FrameNet: Tabular Image Preprocessing Based on UNet and Adaptive Correction

Yufei Wang[1,2], Chen Du[1,2], and Baihua Xiao[1]

[1] The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[2] University of Chinese Academy of Sciences, Beijing, China
{wangyufei2020, duchen2016, baihua.xiao}@ia.ac.cn

Abstract. Detecting and recognizing objects in images with complex backgrounds and deformations is a challenging task. In this work, we propose FrameNet, while a deep table lines segmentation network based on our Res18UNet with an adaptive deformation correction algorithm for correcting the table lines. We use Itinerary/Receipt of E-ticket for Air Transport to evaluate our methods. The experiment results show that our Res18UNet can reduce the number of parameters and improve the speed of image segmentation without significantly reducing the segmentation accuracy, and our correction method can better correct the perspective deformation and some distorted tablular images with no dependence on pixel-level ground truth image. In addition, we also apply our model and method to VAT invoice dataset and prove that they also have better transfer ability.

Keywords: Computer vision · Deep learning · Image rectification.

1 Introduction

The complex background and the deformation of the object of the image is usually the difficulties in detection and recognition tasks. In the real world, such problems exist on scanning QR code, extracting information from the picture of invoices. Once a good preprocessing method can be designed, the subsequent steps such as detection and recognition will be more efficient.

Invoices usually have texture, shade and red seal, and may be distorted since the improper operation of photographers(shown in Fig. 1), which leads to difficulties to obtain the information on this kind of invoice image effectively. However, most invoices are structured, the horizontal and vertical lines of the invoices will divide the information into specific areas. Therefore, the performance of detection and correction of table blocks becomes a significant step in preprocessing. Traditional line detection methods are usually based on edge detection and Hough transform[1][2], but it is not robust due to the interference from complex background and deformation of the image. In the recent years, researchers tackled this problem using deep neural networks, which resulted in huge performance leaps in computer vision tasks[3] [4] [5] and was extended to other

Fig. 1. Examples of Itinerary/Receipt of E-ticket for Air Transport, it can be seen that since the invoices were bound into a book, the image has some deformation.

fields[6][7]. Thus, detecting table lines using deep neural network from complex background and deformable image become realizable.

In this work, we propose FrameNet, which contains a deep neural network for table line segmentation and an adaptive deformation correction for table line rectification, can be used as a preprocessing step for distorted tablular image. The contributions of this work are two-fold.

1. We propose a table line segmentation network based on Res18UNet to predict the table line on pixel level. The network can effectively reduce the amount of parameters and computation without a significant drop of localization accuracy.
2. An adaptive deformation correction method is proposed to correct the table line without pixel-level corrected ground truth images. Results show that our method can better deal with the obstacles of perspective deformation and slight distortion in images and is suitable for tablular image than deep neural network method DewarpNet[8].

2 Methods

2.1 Res18UNet for Table Line Segmentation

UNet[9] is an end-to-end semantic segmentation network that consists of a contracting path and an expansive path. The contracting path extracts feature and downsample the feature map, while the expansive path upsamples the feature map and combines it to the origin feature map from contracting path at the same stage.

However, UNet has a huge number of parameters and mass computation. To this end, following ResNet[5], we propose Res18UNet where use ResNet18 as the downsampling backbone of UNet(shown in Fig. 2) for our table lines segmentation task. Unlike ResNet, we remove max pooling layer in our Res18UNet, and build a convolutional block by stacking two BasicBlocks. Our contracting path consists of four convolutional blocks, the output of each convolutional block is concatenated with the output of the upsampling layer at the same stage. Then

