

Towards Semi-Supervised Text Recognition in Natural Scene

Student PUID: 29096797

Abstract

Detecting and recognizing multi-line texts in natural scene is an important and very useful task in computer vision and many other areas. For example, it allows the robot to read not only books but also texts with complex background images, like the street name. In the report, I will introduce three published papers proceeding towards semi-supervised text recognition in natural scene. The first paper introduced the spatial transformer networks, a meaningful module that provides the capability of leaning invariance to spatial transformations like rotation and scale. The second paper proposed a semantic segmentation via multi-task network cascades, which allows neural networks to localize different scenarios. The third paper utilized above two methods, together with previous text recognition networks, then gave the method to detect and recognize texts in natural scene.

Paper 1: Spatial Transformer Networks

The first paper (Jaderberg et al. 2015) is about spatial transformer networks. The author proposed spatial transformer as an independent layer that could be inserted to any existing neural networks. The key advantage of spatial transformer is to disentangle object pose and part deformation from texture and shape. Even though this objective could be partially fulfilled by local max-pooling layers in Convolution Neural Network (CNN), due to the typically small spatial support for max-polling (e.g. 2×2 pixels), learning invariance to large transformations of the input data is hard to realized unless over a deep hierarchy of max-pooling and convolutions.

In this paper, the author argued that the spatial transformer surpassed the fixed and local pooling layers by a dynamic mechanism that can actively spatially transform an image by producing an appropriate transformation on the entire feature map. Typical transformations including scaling, cropping, rotations, and non-rigid deformations can be achieved within this simple spatial transformer layer, tuning desired region to a canonical, expected pose to simplify inference in the subsequent layers.

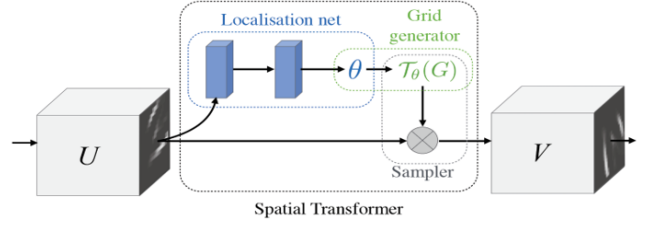


Figure 1: The architecture of a spatial transformer module. The input feature map U is passed to a localization network which regresses the transformation parameters θ . In the sampling step, the output grid G is sampled from the input sampling grid $T_\theta(G)$, from which, we can produce the warped output feature map V .

Formulation of Spatial Transformer

The spatial transformer architecture is composed of three parts, as shown in Fig. 1. The first one is localization network, which takes the input feature map U , localizes each small components, and outputs θ , the parameters to be used in the second part, parameterized sampling grid. In the second step, the output pixels are defined to lie on a regular grid $G = \{G_i\}$ with pixels $G_i = (x_i^t, y_i^t)$, forming the output map V . The transformation between the source coordinates (x_i^s, y_i^s) in the input feature map and the target coordinates (x_i^t, y_i^t) in the output feature map is

$$\begin{bmatrix} x_i^s \\ y_i^s \end{bmatrix} = T_\theta(G_i). \quad (1)$$

For a 2D affine transformation $T_\theta = A_\theta$, (1) yields

$$\begin{bmatrix} x_i^s \\ y_i^s \end{bmatrix} = A_\theta \begin{bmatrix} x_i^t \\ y_i^t \\ 1 \end{bmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{bmatrix} x_i^t \\ y_i^t \\ 1 \end{bmatrix}, \quad (2)$$

from which we can directly see the physical meaning of these transformation parameters θ . One advantage of (2) is that only 6 parameters are required to express cropping, translation, rotation, scale and skew! The third part of the spatial transformer is to perform a differentiable image sampling to generate output feature map. The easiest way is bi-

