Discrimination between Handwritten and Computer-Generated Texts using a Distribution of Patch-Wise Font Features

Naoki Hamasaki, Kazuaki Nakamura, Naoko Nitta, and Noboru Babaguchi
Graduate School of Engineering, Osaka University, Yamadaoka 2-1, Suita, Osaka, 565-0871 Japan.
E-mail: hamasaki@nanase.comm.eng.osaka-u.ac.jp

***Abstract*—Recently developed deep learning techniques allow us to generate images of handwritten-like texts that closely resemble a target writer's actual handwriting. Although these models are applicable to useful systems (e.g. communication tools for hand-impaired people), they also could be misused for document forgery by a malicious user. To cope with this problem, in this paper, we propose a text-independent method for discriminating between the computer-generated texts (CGTs) and actual handwritten texts (HWTs). Our proposed method only takes a single text image and recognizes whether the image is a CGT or a HWT. Characters in HWT images have various shapes even when the same writer writes the same sentence. This property is difficult to perfectly mimic even by state-of-the-art CGT generation methods. To capture this difference between HWTs and CGTs, we use the distribution of patch-wise font features. The proposed procedure for discriminating HWTs and CGTs is as follows: First, we divide a given text image into several patches and classify each patch into one of pre-determined standard font classes. Then we compute the histogram of the standard fonts, which is finally fed into the recognizer that discriminates between HWTs and CGTs. In our experiments, the proposed method achieved more than 96% of discrimination accuracy, which demonstrates the effectiveness of the proposed method.**

I. Introduction

Recently, with the development of deep learning techniques such as autoencoders (AEs) [1], generative adversarial networks (GANs) [2], and recurrent neural networks with long short-term memory (RNN-LSTM) [3], there have been proposed a lot of methods for automatically generating images of a handwritten-like text that closely resembles a target writer's actual handwriting. These methods are useful both socially and academically; they are applicable to rich communication tools, especially for hand-impaired people, as well as offer a huge amount of training dataset for handwriting text recognition and transcription.

On the other hand, automatically generated text images could be misused for document forgery, e.g., forging a testament, receipt, and so on. This is a serious threat for our society since handwritten texts are widely used for identity verification even today. Hence, in this paper, we tackle the problem of discriminating between the images of the computer-generated texts (CGTs) and those of handwritten texts (HWTs) actually written by humans. Our proposed method takes an image of a single-line sentence consisting of several words as an input. We refer to such a single-line sentence as "text" in the remainder of this paper. We aim to make the proposed method text-independent because CGT generation methods allow us to obtain images of an arbitrary text.

In the past two decades, several studies have focused on the task of separating HWTs and machine-printed texts that are not handwritten-like ones [4], [5], [6]. Unlike them, our focusing CGTs are quite similar with HWTs owing to deep learning techniques; they are more difficult than the printed texts to separate from HWTs. In this sense, to the best of our knowledge, this is the first work that considers a threat of CGT-based document forgery and proposes a method for discriminating between CGTs and HWTs. This is the main contribution of this work.

This paper is organized as follows. After describing about the related work in Section II, we make a detailed explanation of our proposed method in Section III. Then we experimentally evaluated the performance of the proposed method in Section IV and finally conclude this paper in Section V.

II. Related Work

Our target task is somewhat related to signature verification, because the problem of forged texts has been partially considered in most of the existing signature verification studies. Hence, we first review them and clarify the difference between signature verification and our target task in Section II-A. After that, we describe about CGT generation methods in Section II-B in order to clarify the characteristics of CGTs generated by deep learning techniques.

A. Methods for Signature Verification

Signature verification is the task of judging whether a given signature data of a target person is actually obtained from him/herself or not. Methods for signature verification can be roughly divided into two categories: offline and online. In offline methods, the signature data is just an image. On the other hand, in online methods, a sequence of two-dimensional pen-tip locations is given as the data. For both categories, skilled forgeries are assumed as negative samples. Comparing a given data with the target person's template, the methods judge whether it is a skilled forgery or a genuine signature.

