the local threshold T . If the value of a pixel in the current window is less than the local threshold $T/100$, the pixel is assigned a value of 0; if it is greater than the local threshold $T/100$, it is assigned a value of 255, as shown in **Figure 22**. NiBlack calculates the mean m and the variance s of the pixels in the local region of the image, then calculates the local threshold through the formula $T = m + k * s$, k represents the correction factor with a value between 0 and 1. Finally, binarization is performed according to the threshold T . Sauvola is improved from the NiBlack algorithm. It calculates the local threshold through $T = m - [1 + k - (\frac{s}{R} - 1)]$, R represents the dynamic range of the variance, and if the

current input image is an 8-bit grey-scale image, $R = 128$. Sauvola performs better than NiBlack in situations such as the uneven illumination.



Fig.22 The effect of adaptive binarization

10.3.3 Techniques Based on Deep Learning

It is difficult for traditional global and local binarization methods to set appropriate thresholds, which results in poor image binarization effect. With the continuous development of deep learning, some scholars have begun to try to use neural networks to binarize images.

Pratikakis I et al [14] has used U-Net convolutional neural network architecture for document image binarization in ICDAR2017 DIBCO competition and won the championship. Chris Tensmeyer et al [15] use a multi-scale full convolutional neural network to binarize document images and the results are shown in **Figure 23**. Vo QN et al [16] propose a hierarchical deep supervised network DSN for document binarization and the results are shown in **Figure 24**. Westphal F et al [17] use a Grid Long Short Term Memory (Grid LSTM) network for binarization. However, its performance was lower than that of the method [16]. Calvo-Zaragoza J et al [18] use deep encoder and decoder architecture to achieve binarization.

