Chinese/English mixed Character Segmentation as Semantic Segmentation

Huabin Zheng
Sun Yat-sen University
Guangzhou, China
zhhuab@mail2.sysu.edu.cn

Jingyu Wang
iPIN
Shenzhen, China
wangjingyu@ipin.com

Zhengjie Huang
Sun Yat-sen University
Guangzhou, China
hzhengj@mail2.sysu.edu.cn

Yang Yang
iPIN
Shenzhen, China
yangyang@ipin.com

Rong Pan
Sun Yat-sen University
Guangzhou, China
panr@sysu.edu.cn

Abstract

OCR character segmentation for multilingual printed documents is difficult due to the diversity of different linguistic characters. Previous approaches mainly focus on monolingual texts and are not suitable for multilingual-lingual cases. In this work, we particularly tackle the Chinese/English mixed case by reframing it as a semantic segmentation problem. We take advantage of the successful architecture called fully convolutional networks (FCN) in the field of semantic segmentation. Given a wide enough receptive field, FCN can utilize the necessary context around a horizontal position to determinate whether this is a splitting point or not. As a deep neural architecture, FCN can automatically learn useful features from raw text line images. Although trained on synthesized samples with simulated random disturbance, our FCN model generalizes well to real-world samples. The experimental results show that our model significantly outperforms the previous methods.

1. Introduction

Character segmentation plays an important role in optical character recognition (OCR) pipeline [18]. One major reason for poor recognition accuracy in OCR system is the error in character segmentation. Some previous researches [1, 2, 5, 6, 13, 16, 17, 19, 23] achieve high performance on monolingual texts, but rely on feature engineering specific to single character style. Other researches [4, 10, 24, 25] work on multilingual cases but introduce complex processing pipelines. Actually, it's difficult to manually design a set of features suitable for multilingual scene. Thus a mixture of multiple languages presents a challenge for existing character segmentation methods. Chinese/English

mixed case is especially difficult due to the coexistence of touching characters and Chinese disconnected structure, as shown in Figure 1. For those ignorant of both languages, it's confusing that a Chinese character with disconnected structure (*e.g.* those in Figure 1) should *not* be splitted apart, but a pair of touching neighboring English characters (*e.g.* "DL" or "AI") should be splitted apart. Traditional projection based method will falsely break up an intact Chinese character with disconnected structure. A more advanced method [13] with a connected regions merging phase, tends to falsely take "DL" as an intact character. In order to correctly perform segmentation, a model should implicitly or explicitly remember all valid characters in both languages. Moreover, in order to deal with various possible font types and sizes, the model should automatically learn necessary features to recognize a valid character since it is cumbersome to specifically design features for every font.

Nowadays we know the ability of deep neural networks to perform automatic feature learning on raw data has significantly advanced the research in various fields of computer vision. Semantic segmentation is among these fields. Fortunately, the problem of multilingual character segmentation can be reframed as the problem of two-class semantic segmentation. To be specific, given a text line image, we classify each horizontal pixel into two categories: splitting point or not. With this problem re-defined, we can utilize those successful deep architectures in recent progress of semantic segmentation. Among them we choose fully convolutional networks (FCN).

In this paper, we reframe OCR character segmentation as semantic segmentation and propose a FCN architecture to solve it. We train our FCN model on synthesized samples with simulated random disturbance and show that it is able to

1. significantly outperform previous methods on Chi-

Figure 1. The main challenge of Chinese/English mixed character segmentation arises from the coexistence of disconnected structure in Chinese and touching characters. Correct and wrong segments are colored as blue and red, respectively.

nese/English mixed printed document images;

2. generalize well from simulated disturbance to real-word disturbance introduced by photographing;

3. generalize well across different text content styles;

4. generalize well across different font styles in most cases;

5. nicely handle disconnected structure and touching characters.

2. Related Work

Previous Approaches Projection based method is among the simplest approach for OCR segmentation. It calculates the average grey value for each pixel column then split every blank region in the middle, making it vulnerable to disconnected structure and touching characters. Recently, improved methods have been proposed but are only specific for single language [1, 2, 5, 6, 13, 16, 17, 19, 23]. Other researches exploit complex processing pipelines and hand-crafted rules to tackle multilingual cases [4, 10, 24, 25]. There are also researches on handwritten character segmentation [26, 22, 9, 3]. Compared with printed characters, handwritten characters often require nonlinear splitting paths rather than vertical splitting lines, which is not necessary for regular font types in normal printed documents.

Semantic Segmentation Semantic segmentation is a sub-field of computer vision. Compared with recognition problem, it progresses from coarse to fine inference by making a prediction directly at every pixel [12, 15]. To this end it requires the output size of model to match original input size. However, normal convolutional layer used in recognition only maintains or reduces the size of feature maps, so comes the deconvolution layer.