As an example of the offline methods, Bhattacharya et al. proposed to use a pixel matching technique [7]. In this method, a given signature image is first rotated and resized

so that its angle and scale are exactly matched to those of the template image. Then both the images are binarized and compared with each other pixel-by-pixel. If the number of pixels whose color is consistent in the two images exceeds a threshold, the given image is judged as genuine data. This method is computationally efficient but not so powerful in terms of verification accuracy due to its simplicity. To improve the performance, Hafemann et al. proposed to automatically learn a good feature representation by a convolutional neural network (CNN) [8]. In this method, a given image is first pre-processed (e.g., resized) and fed into the trained CNN to extract a feature vector. Then the feature is further fed into a support vector machine (SVM) to perform final verification process. Since CNNs are a recent trend due to its high performance, a lot of offline signature verification methods using CNNs have been proposed in the last five years [9], [10], [11]. In CNN-based methods, signature images first have to be resized to the fixed size. Maergner et al. regarded this point as a problem and proposed to use two kinds of graph-based representations called keypoint graph and inkball model, which are flexible in size [12]. With these representations, their method achieved a good performance.

For online methods, Sharma et al. proposed a Gaussian mixture model (GMM)-based descriptor to represent online signature data [13]. In their method, they first extract the pen-tip location, velocity, acceleration, and so on as temporally local features from a template signature data. With this process, the template is represented as a distribution of the local features. Next, the distribution is approximated by a GMM, whose parameter is used as the descriptor. Then, the same kind of local features are extracted from a given signature data, which are matched to the GMM in order to verify whether it is a genuine data or not. Instead of directly using a sequence of pen-tip locations, Diaz et al. converted it to a sequence of anthropomorphic robot poses that can reproduce the pen-tip location sequence, and extracted a new feature from the robot pose sequence, which is used as a feature for verification [14].

As reviewed above, there are a lot of offline and online methods for signature verification. However, most of them only consider the skilled forgeries written by another person; they do not consider the forged signature data automatically generated by deep networks. Moreover, a template data can be used in signature verification because a person's signatures always consist of the same characters. Unlike this, since our focus is a text-independent method for discriminating CGTs from HWTs, we cannot use the template whose original text is same with an input image. These are the main difference between our target task and the signature verification.

B. Methods for Generating CGTs

There are two types of CGT generation strategy: sentence-wise generation and character-wise generation. In sentence-wise generation, an image of a single-line text is generated at once. On the other hand, in character-wise generation, images of each characters are separately generated and then integrated into a single image of a sentence.

Sentence-wise generation is often employed for generating alphabetical text images. This is because neighboring characters tend to be continuously written in alphabet-based languages including English. A typical example of this kind of methods was proposed by Graves [15], where a sequence of two-dimensional pen-tip locations is generated by RNN-LSTM. Chung et al. extended this method by using variational RNN [16], which is a combination of a variational autoencoder [17] and RNN-LSTM, to achieve better generation performance. Aksan et al. further extended the variational RNN by introducing a variable representing the type of font style [18]. This allows us to generate CGTs with a specified font style. There are also several methods directly generating images of CGTs instead of pen-tip locations. The method of Haines et al. [19] first constructs a database of sample glyphs actually written by a target writer. Next, for each character in a given arbitrary text, the method selects the optimal glyph from the constructed database considering the neighboring characters' position, ligature, texture, and so on. Finally, the selected glyphs are connected into a single CGT image after a slight deformation.

Character-wise generation is suitable to the language in which each character tends to be independently written, such as Chinese. In Chinese, most characters can be regarded as a combination of limited kinds of components. Based on this property, Lin et al. proposed to construct a set of sample glyphs for each component from a training dataset and appropriately combine them to generate a handwritten-like image of each character [20]. Zong et al. also employed the similar strategy [21]. In their method, each Chinese character is decomposed into strokes, which are then deformed by affine transforms. After that, the deformed strokes are combined to generate a handwritten-like image of the character.

Some of the above methods always generate the exact same image for the same character. This is not the case with HWTs; the shapes of handwritten characters are at least slightly different with each other even if the same writer writes the same character. We refer to this property as *within-person variety* of handwriting, which can be an important clue for discriminating between HWTs and CGTs. Note that there are also some CGT generation methods employing a probability process for mimicking the within-person variety of handwriting. However, since the probability process is chosen in an ad hoc manner, the shape distribution of the characters generated by these methods is significantly different from that of HWTs. Hence, the shape distribution resulting from the within-person variety of handwriting is expected to be still a good feature.

III. Discrimination between HWTs and CGTs using Font Feature Distribution

In this section, we propose a method for discriminating between HWTs and CGTs. After describing the overview in Section III-A, we describe each step of the proposed method in detail in the subsequent subsections.

