Braille Recognition Using Deep Learning

CHANGJIAN LI

Auckland University of Technology, Auckland 1010, New Zealand

WEI QI YAN

Auckland University of Technology, Auckland 1010, New Zealand

Text is the media to convey and transmit information. Braille is extremely important for vision impaired people to exchange information through reading and writing. A braille translator is crucial tool for aiding people to understand braille messages. In this paper, we implement character-based braille translator using ResNet, there are three versions of ResNet we implement for braille classifiers, including ResNet-18, ResNet-34, and ResNet-50. We also implement a word-based braille detector using a novel solution called Adaptive Bezier-Curve Network (ABCNet), which is a Scene Text Recognition (STR) method for detecting word-based text in natural scenes. A comparison is present to evaluate the performance of ABCNet.

CCS CONCEPTS • Computing methodologies~Artificial intelligence~Computer vision

Additional Keywords and Phrases: Braille recognition, Deep learning, Convolutional neural network, Natural scene text detection.

1 INTRODUCTION

Braille is designed for blind persons to operate through touch reading and writing. Braille normally contains 64 characters. Each character is formed in a braille cell, made by six raised or flatten dots. Each cell has two columns of three braille dots managed in a shape of 3x2 matrix. The braille has two levels. The first level is for each cell which only represents one alphabet, number or special character. The second level allows a braille cell to represent a single word, prefix, or suffix [1]. As the braille is made for blind and visually impaired people, if others would like to share information with them by text, they need the knowledge of braille, and transfer the normal text messages into braille cells. But not everyone can well understand the braille characters, and there is a gap of written communication between two worlds. There are numbers of implementations for braille recognition using traditional image processing techniques, all have the similar processes including image acquisition, preprocessing (noise removal, image alignment) and cells segmentation. The methods for matching the segmented braille cells into related alphabets are various. One of the most popular techniques is to convert the cells into either 3x2 binary matrixes or a 6-bit binary values, then match with the existing records in a database to identify them.

There is a disadvantage of these matching methods which is that they only recognize well-defined braille characters. If the characteristic of a braille cell has changed due to distortion and unwanted noises, this might cause mismatching of the cells, or the method cannot identify the cell's pattern. The alternative classification for matching the braille cells into corresponding alphabets is to use artificial neural networks (ANNs). An ANN is formed by a group of connected units or nodes for mathematical calculation called artificial neurons, mimicking the structure of a biological brain. A node can pass a signal to other nodes like a synapse in a brain. When a neuron receives various inputs from other neurons, the values will go through a nonlinear function, the sum of the values will be exported as its output and be passed to other neurons. The neurons in an ANN model are organized into layers, each layer has various transformations depends on its inputs. There are three basic components in a neural network: Input layer, hidden layers and output layers. Signals (inputs) will travel through the network from the input layer to the last layer (output layer). Wong et al. [2] has implemented a neural network with one hidden layer (contained seven radial basis function neurons) for converting seven braille characters into corresponding alphabets and symbols in 2003. Convolutional Neural Network (CNN or ConvNet) is a type of deep neural networks, mainly used for image classification and recognition. The main types of layers in a ConvNet are input layer, convolutional layers, pooling layers, and fully connected layer. Compared to ANN, a ConvNet is able to capture spatial features and offer parametric sharing, thus it is much feasible for analyzing visual imagery.

In this paper, we will introduce our literature review in Section 2. Our methods will be depicted in Section 3, our results are demonstrated in Section 4, our conclusion will be remarked in Section 4.

2 LITERATURE REVIEW

Convolutional neural network[3] was introduced at first. The model is well-known as LeNet-5, which applies gradient-based learning algorithm with ConvNet for document recognition. The gradient-based learning algorithms generate a decision surface for classifying high-dimensional patterns, thus it is very suitable for classifying handwritten characters. The architecture of LeNet-5 has two sets of convolutional layers and average pooling layers, one flattening convolutional layer. At the end of this network, two fully connected layers and one softmax classifier are responsible for the classification. A general ConvNet is structured by using three basic functions, convolution, activation and pooling. The input for ConvNet is an image with the width, height, and depth, for example, an image from the CIFAR-10 dataset has the size 32×32 with a depth of 3 which represents RGB channels [4].

There are a plenty of ConvNet models which have been created in the last decade, such as AlexNet, VGG-Net, GoogLeNet, ResNet, etc. Each model has multiple versions and structures for handling a variety of types of classifications.

