A Machine Learning Framework for Data Ingestion in Document Images

Han Fu†, Yunyu Bai†, Zhuo Li‡, Jun Shen‡, Jianling Sun†
†Zhejiang University, Hangzhou, China
‡State Street Corporation, Hangzhou, China

ABSTRACT

Paper documents are widely used as an irreplaceable channel of information in many fields, especially in financial industry, fostering a great amount of demand for systems which can convert document images into structured data representations. In this paper, we present a machine learning framework for data ingestion in document images, which processes the images uploaded by users and return fine-grained data in JSON format. Details of model architectures, design strategies, distinctions with existing solutions and lessons learned during development are elaborated. We conduct abundant experiments on both synthetic and real-world data in State Street. The experimental results indicate the effectiveness and efficiency of our methods.

CCS CONCEPTS

• **Applied computing → Graphics recognition and interpretation**; *Optical character recognition*; Online handwriting recognition.

KEYWORDS

Document image, region detection, handwriting recognition, neural networks

ACM Reference Format:
Han Fu†, Yunyu Bai†, Zhuo Li‡, Jun Shen‡, Jianling Sun† . 2019. A Machine Learning Framework for Data Ingestion in Document Images. In *Proceedindgs of CIKM'19, November 03–07, 2019, Beijing, China* . ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

CIKM'19, November 03–07, 2019, Beijing, China

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Information ingestion from structured documents has drawn considerable attention from both research and application area of data mining and information retrieval. The substantial variability of layouts and data contents makes it difficult to precisely convert the content into structured representations. Further, it is more challenging when the document contents are not encoded in editable data formats (e.g. PDF) but are captured as images. Document images are common in business world, such as fax, bills and receipts, etc. Paper documents are widely used as a essential information medium, especially in government departments and financial industry for security reasons. Naturally, there comes a great demand to automatically convert contents of document images into structured representation formats. With the rapid development of machine learning techniques, many methods have been proposed for document layout analysis [1, 23, 33]. However, to our knowledge, there is still not a technique to meet the demand of industrial application due to following challenges:

(1) A document image consists of multiple regions with different semantic labels, such as tables, figures, signatures, text blocks, etc. Optical Character Recognition (OCR) methods are not capable to capture the differences between these regions.
(2) The precision of current document analyzers decreases sharply when solving tabular data without form lines.
(3) The accuracy of handwriting verification and recognition can hardly meet the standard for industrial use.

Consequently, the task of information extraction from document images is often manually done.

Motivated by such challenges, we propose a generic framework for document image understanding systems to reduce human intervention. Specifically, the framework involves following sub-modules as shown in Figure 1:

- **The region detection module** segments a document image page into several document regions and predict the corresponding semantic labels. In this paper, we only focus on handwritings, tables, tabular cells and text blocks.
- **The text Rrcognition module** takes the location information of each region as input and outputs text

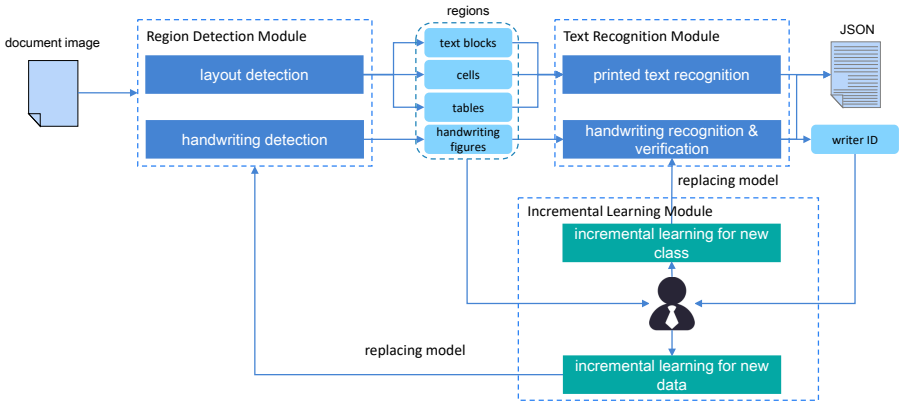

Figure 1: Workflow of data ingestion from document images. The approach takes a document image as input and outputs structured JSON containing fine-grained textual data.

sequences. To reduce coupling, we train independent models to recognize handwritings and printed texts.

- **The incremental learning module**: this module is responsible for fine-tuning the trained models to adapt to new classes or new data.

To improve the efficiency and effectiveness of our system, we propose multiple novel models and training algorithms for each task. Details of design ideas, system architectures and model settings are provided in this paper. The entire system is currently deployed in the development environment of State Street Corporation but not yet in production.

The rest of this paper is structured as follows: Section 2 introduces state-of-the-art document understanding solutions. In section 3, we present the overall design of the data ingestion process and architecture of each components. Section 4 presents the overall system design and pipeline of document understanding process. In section 5, we present the experimental results and discuss the lessons learned during model training and system developing. Last but not the least, we conclude the paper and discuss relative open questions and possible next steps for this solution in section 6.