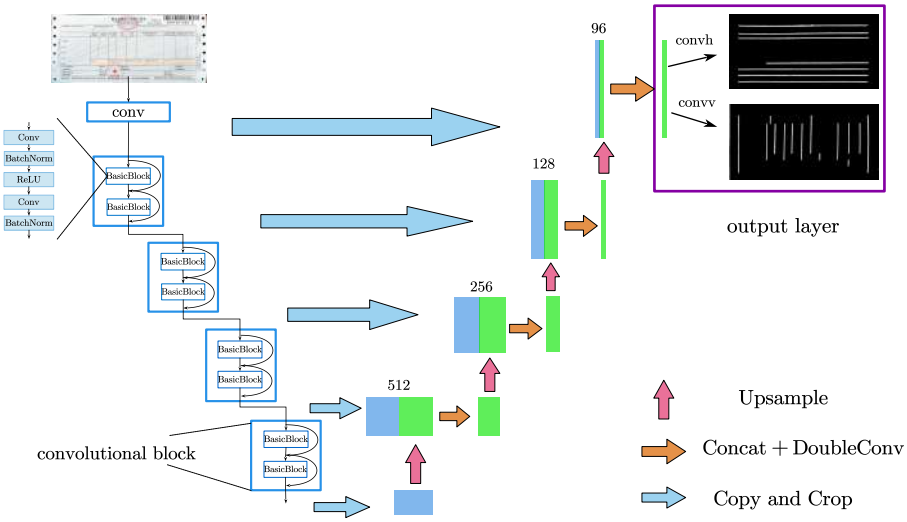

Fig. 2. The architecture of Res18UNet.

the image scale will be expanded through a double convolutional layers and an upsampling layer. In the output layer, we build two convolutional layers followed by a Sigmoid function to predict the probability of horizontal and vertical lines for each pixel respectively.

We use binary cross-entropy loss function(Equation 1) to calculate the horizontal and vertical loss for each pixel respectively, and add them together to obtain the total loss(Equation 2).

$$L\left(x_i, y_i\right) = -\left(y_i \cdot \log x_i + \left(1 - y_i\right) \cdot \log\left(1 - x_i\right)\right) \tag{1}$$

$$L = \sum_i L_h\left(x_i, y_i\right) + \sum_i L_v\left(x_i, y_i\right) \tag{2}$$

where x_i is the output probability of horizontal line or vertical line of one pixel, and y_i is the ground-truth value of the pixel.

2.2 Adaptive Correction based on TPS

After obtaining the prediction of the probability of horizontal and vertical lines for each pixel, we use binarization to get binary images where white pixels represent table lines and black pixels represent the background(shown in the upper right part of Fig. 2). Then we apply *bitwise-and* operation to obtain the intersection points so that we can find the moment for each intersection contour by Equation 3:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y) \tag{3}$$

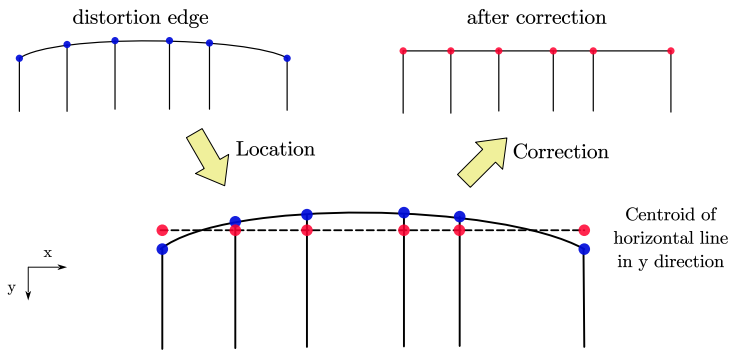

Fig. 3. Main steps of our adaptive correction method, we only show how to correct the horizontal line, correcting the vertical line is similar to this.

where $I(x, y)$ denotes to the value of the pixel at (x, y), and M_{00} is the area (for binary image) or sum of grey level. Therefore, we can calculate the centroid $(\bar{x}, \bar{y})$ for each intersection contour using the following equation:

$$\{\bar{x}, \bar{y}\} = \left\{ \frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right\} \tag{4}$$

We use centroid above to represent the corresponding intersection contour, and the subsequent processes are Adaptive Localization and Correction.

1. **Adaptive Localization**: For the centroid $(\bar{x}_i, \bar{y}_i)$ of the i-th contour, first traverse all horizontal lines and vertical lines to find which one it belongs to(we assume h for the horizontal line and v for the vertical line), then use Equation 4 to calculate the centroid in the x direction of v (called $\bar{x}_i^{(v)}$) and the centroid in the y direction of h (called $\bar{y}_i^{(h)}$), and $(\bar{x}_i^{(v)}, \bar{y}_i^{(h)})$ is the corrected coordinates for $(\bar{x}_i, \bar{y}_i)$.
2. **Correction**: Use thin plane spline(TPS[10]) to transfer image from $(\bar{x}_i, \bar{y}_i)$ to $(\bar{x}_i^{(v)}, \bar{y}_i^{(h)})$.

Our adaptive correction method is illustrated in Fig. 3 and Algorithm 1.