Devonlutional Layers Deconvolution is also called up-convolution [20] or fractionally-strided convolution [8]. It is typically used for expanding the size of feature maps in FCN architecture.

Fully Convolutional Networks FCN is prevalent in the research of semantic segmentation and object detection. The key feature that distinguishes FCN from CNN is that

it is easy to control the output size of FCN via deconvolution. Therefore, FCN is also widely used in tasks where both input and output are images. For example, Simo-Serra *et al.* [20] use FCN to simplify sketch drawing.

(a) Several training samples

(b) Zoom in to a region

Figure 2. (a) Several training samples and corresponding output ground truths. Notice that the model outputs a vector of length W rather than a matrix of shape $H \times W$. (b) For each character in the image, the left and right margins are given as splitting points. Each splitting point is visualized as a vertical blue lines.

3. Proposed Approach

We firstly define the training task in Section 3.1. In Section 3.2 we propose the detailed architecture of FCN. In Section 3.3 we describe the post-processing phase when using the trained model to crop segments. Training data synthesizing process is described in Section 3.4. To deal with imbalanced classes problem, we use a dynamic weighted binary cross entropy loss, which is defined in Section 3.5.

3.1. Training Task Definition

In semantic segmentation form, our model is to classify each horizontal pixel position into two classes: splitting point as positive class and non-splitting point as negative class. Formally, given an image of height H and width W as input, a FCN outputs a probability vector $\mathbf{p}$ of length W,

where

$$\mathbf{p}_i = \begin{cases} 1 & \text{if the image should be splitted} \\ & \text{at the } i\text{-th column;} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

For each character in the image, the left and right margins are given as splitting points. For example, if a character has an extending range from the m-th column to the n-th column, then we have

$$\begin{aligned} \mathbf{p}_i = 1, i = m, n, \\ \mathbf{p}_i = 0, i = m+1, m+2, \ldots, n-1. \end{aligned} \quad (2)$$

See Figure 2 for several input images and corresponding output ground truths. In this paper we have $H = 48$ and $W = 2048$.

3.2. Architecture

In a typical FCN architecture, a down-convolution block and an up-convolution block are to reduce and expand the size of feature maps, respectively [20]. Each down-convolution is composed of a convolutional layer, a batch normalization layer [7], a max-pooling layer and an activation layer. Each up-convolution is composed of a deconvolutional layer, a batch normalization layer and an activation layer. In research of semantic segmentation, FCN is used to restore both the width and height of original input images [12, 15].

Our FCN architecture is almost the same as typical ones except that only the width of input images need to be restored. As defined in Section 3.1, Eq. (1), it simply outputs a vector of length W, which is equivalent to an image of shape $1 \times W$. To this end, deconvolutional layers in our FCN expand the width of feature maps but maintain the height. See Figure 3 for details.

All activation layers except the last one are the ReLU layer [14]. The last one is a sigmoid layer producing a probability output vector.

3.3. Post-processing for Cropping

As described in Section 3.1, during training phase only two splitting points are given as positive ground truth for each character. During prediction, however, points in adjacent region of a true positive point are usually classified to positive as well. It's inevitable because neural networks cannot fit the data exactly. In fact, those surrounding points could also be valid splitting points.

Another problem arises when we want to really crop out characters for downstream recognition, because a segment bounded by two adjacent splitting points could contain a character or just the blank between characters. The blank segments should be discarded.

Therefore, we propose a simple post-processing procedure to deal with the issues above. Firstly, the probability