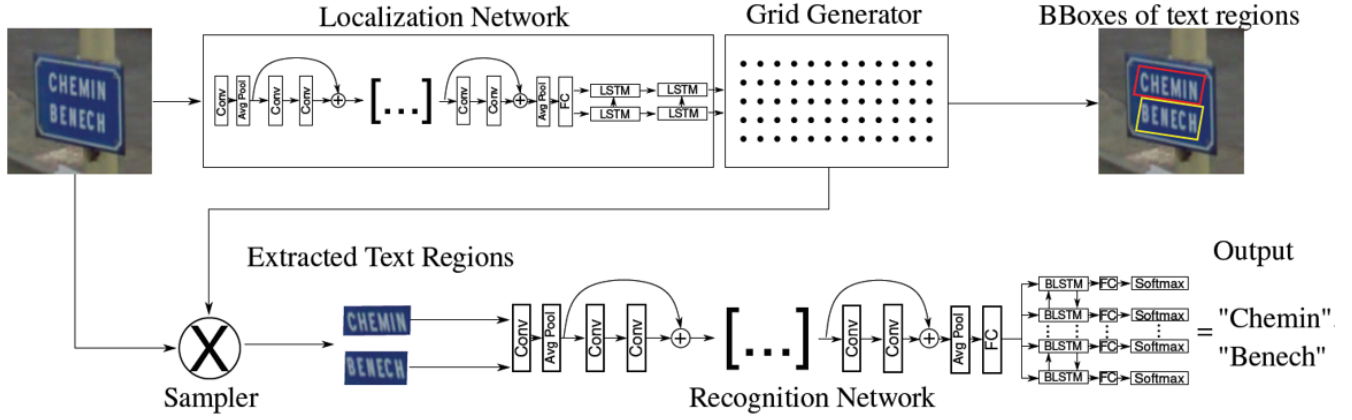


Figure 4: Architecture of text detection and recognition networks. The localization network takes the input images, predicts N transformation matrices, and creates N sampling grids for N predicted text locations. Then, the extracted text regions are supplied to ResNet CNN text recognition networks. The whole system is trained end-to-end by only supplying information about the text labels for each text region.

Architecture and Formulations

The proposed system can be generally divided into two sub-systems: text detector and text recognizer, as shown in Fig. 4. The text detector uses semantic segmentation network in paper 2 to localize the text position with bounding boxes, then uses the spatial transformer module to further improve the performance. The text recognizer uses ResNet-based CNN to learn text recognition from training data set. The training for this whole network is end-to-end, with the text detection part being jointly trained in the recognition part.

Since the text detecting stage has already been described in Paper 1 and Paper 2, I would like to include more details about ResNet CNN text recognition stage here. The CNN of the recognition stage predicts a probability distribution over the label space $L_\epsilon = L \cup \{\epsilon\}$, where L is the alphabet text to be recognized, and ϵ is the blank labels. The network is then trained with LSTM for a fixed number of time steps. In each time step, the cross-entropy loss for the output is calculated to adjust the learning. The loss function for the n -th recognition predicted grid G^n is computed as

$$\mathcal{L}^n(G^n) = \lambda_1 \times \mathcal{L}_{ar} + \lambda_2 \times \mathcal{L}_{as} + \mathcal{L}_{di}. \quad (4)$$

Here λ_1 and λ_2 are scaling parameters that are chosen freely but typically between 0 and 0.5. With \mathcal{L}_{ar} , \mathcal{L}_{as} , \mathcal{L}_{di} in this loss function, the ResNet CNN recognition networks count regulations from grid area, grid aspect ratio, and grid direction, respectively.

Experiments

For validation purpose, this paper first tested the proposed method on the SVHN dataset, yielding a competitive 95.2% recognition accuracy, similar to peer works. Then, this paper tested the challenging French Street Name Signs dataset, giving a 78.0% sequence accuracy. Overall, the proposed method demonstrated a fair performance for text recognition in natural scene.

Comments

This paper proposed a single deep neural network, which learns to detect and recognize text from natural images in a semi-supervised way. Experiments on complex dataset proved the feasibility of proposed method, though further optimization should be done in terms of increasing the maximum words capability and enhancing training robustness.

