Enhancement Spatial Transformer Networks for Text Classification

Ka-Hou Chan
School of Applied Sciences,
Macao Polytechnic Institute
Macau, China
chankahou@ipm.edu.mo

Sio-Kei Im
Macao Polytechnic Institute
Macau, China
marcusim@ipm.edu.mo

Vai-Kei Ian
School of Applied Sciences,
Macao Polytechnic Institute
Macau, China
p9700341@ipm.edu.mo

Ka-Man Chan
School of Applied Sciences,
Macao Polytechnic Institute
Macau, China
p0508326@ipm.edu.mo

Wei Ke
School of Applied Sciences,
Macao Polytechnic Institute
Macau, China
wke@ipm.edu.mo

ABSTRACT
This paper introduces a 2D transformation based framework for arbitrary-oriented text detection in natural scene images. We present the localization networks within Spatial Transformer Networks (STN), which are designed to generate proposals with text orientation affine information including translation, scaling and rotation. This information will then be adapted as learning parameters to make the proposals to be fitted into the text regular form in terms of the orientation more accurately. Localization network is proposed to project arbitrary-oriented proposals to a feature map for a text region classifier. Compared with any previous text detection systems, this work ensures the relationship between the learning parameters, which can lead to a better approximation for orientation. As a result, this new layer greatly enhances the training accuracy. Moreover, the design and implementation can be easily deployed in the current systems built upon the standard CNNs architecture.

CCS CONCEPTS
• **Computing methodologies → Machine learning-Machine learning approaches-Neural networks.**

KEYWORDS
Spatial Transformer Networks, Learning Parameters, Affine Transformation, Homogeneous Matrix

1. INTRODUCTION
Deep learning models have a great impact on the research in image analysis, hence, performance can significantly be improved in many standard evaluations. Convolutional neural networks (CNN) [1], is a powerful model especially for the analysis of images, which excel at the learning of meaningful representations of features and concepts within the images. These capabilities make CNN extremely valuable for solving problems in the image analysis domain, but it is still limited by the lack of analysis ability to 3D projective changed images. Most methods based on the sliding windows, connected components and the up-down strategies are designed to handle horizontal based text detection [2] [3]. As we know that the appearance of a symbol in an image is subject to various deformations: this (geometry) transform is likely due to the different handwriting and perspective of photography, and CNN typically does not take any advantages of such unexpected transformations on the invariants of this domain. In recent years, Spatial Transformer Networks (STN) [4] was proposed to be included into the CNN architecture for the provision of spatial transformation capabilities. It makes use of the 2D transformation matrix concept for image regulating, and the transformation can be presented by a homogeneous matrix as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (1)$$

here w_{ij} are the learning parameters. Thus, each STN layer allocate six learning parameters to approximate this homogeneous matrix for the machine training. After STN sampling, the text image will be projected into a regular result by affine transformation, which will be well used in accordance with the CNN classification. Furthermore, the STN can also be used in CNN feature capturing because the max-pooling layers in CNN allow containing the maximum feature to spatially invariant to the position. In contrast to the current standard practice, heuristically incorporates these priors during training via data augmentation (such as by applying a random rotation or scaling to each of the training images). The STN module can be inserted into the existing CNN architecture, so that the neural network can actively transform the feature map according to the feature map itself. However, while STN can help reduce the test error of CNN-based models, it does not guarantee that transformation invariance will be enforced on data not seen during training.

1.1 Related Works
Projective transformations as an alternative to data augmentation, knowledge about geometric transformations can be directly

ICGSP 2020, June 26–29, 2020, Nagoya, Japan
 ACM ISBN 978-1-4503-7781-2/20/06…$15.00
https://doi.org/10.1145/3406971.3406981

embedding into the layers before CNN processing. To better achieve spatial-invariant text classification, [4] first proposed STN to improve the performance through feature transformation by using affine transformation related learning parameters. Similarly, [5] grouped equivariant CNNs' filters under different transformations and pooled their responses to create invariant representations. Then more lightweight alternatives to STNs are deformable convolutional networks [6]. Later, [7] made use of STN repeatedly with an inverse composite matrix by propagating warp parameters rather than features for improved performance. [8] proposed a rotation-invariant network by replacing the grid generation of STN with a polar transformation, which transform input features into the polar coordinates with the origin that was determined by the mass center of the image. [9] presented a global rotation equivariane which used convolution and pooling with weight flips and four rotations with right-angle step-size.