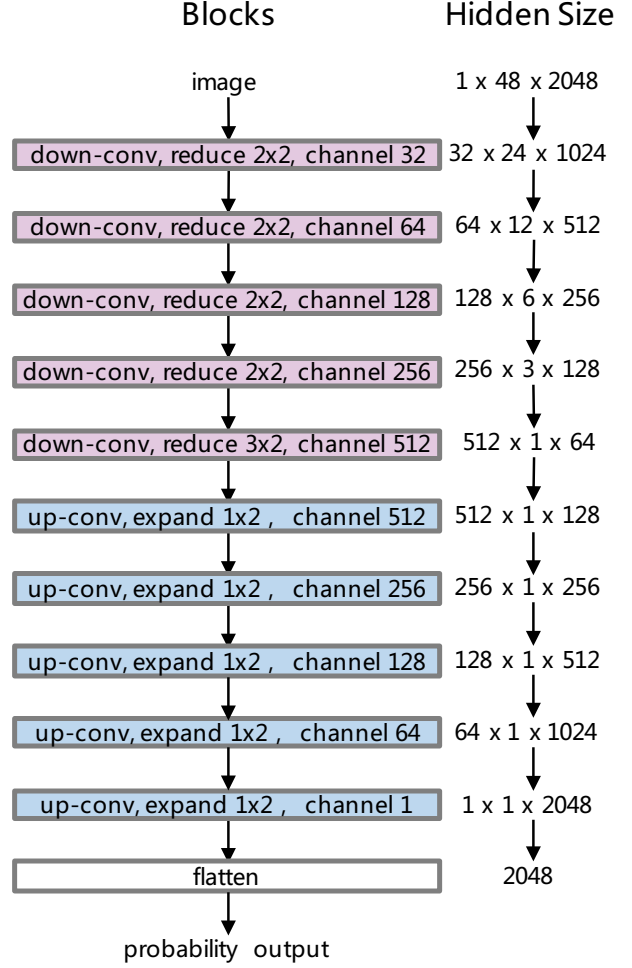| Blocks	Hidden Size
image	1 x 48 x 2048
down-conv, reduce 2x2, channel 32	32 x 24 x 1024
down-conv, reduce 2x2, channel 64	64 x 12 x 512
down-conv, reduce 2x2, channel 128	128 x 6 x 256
down-conv, reduce 2x2, channel 256	256 x 3 x 128
down-conv, reduce 3x2, channel 512	512 x 1 x 64
up-conv, expand 1x2 , channel 512	512 x 1 x 128
up-conv, expand 1x2 , channel 256	256 x 1 x 256
up-conv, expand 1x2 , channel 128	128 x 1 x 512
up-conv, expand 1x2 , channel 64	64 x 1 x 1024
up-conv, expand 1x2 , channel 1	1 x 1 x 2048
flatten	2048
probability output	

Figure 3. Our FCN architecture is composed of 5 down-convolution blocks and 5 up-convolution blocks. The activation layer of the last down-convolution block is a sigmoid layer.

vector output $\mathbf{p}$ of FCN is converted to binary vector according to a threshold (*e.g.* 0.5). Secondly, for each contiguous positive segments, the center point is selected as the splitting point. Thirdly, each pair of adjacent splitting points form a candidate segments. Finally, we discard the blank segments and output a list of bounding line segments. Complete pipeline is shown in Figure 4.

3.4. Synthesizing Training Data

We focus on camera photographs of Chinese/English mixed printed documents with various fonts. Compared with using scanner, camera photographing introduces much more noises, blurring, rotation and distortion, thus makes the problem more challenging. To approximate these real-world disturbance, we first plot clean texts onto blank image with a size of 2048×48 then successively apply four simulated random disturbance: rotation, erosion, dilation and

同时按下Ctrl与C键，产生信号

(a) Partial input image

(b) FCN's output probability

(c) **Step 1:** thresholding

(d) **Step 2:** select middle points

(e) **Step 3:** get candidate segments
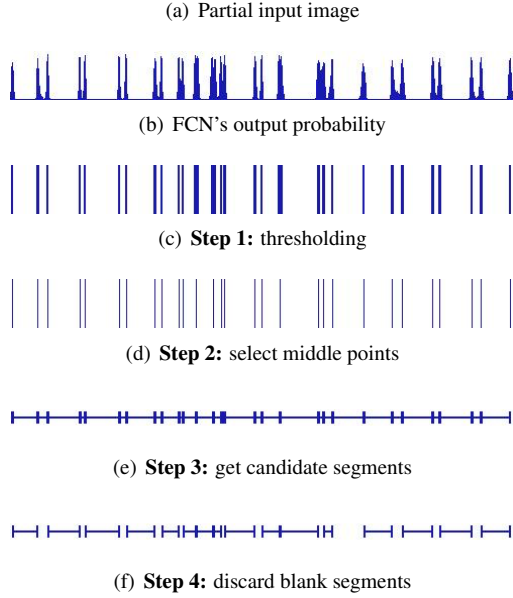
(f) **Step 4:** discard blank segments

Figure 4. Post-processing steps. (a) Partial input image to FCN. (b) Corresponding part of FCN's output probability vector. (c) **Step 1:** convert the probability vector into binary vector according to a threshold. (d) **Step 2:** for each contiguous positive segments, select the center point as splitting point. (e) **Step 3:** each pair of adjacent splitting points form a candidate segment. (f) **Step 4:** discard the blank segments and finally output a list of bounding line segments.

Gaussian blurring. Finally we binarize the grey-scale images according to threshold 160. We keep track of the left and right margins of each character in process above and finally convert them into corresponding binary mask vector as ground truth. Several training input images and output vectors are visualized in Figure 2.