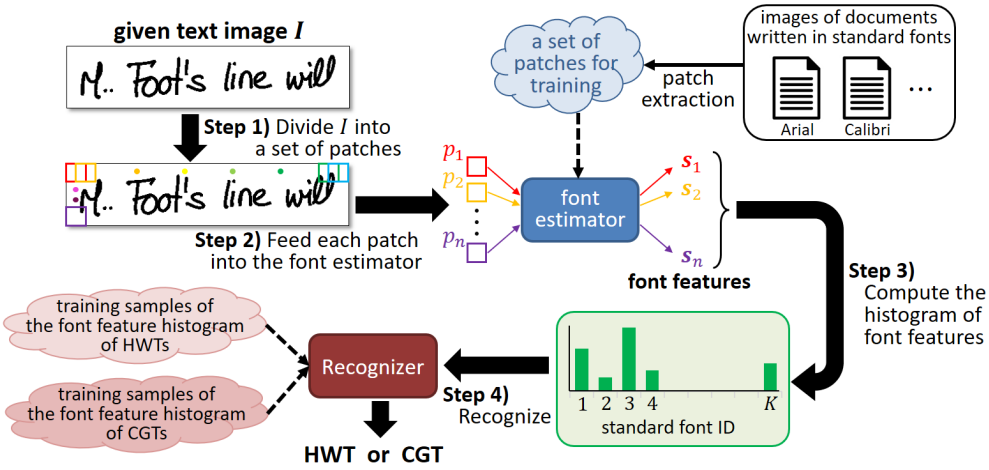

Fig. 1. Overview of the proposed method

A. Overview

As mentioned in Section II-B, the shape distribution of the characters in a given text image would be a good feature for our target task. We propose to extract this as follows: First, we divide a given text image into a set of local patches. Next, for each patch, we extract a local descriptor that represents the shape of the characters in the patch. Then we compute the histogram of the local descriptors like bag-of-visual-words approach, which is finally used as a feature vector of the given text image.

In the above method, it plays an important role which kind of local descriptor is employed in the second step. Of course, we should employ a descriptor that can well capture the shape of characters. In general, the shape of a character is determined by the following two factors: the kind of the character and a font style. Among the two, the local descriptor should only capture the latter. If the local descriptor captures the former, the quite similar histograms would be obtained from a HWT image and a CGT image whose original texts are same, which have less ability to discriminate between HWTs and CGTs. Based on the above consideration, we propose to use a font estimator for local descriptor extraction. The font estimator takes a local patch as an input and tries to classify the font style of the characters in the patch into one of the pre-determined standard fonts. With this font estimator, we can obtain the similarity between an input patch and each standard font, which is used as the local descriptor of the patch.

The overview of the proposed method is shown in Fig. 1, which is summarized as follows:

Step 1) Dividing a given image
Divide a given text image I into a set of patches $\{p_n \mid n = 1, \cdots, N\}$, where p_n is the n-th patch whose size is $w \times w$ pixels and N is the number of patches.

Step 2) Extracting font features
Feed each patch p_n into the font estimator and get its output $\boldsymbol{s}_n$, where $\boldsymbol{s}_n = (s_{n1} \ \cdots \ s_{nK})^\top$ is a K-dimensional vector each of whose elements s_{ni} indicates the similarity between the characters in p_n and the i-th standard font. Note that K is the number of the pre-determined standard fonts. In the remainder of this paper, we refer to $\boldsymbol{s}_n$ as a font feature.

Step 3) Computing font feature distribution
Compute the histogram of the font features based on $\{\boldsymbol{s}_n \mid n = 1, \cdots, N\}$. The computed histogram $\boldsymbol{v}$ represents the character shape distribution well.

Step 4) Recognition
Recognize whether I is a HWT image or a CGT image, using $\boldsymbol{v}$ as a feature vector.

Hereafter, we describe the details of steps 2, 3, and 4 in Section III-B, III-C, and III-D, respectively.

B. Training a Font Estimator

We design the font estimator as a CNN in the proposed method. To train the CNN, a lot of training samples are required. We collect them as follows: First, for each pre-determined standard font, we prepare several document images in which multi-line sentences are written in the font. Fig. 2 shows some examples of the prepared document images. Next, we extract a lot of patches from the document images, where the location of each patch is randomly set and the patch size is fixed to $w \times w$. In this process, patches with no characters (or almost white patches) are sometimes obtained. Since these patches are inappropriate as the training samples, we remove them as seen in Fig. 3. More specifically, we count the number of the black pixels m in each patch and calculate the ratio of m to the total number of pixels, i.e.,

$$R = \frac{m}{w^2} . \tag{1}$$

If the R is less than a certain threshold, we remove the patch.