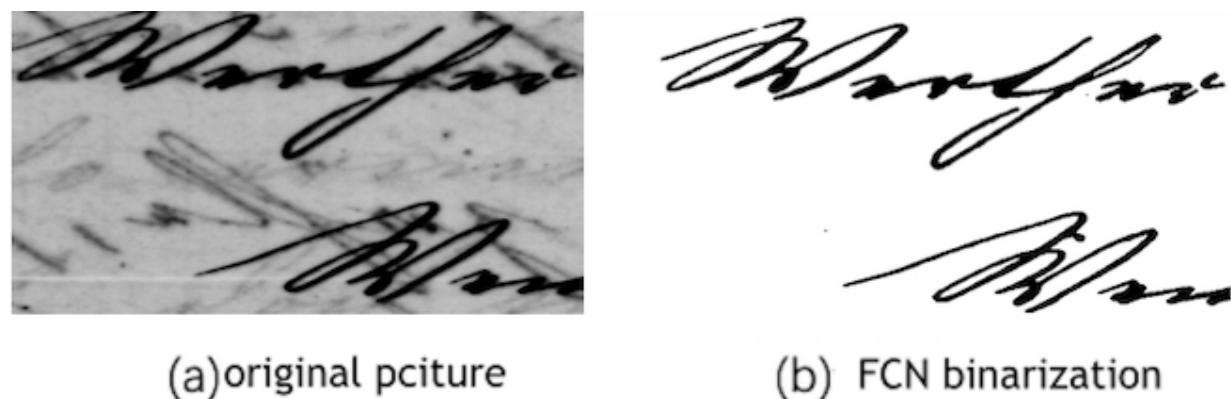


Fig.23 The binarization effect of the full convolutional neural network

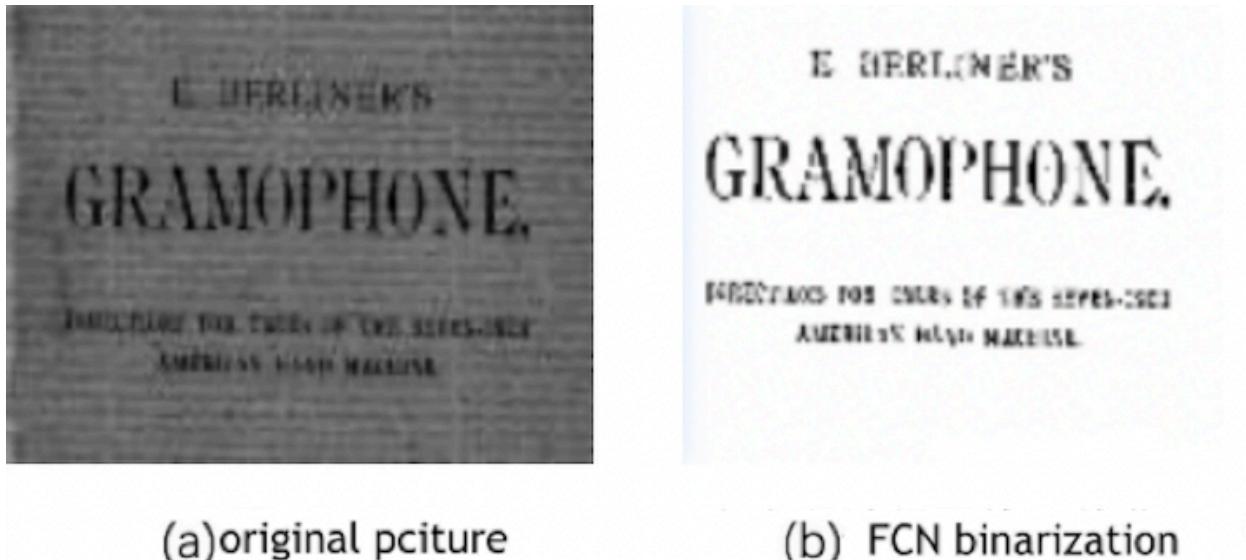


Fig.24 The binarization effect of hierarchical DSN

10.4 Denoising

Image noise refers to unnecessary or redundant interference information existing in image data. It may seriously affect the data quality, so it is often necessary to process the noise. In addition, we need to maintain the details of the images removing noise. The denoising methods will be categorized into four groups: spatial domain filtering, transform domain filtering, non-local filtering, and methods based on neural network.

10.4.1 Spatial Domain Filtering

Spatial domain filtering refers to the processing of pixel values by performing data operations directly on the original image. Common spatial domain filtering denoising algorithms include mean filter, Gaussian filter, median filter, bilateral filter [19], Non-Local Means (NLM) algorithm [20], etc.

The mean filter uses the average pixel value in the field of pixel point A to replace the original pixel value of pixel point A. It is a typical smoothing linear filter. The mean filter is simple to calculate and plays a smoothing role in the whole image. However, it cannot preserve image details, which makes the image blurred. Gaussian filter is also a linear filter and a commonly used filtering algorithm. After Gaussian filter, the value of each image pixel is replaced by a weighted average value of itself and other pixels values in the field. Compared to the mean filter, the Gaussian filter performs better on smoothing and can better preserve the edge information, and suppress Gaussian noise.

The median filter replaces image pixel value with the median of the neighbourhood grey value of target pixel. It is a non-linear filter, suitable for dealing with salt-and-pepper noise and preserving the image edge details. The bilateral filter takes into account not only the spatial distance of pixels, but also the similarity between pixels, and the colour intensity.

Four filters are implemented with the OpenCV library.

```
noise_img = cv2.imread('preprocess/noise.png')
# Mean filter
img_mean = cv2.blur(noise_img, (5,5))
show_img(noise_img, img_mean)
# Gaussian filter
```

(continues on next page)

(continued from previous page)

```
img_gussian = cv2.GaussianBlur(noise_img, (5,5), 1)
show_img(noise_img, img_gussian)
# Median filter
img_median = cv2.medianBlur(noise_img, 5)
show_img(noise_img, img_median)
# Bilateral filter
img_bilater = cv2.bilateralFilter(noise_img, 3, 15, 15)
show_img(noise_img, img_bilater)
```