The design of ResNet models was proposed [8]. As the development of deeper neural networks is growing fast, a converging degradation problem is occurred. When the depth of a network on the raise, the accuracy of the model gets saturated and degrades rapidly, and overfitting is not the reason for degradation. With more layers added into the network might bring higher training error to the model. The accuracies of VGG-18 (18 layers) and VGG-34 (34 layers) are attained by classifying CIFAR-10 dataset, the results show that VGG-18 was fully trained in 5 minutes and got 80% accuracy, VGG-34 spent 8 minutes for training and got 72% accuracy. Thus, ResNet was applied to solve this problem.

The ResNet implements a deep residual learning framework in this model. Considered $H(x)$ represents an underlying mapping that is fit for a few stacked layers, x denotes the input to the first layer, and the stacked nonlinear layers fit the mapping of $F(x):=H(x)-x$. The original mapping is described as $F(x)+x$. If an identity mapping is optimal, the loss rate will increase when x is changed, $F(x)$ will approach to zero. A shortcut connection is employed for $F(x)+x$ with a feedforward neural network. Based on stacks of residual blocks, the output of a hidden layer is passed to another layer without changes, this ensures the loss will not dramatically be risen in a deeper stacked network, thus the network is stacked into even 1,000 layers.

For word-based braille recognition, the techniques will be involved with natural scene text detection and recognition. Scene text detection and recognition have attracted more attention in recent years. The texts within natural scene have a diversity of font styles and sizes, ratios, shapes, and distortion that make the detection and recognition tasks become much difficult. Recently, there are a plenty of end-to-end models which are made for detecting arbitrarily shaped texts in natural scenes, such as *Mask textspotter* [9], TextNet [10], and TextDragon [11]. Although these methods have various problems, the segmentation-based methods require complex model pipelines, the character-based methods need many character-level annotations while training the models. These issues need expensive efforts while proceeding real-time detection tasks. The adaptive Bezier Curve Network (ABCNet) was proposed [12], which is an e2e trainable, single-shot, anchor-free convolutional neural network for detecting arbitrarily shaped text in natural scene. The ABCNet carries the hope of the team of creating a simple e2e model which efficiently spots arbitrarily shaped text in real time, the performance is good enough to compare with the state-of-art methods.

In ABCNet, a novel solution was proffered to handle arbitrarily shapes and curved text by implementing Bezier curve adaption, adding alignment layer – BezierAlign into the network, which adds negligible computation cost into detection process. There are vast number of components in the network: Bezier curve detection, Bezier-align and recognition branch.

That regression-based methods are more direct methods for detecting arbitrarily shaped text. The Bezier curve *c*(*t*) is represented by using Bernstein Polynomials [13]

$$c(t) = \sum_{i=0}^{n} b_i B_{i,n}(t), 0 \le t \le 1, \tag{1}$$

where *n* stands for polynomial degree, b_i refers to the *i-th* control points, $B_{i,n}(t)$ means the Bernstein basis polynomials:

$$B_{i,n}(t) = \binom{n}{t} t^i (1-\mathrm{t})^{n-i}, i = 0, \dots, n, \tag{2}$$

where $\binom{n}{t}$ stands for binomial coefficient. A cubic Bezier curve (n = 3 for instance) is suitable in most of the cases to fit the arbitrarily shaped text, the text detection is implemented with eight control points. For the model to learn the coordinates of all control points, it is necessary to generate the Bezier curve as the ground truths.

For the points on the curved boundary $\{p_i\}_{i=1}^{n}$, p_i stands for *i-th* annotation points. In order to generate the result *c(t)* in Equation (8), the transformation is needed as shown in eq.(3)

$$\begin{bmatrix} B_{0,3}(t_0) & \cdots & B_{3,3}(t_0) \\ \vdots & \ddots & \vdots \\ B_{0,3}(t_m) & \cdots & B_{3,3}(t_0) \end{bmatrix} \begin{bmatrix} b_{x0} & b_{y0} \\ b_{x1} & b_{y1} \\ b_{x2} & b_{y2} \\ b_{x3} & b_{y3} \end{bmatrix} = \begin{bmatrix} p_{x0} & p_{y0} \\ p_{x1} & p_{y1} \\ \vdots & \vdots \\ p_{xm} & p_{ym} \end{bmatrix} \tag{3}$$

where *m* represents the total number of the annotated points on a boundary, *t* will be calculated by the ratio of the accumulative length to the polyline perimeter. By using Eq. (7) and Eq. (9), the original annotation will be converted into parameterized Bezier curve. After the curve is generated, a regression method is given to regress each target as shown in Eq.(4)