Using the training dataset collected by the above procedure, we train a CNN-based font estimator. The network structure is shown in Fig.4, which is inspired by *Half DeepWriter* [22].

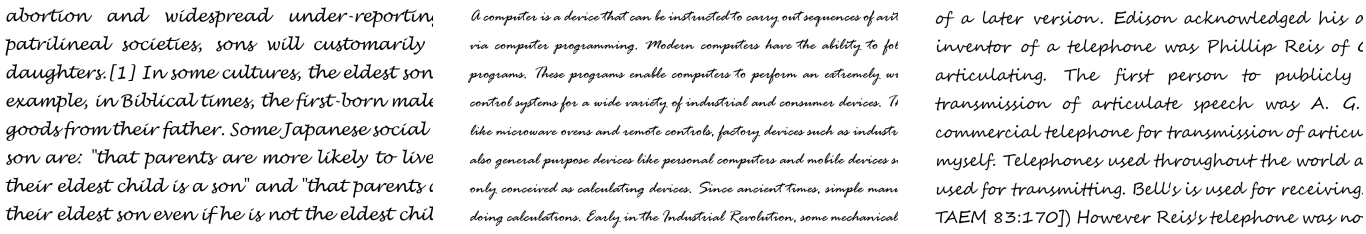

Fig. 2. Examples of the document images used to collect the training sample for the font estimator.

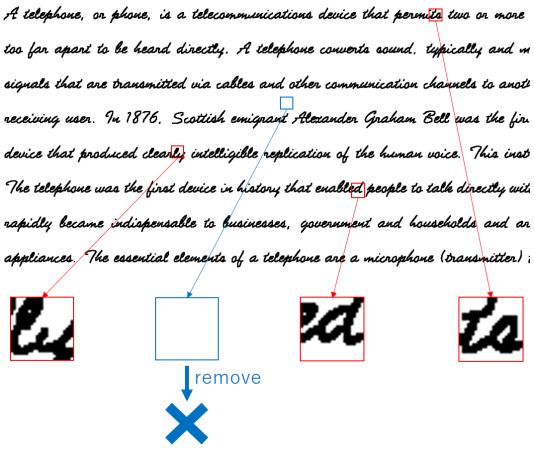

Fig. 3. Patch extraction and removal from the document images.

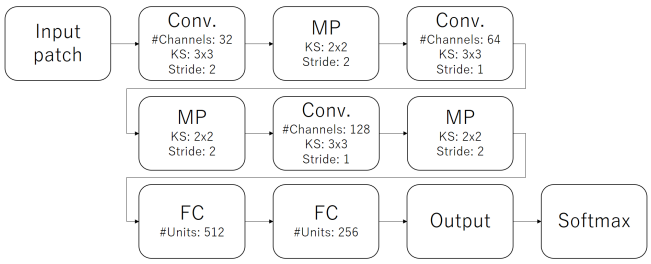

Fig. 4. Network structure of the font extimator.

In this figure, Conv. means a convolutional layer, MP means a max pooling layer, and FC means a fully-connected layer. In addition, #Channels and KS means the number and the kernel size of convolution filters, respectively, and #Units means the number of units in each FC layer. We employ ReLU as the activation function of the convolutional layers and FC layers. Moreover, we employ the dropout technique in FC layers, where the dropout ratio is set as 0.5. The number of units in the output layer is K. Applying the softmax function to the K-dimensional output vector, we can obtain $\boldsymbol{s}_n$ in the form of probability.

C. Computing Font Feature Distribution

Before describing how we compute the font feature histogram $\boldsymbol{v}$, we first describe how we divide an input text image into patches. In the proposed method, we slide a $w \times w$ window

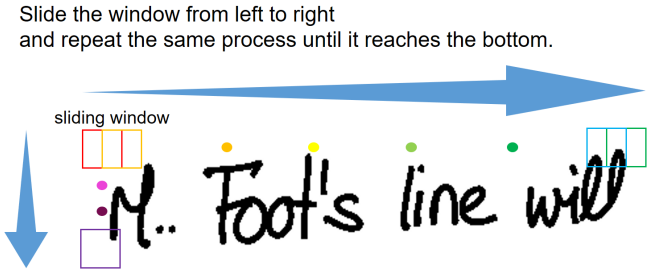

Fig. 5. The way to divide an input image into patches.

in the input image as shown in Fig. 5 and densely extract fixed-size patches. We set the stride of the sliding window as $\frac{w}{2}$, thus the neighboring patches are half-overlapped with each other.