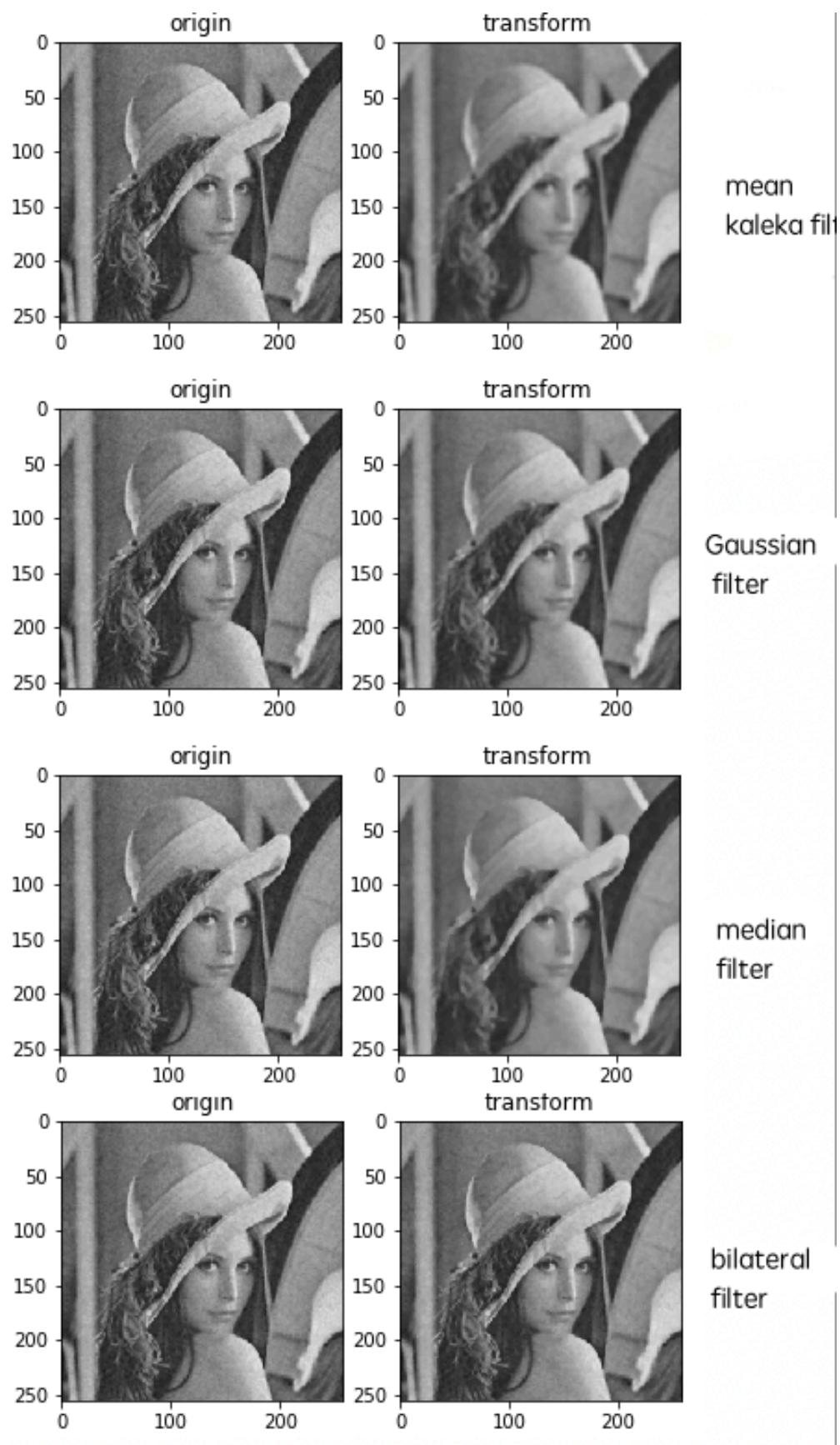


Fig. 25 Spatial Domain Filtering

The filters based on the neighbourhood pixels basically only considers the gray value information of pixels within the sliding window, without considering the statistical information of pixels within the window, such as variance, the pixel distribution characteristics of the whole image, the pixel distribution characteristics of the whole image, and the prior knowledge of noise. In view of these limitations, NLM algorithm has been proposed, which uses redundant information in the image to remove the noise and retain the maximum detailed features in it. NLM method uses the whole image for denoising by finding similar regions in the image in blocks, and calculating the weighted average of these regions to get the denoised image. The similarity is calculated with the weighted Euclidean distance, and it is in positive correlation with the weight.

10.5 Transform Domain Filtering

Transform domain filtering methods convert the image from the original spatial domain to transform domain; In the transform domain, the noise can be divided into high, medium and low frequency noises, and the different frequency noise can be separated by the transform domain method; and then the image is converted from transform domain to original space domain by inverse transformation, so as to remove the image noise. There are many ways for conversion, including Fourier transform, wavelet transform algorithms and so on.

Fourier transform converts the input image from the spatial domain to the frequency domain, containing both low and high frequency information. The points on the image where the grey values change rapidly are often the high frequency noise in the image. Then a low-pass filter with Fourier transform removes the high-frequency component of the image and only allows the low-frequency information to pass through the filter, so as to achieve the purpose of removing the image noise.

