Introduction to OCR

Xinyun Zhang
CSE Department, CUHK
xyzhang21@cse.cuhk.edu.hk

Fall 2021

Outline

- Background
- Text Detection
- Text Recognition
- Conclusion

Background

- **What is OCR**?

OCR stands for Optical Character Recognition, which is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text.

- **Application Scenarios**

ID recognition

Bank card recognition

Text recognition

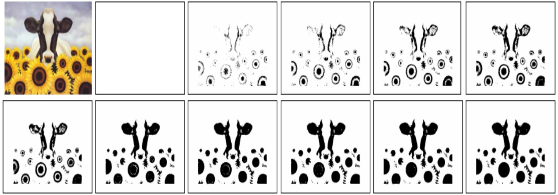
Background

- **The story of OCR**
  - ➢ Traditional algorithms
    - Pipeline

      Text region location → Text rectification → Character segmentation → Character recognition → Post processing

    - Text region location

      Maximally Stable Extremal Regions (MSER)

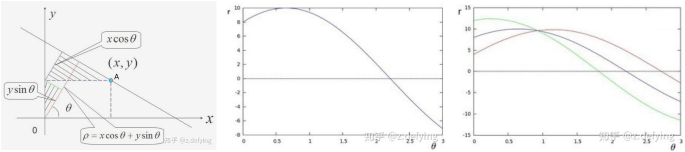


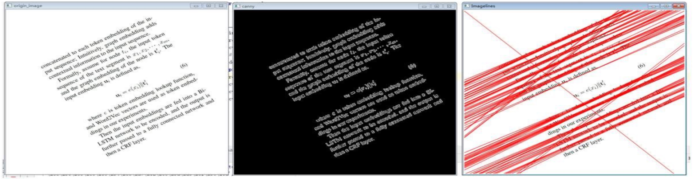
- Apply a series of thresholds to binarize the image
- Extract connected components
- Find a threshold when an extremal region is Maximally Stable, i.e., local minimum of the relative growth of its area
- Approximate a region with a bounding box (ellipse or rectangle)
- Non-maximum suppressing

Background

- **The story of OCR**
  - ➢ Traditional algorithms
    - Text rectification

Hough Line detection + rotation





Background

- **The story of OCR**
  - Traditional algorithms
    - Character segmentation

Connected Component Labeling： find connected regions then split

Vertical Histogram Projection

- Calculate the number of white pixels in each column
- Draw the vertical projection map
- Split the characters based on the values

Background

- **The story of OCR**
  - ➢ Traditional algorithms
    - Character recognition

      Handcrafted features + machine learning algorithms

      - Possible features: HOG, SIFT, LBP, …
      - Machine learning algorithms: SVM, Decision Tree, Adaboost, …

    - Post processing

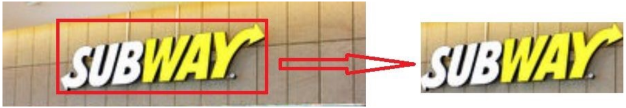
      Design some rules based on the application scenario to refine the results.

Traditional algorithms require complicated pipelines to process the images, and they highly rely on the handcrafted features for different scenarios.

Background

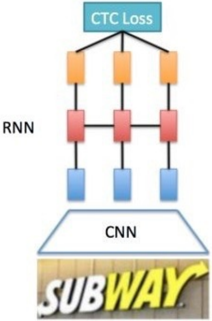
- **The story of OCR**
  - ➢ The deep learning era

text detection: extract the part of image that contains the text



- Region-proposal based methods
- Segmentation-based methods

text recognition: convert the text image into text



Background

- **The story of OCR**
  - ➢ Traditional algorithms vs. deep learning algorithms
    - Both consist of text detection part and text recognition part
    - Bottom-up perspective vs. top-down perspective
    - Deep learning frees us from designing handcrafted features and has reshaped compute vision.
    - Methods based on deep learning also borrows ideas from traditional algorithms.

Text Detection

- **Semantic Segmentation**

The task of assigning a semantic label, such as “road”, “cars”, “person”, to **every pixel** in an image.

blue pixels: cars

red pixels: people

purple pixels: road

Text detection: a semantic segmentation task with labels “**text**” and “**background**”, plus a bounding box to select the text pixels.