3 Experiment

3.1 Dataset

We use Itinerary/Receipt of E-ticket for Air Transport dataset in our experiment. In order to show the correction effect, we also made some synthetic data by distorting and deforming the invoice template image. The training set contains 51 images, 47 of which are real invoice images and 4 of which are synthetic invoice images. Test sets are divided into real test set and synthetic test set, where real test set contains 46 real images and synthetic test set contains 13 synthetic images.

Algorithm 1: Adaptive Correction based on TPS.

Input: I denotes the original image, I_h, I_v denotes the prediction feature map of horizontal and vertical lines respectively.

I_p = `bitwise_and`(I_h, I_v) ; `// Extract intersection points.`

C = `findContours`(I_p) ; `// Find set of points for each contour.`

`// Use Eq4 to get the centroid of each contour.`

for $i = 1$ **to** `length`(C) **do** $(\bar{x}_i, \bar{y}_i)$=`moment`$(C[i])$;

for *each* $(\bar{x}_i, \bar{y}_i)$ **do** `// Adaptive Localization.`

  `// Find the contour mask of horizontal and vertical line where` $(\bar{x}_i, \bar{y}_i)$ `belongs to and calculate its centroid` (h, v).

  h=`findHContour`$((\bar{x}_i, \bar{y}_i), I_h)$;

  v=`findVContour`$((\bar{x}_i, \bar{y}_i), I_v)$;

  $(\_, \bar{y}_i^{(h)})$ = `moment`(h);

  $(\bar{x}_i^{(v)}, \_)$ = `moment`(v);

end

I' = `tps`$(I, (\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}), (\bar{\boldsymbol{x}}^v, \bar{\boldsymbol{y}}^h))$; `// Use TPS to correct the image.`

return I' ;

Table 1. Comparison of FLOPS(Floating-point operations per second) and parameters of different networks. Res18UNet has far fewer parameters and computations than the other two networks.

Network	GFLOPS	Parameters($\times 10^6$)
UNet	1371.8	17.27
MobileUNet	1056.92	10.04
Res18UNet	655.4	7.25

3.2 Experiment Settings

Applying data augmentation is essential to help network learn the desired invariance and robustness properties when few training samples are available. In our experiment, we randomly flipped horizontally and vertically with a probability of 50%, and randomly added motion blur. At last, we resized the training images uniformly to 256×512.

We trained by RMSprop with a minibatch size of 5 images, learning rate of 0.0001 with a decay of 1/2 at 50 and 100 epoch, weight_decay of 0.5 and momentun of 0.9. Our model was trained and tested on a single NVIDIA GTX 1080Ti. We trained for 1000 epochs and saved the checkpoint for testing.

3.3 Results

Backbone We compared our Res18UNet to UNet and MobileUNet (use MobileNetv1[11] as the backbone of the subsampling section in UNet). The comparison of parameters and computation operations of different backbone is shown in Table. 1.

Table 2. Comparison of segmentation accuracy of different networks, where f.w. IU denotes the frequency weighted intersection over union.

Network	Pixel acc.	Mean acc.	Mean IU	f.w. IU
UNet	97.1	97.4	81.8	95.1
MobileUNet	96.4	96.8	81.0	94.5
Res18UNet	96.8	97.2	81.0	94.8

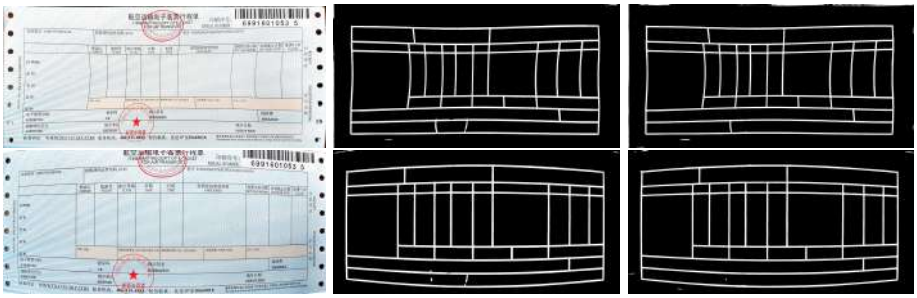

Fig. 4. Comparison of UNet and Res18UNet on table line segmentation task. The first column shows the original invoice image, the second and third columns show the segmentation result from UNet and Res18UNet respectively. We can find from midlle column that UNet may predict the red seal at the bottom of invoice to be lines.

Table 3. Comparison of different loss function.