Wavelet transform denoising can be carried out in three steps:

- 1 Wavelet decomposition of the image: selects a wavelet and the number of layers N of wavelet decomposition and applied N layers of wavelet decomposition to the signal s .
- 2 Non-linear threshold quantization of wavelet transform coefficients: selects a threshold for each layer from 1 to N and the high-frequency coefficient of that layer is quantized and the low-frequency coefficient of each layer is not processed.
- 3 Wavelet coefficient reconstruction: Based on the low frequency coefficient of the N th layer and the high frequency coefficients of the $1 \sim N$ layers after processing, the wavelet reconstruction of the original signal is processed. The key to wavelet transform denoising lies in the threshold value. The threshold function can be divided into the hard threshold function and the soft threshold function.

10.5.1 BM3D

BM3D (Block-matching and 3D filtering) [21] combines spatial domain filtering and transform domain filtering techniques. Firstly it uses the method of computing similar blocks in the NLM, and then integrates wavelet transform denoising method. The algorithm finds similar blocks by similarity determination and combines them into 3D groups. The 3D groups are then transformed into the wavelet domain, where hard thresholding or Wiener filtering is used to reduce the noise. Finally, an inverse transformation process is performed to aggregate all the image blocks to obtain the noise-reduced image.

10.5.2 Methods based on Deep Learning

With the development of deep learning, denoising algorithms based on deep learning also keep emerging, and CNN-based denoising methods have improved the denoising effect. Commonly methods based on deep learning include DnCNNs[22], FFDNet[23], MPRNet[24] and so on.

The DnCNNs (Denoising CNNs) introduces residual learning and batch normalisation into image denoising for the first time, and the combination of the two can enhance each other, and effectively improve training speed and denoising performance. FFDNet(Fast and Flexible Denoising NetWork) works on downsampled sub-images, achieving a good balance between the training and inference speed ,and the enlarging the receptive field. Also, it adopts orthogonal regularization to enhance generalization. What's more, MPRNet first learns the contextualized features using encoder-decoder architectures with a large receptive field. And then combines them with a high-resolution branch that retains local information needed in the image. MPRNet can be applied in multiple scenarios such as rain removal, deblurring, denoising and so on.

10.6 Summary

This section focuses on common methods of data augmentation, binarization and denoising in image pre-processing. First, in order to improve the robustness of the model, data augmentation is usually used on the training samples. Four kinds of data augmentation techniques are introduced here:

1. Standard data augmentation techniques: rotation, perspective transformation, blurring, Gaussian noise, random cropping, etc;
2. Image transformation techniques: transform images after RandCrop. It mainly contains AutoAugment and RandAugment;
3. Image cropping techniques: crop images after transposition and set the pixel values of the cropped region to a particular constant (the default value is 0). It mainly contains CutOut, RandErasing, HideAndSeek and GridMask;;
4. Image mixing techniques: mix the data after batch processing. It mainly contains Mixup and Cutmix.

Secondly, high quality binary images can effectively improve text recognition performance. Global thresholding (Fixed Thresholding, Otus), local thresholding (Adaptive Thresholding, NiBlack, Sauvola, Bernsen) and methods based on deep learning (U-Net, Grid LSTM, Full Convolutional Neural Network, etc.) are mainly introduced.

Thirdly, in the era of big data image quality is uneven, image filtering is increasingly used and more and more demanding. So this section also briefly introduce some image denoising methods, including spatial domain filtering, transform domain filtering, BM3D and filtering based on deep learning.

10.7 References

- [1] Luo C, Zhu Y, Jin L, et al. Learn to augment: Joint data augmentation and network optimization for text recognition[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 13746-13755.
- [2] Cubuk E D, Zoph B, Mane D, et al. Autoaugment: Learning augmentation strategies from data[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2019: 113-123.
- [3] Cubuk E D, Zoph B, Shlens J, et al. Randaugment: Practical automated data augmentation with a reduced search space[J]. arXiv preprint arXiv:1909.13719, 2019.
- [4] DeVries T, Taylor G W. Improved regularization of convolutional neural networks with cutout[J]. arXiv preprint arXiv:1708.04552, 2017.
- [5] Zhong Z, Zheng L, Kang G, et al. Random erasing data augmentation[J]. arXiv preprint arXiv:1708.04896, 2017.