Different from these global approaches, [10] (R-CNN) and [11] introduced a local network for generating region proposals to object detection, which was to find some text-like objects first. That way, better region proposals could be generated in a shorter period of time. Also, this enabled a higher level of recognition accuracy and better real-time performance. The advantage of these approaches is that multi-text within an image could be found. For using local rotation-invariant object detection, [12] used a rotation region proposal network to transform regions for classification using rotation region-of-interest pooling. [13] investigated object detection using a progressive calibration network that predicted rotation by half of right-angle step-size after sliding window. [14] proposed a novel bilinear pooling operation to combine pairwise local features and improved classification performance.

However, the CNN-based approaches usually have the fixed-size (predefined) image patches, which confine the shapes of CNN inputs for the subsequent steps. In addition, how to transform the original image to regular presenting is another time-consuming problem. In this work, we are going to propose a 2D transformation layer that is similar to the idea of STN and [12]. These methods introduce a neural network model that rectifies an image by learning its affine transformation matrix. Compare to others, our proposed layer will be applying before the CNN layer since the CNN results may causes data losing which might misdirect the transformed results. Another benefit is that the part of CNN classification does not need to be re-trained or can be replaced by any other classification approaches.

1.2 Outline

The rest of this article is organized as follows. Section 2 discusses the concept of spatial transformation in the making use of neural networks. A couple of instances are indicated to give an analysis of affine matrix construct and conditions, including that how the localization network can be found by our proposed learning parameters. Next, in Section 3, we give the idea of how this approach can be integrated, followed by the configurations of our experiments and evaluation environments. The results and improvements are illustrated and discussed further in details in Section 4 as well. Finally, Section 5 concludes the papers.

2. PROPOSED SPATIAL TRANSFORMATION

We consider the problem of recognizing the artificially generated patterns that are subjected to projective transformations which simulate changes in the camera view point. With STN method which allows for translation, rotation and scaling transformation with only six learning parameters w_{ij}, they are independent to each other in Eq.1 of using somewhat inconsistent with spatial transformation. To overcome this risk, we take into the standard transformation formula, as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = R(\theta)S(s_x, s_y)T(t_x, t_y)\begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (2)$$

here $R(\theta)$, $S(s_x, s_y)$ and $T(t_x, t_y)$ allow rotation, scaling and translation respectively. Therefore, we propose to make use of this constraint into the training processing. Although the structure is similar to STN module, Eq.1 will be replaced by Eq.2 and the five learning parameters would be allocated for machine training as well. Based on the predicted transformation parameters, a sampling grid is created and applied to the input image that directly learn angle and offsets to the regular sampling grid correspondingly. As the Cartesian coordinate representation can be easily deduced from the input, these approaches increase the flexibility of neural networks in the handling of geometric transformations.

2.1 Localization Network

The localization network is implemented similarly to STN, but with Eq.2 instead. This network takes an image or its feature map as the input. The output of the localization network is produced by applying an affine transformation and bilinear interpolation. As mentioned above, the five transformation parameters are predicted by using the localization network as:

$$A(\theta, s_x, s_y, t_x, t_y) = R(\theta)S(s_x, s_y)T(t_x, t_y) \quad (3)$$

where A specifies the affine homogeneous matrix. Hence, the range of these training parameters: $t_x \in [-0.5, +0.5]$, $t_y \in [-0.5, +0.5]$, $s_x, s_y \in [1.0, 2.0]$ and $\theta \in (0, 2\pi]$ (see Fig.1) can be predicted. Note that we have use height and width to normalize the coordinates of the map.

