Convolutional Neural Networks for Page Segmentation of Historical Document Images (2017)

Kai Chen and Mathias Seuret
Resume

December 11, 2018

Abstract
Using a simple CNN architecture with one convolutionnal layer, they show the this simple architecture achieves competitive results agains other deep architecture on different public datasets.

1 Introduction

Page segmentation is an important prerequisite step of document image analysis and understanding. The goal is to split a document image into regions of interest. Compared to segmentation of machine printed document images, page seg- mentation of historical document images is more challenging due to many variations such as layout structure, decoration, writing style, and degradation.

In this work, the authors focus on developing an end-to-end method. Combining feature learning and classifier training into one step. Image patches are used as input to train a CNN for the labeling task. During training, the features used to predict labels of the image patches are learned on the convolution layers of the CNN. Instead of using deep architectures, they propose a simple CNN of one convolutionnal layer.

2 Proposed method

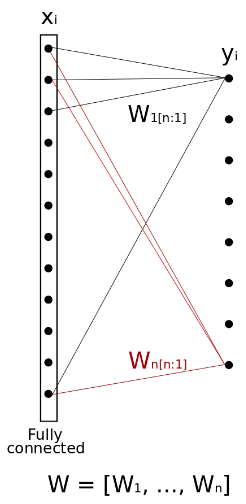
The main idea is to learn a set of feature detectors and train a nonlinear classifier on the features extracted bu the feature detectors.

Preprocessing In order to speed up the pixel labeling process, for a given document image, we first applying a superpixel algorithm to generate superpixels. A superpixel is an image patch which contains the pixels belong to the same object. Then instead of labeling all the pixels, we only label the center pixel of each superpixel and the rest pixels in that superpixel are assigned to the same label.

SLIC algorithm is applied as a preprocessing step to generate superpixels for a given document images.

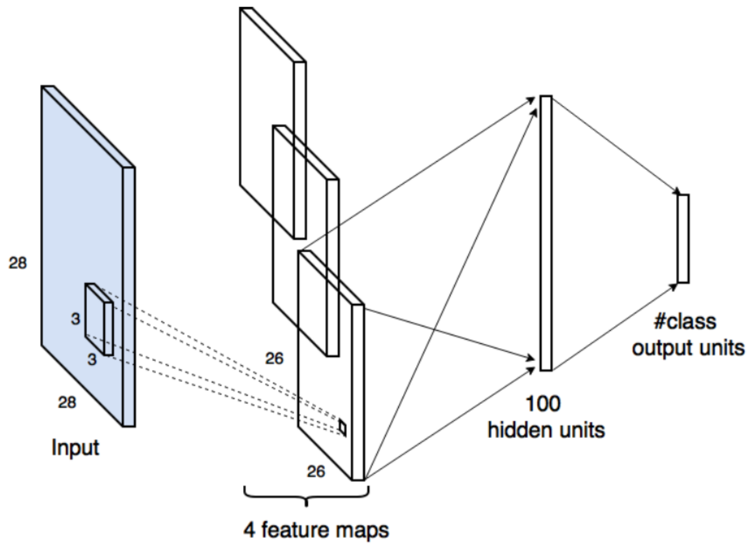
CNN Architecture The input is a grayscale image patch. The size of the image patch is 2828 pixels. Our CNN architecture contains only one convolution layer which consists of 4 kernels. The size of each kernel is 3 3 pixels. No pooling layer is used in our architecture. Then one fully connected layer of 100 neurons follows the convolution layer. The last layer consists of a logistic regression with softmax which outputs the probability of each class, such that

$$P(y = i|x, W_1, \cdots, W_M, b_1, \cdots, b_M) = \frac{e^{W_i x + b_i}}{\sum_{j=1}^{M} e^{W_j x + b_j}}$$



where x is the output of the fully connected layer, Wi and bi are the weights and biases of the i-th neuron in this layer, and M is the number of the classes. The predicted class $\hat{y}$ is the class which has the max probability, such that $\hat{y}$

$$\hat{y} = \arg\max_i P(y = i | x, W_1, \cdots, W_M, b_1, \cdots, b_M)$$



3 Metrics

the metrics used in the experiments are: pixel accuracy, mean pixel accuracy, mean IU, and frequency weighted IU (f.w. IU).

In order to obtained the metrics, we define the variables:

- n_c the number of classes
- n_{ij} the number of pixels of class i predicted to belong class j. For class i:
  - n_{ii} the number of correctly classified pixels (true positives).
  - n_{ij} the number of wrongly classified pixels (false positives).
  - n_{ji} the number of wrongly not classified pixels (false negatives).

- t_i the total number of pixels in class i, such that: $t_i = \sum_j n_{ji}$

With the defined varibles above we can compute:

- pixel accuracy: $acc = \frac{\sum_i n_{ii}}{\sum_i t_i}$
- mean accuracy: $acc_{mean} = \frac{1}{n_c} \times \sum_i \frac{n_{ii}}{t_i}$
- mean IU: $iu_{mean} = \frac{1}{n_c} \times \sum_i \frac{n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}}$
- f.w. IU: $iu_{\text{weighted}} = \frac{1}{\sum_k t_k} \times \sum_i \frac{t_i \times n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}}$

4 Training and results

To train the CNN, for each superpixel, we generate a patch which is centred on that superpixel. The patch is considered as the input of the network. The size of each patch is 2828 pixels. The label of each patch is its center pixel's label. The patches of the training images are used to train the network. In the CNN, the cost function is defined as the cross-entropy loss, such that:

$$\mathcal{L}(X, Y) = -\frac{1}{n}\sum_{i=1}^{n}\left(\ln a\left(x^{(i)}\right) + \left(1 - y^{(i)}\right)\ln\left(1 - a\left(x^{(i)}\right)\right)\right)$$

where $X = \left\{x^{(1)}, \cdots, x^{(n)}\right\}$ is the training image patches and $Y = \left\{y^{(1)}, \cdots, y^{(n)}\right\}$ is the corresponding set of labels.

The CNN is trained with Stochastic Gradient Descent (SGD) with the dropout technique. The goal of dropout is to avoid overfitting by introducing random noise to training samples. Such that during the training, the outputs of the neurons are masked out with the probability of 0.5.

Table II: Performance (in percentage) of superpixel labeling with only local MLP, CRF, and the proposed CNN.

	G. Washington				*Parzival*				*St.Gall*			
	pixel acc.	mean acc.	mean IU	f.w. IU	pixel acc.	mean acc.	mean IU	f.w. IU	pixel acc.	mean acc.	mean IU	f.w. IU
Local MLP [17]	87	89	75	83	91	64	58	86	95	89	84	92
CRF [18]	**91**	90	76	85	93	70	63	88	97	88	84	94
CNN	**91**	**91**	**77**	**86**	**94**	**75**	**68**	**89**	**98**	**90**	**87**	**96**
CNN (max pooling)	**91**	90	**77**	**86**	**94**	**75**	**68**	**89**	**98**	**90**	**87**	**96**
	CB55				*CSG18*				*CSG863*			
	pixel acc.	mean acc.	mean IU	f.w. IU	pixel acc.	mean acc.	mean IU	f.w. IU	pixel acc.	mean acc.	mean IU	f.w. IU
Local MLP [17]	83	53	42	72	83	49	39	73	84	54	42	74
CRF [18]	84	53	42	75	86	47	37	77	86	51	42	78
CNN	**86**	59	47	**77**	**87**	**53**	41	79	**87**	**58**	**45**	**79**
CNN (max pooling)	**86**	**60**	**48**	**77**	**87**	**53**	**42**	**80**	**87**	57	**45**	**79**