All patches obtained by the above process are then fed into the trained font estimator, resulting in a font feature set $\{\boldsymbol{s}_n \mid n = 1, \cdots, N\}$. As mentioned in the previous section, we can obtain each $\boldsymbol{s}_n$ in the form of probability. This means

$$\forall i \in \{1, \cdots, K\} \quad s_{ni} \geq 0 \tag{2}$$

and

$$\sum_{i=1}^{K} s_{ni} = 1 \tag{3}$$

are satisfied for all $\boldsymbol{s}_n$. Using this property, we compute the font feature histogram $\boldsymbol{v}$ as

$$\boldsymbol{v} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{s}_n \ . \tag{4}$$

This is so called a soft histogram, which is computed by voting s_{ni} for i-th bin of the histogram and normalizing the final voting result.

D. Discrimination between HWTs and CGTs

To recognize whether the computed $\boldsymbol{v}$ is obtained from a HWT image or a CGT image, we employ a supervised machine learning technique again. More specifically, we employ a support vector machine (SVM) with a radial-basis function (RBF) kernel. To train the SVM, we collect a number of HWT images and CGT images, from which a font feature histogram $\boldsymbol{v}$ is extracted by the same process with Sections III-B and III-C. After that, using the extracted histograms as a training dataset, we train the SVM for discriminating between HWTs and CGTs.

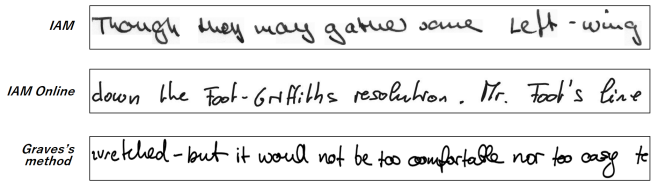

Fig. 6. Example images of HWT and CGT used in the experiments

IV. Experiments

To evaluate the performance of the proposed method, we conducted two experiments. In this section, we report the results of the experiments in detail.

A. Experimental Setup

In general, there are two approaches for acquiring HWT images: scanning papers and using touch-pen devices. We collected HWT images acquired by both approaches. For the former, we used *IAM Handwriting Database* (abbreviated as *IAM*) [23]. This dataset has 16,752 HWT images written by 657 writers that were acquired by scanning papers. HWTs in these images are written in various types of pens (e.g. color and thickness) and their background color is not always white. On the other hand, for the latter, we used *IAM On-Line Handwriting Database* (abbreviated as *IAM Online*) [24]. This dataset has 13,017 HWT images written by 217 writers that were acquired by using a touch-pen device. These images were drawn by connecting the pen-tip locations, which were acquired by the touch-pen device, with a black line of the fixed thickness. Hence, the character color and thickness are always same as well as the background color is always white.

For collecting CGT images, we used Graves's method [15], which is the most famous CGT generation method and whose source code is publicly available. To train Graves's generator, a lot of sequences of pen-tip locations are required as a training dataset. For this purpose, we used all data samples in *IAM Online*. Once the generator was trained, it can mimic any human writer's style only by initializing the generator's internal state with his/her actual handwriting data. This is a strength of Graves's method. Based on this property, we used the trained generator to obtain 10,100 CGT images that mimic the actual handwriting of 202 different writers in *IAM Online*. The character color and the background color of these images are always same. This is because the output of Graves's generator is a sequence of pen-tip locations, similar with *IAM Online*. For the thickness of characters, we adopted the following two settings. One is the fixed thickness, which is the same condition with *IAM Online*. The other is variable thickness, which is the similar condition with *IAM*. In this condition, we randomly set the character thickness for each CGT image. For both conditions, we separately generated 10,100 CGT images with the above procedure. We refer to these two datasets as *Fixed-thickness CGT* and *Variable-thickness CGT* in the remainder of this section. Fig. 6 shows an example image of HWTs in *IAM* and that in *IAM Online* as well as an example of CGTs generated by Graves's method.