Text Detection

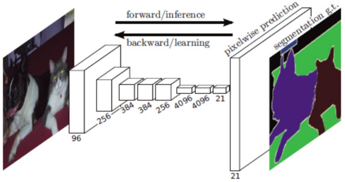
- **Fully Convolutional Network (FCN)**
  - ➢ Main idea: convolution + upsampling + dense prediction

image classification

replace the FC layer with 1*1 conv layer

without resize operation

add upsampling operation

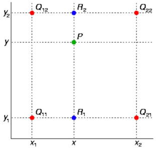


Text Detection

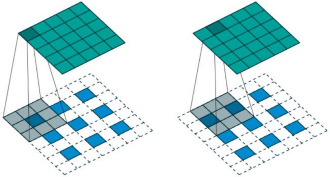
- **Fully Convolutional Network (FCN)**

  - Upsampling:

    Interpolation: bilinear interpolation, cubic interpolation, ...



Transposed convolution:



input size: (3, 3)

output size: (5, 5)

- Add paddings to the input feature map, then the feature map size becomes (7, 7)
- Use a conv layer (3*3, stride 1) to get the output

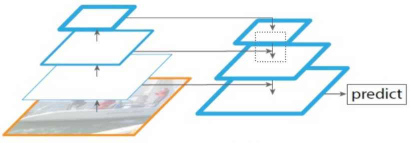
Text Detection

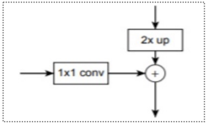
- **Feature Pyramid Network (FPN)**

  ➢ Motivation

  1. Feature maps with different resolution for objects with different sizes
  2. Different feature maps contain different information (spatial information vs. semantic information)

  ➢ Main idea: merge features of different scales

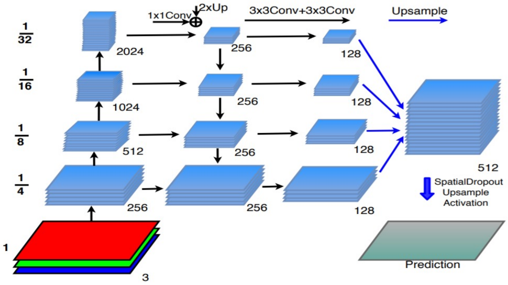




Text Detection

- **Text Detection Model**

Feature extractor (backbone+FPN) -> upsampling -> dense prediction(text/background) -> bounding box



(H, W, 3) $\xrightarrow{\text{feature extractor}}$ (H/4, W/4, 512) $\xrightarrow{\text{upsampling}}$ (H, W, 512) $\xrightarrow{\text{1*1 conv}}$ (H, W, 2)

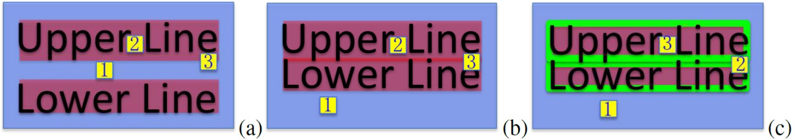
- text
- background

Text Detection

- **Improved Text Detection Model**

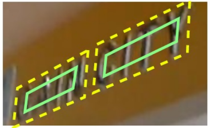
  - Motivation

When two text instances are too close, it is hard to separate them.



In addition to "text" and "background", we add the third class "border" to separate the crowded text instances.

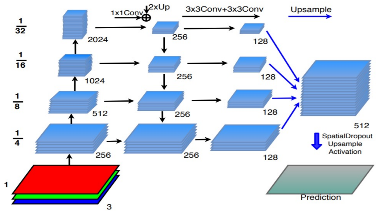
Shrink the text region to generate the border label.



Text Detection

- **Improved Text Detection Model**

Feature extractor (backbone+FPN) -> upsampling -> dense prediction(text/**border**/background) -> bounding box

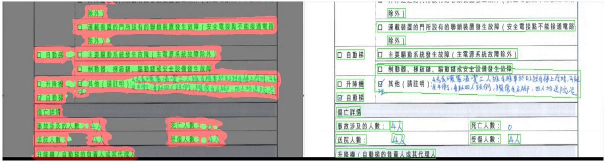


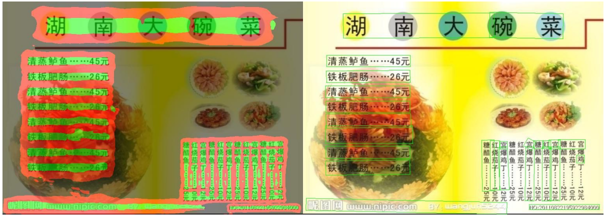
(H, W, 3) $\xrightarrow{\text{feature extractor}}$ (H/4, W/4, 512) $\xrightarrow{\text{upsampling}}$ (H, W, 512) $\xrightarrow{\text{1*1 conv}}$ (H, W, 3)