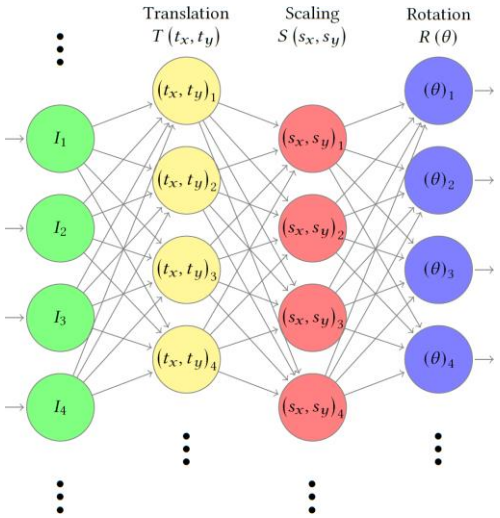

Figure 1. The localization network takes the input feature map with width and height. The training parameters of translation $t_x \in [-0.5, +0.5]$, $t_y \in [-0.5, +0.5]$, scaling $s_x, s_y \in [1.0, 2.0]$ and rotation $\theta \in (0, 2\pi]$ to be applied to the feature map according to this order.

The purpose of this network is to find the relationship between input image and its affine matrix (A). Therefore, we can make use of the features of neural networks to allocate lots of neural cells to cover all the situations. Because our network must satisfy the 2D transformation in Eq.3, we will input the source image to localization network directly instead of CNN. Furthermore, the translated parameters (t_x, t_y) have to be predicted first. Since $T(t_x, t_y)$ is purposed to find the origin and move it to the center of current feature, $S(s_x, s_y)$ and $R(\theta)$ which is then applied to the Cartesian coordinates is also based on the feature center as well. Thus, we have to determine the origin started from the beginning, which is beneficial for later works.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x \cos\theta & -s_y \sin\theta & s_x t_x \cos\theta - s_y t_y \sin\theta \\ s_x \sin\theta & s_y \cos\theta & s_x t_x \sin\theta + s_y t_y \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

These parameters vary depending on the transformation type that is parameterized: e.g. translate is dependent on the size of input feature map. Every input image passes through a localized network, which will determine (t_x, t_y), (s_x, s_y) and (θ) to construct $T(t_x, t_y)$, $S(s_x, s_y)$ and $R(\theta)$ respectively, and application to source image to producing a transformed regular result.