TABLE I
Standard fonts used for training the font estimator.

ID	example	ID	example	ID	example
1		11		21	
2		12		22	
3		13		23	
4		14		24	
5		15		25	
6		16		26	
7		17		27	
8		18		28	
9		19		29	
10		20		30	

TABLE II
Accuracy of font estimation and that of the main task with various settings of w.

w	acc. of font estimation	acc. of the main task
32	75.3%	92.4%
40	82.1%	92.6%
48	84.9%	92.2%

For constructing the font estimator, we selected 30 kinds of handwritten-like fonts from *Microsoft Office 2016 fonts* and used them as the pre-determined standard fonts. This means $K = 30$ in our experiments. TABLE I shows example images of the selected fonts with their font IDs. Note that it is still unclear whether $K = 30$ is the optimal setting or not. Theoretically, larger K allows us to better represent the character shape distribution of text images, resulting in higher discrimination accuracy between HWTs and CGTs. We will experimentally examine the relationship between K and the accuracy in our future work.

B. Setting of Patch Size

We first examined the effect of the patch size w in order to appropriately set its value.

Since a smaller patch is less informative, the performance of the font estimator is expected to degrade with smaller w. This might make a negative effect on the performance of the main task, i.e., discrimination between HWTs and CGTs. On the other hand, with larger w, only a fewer number of patches can be extracted from an input text image. This also might make a negative effect on the performance of the main task. To test these hypotheses and find a good value of w, we evaluated the accuracy of the font estimation and that of the main task. For this purpose, we used 500 HWT images from *IAM* and 500 images from *Variable-thickness CGT* as training samples for the main task. To evaluate the accuracies, we used other 500 HWT images and 500 CGT images from the same dataset as test samples. For constructing the font estimator, we prepared 2,500 patches as training samples for each of the 30 standard

TABLE III
DISCRIMINATION ACCURACY ON *IAM* AND *Variable-thickness CGT*

		Recognizied label	
		HWT	CGT
Ground truth label	HWT	97.7%	2.3%
	CGT	3.7%	96.3%

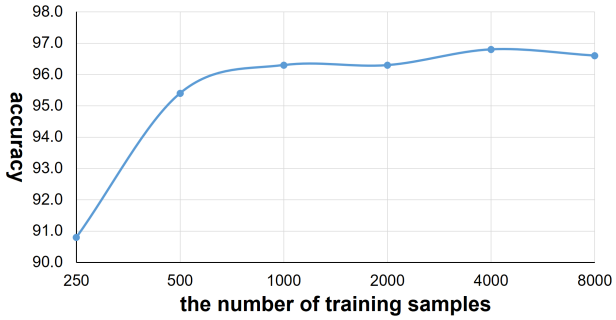

Fig. 7. Discrimination accuracy with various number of training samples on *IAM* and *Variable-thickness CGT*

TABLE IV
DISCRIMINATION ACCURACY ON *IAM Online* AND *Fixed-thickness CGT*

		Recognizied label	
		HWT	CGT
Ground truth label	HWT	100.0%	0.0%
	CGT	0.4%	99.6%

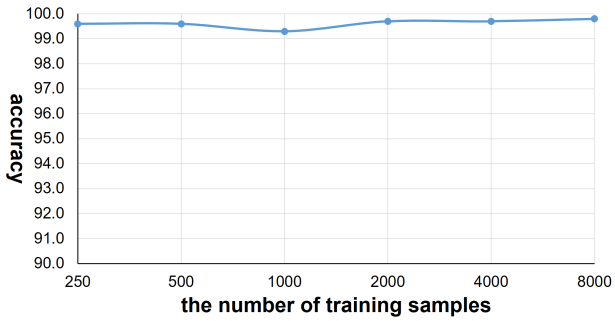

Fig. 8. Discrimination accuracy with various number of training samples on *IAM Online* and *Fixed-thickness CGT*

fonts. The result is shown in TABLE II.

As seen in TABLE II, higher performance of the font estimation is achieved with larger w. This is consistent with the above hypothesis. On the other hand, the performance of the main task is almost same for any settings of w. This is because the negative effect of smaller patches and the positive effect of the more number of patches are balanced with each other. This result indicates that the setting of w is not so important in the proposed method. Hence, we used the setting of $w = 32$ in the subsequent experiments, considering that the processing time becomes shorter with smaller w.