- [6] Singh K K, Lee Y J. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization[C]//2017 IEEE international conference on computer vision (ICCV). IEEE, 2017: 3544-3553.
- [7] Chen P. GridMask Data Augmentation[J]. arXiv preprint arXiv:2001.04086, 2020.
- [8] Zhang H, Cisse M, Dauphin Y N, et al. mixup: Beyond empirical risk minimization[J]. arXiv preprint arXiv:1710.09412, 2017.
- [9] Yun S, Han D, Oh S J, et al. Cutmix: Regularization strategy to train strong classifiers with localizable features[C]//Proceedings of the IEEE International Conference on Computer Vision. 2019: 6023-6032.
- [10] Otsu N. A threshold selection method from gray-level histograms[J]. IEEE transactions on systems, man, and cybernetics, 1979, 9(1): 62-66.
- [11] Bradley D, Roth G. Adaptive thresholding using the integral image[J]. Journal of graphics tools, 2007, 12(2): 13-21.
- [12] Niblack W. An introduction to digital image processing[M]. Strandberg Publishing Company, 1985
- [13] Lazzara G, Géraud T. Efficient multiscale Sauvola's binarization[J]. International Journal on Document Analysis and Recognition (IJDAR), 2014, 17(2): 105-123.
- [14] Pratikakis I, Zagoris K, Barlas G, et al. ICDAR2017 competition on document image binarization (DIBCO 2017)[C]//2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2017, 1: 1395-1403.
- [15] Tensmeyer C, Martinez T. Document image binarization with fully convolutional neural networks[C]//2017 14th IAPR international conference on document analysis and recognition (ICDAR). IEEE, 2017, 1: 99-104.
- [16] Vo Q N, Kim S H, Yang H J, et al. Binarization of degraded document images based on hierarchical deep supervised network[J]. Pattern Recognition, 2018, 74: 568-586.
- [17] Westphal F, Lavesson N, Grahn H. Document image binarization using recurrent neural networks[C]//2018 13th IAPR International Workshop on Document Analysis Systems (DAS). IEEE, 2018: 263-268.
- [18] Calvo-Zaragoza J, Gallego A J. A selectional auto-encoder approach for document image binarization[J]. Pattern Recognition, 2019, 86: 37-47.
- [19] Tomasi C, Manduchi R. Bilateral filtering for gray and color images[C]//Sixth international conference on computer vision (IEEE Cat. No. 98CH36271). IEEE, 1998: 839-846.
- [20] Buades A, Coll B, Morel J M. A non-local algorithm for image denoising[C]//2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). IEEE, 2005, 2: 60-65.
- [21] Dabov K, Foi A, Katkovnik V, et al. Image denoising by sparse 3-D transform-domain collaborative filtering[J]. IEEE Transactions on image processing, 2007, 16(8): 2080-2095.
- [22] Zhang K, Zuo W, Chen Y, et al. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising[J]. IEEE transactions on image processing, 2017, 26(7): 3142-3155.
- [23] Zhang K, Zuo W, Zhang L. FFDNet: Toward a fast and flexible solution for CNN-based image denoising[J]. IEEE Transactions on Image Processing, 2018, 27(9): 4608-4622.
- [24] Zamir S W, Arora A, Khan S, et al. Multi-stage progressive image restoration[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 14821-14831.

DATA SYNTHESIS ALGORITHMS

11.1 Background

The performance of deep learning methods is closely related to the quality and quantity of training data, but the acquisition of massive annotated data has become a bottleneck for efficient development of many deep learning tasks. Images of different scenes in OCR tasks have unique styles in fonts, blurs, lighting, and so on. Collecting enough data is challenging and costly, and manual annotation is also time-consuming and error-prone. Therefore, in the absence of access to sufficient annotated data, automatic synthesis of annotated images is gaining more and more attention, for it successfully alleviating the shortage of data and the problem of manual data annotation. In the case of OCR, for example, synthesised data is important in training OCR text detection and recognition models and has proved effective in numerous algorithms. In this section, some representative data synthesis methods in recent years will be presented, such as SynthText, Verisimilar, SynthText3D,SF-GAN, SRNet, ScrabbleGAN, UnrealText.

11.2 Data Synthesis Algorithms

Ankush Gupta et al. (2016)[1] propose SynthText, a new method to synthesize text images by overlaying synthetic text onto background images to generate text images in natural scenes. The synthetic process consists of five steps: 1) collect a large number of text-free background images, fonts and text corpora; 2) the image is segmented into contiguous regions based on local colour and texture cues, and use CNN to obtain a dense pixel-wise depth map; 3) get the candidate region according to semantic and depth information;(4) choose the colour of text and its outline (optional) according to the colour of the candidate region; 5) render the text with a randomly selected font, and the rendered image is shown in Figure 1.