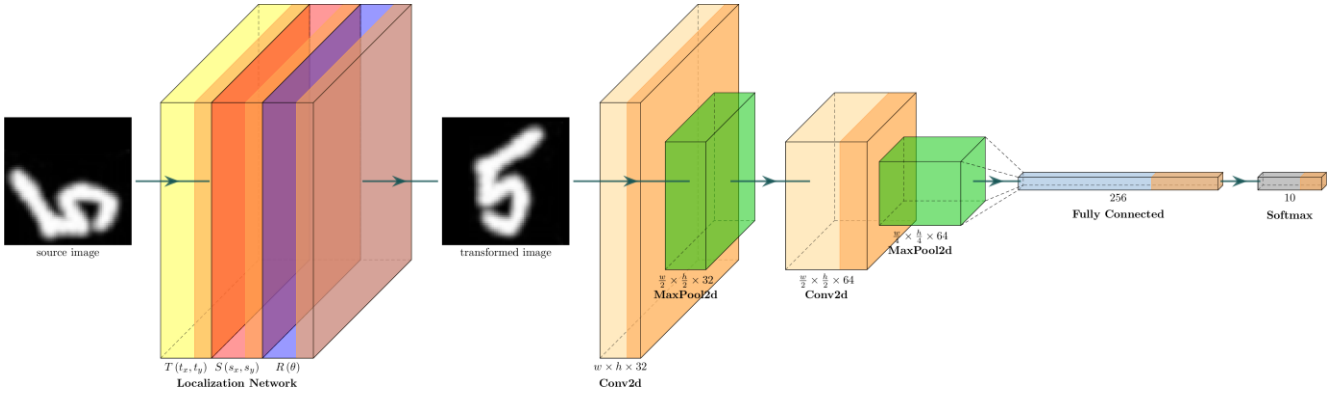

Figure 2. Network architecture, localized network is composed of $T(t_x, t_y)$, $S(s_x, s_y)$ and $R(\theta)$. The inherent translational equivalence of CNNs is independent of the convolution kernel and evident in the corresponding translation of the output in response to translation of the input. The transformed map representation is centered with respect to the original object location, rotations and scaling are now regulated.

As shown in Fig.2, a (pre-trained) convolutional network is connected to the localization network and the transformed map for classification processing is received, each of these results will find the probability of belonging to every class, such that we can calculate the cross entropy as the final Softmax input for our final determination. This localization network can also be connected to a fully-connected network or other classification approach.

2.2 Learning of Transformation Parameters

As mentioned in the previous sections, we are not going to connect CNN before localization network directly. In contrast, connection with source image is made and these transformation parameters are trained from the image data instead. In general, a linear layer that combines with an activation function can connect multiple tensors of different dimensions, as shown below,

$$x' = f(\sum weight_i x_i + bias)$$

here $weight$ and $bias$ are the internal training parameters constructing the linear layer, and an activation function $f(x)$ will distort the result into its nonlinear form. However, much of the activation function does not generalize well to multivariate approximation problems [15]. Since most multivariate approximation problems involve tensor product spaces, they are highly dependent to the features of the activation function being used and require lots of neural cells to increase the layer progression [16]. Thus, we recommend to apply the self-adaptive layer [17] for the tensor conversion as illustrated as,

$$x' = \sum c_i \cos(i \arccos(x)) \quad (4)$$

here coefficient (c_i) is the training parameter. The advantage of it is that it can approximate any unknown form, thus the necessity for any activation function for nonlinear consideration is not required. Back to the localization network, we will use three separate data flows for the prediction of $T(t_x, t_y)$, $S(s_x, s_y)$ and $R(\theta)$. There are three sets of internal training parameter (c_i) and self-adaptive (hidden) layer to be allocated, and the initialize value will become $T(0,0)$, $S(1,1)$ and $R(0)$ as their initial states according to the acknowledgment of the identity matrix of transformation.

3. IMPLEMENTATION

Since the ideas and algorithms have been determined and decided, our architecture can be implemented as shown in Fig.2. Notice that each parameter of the matrix only requires a simple calculation of Eq.4 and no complicated activation function is needed. In order to achieve our network, we make use of the scientific deep learning framework PyTorch [18] for our implementation. As mentioned in the previous sections, the localization network consists of three matrices, and we must determine five parameters for the construction. By passing through the network, we can obtain the transformed features accordingly. This process can be seen as a set of sequential layers with a unique activation function. The computation can be achieved in the Sequential container and Forward function, which are the implementation interface provided by PyTorch. We are going to explain these in details through Algorithm 1, 2 and 3 below.

Algorithm 1: *Sequential* container for (t_x, t_y) prediction and *Forward* function for $T(t_x, t_y)$ construction.

Sequential `Translation`:
  SelfAdaptive(*degree* ← 16) ▷ Eq. 5
  Linear(*in_features* ← $w \times h$, *out_features* ← 2)
end Sequential

def `Forward`:
  $(t_x, t_y) \leftarrow$ `Translation`(*source image*)
  return $T \leftarrow \begin{bmatrix} 1.0 & 0.0 & t_x \\ 0.0 & 1.0 & t_y \end{bmatrix}$
end def

Algorithm 2: *Sequential* container for (s_x, s_y) prediction and *Forward* function for $S(s_x, x_y)$ construction.