Note that the optimal value of w might depend on the size of the characters in a given text image. This will also be experimentally examined in our future work.

C. Results and Discussion

1) IAM vs. Variable-thickness CGT: Using the above setting of w, we evaluated the performance of discriminating between HWTs in *IAM* and CGTs *in Variable-thickness CGT*.

As previously mentioned, the character color and the background color of HWT images in *IAM* have a variety, whereas those of CGT images in *Variable-thickness CGT* are fixed. To eliminate this difference, we binarized each image as a preprocessing. After the preprocessing, we equally divided *IAM* and *Variable-thickness CGT* into two subsets, one of which was used as a training set and the other was used as a test set. Note that we made the training set and the test set not contain the same writer's images. The result is shown in TABLE III, in which more than 96% of accuracy is achieved for both HWTs and CGTs.

To check the effect of the size of the training set, we variously changed the number of training samples and evaluated the performance under each setting. Fig. 7 shows the result, where the almost same performance is achieved in any cases of using more than 1,000 training samples. This indicate that 1,000 training samples are enough for constructing a good recognizer that can successfully discriminate between HWTs and CGTs.

2) IAM Online vs. Fixed-thickness CGT: Next, we conducted another experiment using HWTs in *IAM Online* and CGTs in *Fixed-thickness CGT*.

In this experiment, we did not perform any preprocessing. Similar with the previous experiment, the datasets were equally divided into two subsets, one of which was used as a training set and the other was used as a test set. TABLE IV shows the result, in which more than 99% of accuracy is achieved for both HWTs and CGTs. Moreover, the effect of the size of the training set was also evaluated, whose result is shown in Fig. 8. In this result, a very high accuracy is achieved even in the case of using only 250 training samples. This indicates that HWTs in *IAM Online* is easier than those in *IAM* to discriminate from CGTs.

3) Comparison of Font Feature Distributions: Finally, we qualitatively compared the distribution of the font features in *IAM*, that in *IAM Online*, and that in *Variable-thickness CGT*. To this end, we computed the average of the font feature histogram $\boldsymbol{v}$ for each dataset. Fig. 9 shows the result. As seen in this figure, the averaged distribution of *IAM* and that of *IAM Online* have a common characteristic. For instance, both datasets have a lot of patches classified into the font ID 1, 9, 10, 13, 19, and 28. In contrast, *Variable-thickness CGT* rarely has these kinds of patches; instead, it has a lot of patches classified into the font ID 4, 5, 6, 20, and 24, which rarely appear in *IAM* and *IAM Online*. From this result, we can conclude that HWT images have a certain common characteristic regardless of its acquiring approach, which is not common in CGT images.

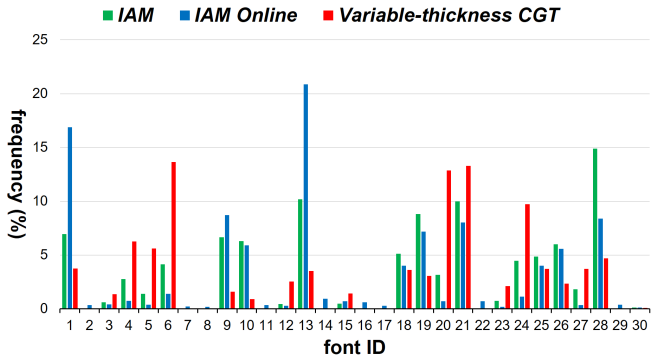

Fig. 9. Comparison of font feature distributions of *IAM*, *IAM Online*, and *Variable-thickness CGT*

V. Conclusion

In this paper, we proposed a method for discriminating between HWTs and CGTs using the distribution of patch-wise font features. Human's actual handwriting has within-person variety, which is difficult to perfectly mimic even by state-of-the-art CGT generation methods. This means the distribution of patch-wise font features is a good feature for our target task. We demonstrated this through several experiments, in which more than 96% of discrimination accuracy was obtained. Moreover, we qualitatively analyzed the difference between the font feature distribution of HWT images and that of CGT images. We found from the result of the analysis that HWT images have a common characteristic regardless of its acquiring approach.

We only focused on Graves's CGT generation method in our experiments. However, there are many other CGT generation methods proposed recently. It is an important future work to conduct further experiments using these CGT generation methods with various hyper parameters in order to examine the generality of the proposed method.