Loss	Pixel acc.	Mean acc.	Mean IU	f.w. IU
dice loss	95.6	97.1	79.4	93.4
bce loss	96.8	97.2	81.0	94.8

We use four metrics proposed in FCN[12] for our table lines segmentation: pixel accuracy, mean accuracy, mean IU and frequency weighted IU. The comparison of segmentation accuracy is shown in Table. 2.

Combined with the above results, we note that in the case of little difference in accuracy, the number of parameters and computation of UNet is at least twice than that of Res18UNet, which make it better to distinguish the edge area. However, since the lack of training data and residual module, UNet tends to overfit and predict the edge of red seal at the bottom of invoice to be lines in some images (shown in Fig. 4). While compared with MobileUNet, our Res18UNet is superior in computation, storage and accuracy.

Loss function We also compared binary cross-entropy(bce) loss with another loss function, dice loss. Results in Table. 3 show that bce loss is more suitable for the task.

Correction After performing the adaptive correction steps, we send the corrected image into Res18UNet again to detect table lines, then calculate the

Table 4. Change in variance before and after correction in test set.

Test set		σ_y	σ_x
Real images	Before correction	77.06	73.31
	After correction	44.99	55.16
Synthetic images	Before correction	62.26	69.47
	After correction	42.77	62.8

variance in the y direction of the obtained horizontal lines as well as the variance in the x direction of the obtained vertical lines, and finally compare them with lines variance before correction.

The details of calculating variance for horizontal lines are as follows: Once we obtain the horizontal segmentation map(the horizontal output of Res18UNet), we first find the contour for all horizontal line segmentation results, then for each contour, we calculate the variance of all pixels in the y direction using Equation 5, finally we sum up variance(Equation 6) from different contours so that we obtain the variance of horizontal lines(σ_y).

$$\sigma_y^{(i)} = \frac{1}{N_i} \sum_{j=1}^{N_i} \left(y_j - \mathrm{E}\left[\boldsymbol{y}^{(i)} \right] \right)^2, (x_j, y_j) \in \Omega_i \tag{5}$$

$$\sigma_y = \sum_{i=1}^{N} \sigma_y^{(i)} \tag{6}$$

where Ω_i denotes to the i-th contour(N contours in total), $\boldsymbol{y}^{(i)}$ denotes to the y coordinate values of all points in Ω_i, N_i and $\sigma_y^{(i)}$ denotes to the number of pixels and the variance of the y coordinate in Ω_i respectively. Analogous to this, we can also calculate the variance of vertical lines (σ_x).

We test our correction algorithm on both real dataset and synthetic dataset, and use variance to evaluate our algorithm. On the test set of real images, the variance is reduced by 41% and 30% in the y and x direction respectively, where on the test set of synthetic images, the variance is reduced by 31% and 10% in the y and x direction respectively. The reason why the variance is smaller on the synthetic test set than on the real test set is that the perspective deformation on synthetic images are less obvious than that of real images. Although the edges of the image are affected during the correction process and will result in the information lost of some pixel(shown as black edge area in the middle column of Fig. 5 and 6), the table line areas can be well corrected and the image looks flat. So the result shows that our correction method can handle the the perspective deformation and slight distortion well, and the performance is better reflected in the synthetic test set than real test set. The variance of real dataset and synthetic dataset are shown in Table 4, while the performance of correcting on real and synthetic dataset is shown in Fig. 5 and 6.

We also compared our correction method with a famous document unwarping network, DewarpNet[8]. We use the pre-trained proposed in [8] on our test set.

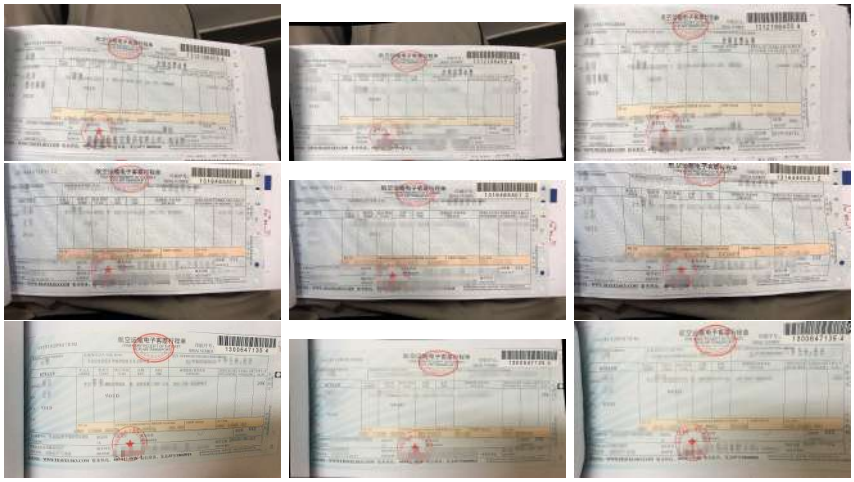

Fig. 5. Comparison of different correction methods for real test images. First column: original image. Second column: image processed by our FrameNet. Third column: image processed by DewarpNet[8].