Sequential `Scaling`:
  SelfAdaptive(*degree* ← 16) ▷ Eq. 5
  Linear(*in_features* ← $w \times h$, *out_features* ← 2)
end Sequential

def `Forward`:
  $(s_x, s_y) \leftarrow$ `Scaling`(*source image*)
  return $S \leftarrow \begin{bmatrix} s_x & 0.0 \\ 0.0 & s_y \end{bmatrix}$
end def

Algorithm 3: *Sequential* container for (θ) prediction and *Forward* function for $R(\theta)$ construction.

Sequential `Rotation`:
  SelfAdaptive(*degree* ← 16) ▷ Eq. 5
  Linear(*in_features* ← $w \times h$, *out_features* ← 1)
end Sequential

def `Forward`:
  $\theta \leftarrow$ `Rotation`(*source image*)
  return $R \leftarrow \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$
end def

As shown in these algorithms, width (w) and height (h) are the received image sizes. Their Forward function will produce our purpose matrices that form the affine matrix (A) as Eq.3. It is worth mentioning that the size of T is different to S and R, therefore T must be assigned on the right side, according to matrix multiplication, this also complies with our idea to dealing with translation first with homogeneous matrix. Moreover, in order to solve the vanishing gradient problem, we would like to make use of a novel activation function SinPN, which is proposed in our other work [19] as follows,

$$SinPN(x) = \sin(x) + Nx \tag{5}$$

The vanishing gradient problem can effectively be alleviated by adjusting the kernel parameter N. We expect that having any $N > 1$ is effective by theory. This allows SinPN to be used in the fully connected processing instead of ReLU [20]. Algorithm 4 below contains more implementation details about the STN sampling and CNN classification.

Algorithm 4: Correspond to Fig. 1, pseudo code for network architecture.

Sequential `Convolution`:
  Conv2d(*in_features* ← 1, *out_features* ← 32)
  ReLU
  MaxPool2d(*kernel_size* ← 2)
  Conv2d(*in_features* ← 32, *out_features* ← 64)
  ReLU
  MaxPool2d(*kernel_size* ← 2)
end Sequential

Sequential `FullyConnected`:
  Linear(*in_features* ← $\frac{w}{4} \times \frac{h}{4} \times 64$, *out_features* ← 256)
  SinPN($N \leftarrow 2.0$) ▷ Eq. 6
  Linear(*in_features* ← 256, *out_features* ← *classes*)
end Sequential

def `Forward`:
  ...
  $A \leftarrow R \times S \times T$ ▷ Eq. 3
  transformed image ← $A \times$ *source image*
  feature ←`Convolution`(*transformed image*)
  return `FullyConnected`(*feature*)
end def

4. EXPERIMENTATION RESULTS AND DISCUSSION

In this section, we evaluate our presented network architecture on standard scene of text detection/recognition datasets. We will apply our approach into the EMNIST [21] training models, which are several different splits provided in this dataset, including various classes for training and testing:

BALANCED contains a set of characters with an equal number of samples per class;

LETTERS merges a balanced set of the uppercase and lowercase letters into a class;

DIGITS provide balanced handwritten digit datasets directly compatible with the original MNIST dataset, and;

MNIST references original MNIST [22].

We use the image data directly without any pre-processing. The result is from the original view test without any enhancement as well. All our experiments were conducted on an NVIDIA GeForce GTX 1080 with 8.0GB of video memory. We evaluate the classification performance on the same environment with and without the usage of our approach. In order to show the gradient activity, we use the adaptive gradient related optimizer, the Adagrad [23] method, for accuracy. We conducted our experiments four times, each of which processed 100,000 iteration steps for each test, on different STN models and took the average of the four to evaluate the overall performance. We used the same dataset in each test with a batch size of 100 per iteration step. For more detail, the complete python source code can be found at:

github.com/ChanKaHou/Enhancement-STN

In these experiments, we evaluate the classification performance by applying our approach and CNNs [1] with or without STN [4] localization network. What they all have in common is the final classification with convolutional network, which performs two sets of convolutional and pooling layers. They also contain 7×7 and 5×5 convolution respectively following by 2×2 max pooling layer with stride 2, and ReLU is used as activation function for each convolutional layer. We then compare the accuracies of the different models. Our model configuration is the same as the previous section and we find the test accuracy (%) after completion of 100 epochs.