$$\Delta_x = b_{ix} - x_{min}, \Delta_y = b_{iy} - y_{min.} \tag{4}$$

In Eq. (4), x_{min} and y_{min} represent minimum values of *x* and *y* of the four vertexes. If the input feature map and control points of Bezier curve are formally given, the pixels from the rectangular output feature map in size of $h_{out} \times w_{out}$ is processed. For example, given g_i from output feature map in position (g_{iw}, g_{ih}), t is represented as

$$t = \frac{g_{iw}}{w_{out}} \quad (5)$$

The points of upper Bezier curve boundary *tp* and lower boundary *bp* are calculated by using *t* and Eq. (7). The sampling points are indexed linearly by using *tp* and *bp*

$$op = bp \cdot \frac{g_{ih}}{h_{out}} + tp \cdot (1 - \frac{g_{ih}}{h_{out}}) \quad (6)$$

In ABCNet, a light-weight recognition branch is implemented for improving execution speed. The branch contains six convolutional layers, one bidirectional LSTM layer [14], and one fully connected layer.

3 OUR METHODS

For character-based braille classification, we take use of a dataset associated with the project AI4SocialGood. The dataset contains 1,404 images of braille characters, there are 27 classes of braille characters corresponding to 26 English alphabets and space symbol. The sizes of the images are various from 91×96 to 509×598. The characters are printed having various colors, brightness of backgrounds, having added noises.

For preprocessing, we have completed data augmentation for enhancing the robustness of the model, rotating each image in random angle within the range [-45°,45°], combining the rotated images with original dataset, the final dataset contains 2,808 images. OpenCV was used to import the images, and resize them into 52×52, then normalize the pixels to 0-1 (pixel value divided by 255). The processed data will be randomly shuffled and separated into training set, test set, and validation set. Training set has 2,248 records with batch size of 40, test set has 280 records with batch size of 10, and validation set has 280 records with batch size of 10 as shown in Fig.1.

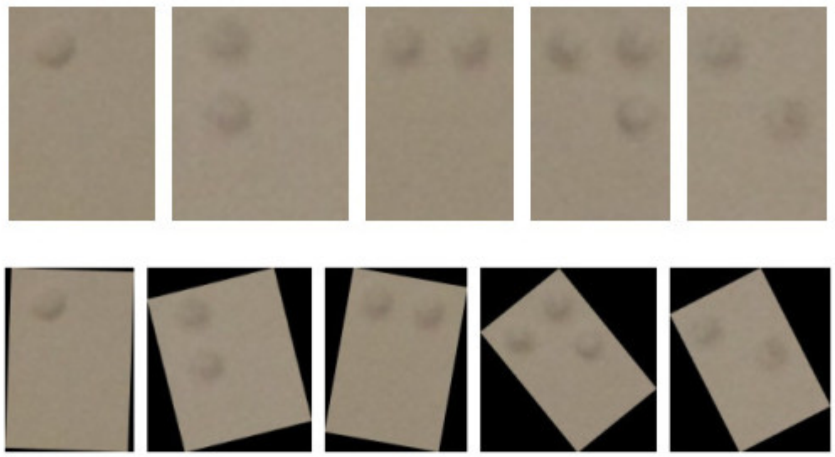

Figure 1: The samples of original images and rotated images. The images from left to right represent English letters A, B, C, D, E.

Pertaining to this project, we chose ResNet-18, ResNet-34, and ResNet-50 for the trials. The sizes of the images are not very big; thus, we test whether a deeper network can perform better than a shallower one. The structures of the chosen models will follow the structures in Table 1. We set 27 outputs classes for the last linear fully connected layer. We created the models from scratch using PyTorch framework, and only used the Dataset I for training, testing and validation.

The experiments were conducted using a laptop, the models were trained by GPU for time saving and efficiency. For meta-parameters, we set the learning rate as 0.001, each model takes 20 epochs of iterations. The loss function for model convergence is cross entropy loss function as shown in equation (9), *M* represents the number of classes, *y* represents a binary indicator (1 or 0) if a class label *c* is classified correctly for observation *o*, *p* stands for the predicted probability observation *o* of class *c*. The optimizer for models is Adam optimization algorithm.