Implementations on Paper 1

The implementation of spatial transformer module actually requires a full network providing the entire training and learning environment. For example, the authors of paper 1 tested this module in FCN and CNN. To give the readers a better sense on the effects of spatial transformer module, I will first exhaustively introduce one experiment in paper 1. Then, I will show my implementation based on the Two-Layer-Perceptron (TLP) codes used in the class. Instead of directly jumping into fancy application like the text recognition in the paper, I would rather like to quantitatively check the this module's effects on converging speed, final output shape, etc. Paper 1 said this module could speed up the classification depending on the networks, however, no experiment data are shown in the paper, so, I mainly tested the converging speed in my implementations.

Experiments on Classifying Distorted MNIST Data

MNIST is a handwriting dataset. By training different neural network models with and without spatial transformer module, paper 1 demonstrated the ability of spatial transformer layers to improve classification performance through actively transforming the input image. Fig. 5 gives the classification error and some distorted example images. Two different network models FCN and CNN are tested with and without spatial transformers. For each transformer, the authors tested three different transformations as described in (1): affine transformation (Aff) in (2), projective transformation (Proj), and 16-point thin plate spline transforma-







Model		MNIST Distortion						
		R	RTS	P	E			
FCN		2.1	5.2	3.1	3.2	E		
CNN		1.2	0.8	1.5	1.4			
ST-FCN	Aff	1.2	0.8	1.5	2.7	E		
	Proj	1.3	0.9	1.4	2.6			
	TPS	1.1	0.8	1.4	2.4			
ST-CNN	Aff	0.7	0.5	0.8	1.2	RTS		
	Proj	0.8	0.6	0.8	1.3			
	TPS	0.7	0.5	0.8	1.1			

Figure 5: Left: The percentage error for different models on different distorted MNIST datasets. Right: some example test images and their transformations within the spatial transformer layer.

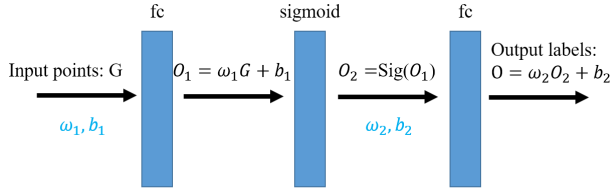


Figure 6: Architecture of TLP networks.

tion (TPS). The tested MNIST data are distorted in different ways: rotation (R); rotation, scale and translation (RTS); projective transformation (P); and elastic warping (E).

Overall, the main effect of this spatial transformer module is to actively transform images within the network, learn the transformation invariance, and improve the classification performance.

Implementations on TLP Networks

The TLP networks classify two groups of points, saying red and blue points, with respect to given training data. The source TLP codes are used in class and published in Github.

TLP Networks without Spatial Transformer The original architecture of TLP networks is formed by three layer, as shown in Fig. 6. The first fully-connected layer accepts input data point G and outputs

$$O_1 = \omega_1 G + b_1 \quad (5)$$

for each input data point. The second Sigmoid layer accepts O_1 and projects O_1 to the range of $[0, 1]$ using Sigmoid function

$$O_2 = \text{Sig}(O_1) = [1 + \exp(-O_1)]^{-1}. \quad (6)$$

The third fully-connected layer transforms O_2 into a number corresponding to the labels. After such three layers, the output predicted label is

$$O = \omega_2 O_2 + b_2. \quad (7)$$

The TLP networks have four set of parameters: w_1, b_1, w_2, b_2 , which are used in two fully-connected layers, and learned through the given labeled training data. Since the output O is predicted labels for the input data points G , we

can define the cost, or loss function \mathcal{L} as the difference between the predicted labels O and the true labels l . Therefore,

$$\mathcal{L} = \sum_{i=1}^{\text{all points}} (O_i - l_i)^2. \quad (8)$$

One obvious advantage is that, at the training stage, the loss function \mathcal{L} counts all the labeled training data points. Based on the loss function (8), we can learn the parameters with naive-gradient-descent method. Eventually, the optimized parameters should minimize the loss function \mathcal{L} .