3.5. Dynamic Weighted Binary Cross Entropy

Our model performs binary classification at each of the 2048 horizontal positions. Therefore, given FCN's output probability vector $\mathbf{p}$ and the ground truth vector $\mathbf{q}$, we can simply define the binary cross entropy loss function as

$$\mathcal{L}(\mathbf{p}, \mathbf{q}) = - \sum_{i, \mathbf{q}_i=1} \log \mathbf{p}_i - \sum_{i, \mathbf{q}_i=0} \log(1 - \mathbf{p}_i). \quad (3)$$

However, the ratio of positive and negative ground truths is not balanced. Most of the ground truths are negative as shown in Figure 2. In our experiments unbalanced classes slow down model convergence or even make the model stuck in a local optimum. When stuck, the model predicts negatively at each horizontal position. To tackle this issue,

we define a weighted binary cross entropy loss function

$$\mathcal{L}(\mathbf{p}, \mathbf{q}) = -\alpha \sum_{i, \mathbf{q}_i=1} \log \mathbf{p}_i - \beta \sum_{i, \mathbf{q}_i=0} \log(1 - \mathbf{p}_i), \quad (4)$$

where $\alpha + \beta = 1$.

We initialize α to 0.9 and β to 0.1. After each iteration we use a heuristic rule to dynamically adjust the weights according to the average positive accuracy acc_{pos} and the average negative accuracy acc_{neg} of the last mini-batch, where

$$acc_{pos} = \sum_{i, \mathbf{q}_i=1} \mathbb{1}(\mathbf{p}_i > 0.5) / \sum_{i, \mathbf{q}_i=1} 1, \quad (5)$$

$$acc_{neg} = \sum_{i, \mathbf{q}_i=0} \mathbb{1}(\mathbf{p}_i < 0.5) / \sum_{i, \mathbf{q}_i=0} 1. \quad (6)$$

If $acc_{pos} < acc_{neg}$ then we increase α and decrease β, otherwise we increase β and decrease α. Our strategy can balance the model performance on positive and negative classes throughout training and speed up convergence. See Algorithm 1 for details.

Algorithm 1 Heuristic Rules for Dynamic Loss

1: $\alpha \leftarrow 0.9$
2: $\beta \leftarrow 0.1$
3: **while** training **do**
4:      get mini-batch data for this iteration
5:      update model towards minimizing Eq. (4)
6:      compute acc_{pos} according to Eq. (5)
7:      compute acc_{pos} according to Eq. (6)
8:      **if** $acc_{pos} < acc_{neg}$ **then**
9:          $\delta \leftarrow min(\beta, 0.001)$
10:          $\alpha \leftarrow \alpha + \delta$
11:          $\beta \leftarrow \beta - \delta$
12:      **else**
13:          $\delta \leftarrow min(\alpha, 0.001)$
14:          $\alpha \leftarrow \alpha - \delta$
15:          $\beta \leftarrow \beta + \delta$
16:      **end if**
17: **end while**
---

4. Experiments

4.1. Datasets

In this section, we describe the datasets for experiments, including those built by photographing printed documents and those synthesized as described in Section 3.4.

4.1.1 Photographed Dataset

The first dataset is built by photographing as follows. Firstly, text contents are randomly extracted from Baidu

Baike corpus, printed with various font types, and photographed with normal phone camera. Secondly, we apply a series of traditional OCR techniques of denoising, binarization, line segmentation *etc.* to collect a set of text line images. Finally, we hand label the bounding line segment annotations for each character.

Because each text line image sample typically contains several dozens of characters, it takes a long time to annotate even one sample. Thus only 50 text line image samples are finally collected. Nevertheless, they totally contains 2710 characters, which are enough for reliable evaluation. In the following sections, we refer this dataset as "Photo-Normal". Several samples are shown in Figure 5.

Figure 5. Several samples of Photo-Normal Dataset.

4.1.2   Synthesized Datasets

The quantity and diversity of Photo-Normal dataset is somewhat limited. Moreover, it can only evaluate our model's generalization ability from simulated disturbance to real-world disturbance. To further evaluate its generalization ability, we synthesize a series of datasets with the combination of two text content styles and 36 font types (specified in next paragraph). Each combination is splitted into training and evaluation parts, producing totally 144 ($2 \times 2 \times 36$) datasets. Among them, one training set contains 3000 samples and one evaluation set contains 30 samples. Total number of characters is more than 10 million.

As for text content styles, the first style called "normal" simply refers to normal content in Baidu Baike corpus, and the second style called "chaotic" is acquired by randomly shuffling normal text characters. As for font styles, see Table 3 for totally 36 font types used in our experiments.