$$L = -\sum_{c=1}^{M} y_{o,c} \log\left(p_{o,c}\right) \tag{7}$$

For evaluating the models, training and validation losses are used for evaluating the convergence of the models during training and validation. The accuracies are overall accuracies of the models. The model time consumptions are recorded for the purpose of comparisons. Precession, Recall, and F1 values are used as the metrics to show how models work:

$$Precision = \frac{TP}{TP+FP} \tag{8}$$

$$Recall = \frac{TP}{TP+FN} \tag{9}$$

$$F1_{measure} = \frac{2\times Recall \times Precision}{Recall+Precision} \tag{10}$$

The word-based ABCNet model was implemented by using ResNet-50 and Feature Pyramid Network (FPN) [12]. The detection branch capitalizes on RoIAlign with 5 feature maps as 1/8, 1/16, 1/32, 1/64 and 1/128 resolution for utilizing with input imag. Bezier-Align as 3 feature maps with 1/4, 1/8. And 1/16 ratios. The images of word-based dataset are collected from the internet public resources. Due to the time limitation, there are only 250 images in total, with 200 images as training set, 30 as validation set, and 20 as the test set.

4 RESULTS

The losses of ResNet-18 and ResNet-50 are converged rapidly from the second or third iteration, the loss of ResNet-34 starts converging from the 5-th iteration, the difference might be caused by the model initial initialization where the weights and the bias values are initialized into random values following the Gaussian distribution. Overall, the losses of all models are converging nicely, the training loss and validation loss of ResNet-18 dip to 0.0058 and 0.2114, ResNet-34 losses slash to 0.0365 and 0.2342, and for ResNet-50, the losses are dropped to 0.082 and 0.195.

Table 1: Results of multiple deep learning models

Model Type	Accuracy	Training Error	Validation Error	Time Used/secs
ResNet18	98.2143%	0.0058	0.2114	590.8
ResNet34	97.5%	0.0365	0.2342	1043
ResNet50	96.4286%	0.082	0.1950	3638

Table 2. Evaluation metrics for all three models with precision, recall and F1 score.

Model Type	Precision	Recall	F1
ResNet18	0.97	0.855	0.91
ResNet34	0.93	0.72	0.81
ResNet50	0.92	0.77	0.83

Table 1 shows the accuracies of three models. ResNet-18 has the highest accuracy 98.2143%, and accuracy of ResNet-34 has slightly down to 97.5%, and ResNet-50 has the lowest accuracy of 96.42%. As the model becomes deeper, it takes more time for training, 9 minutes for training ResNet-18, 17 minutes for ResNet-34 and one hour for ResNet-50.

Table 2 shows the evaluation metrics of the models. The metrics of precision, recall and F1 are used to measure the classification performance. ResNet-18 has the highest precision, recall and F1 value as 0.97, 0.855 and 0.91. The F1 values of ResNet-34 and ResNet-50 are 0.81 and 0.83 which are lower than 18 layers model, the performance of ResNet-34 and ResNet-50 are the same regardless of the 0.02 difference between these two models. ResNet-18 has a better performance compared to the deeper networks. This may be due to that the image sizes of the dataset are not very big, all images are resized in 52×52, noises were added into the images, which have similar sizes compared to braille dots, applying the data for training models, the deeper model might generate the errors from classifying the noises into braille characters. The ResNet models contain convolutional layers to prevent the images getting too small during the process of convolution. Another possible reason might be the deeper network contains more parameters than a shallow one. ResNet-50 has 2,355k parameters and ResNet-18 has 1,118k parameter only.

All the evaluation metrics are generated by using our own models, ResNet-18 was employed for a test of recognizing our own handwritten braille characters. we write the word “hello” onto a plain paper. The braille words are shown in Fig. 3, the image was taken by using a smartphone. After stored the image into a laptop, the words are segmented into five single characters. The segmented images are input into ResNet-18 for classification.

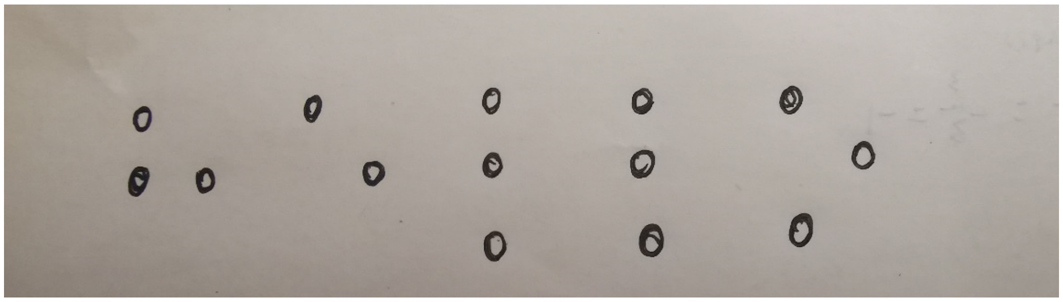

Figure 2: Our handwritten braille word “Hello”.