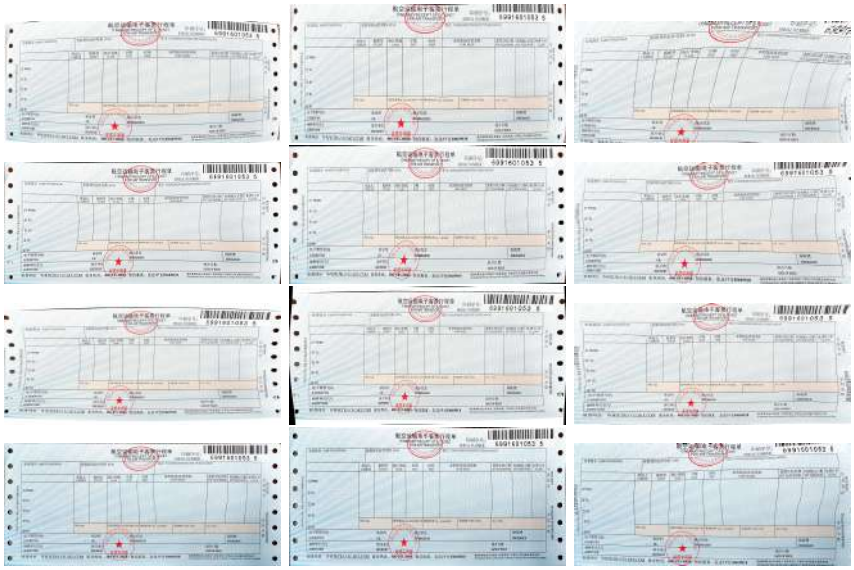

Fig. 6. Comparison of different correction methods for synthetic test images. First column: original image. Second column: image processed by our FrameNet. Third column: image processed by DewarpNet[8].

Results show that DewarpNet may fail to solve the correction for table line images while our method can better deal with it(shown in Fig 5 and 6).

Fig. 7. Results of FrameNet applied to VAT invoices. The left column shows the origin VAT image while the right column shows the corrected results.

3.4 Extended experiment

We extended our FrameNet to VAT invoices. Based on the previous training set, we added 10 labeled VAT samples for training, and tested on our VAT image test set. As seen in Fig. 7, our Res18UNet model and correction method show relatively good results, which implies that our method has the potential to detect and correct different types of tabular images.

4 Conclusion

In this work, we propose FrameNet, a preprocessing method for tablular image. We first build a table line segmentation network, Res18UNet, then an adaptive correction algorithm is proposed to overcome the deformation interference problem. Experiment on Itinerary/Receipt of E-ticket for Air Transport first shows that our segmentation network can effectively reduce the number of parameters and have less computation without highly affecting the segmentation accuracy, then our correction method can correct the perspective deformation and slight distortion case. Extend experiment implies that our method can be extended to other tabular images. Therefore, our model could be helpful to perform downstream tasks such as detecting and recognizing words for deformable tabular image.

Acknowledgements: This work was supported by the National Natural Science Foundation of China under Grant 71621002, Grant 62071469 and Grant 62073326.

Reference

1. F. Zheng, S. Luo, K. Song, C.-W. Yan, and M.-C. Wang, "Improved lane line detection algorithm based on Hough transform," *Pattern Recognition and Image Analysis*, vol. 28, no. 2, pp. 254–260, 2018.
2. J. Tong, H. Shi, C. Wu, H. Jiang, and T. Yang, "Skewness correction and quality evaluation of plug seedling images based on Canny operator and Hough transform," *Computers and Electronics in Agriculture*, vol. 155, pp. 461–472, 2018.
3. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.
4. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.
5. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
6. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, \. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

7. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
8. S. Das, K. Ma, Z. Shu, D. Samaras, and R. Shilkrot, "DewarpNet: Single-image document unwarping with stacked 3D and 2D regression networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 131–140.
9. O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.
10. F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 11, no. 6, pp. 567–585, 1989.
11. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
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*, 2015, pp. 3431–3440.