2 RELATED WORKS

The problem of data extraction from documents images has drawn attention over decades [21]. Generally, it involves two types of tasks: text recognition and document parsing. With the application of deep neural networks, OCR engines such as Tesseract [30] have achieved good performance in dealing documents with simple textual layouts. [13] firstly employ convolutional neural networks to extract text features and train end-to-end learnable recognition models. Following this work, several advanced approaches were proposed. Broadly speaking, these approaches can be categorized into two classes: separated text detection and recognition processes [3, 11, 20, 27, 35], or joint text detection and recognition processes such as [4, 16, 19]. However, document images have natural structured regions to represent various data, such as tables, figures and text blocks. One single OCR step is not enough for such analysis problem. Consequently, there are some works concentrating on extracting document data directly into structured formats [5, 6, 15, 22]. However, such works usually need accurate textual information and expertise knowledge from users, which limits their adaptability in industry applications. Moreover, Raoui-Outach et al. and Augusto Borges Oliveira and Palhares Viana recently propose to leverage deep convolutional networks to recognize different regions of document images. However, these approaches can hardly handle document images with complex data representations, such as tables without form lines. Different from the related methods, we propose to formulate the layout recognition problem as a object detection task. With various scale of view, out systems is capable to locate each table cell, and the tabular contents will be simply converted to structured representations with a table construction algorithm. Object detection is a hot research topic in computer vision area and state-of-art approaches [7, 9, 24] have been utilized for words detection [3, 33, 35]. In this work, we use Light-Head RCNN [17] for both precision and efficiency consideration.

Besides the high performance, the reason why we choose deep neural network-based methods in our system is that, the models are expected to be fine-tuned to quickly adapt to the new coming data or new class, which is natural for a online system. There exists several researches working on incrementally improving trained models with online data such as [28, 32]. Inspired by such works, we design an interactive learning strategy to train online models with author feedbacks and avoid the catastrophic forgetting problem.

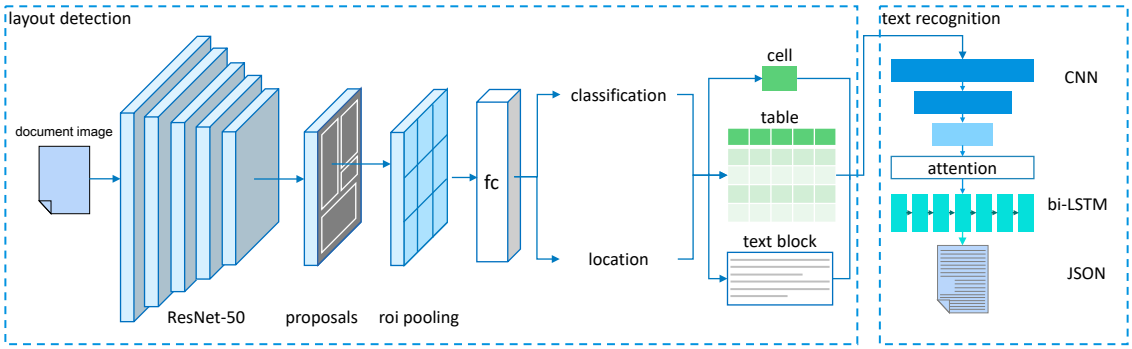

Figure 2: A diagram of data ingestion process for printed texts. The architecture of region detection model is on the left, and the right part comes the recognition model for printed texts.

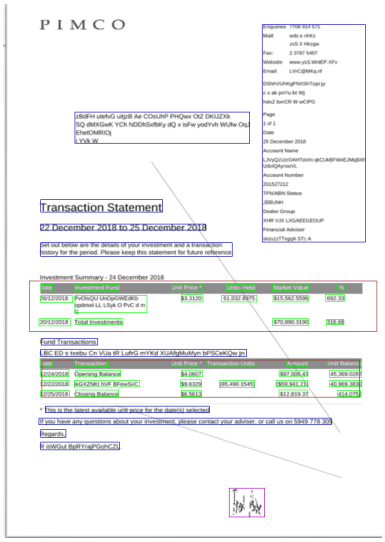

Figure 3: Regions detected on a sampled document image, where the tables, tabular cells, text blocks and handwritings are labeled with red, green, blue and purple boxes.

3 DATA INGESTION FROM DOCUMENT IMAGES

In this section, we introduce the proposed data ingestion approaches and point out how our methods differ from the existing solutions.

As illustrated by Figure 1, the system comprises of three major function components: 1) a region detection module, 2) a text recognition module and 3) a incremental learning module.

Given a document image, the first step is to detect rectangle regions of data, and predict the corresponding layout labels including handwriting blocks, tables, tabular cells and text blocks. In the second step, we apply text recognition, where, the detected regions are fed to a attention-based convolutional recurrent (ACRNN) model to recognize the texts. This workflow enjoys several advantages. First, once the regions are detected, text recognition can be performed in parallel over all regions. Second, the detection module and recognition module can be updated independently. Furthermore, since the models are decoupled, they can be shared by other systems as individual interfaces.

Moreover, different from existing solutions, the machine learning models in this system are not statically pre-trained before deployment, but can be dynamically updated according to new data or added classes. To this end, we propose a knowledge distillation-based incremental learning module.

The whole process of document understanding is conducted semi-automatically as it is too risky to allow full straight through processing in document ingestion when the image quality is impacted by many factors. A UI is provided for operational staff to review and edit results. At every step where manual check is involved, any correction made will be fed into relative incremental training process to continuously refine the model.