In the following sections, we refer a dataset of normal text content and font SIMYOU for training as "Train-Normal-SIMYOU", and so on.

4.2. Evaluation Metric

We evaluate segmentation accuracy by matching predictions and ground truths. Given a text line image, FCN and post-processing procedure output a list of M bounding line segments. They are aligned with the ground truth, which is a list of N bounding line segments. For each predicted segment P_i and each true segment T_j, we denote the number of T_j's horizontal pixels covered by P_i as $c_{i,j}$ and the number of T_j's horizontal pixels not covered as $u_{i,j}$. Then we say

P_i matches T_j if and only if

$$min(u_{i,1}, u_{i,2}, \ldots, u_{i,N}) = u_{i,j}, \quad (7)$$
$$max(c_{i,1}, c_{i,2}, \ldots, c_{i,N}) = c_{i,j}, \quad (8)$$
$$u_{i,j} < t_1, \quad (9)$$
$$c_{i,j} > t_2, \quad (10)$$
$$max(c_{i,1}, c_{i,2}, \ldots, c_{i,j-1}, c_{i,j+1}, \ldots, c_{iN}) < t_3, \quad (11)$$

where t_1, t_2, t_3 are thresholds.

Taking $t_1 = t_3 = 0$ is equivalent to requiring P_i exactly matched with T_j. However, exact matching is not necessary in practice due to some blank space between characters. Therefore we take $t_1 = 8$, $t_2 = 0$ and $t_3 = 5$ in our experiments for all the methods we compare, which is fair.

Given the number of matched pairs K, we define the segmentation accuracy as

$$acc = \frac{K}{max(M, N)}. \quad (12)$$

4.3. Hyperparameters

We use mini-batch stochastic gradient descent for 50000 iterations with batch size 8 and momentum coefficient 0.9. For each iteration a mini-batch of samples are randomly selected from training samples. Learning rate is initialized to 0.0001 and divided by 10 at the 20000-th and 40000-th iteration. Training takes approximately 50 minutes on a single GPU (NVIDIA GeForce GTX TITAN X).

4.4. Quantitative and Qualitative Evaluation

In this section, we quantitatively evaluate whether our model can generalize

1. from simulated disturbance to real-word disturbance,

2. between normal text content and chaotic text content,

3. among different font types.

We also qualitatively evaluate its performance on the hardest part of Chinese/English mixed case: coexistence of disconnected components and touching characters.

The generalization over font sizes is trivial and already included throughout these experiments, thus not specifically evaluated.

4.4.1   Generalization from Simulation to Real-World

Our FCN is trained on synthesized data because it's difficult to collect a large amount of real word samples with segment annotations. However, simulated disturbance in synthesized samples is definitely not identical to real-world disturbance. FCN must generalize well beyond simulation to deal with photographed printed document images.

To verify this, we train three FCN instances on three datasets and evaluate them on Photo-Normal dataset. Training sets are

1. Train-Normal-All: training samples of normal text,

2. Train-Chaotic-All: training samples of chaotic text,

3. Train-All-All: union of both above.

Sample counts are 108000, 108000 and 216000, respectively. Each of them contains all the 36 font types.

In this experiment we compare our approach with four baselines: the traditional projection based method, the region-merging based method designed for Chinese [13], the connected component based method designed for English [2] and Tesseract [21], an open source OCR engine still in active development[1]. In the following sections they are referred to as PROJ, CN, EN and Tesseract, respectively.

The results are shown in Table 1. Our FCN instances significantly outperform the baselines. Among the first three baselines, PROJ is the simplest model but outperforms CN and EN, because it is not specifically designed for one single language. As a morden OCR engine targeting various languages, Tesseract achieves a decent accuracy but is still outperformed by a large margin. Among the three FCN instances, the one trained on Train-Normal-All achieves the best result, because it has the most similar text content style with Photo-Normal dataset.

All FCN instances achieve over 98% accuracy. Thus we conclude that they generalize well from simulated disturbance to real-world disturbance.

Model	Accuracy
PROJ	83.5
CN [13]	72.7
EN [2]	76.0
Tesseract [21]	90.4
Train-All-All	98.1
Train-Normal-All	**98.6**
Train-Chaotic-All	98.2

Table 1. Segmentation accuracy on Photo-Normal dataset. "Train-XXX-All" refers to which dataset our FCN model is trained on. Our model significantly outperforms baselines. The best result is acquired by training on Train-Normal-All, which has the most similar text content style with Photo-Normal dataset.