Figure 1 The synthetic image using SynthText

Fangneng Zhan et al. (2018) [2] propose another novel image synthesis technique, Verisimilar, that aims to generate massive annotated scene text images for training accurate and robust scene text detection and recognition models. The authors combine semantic segmentation and visual saliency in text synthesis, introducing semantic segmentation to make the text appear only on a perceptible object. For example, scene texts tend to appear over the wall or table surface instead of the food or leaves. It uses visual saliency to determine the embedding locations of the text, and distinguish the text and the background. Finally, It designs a novel scene text appearance model that determines the color and brightness of source texts by learning from the feature of real scene text images adaptively. The method goes well in training accurate and robust scene text detection and recognition models, and realizes the synthesis of verisimilar scene text images. And the results are shown in Figure 2.

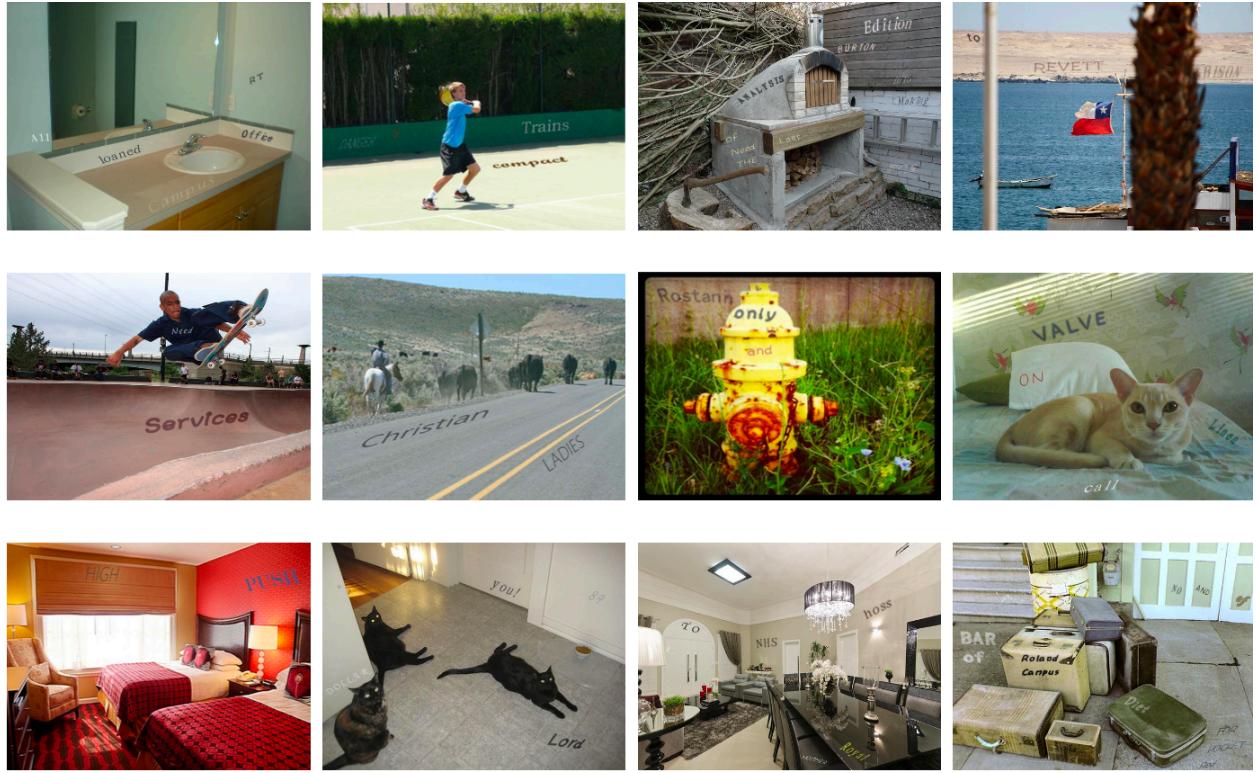


Figure 2 The results of Verisimilar

Minghui Liao et al. (2019) [3] propose SynthText3D, a model of synthesizing scene text images in 3D virtual worlds. In this way, real-world variations, including complex perspective transformations, various illuminations, and occlusions, can be realized in our synthesized scene text images. Specifically, text instances in various fonts are firstly embedded into suitable positions in a 3D virtual worlds. The synthetic images produced from 3D virtual worlds yield fantastic visual effects, including various illuminations, and occlusions, and text and the background scene are rendered together. Finally, the authors set up camera with different locations and orientations to produce images of the same text of different viewpoints. The synthesis results are shown in Figure 3.