3.1 Region Detection Module

We train two independent models to detect handwritings and other regions (tables, cells, and text blocks) respectively for the purpose of task decoupling. Another important reason is that it is hard to train a single model to capture heterogeneous features of handwriting and printed texts simultaneously. For ease in writing, we refer the process of locating tables and text blocks as *layout detection*. An example of regions we are interested in is shown in Figure 3.

Handwriting Detection. For handwriting detection, we build a object detection model based on Light Head R-CNN [17]. The model is trained on a two class problem, namely, "handwriting" and "background". The detection process can

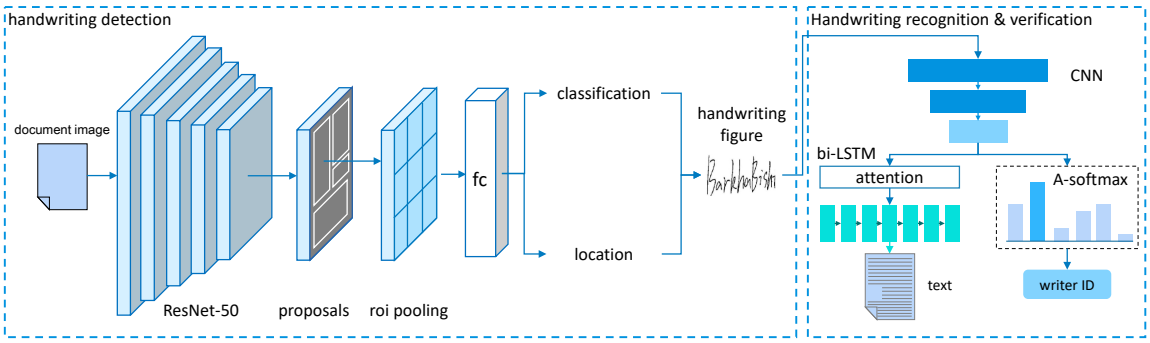

Figure 4: The architecture of handwriting detection model (left) and recognition model (right). The handwriting recognition model is trained to jointly learn sequence recognition and handwriting verification simultaneously.

be summarized as following steps as shown in Figure 4, 1) processing the entire image with a ResNet-based basic feature extractor [10] and a large separable CNN [34] sequentially to extract the feature representation, 2) generating several regionals of interest (RoI) via a region proposal network (RPN) by feeding the feature map as input, and 3) leveraging a R-CNN subnet to predict class label and location of each region candidate. Our detection model uses typical architecture of Light Head R-CNN but replaces the ResNet-101 CNN extractor with ResNet-50 for efficiency. The entire model is trained end-to-end using stochastic gradient methods.

Layout Detection. To construct a structured data representation, we need to find the bounding boxes of each minimum area with a independent semantic label, such as a text block or a tabular cell. However, since texts in tables and other snippets often share fonts and sizes, it is extremely challenging to capture all such regions by training a single model. Concretely, all existing solutions segment a document image into tables and text blocks, but fail to provide more find-grained data representations. In this work, we address this challenge by formulating the layout detection task as a four class problem, including text blocks, tabular cells, tables and the background. The introduction of class *table* can improve the detection accuracy of cells by explicitly providing the relative spatial relationship between tables and text blocks. The model architecture is similar to that of the handwriting detection model by adopting Light Head R-CNN on a four class problem as shown in the left part of Figure 2. Moreover, since the layout detection model should be able to recognize both coarse-grained (tables and text blocks) and fine-grained regions (tabular cells), we modify the original RPN in Light Head R-CNN to generate anchors of wider range of scales. To be specific, the tweaked RPN generates 56 anchor boxes for each RoI with eight scaling ratios ($\{0.25, 0.5, 1, 2, 4, 8, 16, 32\}$) and seven aspect ratios ($\{0.25, 0.33, 0.5, 1, 2, 3, 4\}$). Once the final regions are determined, we apply a cell combination algorithm to construct table-structured data representations. This algorithm constructs rows and columns according to y-coordinates and x-coordinates respectively with the similar manner. The algorithm for data rows is detailed in Algorithm 1. Once all regions are extracted, the location information is constructed as a JSON file, ordered by y-coordinate of the top edge.

Algorithm 1: Table Construction for Data Rows

Input: $C = \{c_1, \cdots, c_N\}$, where C is the list of tabular cells sorted by the y-coordinates of the top edges; the height threshold H.
Output: $\mathcal{R} = \{r_1, \cdots, r_N\}$, where r_i denotes the set of rows that c_i occupies.

```
n = 0;
y_last = −∞;
for i = 1; i <= N; i++ do
    Get the y-coordinates of top and bottom edge of
      c_i: ymin_i, ymax_i;
    if ymin_i − y_last > H then
        n++;
        y_last = ymax_i;
        Add n to r_i;
        foreach c_j in Row n − 1 do
            if ymax_j > ymin_i then
                Add n to r_j;
            end
        end
    else
        Add n to r_i;
    end
end
```

Handling Overlapped Regions. Since several RoI can be overlapped, non-maximum suppression (NMS) is employed

by typical object detection models to reduce redundant proposals. For handwriting blocks, we simply set the intersection-over-union (IoU) threshold of NMS as 0.3 to reduce overlapped region candidates. However, the situation is complicated when applying layout detection. We observe that one single tabular cell or text block is usually detected as several overlapped regions due to the existence of spaces and empty row, which severely harms the data integrity. And simply setting the IoU threshold to a small value will lead to missing valid regions. To address this problem, we propose a region combination approach (RC) by extending the standard NMS. Specifically, the first step is to apply standard NMS with a large IoU threshold (0.7 in this work) to reserve as many candidate regions as possible. In the second step, all overlapped rectangles are combined as one. The process is detailed in Algorithm 2.