4.4.2 Generalization across Text Content Styles

FCN has the advantage to utilize a wide receptive field on input image to predict at a horizontal pixel. This advantage could be a disadvantage, because it introduces the risk that

---

[1]https://github.com/tesseract-ocr/tesseract

Training Set	Evaluation Set	
	Eval-Normal-All	Eval-Chaotic-All
Train-Normal-All	**97.8**	96.0
Train-Chaotic-All	97.3	**97.4**
Train-All-All	97.5	97.2

Table 2. Evaluate generalization ability between normal and chaotic text content styles.

FCN overfits certain style of text content. For example, if character A is always surrounded by B and C in training text content, FCN may fit such pattern. When it comes to testing text content in which A is surrounded by D end E, FCN will probably make a mistake. We want to know how serious this problem is.

In this section, five datasets are used for training and evaluation:

1. Train-Normal-All: training samples of normal text,

2. Train-Chaotic-All: training samples of chaotic text,

3. Train-All-All: union of above,

4. Eval-Normal-All: evaluation samples of normal text,

5. Eval-Chaotic-All: evaluation samples of chaotic text.

Sample counts are 108000, 108000, 216000, 1080 and 1080, respectively.

The results are shown in Table 2. The best performance on Eval-Normal-All and Eval-Chaotic-All are achieved by training on Train-Normal-All and Train-Chaotic-All, respectively. The first two rows show that our model generalizes well from chaotic style to normal style and only slightly worse from normal style to chaotic style.

This experiment suggests that in practice, if the text content style we want to finally work on can be accessed, it is optimal to train on the same content style. If not, training on chaotic text content still works well.

Note that all of the results in Table 2 are worse than results in Table 1 because the our simulated disturbance setup is actually more difficult than real-world disturbance.

4.4.3 Generalization across Font Types

Real-world documents contain various font types. In practice we can include as much font types as possible in training sets to improve generalization. However, some particular fonts of interest may still not be included, which requires the model to generalize across different font types.

To evaluate this generalization ability, three groups of datasets are used in this section:

1. Train-Normal-All: training samples of normal text,

Font	Exclude	All
AdobeFangsongStd-Regular	98.7	99.5
AdobeHeitiStd-Regular	98.6	98.6
AdobeKaitiStd-Regular	98.1	99.1
AdobeSongStd-Light	99.0	99.4
Baoli	97.5	98.1
Hannotate	98.4	98.8
Hanzipen	95.9	96.3
Hiragino-Sans-GB-W3	98.7	98.9
Hiragino-Sans-GB-W6	98.9	98.3
Kaiti	98.6	98.3
Lantinghei	96.9	97.6
Libian	97.9	98.1
SIMLI	98.8	99.3
SIMYOU	99.2	99.3
STCAIYUN	**51.9**	**94.7**
STFANGSO	98.0	98.2
STHUPO	93.2	96.9
STHeiti-Light	98.3	97.7
STKAITI	98.8	99.3
STLITI	**74.0**	**83.8**
STSONG	98.2	98.8
STXIHEI	96.4	96.7
STXINGKA	**72.2**	**91.6**
STXINWEI	98.2	98.6
Songti	98.6	98.8
WawaSC-Regular	95.2	97.1
WeibeiSC-Bold	98.9	98.6
Yuanti	97.2	98.0
YuppySC-Regular	97.6	98.2
msyh	96.8	98.7
msyhbd	97.1	98.5
simfang	99.7	99.9
simhei	98.5	99.1
simkai	98.3	98.4
simsun	98.7	98.9
stzhongs	99.6	99.8

Table 3. Evaluate generalization ability across font styles. For each font XXX in the first column, we train a FCN instance on Train-Normal-exclude-XXX dataset and another on Train-Normal-All dataset. Then we evaluate both on corresponding Eval-Normal-only-XXX dataset. Results are shown in the second and third column, respectively. Bad cases are highlighted.

2. Train-Normal-exclude-XXX: training samples of normal text, containing font types except XXX,

3. Eval-Normal-only-XXX: evaluation samples of normal text, containing only one font type XXX.

Sample counts of each dataset in the three groups are 108000, 105000 and 1080, respectively.

The results are shown in Table 3. For each font type, we train FCN on dataset that does not include this font type and dataset that does, corresponding to the second column and third column. The second column shows that FCN generalizes well on unseen font types for the most cases. The third column shows that including corresponding font type during training further improves accuracy.

Nevertheless, there are several bad cases highlighted in the table: STCAIYUN, STLITI and STXINGKA. Their font styles are illustrated in Figure 6.

(a) STCAIYUN

(b) STLITI

(c) STXINKA

Figure 6. Partial samples of three particular font types on which FCN generalizes badly.