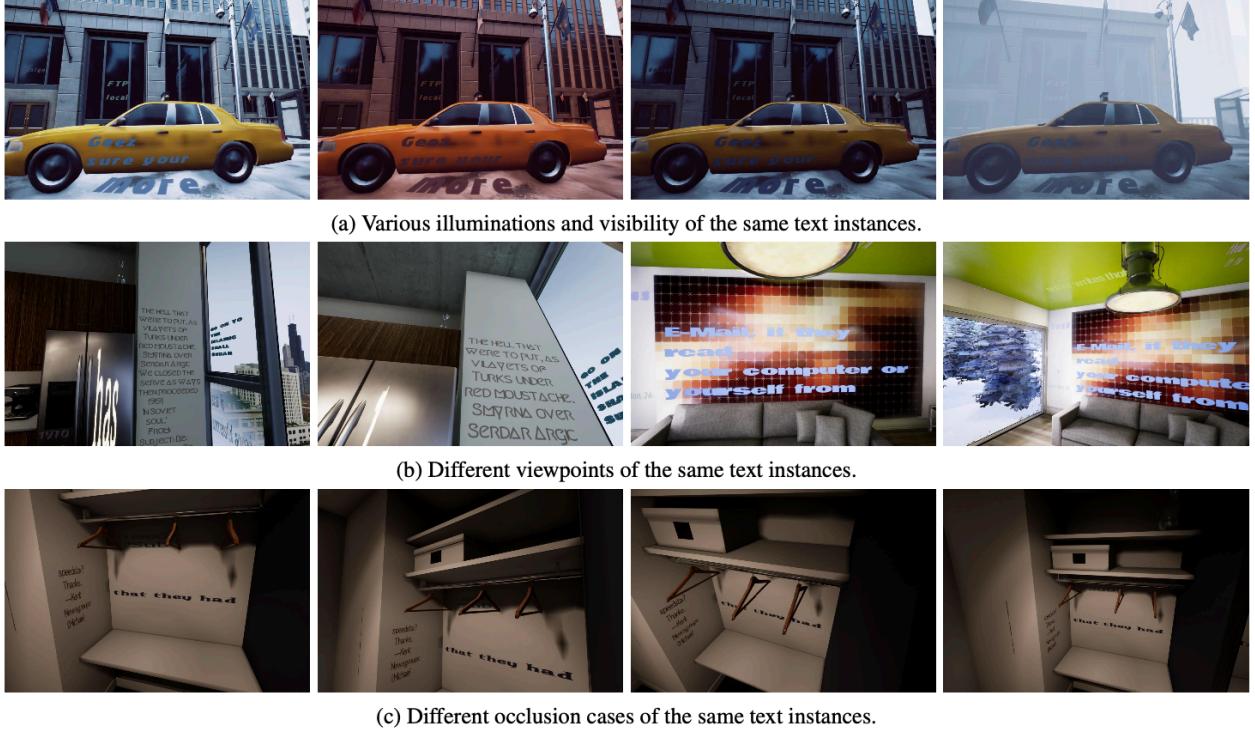


Figure 3 The 3D synthetic images with SynthText

Fangneng Zhan et al. (2019) [4] propose the Spatial Fusion GAN (SF-GAN) that combines a geometry synthesizer and an appearance synthesizer to generate images that are approximately realistic in both geometry and appearance spaces. The geometry synthesizer learns contextual geometries of background images and transforms and places foreground objects into the background images unanimously. The appearance synthesizer adjusts the color, brightness and style of the foreground objects. The authors also design a fusion network which introduces detail-preserving guide filters to preserve realistic appearance. The synthetic results are shown in Figure 4.



Figure 4 Synthetic results of SF-GAN

Liang Wu et al. (2019) [5] propose an end-to-end trainable network: style retention network (SRNet) that changes the content of the source image into target text while keeping the original image style. The framework of Style-Text consists of three modules: 1) the style migration module of the text foreground, 2) the background extraction module, and 3) the fusion module. The first module replaces the text content of the source image with the target text, and preserve the original text region at the same time. The second module erases the original text and fills the text region with appropriate texture. The fusion module combines the information from the two former modules, and generates the edited text images. After these three steps, the image text style can be quickly migrated. The results are shown in Figure 5.