Algorithm 2: Region Combination Algorithm

Input: The set of initial candidate regions $\mathcal{H}$ of the same class.
Output: The set of final regions $\mathcal{L}$ initialized as $\varnothing$.

```
Reduce overlapped region proposals by applying NMS with a large IoU threshold (0.7 in this work). The intermediate set of all reserved regions is denoted by M;
while M ≠ ∅ do
    Find the region r from M with the highest confidence score;
    Remove r from M;
    foreach region candidate c in M do
        if IoU(r, c) > 0 then
            Combine r and c, and the combined region is denoted by s;
            Add s to N;
            Remove c from M;
        end
    end
end
```

Training Strategy. The handwriting detection model can be trained end-to-end directly. In the contrary, the layout detection model is hard to train on document images with complicated styles. To address this challenge, we propose a warm-up training strategy to pre-train the model on synthesized data, inspired by [2]. For model training's efficiency and effectiveness concern, we artificially synthesize a large amount of mock data, which has similar data structures (e.g., tables and text blocks) with the real document images but the visual styles (e.g., fonts and shading) are much more simplex. Once the pre-trained model is converged, we set it as the initialization weights and further train the detection model on document images which are collected in the real scenario with sensitive information hidden. Such document images are named as *true data* for convenience in the rest of this paper. This easy-to-hard training strategy makes model converging process faster. 5 shows both an example synthesized document image and samples from the true data set.

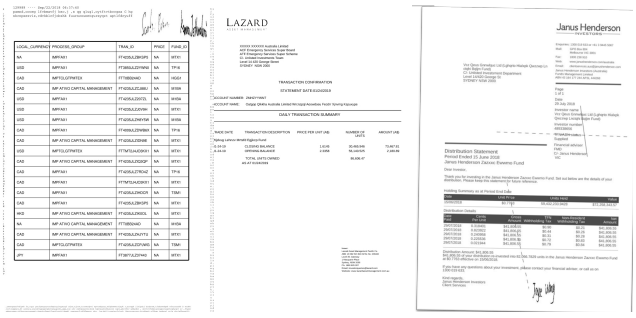

Figure 5: Comparison between real scene documents (middle and right) and a synthesized one.

3.2 Text Recognition Module

Once the regions are detected, the relative location information is then passed to the text recognition module which extracts characters sequentially from the regions. Similar to the decoupling strategy implemented in the region detection module, we also train two independent models to recognize printed texts and handwritings respectively.

Printed Text Recognition. Typically, a recognition system for printed texts involves two components, a line detector and a character extractor. The SOTA document image analyzers [14, 33] use deep CNN-based detection models such as Fully Convolutional Network (FCN) [36] and Connectionist Text Proposal Network (CTPN) [8] to detect text lines. Different such solutions, we utilize the Tesseract line segmentation algorithm [29, 30] based on connected component analysis [25]. The reasons can be summarized as two points. First, we have already predicted the rectangles which precisely cover each text block or tabular cell. With this step, we can assume that all characters in a region share the same font and size. Therefore, the text lines can be segmented precisely according to the blank areas. The second point is that CNN-based models undoubtedly require more computation resource and inference time.

After text line segmentation would follow a sequence recognition model. Since the length of text in a line is not fixed, the recognition model should be capable to handle sequences with arbitrary lengths. Existing solutions [3, 16] employ Connectionist temporal classification [26] output

layer or lstm encoder-decoder architecture to address this problem. Different from these works, we only use a convolutional network to extract sequential visual features and an attention-based lstm decoder character extraction as shown in Figure 2. We refer this model as attention-based convolutional recurrent neural network (ACRNN) in the rest of this paper. Considering that document images are usually clear and easy to decipher compared with scene texts, we choose a much smaller architecture for efficiency. Specifically, we use Google BN-Inception [34] as the basic feature extractor and the dimension of lstm hidden state is set as 256.

Handwriting Recognition. Handwriting recognition is a challenging research topic which has been studied for decades. There are two major distinctions between printed texts and handwritings. First, the features of handwritten texts are influenced by the writing styles of different individuals. Second, since the input images of a text recognition model should be resized to the same width during training, the length of output sequence therefore depends on various writing habits. Motivated by these factors, we propose a novel handwriting recognition model, which jointly learns to extract characters from handwriting regions and verify the handwritings simultaneously as shown in Figure 4. In a nutshell, this method consists of two sub-models sharing the same CNN feature extractor. We incorporate a image classification model into the original sequence recognition model, which is responsible for identifying the owner of handwritings. The output layer of the classification model is Angular Softmax (A-softmax) [18], which makes the decision boundary more stringent and separated. In this way each handwriting can be more likely to be categorized into a certain class rather close to the boundary between two classes. Formally, the training object of the whole model is:

$$\min \lambda \mathcal{L}_c + \frac{1}{N}\mathcal{L}_r, \tag{1}$$

where $\mathcal{L}_c$ and $\mathcal{L}_r$ are the loss functions of handwriting and sequence recognition respectively, N is the length of the target text, and λ controls the ratio between two training losses. In this work, we simply set *lambda* to 1. This model enjoys several benefits, including the ability to distinguish the handwriting features belonging to different individuals, and improve the robustness with multi-task learning. In the rest of this paper, we refer this multi-task model as ACRNN-MT.