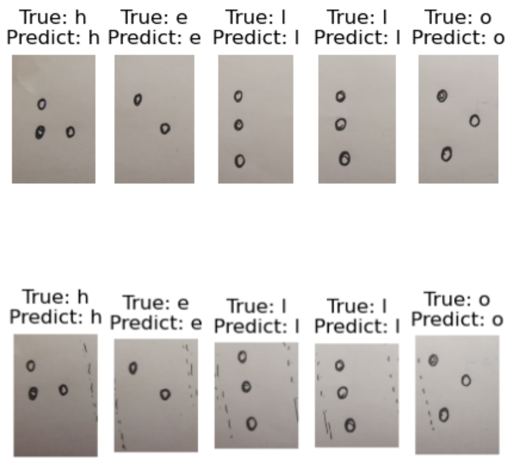

Figure 3: The output of character classification after segmented handwritten characters.

There are two tests of classification conducted as shown in Figure 4. The segmented characters in the first line are captured in absolutely vertical angle with no tilt, the characters in second line are captured with slightly tilted to the left, which are identified by using dash lines. The predicted labels and true labels of each characters are shown on the images. The results for both tests are with 100% accuracy, the characters are correctly identified as the corresponding English alphabets.

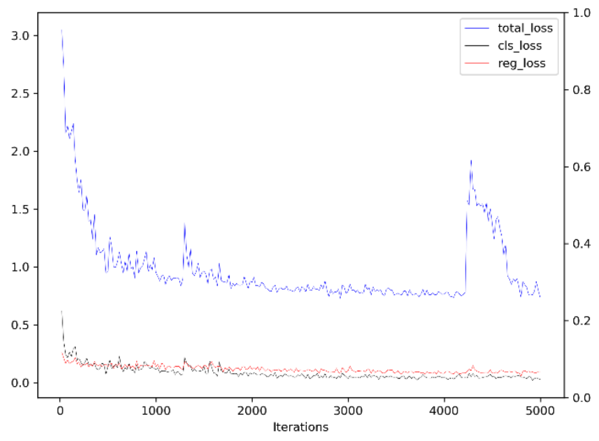

Figure 4: Loss curves for training ABCNet, *cls_loss* represents classification loss, *reg_loss* shows localization loss

After trained with the word-based braille dataset, ABCNet has 5,000 iterations of training. The loss curves are shown in Figure 5. The total loss is 0.73, the classification loss is 0.029, and localization loss is 0.6. Due to the amount of collected images is very limited, the final losses are acceptable.

Table 3: Evaluation Metrics for ABCNet in detection and e2e.

Model		Precision	Recall	H-mean (F1-score)
ABCNet	Detection only	0.83	1.0	0.9
	E2E only	0.41	0.5	0.45
FOTS	Detection only	0.55	0.64	0.71
	E2E only	0.27	0.35	0.39

Table 3 shows the final metrics. All the metrics have performed very well. The results of these metrics only have 0.41, 0.5, and 0.45, the reason why the final values have not reached ideal values is that the training data is not sufficient enough. A comparison on FOTS was conducted, which was trained with 1,000 epochs by using the same dataset. Compared to ABCNet, the detection and e2e results of FOTS are lower, and the time consumption for training FOTS using the same dataset is longer than ABCNet. The reason why FOTS gives worse results than ABCNet is that the iteration of training is not enough, if there are 5000 epochs as same as ABCNet, the results are much better.

5 CONCLUSION AND FUTURE WORK

Implementing character-based braille recognition using ConvNet is witnessed very efficiency. Different from traditional image processing techniques, deep neural networks use noises in the images to train the classifier, allow the model having the ability to detect and classify braille characters with high accuracy. As the traditional techniques limit the images in absolute vertical, ConvNet allows the images to rotate an angle, and classify the characters into correct corresponding letters. The design of residual block in ResNet models allows the losses to be controlled in deeper networks, this is proved by the test results, the training loss of ResNet-50 only increases 0.0762 compared to ResNet-18, the validation loss of ResNet-50 is less than ResNet-18. The overall accuracies of three tested ResNet models are very stable, only have 1%~2% difference. The word-based braille translator using ABCNet gives us the opportunity to translate the braille word.