Here, I will describe how to use the naive-gradient-descent method to learn the parameters w_1, b_1, w_2, b_2 . First, we can write all the parameters in a single vector, saying p ,

$$p = [w_1, b_1, w_2, b_2]. \quad (9)$$

At the very beginning of the training, we assign random numbers to all parameters p . Then, at n -th iteration, we use p_n to calculate the predicted labels O , thereby loss function \mathcal{L} . With this calculated \mathcal{L} , we iteratively update the new parameters

$$p_{n+1} = p_n - \lambda \nabla_p \mathcal{L}|_{p=p_n}, \quad (10)$$

where λ is the constant learning rate. Keep doing the iterations. Until the loss function \mathcal{L} is minimized or reaches desired precision, can the final p and the entire TLP networks be used to classify other unknown points. Otherwise, we need to restart the training with a new set of random numbers, or even to tune the model.

A simple example is given in Fig. 7. Fig. 7(a) shows two groups of training data. Two red points at position $G_1 = [0.3, 0.3]$ and $G_2 = [0.7, 0.7]$ are labeled with $l = -1$. Two blue points at position $G_3 = [0.7, 0.3]$ and $G_4 = [0.3, 0.7]$ are labeled with $l = +1$. With one specific set of random parameters p_0 to start with, the TLP networks converge after 3732 iterations, yielding a final loss $\mathcal{L} = 0.010$ for the four training data points in Fig. 7(a). Using the above trained TLP networks, we can predict the labels for all points in this area, the classification result is shown in Fig. 7(b). Obviously, the training red points are in the predicted red domain whereas the training blue points are in the predicted blue domain.

TLP Networks with Spatial Transformer The simplest way to apply spatial transformer module in TLP networks is to embed a spatial transformer layer before the first fully-connected layer in Fig. 6. This spatial transformer layer requires six new parameters θ and transforms the input data points G into new points G_{st}

$$G_{st} = A_\theta(G) = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{bmatrix} x_i^G \\ y_i^G \\ 1 \end{bmatrix}. \quad (11)$$

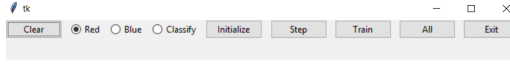
New network architecture is shown in Fig. 8.

After inserting the spatial transformer layer, some formulas in original TLP networks need to be changed. The first fully-connected layer no longer directly accepts input data points but transformed data points G_{st} , so (5) becomes

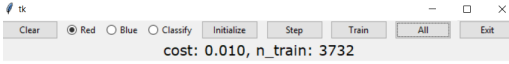
$$O_1 = \omega_1 G_{st} + b_1. \quad (12)$$

New added parameters θ should also be learned dynamically, so including θ in (9) gives

$$p = [w_1, b_1, w_2, b_2, \theta]. \quad (13)$$



(a)



(b)

Figure 7: (a) The given labeled training data points (b) The classification of the entire area based on the predicted labels from the trained TLP networks.

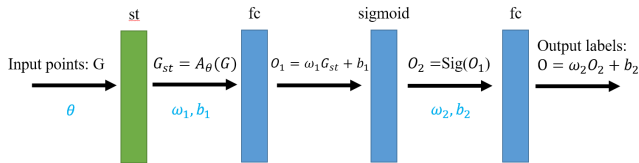


Figure 8: Architecture of TLP networks with a spatial transformer layer.

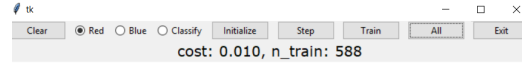


Figure 9: Classification over the entire area using new TLP networks with a spatial transformer layer. The training data are the same as Fig. 7(a)

Table 1: Statistics of Converging Iteration Number

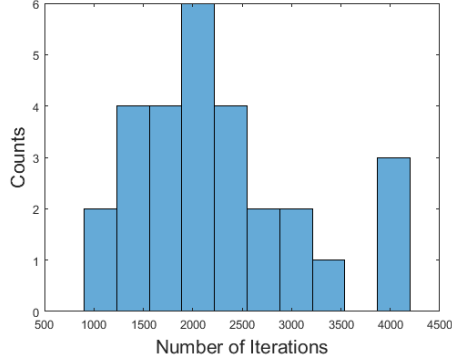
	mean	median	standard deviation
TLP	2291	2118	900
TLP.ST	1535	840	1178