Moreover, the incorporation of handwriting verification brings additional application value. In practice, many financial document images, such as fax, receipt, invoice, contain signatures and handwritten modifications, where the handwritings must be verified to guarantee the data security.

Image Preprocessing. Image preprocessing plays a critical role in text recognition. Following current approaches to text recognition [13], we resize the input images to 32x128 pixels. This process is necessary to train the models in parallel with batches of images. Straight after reshaping the inputs, we apply several preprocessing methods to improve image quality, including grayscaling, brightness enhancing, background cleaning, noise reducing, sharpening, trimming background, and border padding. All these functions are provided by open source ImageMagick [12].

3.3 Incremental Learning Module

One of the crucial distinctions between SOTA systems and ours is that the machine learning models are not *static*. That means, we do not only apply pretrained models for different tasks, but also fine-tune the models to adapt for new coming data or additional class labels. In this paper, we focus on two cases:

(1) Region detection models generate inaccurate bounding boxes on new coming images and users have provided corresponding corrections.
(2) For the handwriting verification model, new classes should be added when new writers are authorized.

The above tasks can be formulated as transfer learning or domain adaptation problems. Ideally, the models are expected to be fine-tuned to adapt to new data or new class. However, neural networks may suffer from *catastrophic forgetting* in the absence of the original training data. The internal representations of neural networks can be severely affected by domain vicissitude of the new coming data. Motivated by such challenges, we propose a knowledge distillation-based incremental learning method inspired by [28].

Specifically, given a trained model M with fixed parameters, the first step is to make a copy M^* from M. We modify the output layer of new copied model only if new classes of handwritings are added, and keep the model unchanged in other cases. At each timestep during training, we first sample a batch of data T_o from D. Then we divide both M and M^* into several groups and compute the distillation loss on T_f as:

$$\mathcal{L}_{dist} = \sum_{l=1}^{L} ||\phi_M^l(T_o) - \phi_{M^*}^l(T_o)||_2^2, \tag{2}$$

where $||\cdot||$ is L2 norm, ϕ_M^l and $\phi_{M^*}^l$ denotes the output layer of the l-th group of M and M^* respectively. For object detection models, the groups are divided according to the function of different components, including feature extractor, region proposal network and R-CNN subnet. The feature extractor CNN is further separated into several blocks with respect to the pooling processing (BN-Inception) or residual blocks

(ResNet). To adapt the model for the new task, we train M^* on T_f by minimizing the standard classification or detection loss $\mathcal{L}_{task}$. The final loss is computed as:

$$\mathcal{L} = \mathcal{L}_{task} + \alpha \cdot \mathcal{L}_{dist} \tag{3}$$

where α is a hyper-parameter to control the ratio between the distillation loss and original loss, which is simply set to 1 in this work. The algorithm is detailed in Algorithm 3.

Algorithm 3: Incremental Learning to adapt to new data or new class.

Input: Trained Model M; Author feedback set D_f; Original training dataset D; Batch size B; New data rate p; Weight of distillation loss α, Training algorithm A

Output: Trained New Model $M^*(\theta)$

while *not converge* **do**
  Random sample a training batch T, including $B \times p$ samples from D_f as T_f and $B \times (1-p)$ samples from D as T_o respectively
  Compute distillation loss $\mathcal{L}_{dist}$ on T_o as Eqn. 2
  Compute task loss $\mathcal{L}_{task}$ on T_f
  Update model paramter θ with A by minimizing training loss $\mathcal{L} = \mathcal{L}_{task} + \alpha \cdot \mathcal{L}_{dist}$
end

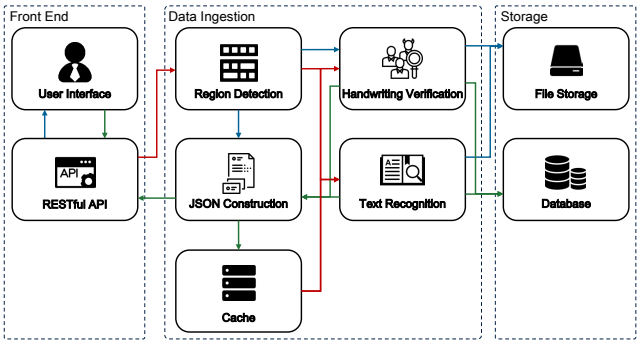

Figure 6: Architecture of the document understanding system.

4 SYSTEM ARCHITECTURE

In this section, we introduce the architecture of the document image understanding system, which has been deployed in the development environment but not yet in production. As shown in Figure 6, the system consists of three components, a front end server providing a user interface, a back end for computation, and a storage layer for data persistence.

Table 1: Averaged time overhead of the machine learning models.

Function Module	Model	Inference Time
Handwriting Detection	Light-head R-CNN	0.32 secs
Layout Detection	Light-head R-CNN	0.79 secs
Handwriting Verification	A-softmax	0.19 secs
Handwriting Recognition	ACRNN	1.15 secs
Printed Text Recognition	ACRNN	2.47 secs

The entire process of document image understanding is as follows:

(1) The user uploads a image through a user interface, and then the image is downloaded to the back-end server.
(2) The image is pre-processed and passed to the region detection module. The location information is constructed as JSON format and returned to UI through a RESTful API. The user is required to check the returned results and correct the mistakes. The JSON data is also written to a in-memory cache, and the raw image is stored in the file storage. In this work, we use Redis[1] as the in-memory storage.
(3) The text recognition module reads JSON data from the cache and fill in the recognized texts and verification information. The parsed data is finally stored in the database for data persistence.
(4) All correction information provided by the users is staged in the cache until the incremental learning is executed regularly.