- text
- border
- background

Text Detection

- **Improved Text Detection Model**
  - ➢ Sample results



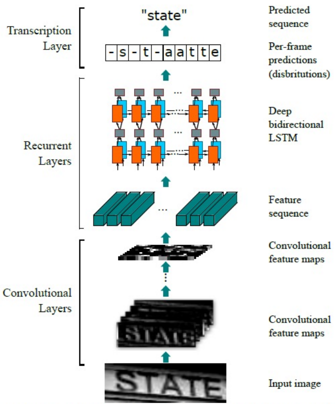


Text Recognition

- **Convolutional Recurrent Neural Network**

  ➢ Main idea

An alphabet contains all the possible characters. For Chinese, the length of the alphabet is approximately 6000.



output (“state”)

↑ transcription layer

alignment/per-frame predictions (1, L, 6000)

↑ recurrent layers

convolutional feature maps (1, L, 512)

↑ convolutional layers

resized input image (32, W, 3)

↑ resize to fixed height

input image (any size)

Text Recognition

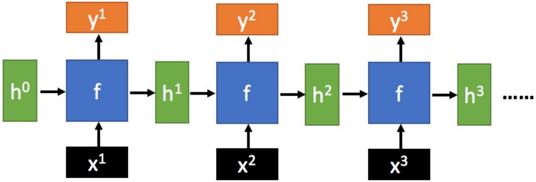
- **Convolutional Recurrent Neural Network**

  ➢ Recurrent Layers

Recurrent neural networks (RNN) are used to encode the sequence information.

- Given function f: $h', y = f(h, x)$

h and h' are vectors with the same dimension

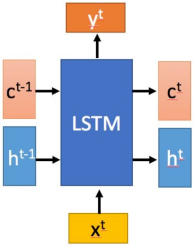


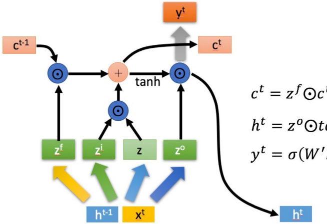
No matter how long the input/output sequence is, we only need one function f

Text Recognition

- **Convolutional Recurrent Neural Network**
  - ➢ Recurrent Layers

    Long short-term memory (LSTM)





$$c^t = z^f \odot c^{t-1} + z^i \odot z$$
$$h^t = z^o \odot tanh(c^t)$$
$$y^t = \sigma(W' h^t)$$

Text Recognition

- **Convolutional Recurrent Neural Network**

  ➢ Transcription layers - CTC

  The alignment problem

  - Approach 1 – merge the repeated characters

x_1	x_2	x_3	x_4	x_5	x_6	input (X)
c	c	a	a	a	t	alignment
c		a			t	output (Y)

What if the alignment is [h, h, e, l, l, l, l, l, o] ?

  - Approach 2 – introduce the blank token (CTC)

h	h	e	ϵ	ϵ	l	l	l	ϵ	l	l	o

First, merge repeat characters.

h	e	ϵ	l	ϵ	l	o

Then, remove any ϵ tokens.

h	e	l	l	o

The remaining characters are the output.

h	e	l	l	o

Text Recognition

- **Convolutional Recurrent Neural Network**

  ➢ Transcription layers - CTC

  loss function

  Suppose the alignment sequence is $X=[x_1, x_2, ..., x_L]$, the target text (label) is $Y = [y_1, y_2, ..., y_U]$, the learning target is to maximize $P(Y|X, \Theta)$.

  e.g.
  Y=[c, a, t]
  Possible alignments: [c, c, ε, a, a, t], [c, ε, a, a, t, t], [c, ε, a, a, ε, t], ...

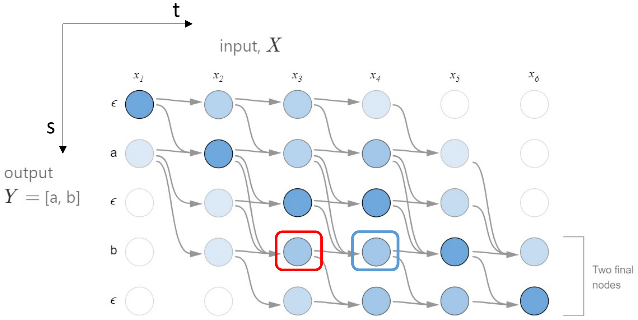
  To calculate $P(Y| X, \Theta)$:
  Intuitive solution – brute force
  Time complexity: O(M^T), M is the length of the alphabet and T is the length of the input sequence.