(a) STLITI

(b) STXINKA

Figure 7. Partial segmentation results on STLITI and STX-INGKA. Bounding line segments above and below are predictions and ground truths, respectively. Both FCN instances are trained on Train-Normal-All dataset but still perform illy. Most errors arise from italic English characters.

The first font type, STCAIYUN, is completely different from others because of its hollow structure. However, when it is included in training set, segmentation accuracy
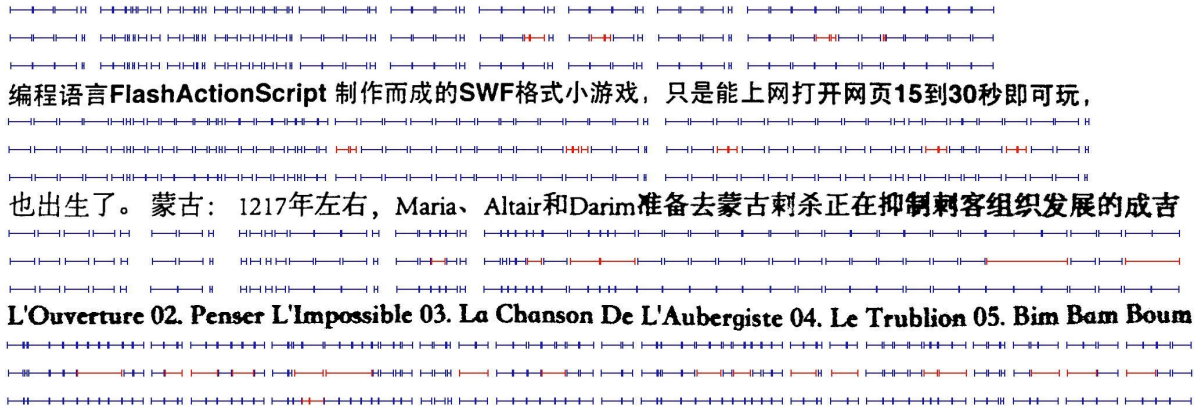
Figure 8. Comparison of projection-based approach and our approach. In each sample there are three rows of bounding line segments, corresponding to ground truths, projection-based method's outputs and our model's outputs, respectively. Correct and wrong bounding line segments are colored as blue and red, respectively. In the last sample FCN falsely splits character "m" apart probably because FCN recognizes it as two touching "n". Nevertheless, our FCN handles disconnected components and touching characters well in most cases.

increases from 51.9% back to 94.7%. In this case, FCN generalizes badly on particular font type but can restore decent accuracy once it is included again.

However, on the second and third font types, accuracy cannot be restored even when they are included in training set. As shown in Figure 7, Most segmentation errors arise from English characters. This is because both STLITI and STXINGKA have italic English character style. To properly segment italic characters, the model should predict oblique lines rather than vertical splitting lines, which is impossible in our FCN architecture.

This experiment shows that our FCN generalizes well on most cases except those with completely different styles or italic styles. The first issue can be fixed by including such special font types in training sets. As for the second issue, we will discuss the possible solution in Section 5.

4.4.4 Handling Difficult Cases

The main challenges of Chinese/English mixed character segmentation are two-fold: first, various character widths inside and across languages and second, the coexistence of disconnected structure and touching characters.

Without the first challenge, we can calculate the widths of each connected components in a text line image then take the mode as unified character width, which is used in traditional OCR techniques. Without the second challenge, we can either tune the threshold in projection-based methods to tackle touching characters, or use a region-merging phase [13] to tackle disconnected structure.

Nevertheless, our FCN architecture handles these difficulties well by automatically utilizing useful features. Typical samples are shown in Figure 8.

5. Conclusion and Future Work

In this paper we tackle Chinese/English character segmentation for printed document images. By reframing it as a two-class semantic segmentation problem, we take advantage of the successful deep neural architecture called fully convolutional networks (FCN) in the field of semantic segmentation. Trained on synthesized samples with simulated random disturbance, FCN can accurately perform binary classification at each horizontal position on text line images to decide whether this position should be a splitting point or not. Our approach significantly outperforms traditional methods on segmentation accuracy. Experiments show that it is able to generalize from simulated disturbance to real-world disturbance, generalize between normal and chaotic text content styles, generalize among various font types and properly handle the coexistence of disconnected structure and touching characters.

The experimental result in Section 4.4.3 shows that our approach performs badly on characters of italic font type because FCN simply predicts vertical splitting lines rather than oblique splitting lines. In addition, there exist even more difficult cases where two characters are so close that they can only be splitted by curved lines. A possible solution and step forward is to reframe character segmentation as an instance segmentation problem. Instance segmentation is also called simultaneous detection and segmentation. In this task, instance-level information and pixel-wise accurate mask for objects are to be estimated [11]. Ideally, with instance segmentation every single characters can be curved out exactly and cleanly. In the future we will work on this possible solution.