To reduce the time spending in communication, we use coroutines for concurrence in the back-end, which is capable to serve at most 1024 manual checkers synchronously with little memory overhead. The detailed time cost of different machine learning models for a single image is listed in Table 1. It should be noted that text recognition is executed concurrently among all regions. From the table, we can observe that the average time to process one single document image is less than 5 seconds, which is much faster than a manual worker does and applicable for financial industry. The step taking most time is the text recognition process and the inference time depends on the text length.

5 EXPERIMENTS

Here, we present the experimental results and describe the lessons learned from model training and system development.

[1] https://redis.io/

Table 2: Statistics of synthetic document images

# Signatures	0	1	2	3	4
# Images	10,000	63,939	28,520	16,784	11,688

5.1 Data Collection

The dataset is constructed by collecting fax images from the real business scenario in State Street Corporation. We remove the sensitive information such as personal details, signatures and fund names. Typical examples of the images are shown in Figure 5. After preprocessing, we obtain 84,000 fax images for training, 18,000 for validation and 18,000 for test respectively. For convenience, we refer such a collection as the *true data*.

Since the collected fax images do not contain handwritten signatures after preprocessing, we manually collected handwriting figures provided by the staff in State Street. To be specific, we manually collected 86,481 signatures written by 486 individuals. In order to get a robust model to precisely distinguish handwriting features from different individuals, we ask the writers not to sign their own names but to pick five to ten names from a given name list which includes 510 English names, 260 Chinese PinYin names and 264 Indian names, with lengths vary from 5 to 32 characters. All names are signed on a blank form with A4 size. Papers with signatures are sent to a scanner and the scanned signatures are split according to form lines. The average height and width of the signature images are about 140 and 400 pixels respectively. The scanned signatures are processed using background elimination, randomly zooming and twisting, and finally pasted to the fax images to replace the original ones. We split the signature dataset as 72833/5000/8648 for training/validation/test of the handwriting recognition task.

Furthermore, as discussed above, we use an easy-to-hard training strategy to warm up the training process with a large synthetic dataset. The images in the synthesized dataset are much simpler in terms of layouts, but vary more in density of table/text cells, fonts and sizes. Since the features in structures at coarse-grained level (layouts and formats) is easy to capture, the model can converge quickly, and the introduced fine-grained variabilities (locations and fonts) can bring robustness to the model. A synthesized image may have multiple text blocks, tables and signatures. The size, location, font and rotation of each text area are all random. In this way, we are able to obtain a large-scale labeled dataset with 80,000 samples. We split the dataset with ratio 70%/15%/15% for training/validation/test respectively. Similar to the true dataset, we paste 0~5 scanned signatures into the white space of synthesized image. The specific statistics are listed in table 2.

5.2 Model Settings

For region detection tasks, the models use Light Head R-CNN architecture, but replace the original ResNet-101 convolutional feature extractor with ResNet-50. The layout detection model is firstly trained on the synthesized data and fine-tuned on the true dataset. In the contrary, the handwriting detection model is trained only on synthesized images. During inference, the confidence threshold is set to 0.5 for handwriting and 0.1 for other regions. For text recognition tasks, the CNN architecture uses BN-Inception with the final pooling layer and fully connection layers removed. The dimensions of sequential visual features and lstm hidden states are 1024 and 256 respectively. Both hyper-parameter α (Equation 3) and λ (Equation 1) are set to 1. Early stopping and Stochastic Gradient Descent (SGD) with momentum term of 0.9 is employed to train all models. The initial learning rate is 0.001 and the batch size is 32 for all neural network models.

5.3 Metrics

We employ multiple metrics for experimental evaluation of different tasks.

Region Detection. For region detection models, we use precision and recall score at certain Intersection-over-Union (IoU) threshold as the metrics. IoU represents the overlap ratio between the generated candidate box and the ground truth area, calculated as the intersection area over the union area. A predicted label is regarded as correct if IoU exceeds a certain threshold. In this paper, the IoU threshold is set to 0.85 which is stricter than the common settings. Formally, the precision and recall is calculated by

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}, \tag{4}$$

where TP, FP, FN are *true positive*, *false positive*, and *false negative* predicted labels.

Handwriting Verification. For handwriting verification, we use top-1 and top-5 error rates for performance evaluation. Specifically, in the case of top-n, the model gives n candidates with highest probabilities and the answer is regarded as correct if the ground truth label is in the n predictions.

Text Recognition. For text recognition, we evaluate the performance of recognition with fragment accuracy, which provides the percentage of score fragments correctly recognized. Specifically, accuracy is computed as:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}, \tag{5}$$

where TP, TN, FP, FN are *true positive*, *true negative*, *false positive*, and *false negative* recognized words.

Table 3: Main results of region detection in document images from real scenes.