Text Recognition

- **Convolutional Recurrent Neural Network**

➢ Transcription layers - CTC

Dynamic Programming



Node (s, t) in the diagram represents $\alpha_{s,t}$ – the CTC score of the subsequence $Z_{1:s}$ after t input steps.

e.g.
the probability that the alignment [x_1, x_2, x_3] can be converted to sequence "ab" and x_3 is b

- Case 1: z_s is not ε, and z_{s-2} != z_s

$$\alpha_{s,t} = (\alpha_{s-1,t-1} + \alpha_{s,t-1} + \alpha_{s-2,t-1}) P_t(z_s|X)$$

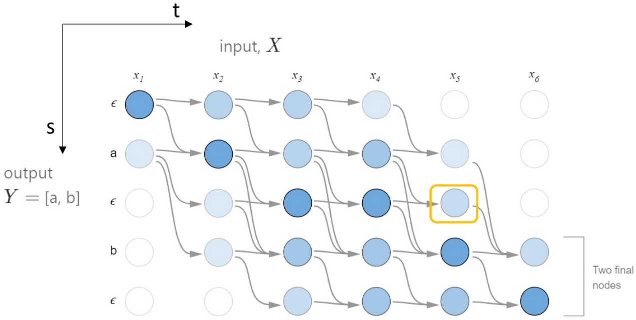
e.g.
If the alignment [x_1, x_2, x_3, x_4] is able to be converted to sequence "ab" and x_4 is "b" , it must come from one of the three cases:

1. [x_1, x_2, x_3] -> "a", x_3="a"
2. [x_1, x_2, x_3] -> "a", x_3="ε"
3. [x_1, x_2, x_3] -> "ab", x_3="b"

Text Recognition

- **Convolutional Recurrent Neural Network**
  - ➢ Transcription layers - CTC

Dynamic Programming



Node (s, t) in the diagram represents $\alpha_{s,t}$ – the CTC score of the subsequence $Z_{1:s}$ after t input steps.

- Case 2: other cases

$$\alpha_{s,t} = (\alpha_{s-1,t-1} + \alpha_{s,t-1}) P_t(z_s|X)$$

e.g.

If the alignment [x_1, x_2, x_3, x_4, x_5] is able to be converted to sequence “a” and x_5 is “ε” , it must be one of the two cases:

1. [x_1, x_2, x_3, x_4] -> “a”, x_4=“ε”
2. [x_1, x_2, x_3, x_4] -> “a”, x_4=“a”

⇒ time complexity: O(ST)

Loss function:

$$\Sigma_{(X,Y)\in D} - log\big(P(Y|X)\big)$$

Text Recognition

- **Convolutional Recurrent Neural Network**

  - Transcription layers - CTC

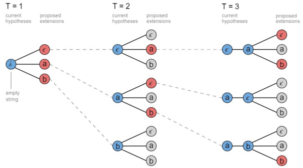
    Inference

- Greedy search

For each t, choose the character with the highest probability.

Problem: many-to-one mapping
e.g.
Alignment 1: [a, b, b, c], P = 0.5
Alignment 2: [b, a, a, c], P = 0.3
Alignment 3: [b, b, a, c], P = 0.3
P(Y = [a, b, c]) = 0.5, P(Y=[b, a, c]) = 0.6

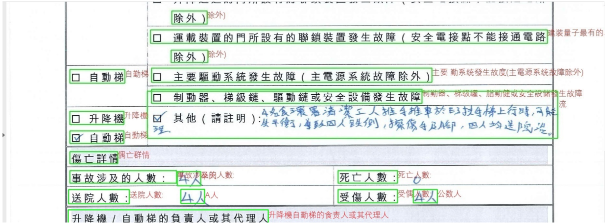
- Beam search



A standard beam search algorithm with an alphabet of $\{\epsilon, a, b\}$ and a beam size of three.

Text Recognition

- **Convolutional Recurrent Neural Network**
  - ➢ Sample results



Take-home message

- OCR is one of the best scenario for the application of computer vision technology .
- Segmentation-based models are effective to detect text. Adding border benefits detecting crowded text instances.
- Incorporating recurrent layers can encode the sequence information to help recognize the text in the images.
- CTC algorithm can be adopted to align the predictions and ground truth.
- Problems to solve: hand-written text recognition, curved text detection, …

Thanks