A typical classification result after embedding the spatial transformer layer, as compared to Fig. 7(b) under original TLP networks, is Fig. 9, where the same training data points Fig. 7(a), same learning rate $\lambda = 0.1$, and same converging criteria $\mathcal{L} \leq 0.010$ are used. From Fig. 9, we can see that converging to the same precision of loss $\mathcal{L} = 0.010$ only requires 588 iterations, about six times faster than the case in Fig. 7(b). The red region and blue region strictly follow the training data pattern as expected.

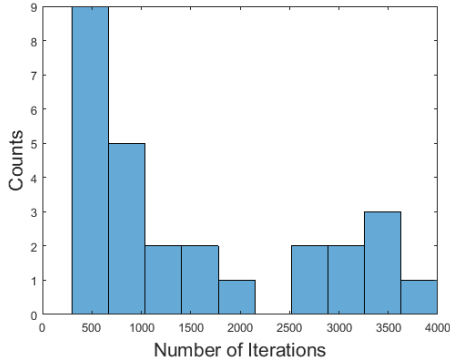
Quantitative Comparison Having seen the spatial transformer layer could give the same classification pattern but with a faster converging speed, I want to quantitatively test the change on converging speed for this TLP networks. Also, I want to see to what extent the converging is affected by the initial random parameters. The second test is due to the fact that we often need to restart the training not only for this TLP networks, but also for many other CNNs. So, I feel the model robustness to initial random condition is very interesting.

To do the comparison, I fix some settings. I use the same training data points in Fig. 7(a), the same learning rate $\lambda = 0.1$, and the same converging criteria $\mathcal{L} \leq 0.010$ or max iteration number $n_{train} < 5000$. Then, I randomly assign the initial parameters p_0 for each training, and count converging iteration numbers over 30 training cases for both TLP networks and spatial transformer TLP networks.

The distribution of the converging iteration number is



(a)



(b)

Figure 10: Distribution of the converging iteration number. (a) TLP networks without spatial transformer layer. (b) TLP networks with spatial transformer layer.

shown in Fig. 10. Table 1 shows the statistics of iteration number. From the distribution pattern, median value and mean value, we can see the spatial transformer layer does accelerate the converging speed. However, this extra layer makes the entire networks more sensible to initial random numbers. From Fig. 10(b), we can see the TLP networks with spatial transformer layer has more polarized iteration number, depending on the initial parameters. Also, the standard deviation is larger. On other words, the TLP networks, after inserting spatial transformer layer, is less stable to initial conditions.

Conclusion

In this report, I choose the spatial transformer network to implement. I implement this module based on the TLP networks used in the class. I quantitatively compare the classification change with and without spatial transformer layer. From my tests, both networks with and without this extra layer could achieve very good classification accuracy. In terms of converging speed, on average, the spatial transformer layer does increase the converging speed. However, this increasing is less stable, and is very much depending on the initial random numbers. In the paper, the authors only mentioned that the spatial transformer layer may increase the speed. Here, my implementations shed some lights on the answer.

References

- Bartz, C.; Yang, H.; and Meinel, C. 2018. See: Towards semi-supervised end-to-end scene text recognition. In *AAAI Conference on Artificial Intelligence*.
- Dai, J.; He, K.; and Sun, J. 2016. Instance-aware semantic segmentation via multi-task network cascades. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3150–3158.
- Hariharan, B.; Arbelaez, P.; Girshick, R.; and Malik, J. 2015. Hypercolumns for object segmentation and fine-grained localization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- He, K.; Zhang, X.; Ren, S.; and Sun, J. 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. In Fleet, D.; Pajdla, T.; Schiele, B.; and Tuytelaars, T., eds., *Computer Vision – ECCV 2014*, 346–361. Cham: Springer International Publishing.
- Jaderberg, M.; Simonyan, K.; Zisserman, A.; and kavukcuoglu, k. 2015. Spatial transformer networks. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc. 2017–2025.
- Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc. 91–99.