Solution	Handwriting		Table		Cell		Text Block		Inference Time
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	
Adlib PDF	99.9	17.7	66.2	4.1	53.9	9.7	77.5	96.9	-
Fast CNN	-	-	89.6	11.2	-	-	92.1	91.7	0.68 secs
Faster R-CNN	99.0	99.9	96.2	99.0	99.9	93.1	99.1	97.1	3.84 secs
Ours	99.0	99.8	95.1	98.8	99.9	97.4	98.9	97.8	0.79 secs

5.4 Results and Analyses

Region Detection. For the region detection tasks, we compare our models with following SOTA solutions:

- **Adlib OCR**: Adlib OCR[2] is a enterprise optical character recognition software. It can convert document images into searchable PDF files. Here, tables and text blocks are parsed by pdfplumber[3]
- **Fast CNN**: Fast CNN [1] is Deep CNN-based solution for document segmentation and content classification.
- **Faster R-CNN**: Faster R-CNN [24] is the SOTA object detection model. [31] proposes to leverage Faster R-CNN for region detection task from document images. We train such a model sharing the same settings with our solutions.

The main results are listed in Table 3. From this table, we can observe that our proposed model outperforms state-for-art solutions on document layout understanding. Adlib PDF and Fast CNN can hardly detect tables and cells since most of the collected images do not contain tabular lines. The performance of Faster R-CNN is competitive with ours. However, it takes almost 5 times longer for inference.

From Table 4, the performance drops sharply when training on the *true data* directly while the warm-up strategy brings about 25% improvements in average. The reason is that the real scene documents has a large amount of complex drawings, making the CNN architecture difficult to map the original image into the latent space and this is the first lesson we learned during model training. Moreover, it should be noted that it is interesting to observe that the test performance of handwriting detection is good enough even only the synthesized images are provided for training. The possible reason is that the features of handwritings significantly differ from the printed texts, which can be captured easily by CNN feature extractors. Another important lesson we learned from this task is that training the model to detect cells and tables jointly can indeed improve the detection accuracy as shown in Table 5.

[2]https://www.adlibsoftware.com

[3]https://pypi.org/project/pdfplumber/

Table 4: Performance on true data test set with different training sets.

Dataset	F-measure			
	Text	Table	Cell	Handwriting
Synthetic Data	58.4	49.6	51.2	99.4
True Data	83.1	76.3	77.0	99.2
Synthetic → True	98.3	96.4	98.6	99.4

Table 5: Performance of variant models on layout detection tasks.

#	Model	F-measure		
		Text	Table	Cell
1	Light Head R-CNN	98.3	96.4	98.6
2	Model #1 without RC (Algorithm 2)	91.8	88.3	79.4
3	Model #1 with class *table* deleted	84.6	-	17.5

Text Recognition. For text recognition, we compare our models with the SOTA system proposed by [14], which combines CTPN and an attention-based encoder decoder model (AED). We refer this system as CTPN-AED for convenience. Comparisons between recognizing accuracy of different models are listed in Table 6. The performance of ACRNN is comparable with CTPN-AED but only needs about half the time for inference. For handwriting recognition, both ACRNN and CTPN-AED achieve quite low scores while the incorporation of handwriting verification in ACRNN-MT brings a significant improvement. This is also an interesting lesson we learned during system designing, that learning to distinguish handwriting features enjoys the ability to boost recognition performance.

We also list the results of handwriting verification in Table 7. We can observe that the Angular Softmax model significant outperforms the conventional softmax classification model and the final Top-5 precision of 97.54 guarantees the practicability of the verification model for industrial application.

Table 6: Recognition accuracy of different solutions.

Solution	Printed Text		Handwriting	
	Accuracy	Speed	Accuracy	Speed
CTPN-AED	92.2	4.2 secs	71.8	2.7 secs
This work				
ACRNN	92.7	2.5 secs	71.3	1.1 secs
ACRNN-MT	-	-	88.6	1.2 secs

Table 7: Performance of handwriting verification

Output Layer	Top-1 error rate	Top-5 error rate
Softmax	19.06	8.57
A-Softmax	9.48	2.46

Table 8: Top-1 precison of handwriting verification when a new class is added.

Method	Top-1 Error Rate	
	A(486 classes)	A + B(487 classes)
Fine-tuning on B	46.80	46.56
Algorithm 3	9.50	9.24

Incremental Learning. It is hard to evaluate the effectiveness of the incremental learning methods since it should be tested in the production environment for a long time, which involves a lot of manual effort. We leave it as an open question for future study. In this paper, we conduct a simulated experiment and, in one sense, prove the effectiveness of the method. To be specific, we artificially synthesize a dataset of 500 images containing short text sequences, and split the data as 400/50/50 for training/validation/text respectively. These images then appended to the signature dataset as a new class. For convenience, we denote the set the original 486 classes as partition A and the new coming class as B. The Top-1 error rate of handwriting classification is listed in Table 8. The baseline in Row 1 is fine-tuned directly only on the new data. According to the results, the model fine-tuned on the new class suffers from performance degradation on the original classes. In contrary, the classification precision of the model trained with Algorithm 3 is similar to the old model in Table 7, indicating the effectiveness of our method.

6 CONCLUSION AND FUTURE WORK

In this paper, we present a system that automatically parses document images into the structured formatted data. The system consists of three major functional components: region detection, text recognition and incremental learning module. We conduct experiments on various tasks, and the results indicate the effectiveness and efficiency of our solutions.