References

[1] Y.-K. Chen and J.-F. Wang. Segmentation of single-or multiple-touching handwritten numeral string using background and foreground analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1304–1317, 2000.

[2] A. Choudhary, R. Rishi, and S. Ahlawat. A new approach to detect and extract characters from off-line printed images and text. *Procedia Computer Science*, 17:434–440, 2013.

[3] N. Dave. Segmentation methods for hand written character recognition. *International Journal of signal processing, image processing and pattern rcognition*, 8:154–164, 2015.

[4] U. Garain and B. B. Chaudhuri. Segmentation of touching characters in printed devnagari and bangla scripts using fuzzy multifactorial analysis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 32(4):449–459, 2002.

[5] A. R. Himamunanto and A. R. Widiarti. Javanese character image segmentation of document image of hamong tani. In *Digital Heritage International Congress (DigitalHeritage), 2013*, volume 1, pages 641–644. IEEE, 2013.

[6] W. L. Hwang and F. Chang. Character extraction from documents using wavelet maxima. *Image and Vision Computing*, 16(5):307–315, 1998.

[7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In D. Blei and F. Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 448–456. JMLR Workshop and Conference Proceedings, 2015.

[8] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *arXiv preprint arXiv:1603.08155*, 2016.

[9] A. Kaur, S. Rani, and P. Singh. Segmentation of isolated and touching characters in handwritten gurumukhi word using clustering approach. 2014.

[10] S.-W. Lee, D.-J. Lee, and H.-S. Park. A new methodology for gray-scale character segmentation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):1045–1050, 1996.

[11] S. Liu, X. Qi, J. Shi, H. Zhang, and J. Jia. Multi-scale patch aggregation (mpa) for simultaneous detection and segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[12] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[13] Y. Mei, X. Wang, and J. Wang. A chinese character segmentation algorithm for complicated printed documents. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 6(3):91–100, 2013.

[14] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In J. Frnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814. Omnipress, 2010.

[15] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[16] S. Nomura, K. Yamanaka, O. Katai, H. Kawakami, and T. Shiose. A novel adaptive morphological approach for degraded character image segmentation. *Pattern Recognition*, 38(11):1961–1975, 2005.

[17] P. P. Roy, U. Pal, J. Lladós, and M. Delalandre. Multi-oriented touching text character segmentation in graphical documents using dynamic programming. *Pattern Recognition*, 45(5):1972–1983, 2012.

[18] P. Sahare and S. B. Dhok. Review of text extraction algorithms for scene-text and document images. *IETE Technical Review*, pages 1–21, 2016.

[19] D. Senapati, S. Rout, and M. Nayak. A novel approach to text line and word segmentation on odia printed documents. In *Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference on*, pages 1–6. IEEE, 2012.

[20] E. Simo-Serra, S. Iizuka, K. Sasaki, and H. Ishikawa. Learning to simplify: fully convolutional networks for rough sketch cleanup. *ACM Transactions on Graphics (TOG)*, 35(4):121, 2016.

[21] R. Smith and G. Inc. An overview of the tesseract ocr engine. In *Proc. 9th IEEE Intl. Conf. on Document Analysis and Recognition (ICDAR*, pages 629–633, 2007.

[22] J. Tan, W.-X. Wang, M.-S. Feng, and X.-X. Zuo. A new approach based on ncut clustering algorithm for signature segmentation. *AASRI Procedia*, 1:14–20, 2012.

[23] J. Tse, C. Jones, D. Curtis, and E. Yfantis. An ocr-independent character segmentation using shortest-path in grayscale document images. In *Machine Learning and Applications, 2007. ICMLA 2007. Sixth International Conference on*, pages 142–147. IEEE, 2007.

[24] K. Wang, J. Jin, and Q. Wang. High performance chinese/english mixed ocr with character level language identification. In *2009 10th International Conference on Document Analysis and Recognition*, pages 406–410. IEEE, 2009.

[25] K. Wang, J.-M. Jin, W.-M. Pan, G.-S. Shi, and Q.-R. Wang. Mixed chinese/english document auto-processing based on the periodicity. In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, volume 6, pages 3616–3619 vol.6, Aug 2004.

[26] S. Wshah, Z. Shi, and V. Govindaraju. Segmentation of arabic handwriting based on both contour and skeleton segmentation. In *2009 10th International Conference on Document Analysis and Recognition*, pages 793–797. IEEE, 2009.