Table 1. The best accuracy of various experiments for EMNIST datasets.

	Our proposed	*STN + CNNs*	*CNNs*
BALANCED	0.8826	0.8808	0.8799
LETTERS	0.9415	0.9385	0.9378
DIGITS	0.9960	0.9954	0.9956
MNIST	0.9938	0.9924	0.9936

As shown in Table 1 and Fig. 3 to 6, the advantages of our proposed method are not very obvious since the feature of these datasets are presented in regular. Nevertheless, it can be seen that less time was spent and our network can provide a faster learning as it requires fewer learning parameters. In all fairness, we believe that the final accuracy of our proposed method, when comparing to CNNs with or without STN localization network, will likely to be identical to each other because our common objectives are to make the approximation as closely as affine transformation as possible and it has to be accomplished by allocating the homogeneous matrix.

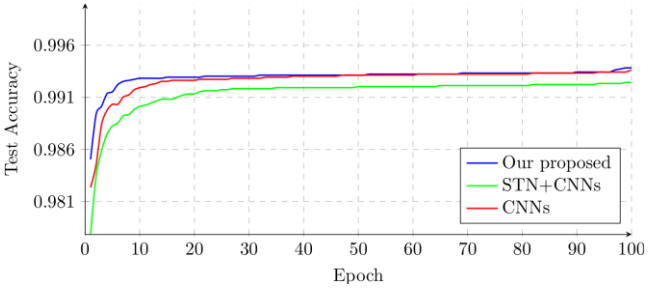

Figure 3. Test accuracy rates for MNIST after 100 epochs of training.

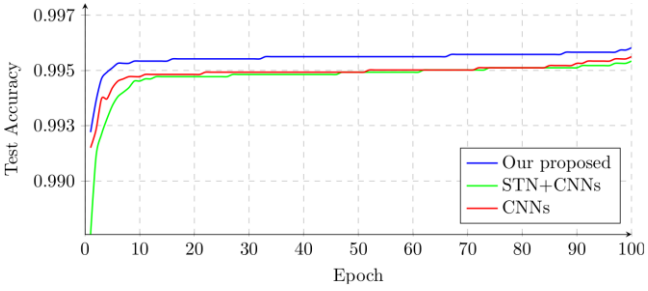

Figure 4. Test accuracy rates for DIGITS after 100 epochs of training.

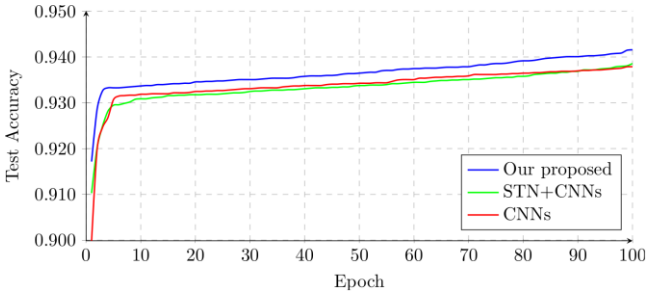

Figure 5. Test accuracy rates for LETTERS after 100 epochs of training.

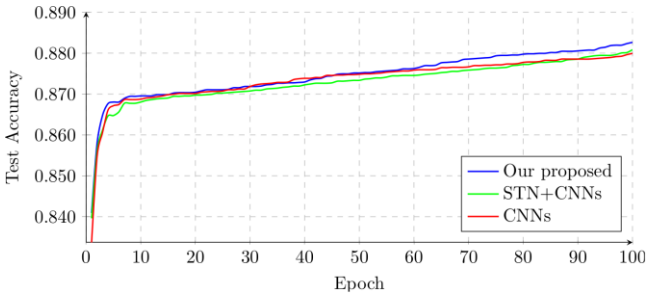

Figure 6. Test accuracy rates for BALANCED after 100 epochs of training.