An interesting topic for future study is to systematically evaluate the performance of our system in the production environment. We are also interested in investigating approaches to extract data from natural language sentences in document files.

REFERENCES

[1] Dario Augusto Borges Oliveira and Matheus Palhares Viana. 2017. Fast CNN-based document layout analysis. In *Proceedings of ICCV*. 1173–1180.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of ICML*. ACM, 41–48.

[3] Fedor Borisyuk, Albert Gordo, and Viswanath Sivakumar. 2018. Rosetta: Large scale system for text detection and recognition in images. In *Proceedings of the 24th ACM SIGKDD*.

[4] Michal Bušta, Lukáš Neumann, and Jiri Matas. 2017. Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In *ICCV*.

[5] Francesca Cesarini, Marco Gori, Simone Marinai, and Giovanni Soda. 1998. INFORMys: A flexible invoice-like form-reader system. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, 7 (1998), 730–745.

[6] Jean-Pierre Chanod, Boris Chidlovskii, Hervé Dejean, Olivier Fambon, Jérôme Fuselier, Thierry Jacquin, and Jean-Luc Meunier. 2005. From legacy documents to xml: A conversion framework. In *International Conference on Theory and Practice of Digital Libraries*.

[7] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. 2016. R-fcn: Object detection via region-based fully convolutional networks. In *Proceedings of NIPS*.

[8] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of ICML*.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of ICCV*. 2961–2969.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of CVPR*.

[11] Wenhao He, Xu-Yao Zhang, Fei Yin, and Cheng-Lin Liu. 2017. Deep direct regression for multi-oriented scene text detection. In *Proceedings of ICCV*. IEEE, 745–753.

[12] LLC ImageMagick Studio. 2014. ImageMagick: Convert, Edit, and Compose Images. *Version* 6 (2014), 9–1.

[13] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227* (2014).

[14] Anh Duc Le, Dung Van Pham, and Tuan Anh Nguyen. 2019. Deep Learning Approach for Receipt Recognition. *arXiv preprint arXiv:1905.12817* (2019).

[15] Chuan Li and Michael Wand. 2016. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *Proceedings of ECCV*.

[16] Hui Li, Peng Wang, and Chunhua Shen. 2017. Towards end-to-end text spotting with convolutional recurrent neural networks. In *Proceedings of ICCV*.

[17] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. 2017. Light-head r-cnn: In defense of two-stage object detector. *arXiv preprint arXiv:1711.07264* (2017).

[18] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. 2017. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of CVPR*.

[19] Xuebo Liu, Ding Liang, Shi Yan, Dagui Chen, Yu Qiao, and Junjie Yan. 2018. FOTS: Fast Oriented Text Spotting with a Unified Network. In *Proceedings of CVPR*. 5676–5685.

[20] Pengyuan Lyu, Cong Yao, Wenhao Wu, Shuicheng Yan, and Xiang Bai. 2018. Multi-oriented scene text detection via corner localization and region segmentation. In *Proceedings of CVPR*. 7553–7563.

[21] George Nagy. 2000. Twenty years of document image analysis in PAMI. *IEEE Transactions on Pattern Analysis & Machine Intelligence* (2000).

[22] Claudio Antonio Peanho, Henrique Stagni, and Flavio Soares Correa da Silva. 2012. Semantic information extraction from images of complex documents. *Applied Intelligence* 37, 4 (2012), 543–557.

[23] Rizlene Raoui-Outach, Cecile Million-Rousseau, Alexandre Benoit, and Patrick Lambert. 2017. Deep Learning for automatic sale receipt understanding. In *Proceedings of IPTA*.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of NIPS*.

[25] Azriel Rosenfeld. 1976. *Digital picture processing*. Academic press.

[26] Baoguang Shi, Xiang Bai, and Cong Yao. 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 (2016), 2298–2304.

[27] Baoguang Shi, Xiang Bai, and Cong Yao. 2017. 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* (2017).

[28] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. 2017. Incremental Learning of Object Detectors without Catastrophic Forgetting. In *Proceedings of ICCV*.

[29] Ray Smith. 1995. A simple and efficient skew detection algorithm via text row accumulation. In *Proceedings of ICDAR 1995*.

[30] Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Proceedings of ICDAR 2007*.

[31] Peter WJ Staar, Michele Dolfi, Christoph Auer, and Costas Bekas. 2018. Corpus Conversion Service: A machine learning platform to ingest documents at scale. In *Proceedings of the 24th ACM SIGKDD*.

[32] Pei-Hao Su, Milica Gasic, Nikola Mrkšić, Lina M Rojas Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line Active Reward Learning for Policy Optimisation in Spoken Dialogue Systems. In *Proceedings of ACL*.

[33] Vishal Sunder, Ashwin Srinivasan, Lovekesh Vig, Gautam Shroff, and Rohit Rahul. 2019. One-shot Information Extraction from Document Images using Neuro-Deductive Program Synthesis. In *the 13th International Workshop on Neural-Symbolic Learning and Reasoning at IJCAI 2019*.

[34] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of CVPR*.

[35] Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao. 2016. Detecting text in natural image with connectionist text proposal network. In *Proceedings of ECCV*.

[36] Zheng Zhang, Chengquan Zhang, Wei Shen, Cong Yao, Wenyu Liu, and Xiang Bai. 2016. Multi-oriented text detection with fully convolutional networks. In *Proceedings of CVPR*.