Figure 5 The synthetic results of SRNet

Sharon Fogel et al. (2020) [6] propose a semi-supervised approach, ScrabbleGAN, to synthesize handwritten text images. A semi-supervised approach can also use unlabelled data to train a handwritten text synthesis framework besides labeled data. ScrabbleGAN relies on a new fully convolutional generator model that can generate images with arbitrarily long words or even complete sentences. In addition, ScrabbleGAN's generator can control the style of the generated text. For example, it allows us to change whether the text is cursive or how thin is the penstroke. The results are shown in Figure 6.

Supercalifragilisticexpialidocious
 Supercalifragilisticexpialidocious
 Supercalifragilisticexpialidocious
~~Supercalifragilisticexpialidocious~~
 Super cali fragi list i cexpia lidocious
 Supercali fragilis t i cexpia lido cious
 Supercali fragilis t i cexpia lido cious
Supercali fragilis t i cexpia lido cious
 Super cali fragi list i cexpia lidocious
 Supercali fragilis t i cexpia lido cious
 Supercali fragilis t i cexpia lido cious
Supercali fragilis t i cexpia lido cious
 Super cali fragi list i cexpia lido cious
 Supercali fragilis t i cexpia lido cious

Figure 6 The synthetic results of ScrabbleGAN

Shangbang Long et al. (2020) [7] propose another effective image synthesis method, UnrealText, to synthetic scene text images from 3D virtual world. UnrealText is based on the famous Unreal Engine 4 (UE4), and is therefore named as UnrealText. Specifically, text instances are regarded as planar polygon meshes with text foregrounds loaded as texture, and they are placed in suitable positions in 3D world. And the text and the scene are rendered together as whole, achieving realistic visual effects, e.g. illumination, occlusion, and perspective transformation. The results are shown in Figure 7. The UnrealText engine achieves realistic rendering and high scalability, significantly improves the performance of text detectors and model generation. Also, a large-scale multilingual scene text dataset is also constructed, which is helpful for further research.

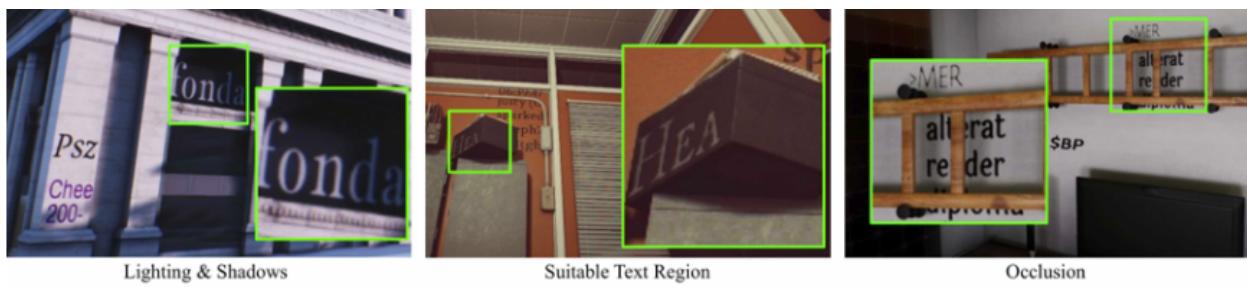


Figure 7 The synthetic results of UnrealText

11.3 summary

Public datasets in the text detection and recognition may not meet the demand of current application scenarios or small number of datasets. What's more, manual annotation of data is time-consuming and labour-intensive, so OCR data synthesis has become a common practice. In this section, we have summarized some data synthesis methods, including SynthText, SynthText3D, SF-GAN, SRNet and so on. OCR data is generated through these methods which improves the performance of text detection and recognition.

11.4 References

- [1] Gupta A, Vedaldi A, Zisserman A. Synthetic data for text localisation in natural images[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2315-2324.
- [2] Zhan F, Lu S, Xue C. Verisimilar image synthesis for accurate detection and recognition of texts in scenes[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 249-266.
- [3] Liao M, Song B, Long S, et al. SynthText3D: synthesizing scene text images from 3D virtual worlds[J]. Science China Information Sciences, 2020, 63(2): 1-14.
- [4] Zhan F, Zhu H, Lu S. Spatial fusion gan for image synthesis[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 3653-3662.
- [5] Wu L, Zhang C, Liu J, et al. Editing text in the wild[C]//Proceedings of the 27th ACM international conference on multimedia. 2019: 1500-1508
- [6] Fogel S, Averbuch-Elor H, Cohen S, et al. Scrabblegan: Semi-supervised varying length handwritten text generation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 4324-4333.
- [7] Long S, Yao C. Unrealtext: Synthesizing realistic scene text images from the unreal world[J]. arXiv preprint arXiv:2003.10608, 2020.