5. CONCLUSION

We present a novel spatial network which is based on 2D transformation theory. Our method can apply to the original architecture as a middle network. By introducing the design mapping expansion forms to neural nodes, a transformation layer can rectify an image by learning its affine transformation matrix. We show that the localization network can improve the effectiveness of classification. To validate the effectiveness, we use models with our approach and CNNs with or without STN localization network. The results show that our approach can outperform both architectures in terms of convergence speed, from the start till the end. Further work such as shearing and reflection can be done to enhance this affine matrix.

6. ACKNOWLEDGMENT

The article is part of the research project funded by The Science and Technology Development Fund, Macau SAR (File no. 0001/2018/AFJ).

7. REFERENCES

[1] 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.

[2] K. Wang and S. Belongie, "Word spotting in the wild," in European Conference on Computer Vision, Springer, 2010, pp. 591-604.

[3] M. Jaderberg, A. Vedaldi and A. Zisserman, "Deep features for text spotting," in European conference on computer vision, Springer, 2014, pp. 512-528.

[4] M. Jaderberg, K. Simonyan, A. Zisserman and others, "Spatial transformer networks," in Advances in neural information processing systems, Curran Associates, Inc., 2015, pp. 2017-2025.

[5] T. Cohen and M. Welling, "Group equivariant convolutional networks," in International conference on machine learning, 2016.

[6] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu and Y. Wei, "Deformable convolutional networks," in Proceedings of the IEEE international conference on computer vision, 2017.

[7] C.-H. Lin and S. Lucey, "Inverse compositional spatial transformer networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2568-2576.

[8] C. Esteves, C. Allen-Blanchette, X. Zhou and K. Daniilidis, "Polar Transformer Networks," CoRR, vol. abs/1709.01889, 2017.

[9] D. E. Worrall, S. J. Garbin, D. Turmukhambetov and G. J. Brostow, "Harmonic networks: Deep translation and rotation equivariance," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5028-5037, 2017.

[10] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580-587.

[11] S. Ren, K. He, R. Girshick and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in Advances in neural information processing systems, 2015, pp. 91-99.

[12] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng and X. Xue, "Arbitrary-oriented scene text detection via rotation proposals," IEEE Transactions on Multimedia, vol. 20(11), no. IEEE, pp. 3111-3122, 2018.

[13] X. Shi, S. Shan, M. Kan, S. Wu and X. Chen, "Real-time rotation-invariant face detection with progressive calibration networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2295-2303.

[14] Y. Gao, O. Beijbom, N. Zhang and T. Darrell, "Compact bilinear pooling," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 317-326.

[15] T. Lyche, K. M{\o}rken and E. Quak, "Theory and algorithms for non-uniform spline wavelets," Multivariate approximation and applications, no. Cambridge University Press Cambridge, p. 152, 2001.

[16] J. a. Y. Q. Fan, Nonlinear time series: nonparametric and parametric methods, Springer Science \& Business Media, 2008.

[17] K.-H. Chan, S.-K. Im and W. Ke, "Self-Adaptive Layer: An Application of Function Approximation Theory to Enhance Convergence Efficiency in Neural Networks," in The 34th International Conference on Information Networking (ICOIN 2020), 2020.

[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, "Automatic differentiation in pytorch," 2017.

[19] K.-H. Chan, S.-K. Im, W. Ke and N.-L. Lei, "SinP [N]: A Fast Convergence Activation Function for Convolutional Neural Networks," in 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), 2018.

[20] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in Proceedings of the 27th international conference on machine learning (ICML-10), 2010.

[21] G. Cohen, S. Afshar, J. Tapson and A. Van Schaik, "EMNIST: Extending MNIST to handwritten letters," in 2017 International Joint Conference on Neural Networks (IJCNN), 2017.

[22] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[23] Z. Zhou, S. Chen and Z. Chen, "FANNC: A fast adaptive neural network classifier," Knowledge and Information Systems, vol. 2, no. 1, pp. 115-129, 2000.