There are few limitations of the project. The original plan is taken ResNet-101 into the controlled trails. But with the limitation of the hardware where the GPU has 6GB of memory, it is not enough to train a ResNet-101 with training batch size of 40, validation batch size of 10, and test batch size of 10. The second limitation is the results of the ACBNet are all implemented based on static images, we have not the tests in real time.

In future, we will continue collecting the word-based braille images and added them into the dataset to uplift the performance. Also, further experiments for real-time detection are needed, we will set up a suitable camera as the source of real-time input and implement the real-time scene text detection for translating the braille words from natural scenes [15,16,17,18,19, 20,21].

REFERENCES

[1] Atul Adya, Paramvir Bahl, Jitendra Padhye, Alec Wolman, and Lidong Zhou. 2004. A multi-radio unification protocol for IEEE 802.11 wireless networks. In Proceedings of the IEEE International Conference on Broadnets Networks (BroadNets'04). IEEE, Los Alamitos, CA, 210–217.

[2] Lisa Wong, Waleed H. Abdulla, and Stephan Hussmann. 2004. A software algorithm prototype for optical recognition of embossed Braille. In Proceedings of the International Conference on Pattern Recognition,586-589.

[3] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. In Proceedings of the IEEE. 86(11), 2278-2324.

[4] Saad Albawi, Tareq A. Mohammed, and Saad Al-Zawi. 2017. Understanding of a convolutional neural network. In Proceedings of International Conference on Engineering and Technology (ICET)

[5] Marina A. Mercioni and Stefan Holban. 2020. The most used activation functions: Classic versus current. In Proceedings of International Conference on Development and Application Systems (DAS). 141-145.

[6] Rafael C. Gonzalez. 2018. Deep convolutional neural networks. IEEE Signal Processing Magazine. 35(6), 79-87

[7] Matthew Y.W. Teow. 2017. Understanding convolutional neural networks using a minimal model for handwritten digit recognition. In Proceedings of IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS). 167-172

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR),770-778

[9] Minghui Liao, Pengyuan Lyu, Minghang He, Cong Yao, Wenhao Wu, and Xiang Bai. 2021. Mask TextSpotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence. 43(2), 532-548

[10] Yipeng Sun, Chengquan Zhang, Zuming Huang, Jiaming Liu, Junyu Han, and Errui Ding. 2018. TextNet: Irregular text reading from images with an end-to-end trainable network. In Proceedings of Asian Conference on Computer Vision, ACCV 2018. 83-99

[11] Wei Feng, Wenhao He, Fei Yin, Xuyao Zhang, and Chenglin Liu. 2020. TextDragon: An end-to-end framework for arbitrary shaped text spotting. In Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV).

[12] Yuliang Liu, Hao Chen, Chunhua Shen, Tong He, Lianwen Jin, and Linagwei Wang. 2020. ABCNet: Real-time scene text spotting with adaptive Bezier-curve network. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

[13] George G. Lorentz. 1953. Bernstein Polynomials. University of Toronto Press. Toronto, Canada

[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural Computation (1997) 9 (8): 1735–1780.

[15] Wei Qi Yan. 2021. Computational Methods for Deep Learning-Theoretic, Practice and Applications, Springer International Publishing.

[16] Jia Lu, Minh Nguyen, Wei Qi Yan. 2021. Sign language recognition from digital videos using deep learning methods, In Proceedings of International Symposium on Geometry and Vision (ISGV), 108-118.

[17] Minh Nguyen, Huy Le, Wei Qi Yan, Arpita Dawda. 2018. A vision aid for the visually impaired using commodity dual-rear-camera smartphones. In Proceedings of International Conference on Mechatronics and Machine Vision in Practice.

[18] Wei Qi Yan. 2019. Introduction to Intelligent Surveillance Surveillance Data Capture, Transmission, and Analytics, Springer International Publishing.

[19] Ricky Li, Minh Nguyen. 2017. Wei Qi Yan. Morse codes enter using finger gesture recognition. In Proceedings of International Conference on Digital Image Computing: Techniques and Applications (DICTA).

[20] Yang Zhang, Wei Qi Yan, Ajit Narayanan. A virtual keyboard implementation based on finger recognition. In Proceedings of International Conference on Image and Vision Computing New Zealand (IVCNZ).

[21] Guangyu Wang, Feng Liu, Wei Qi Yan. 2014. Braille for visual cryptography. In Proceedings of IEEE International Symposium on Multimedia (ISM), 275-276.