This work was supported by JSPS KAKENHI Grant Number JP16H06302.

References

[1] G. Hinton and R. Salakhutdinov: "Reducing the Dimensionality of Data with Neural Networks," Science, Vol.313, No.5786, pp.504–507, 2006.
[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio: "Generative Adversarial Nets," in Proc. of 27th Int'l Conf. on Neural Information Processing Systems, Vol.2, pp.2672–2680, 2014.
[3] A. Graves and J. Schmidhuber: "Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures," Neural Networks, Vol.18, No.5–6, pp.602–610. 2005.
[4] J.K. Guo and M.Y. Ma: "Separating Handwritten Material from Machine Printed Text using Hidden Markov Models," in Proc. of 6th Int'l Conf. on Document Analysis and Recognition, pp.439–443, 2001.
[5] K. Zagoris, I. Pratikakis, A. Antonacopoulos, B. Gatos, and N. Papamarkos: "Distinction between Handwritten and Machine-Printed Text Based on the Bag of Visual Words Model," Pattern Recognition, Vol.47, No.3, pp.1051–1062, 2014.
[6] S. Das, P. Banerjee, B. Seraogi, H. Majumder, S. Mukkamala, R. Roy, and B.B. Chaudhuri: "Hand-Written and Machine-Printed Text Classification in Architecture, Engineering & Construction Documents," in Proc. of 16th Int'l Conf. on Frontiers in Handwriting Recognition, pp.546–551, 2018.
[7] Bhattacharya, P. Ghosh, and S. Biswas: "Offline Signature Verification using Pixel Matching Technique," Procedia Technology, Vol.10, pp.970–977, 2013.
[8] L.G. Hafemann, R. Sabourin, and L.S. Oliveira: "Learning Features for Offline Handwritten Signature Verification using Deep Convolutional Neural Networks," Pattern Recognition, Vol.70, pp.163–176, 2017.
[9] Z. Xing, F. Yin, Y. Wu, and C. Liu: "Offline Signature Verification using Convolution Siamese Network," in Proc. of 9th Int'l Conf. on Graphic and Image Processing, 9 pages, 2017.
[10] A. Rehman, S.U. Rehman, Z.H. Babar, M.K. Qadeer, and F.A. Seelro: "Offline Signature Recognition and Verification System using Artificial Neural Network," University of Sindh Journal of Information and Communication Technology, Vol.2, No.1, pp.73–80, 2018.
[11] S. Lai and L. Jin: "Learning Discriminative Feature Hierarchies for Off-Line Signature Verification," in Proc. of 16th Int'l Conf. on Frontiers in Handwriting Recognition, pp.175–180, 2018.
[12] P. Maergner, N. Howe, K. Riesen, R. Ingold, and A. Fischer: "Offline Signature Verification via Structural Methods: Graph Edit Distance and Inkball Models," in Proc. of 16th Int'l Conf. on Frontiers in Handwriting Recognition, pp.163–168, 2018.
[13] A. Sharma and S. Sundaram: "Histogram-Based Matching of GMM Encoded Features for Online Signature Verification," in Proc. of 16th Int'l Conf. on Frontiers in Handwriting Recognition, pp.169–174, 2018.
[14] M. Diaz, M.A. Ferrer, and J.J. Quintana: "Robotic Arm Motion for Verifying Signatures," in Proc. of 16th Int'l Conf. on Frontiers in Handwriting Recognition, pp.157–162, 2018.
[15] A. Graves: "Generating Sequences with Recurrent Neural Networks," arXiv:1308.0850, 43 pages, 2014.
[16] J. Chung, K. Kastner, L. Dinh, K. Goel, A.C. Courville, and Y. Bengio: "A Recurrent Latent Variable Model for Sequential Data," in Proc. of 28th Int'l Conf. on Neural Information Processing Systems, Vol.2, pp.2980–2988, 2015.
[17] D.P. Kingma and M. Welling: "Auto-Encoding Variational Bayes," in Proc. of 2nd Int'l Conf. on Learning Representations, pp.1–14, 2014.
[18] E. Aksan, F. Pece, and O. Hilliges: "DeepWriting: Making Digital Ink Editable via Deep Generative Modeling," in Proc. of 2018 CHI Conference on Human Factors in Computing Systems, 14 pages, 2018.
[19] T.S.F. Haines, O.M. Aodha, and G.J. Brostow: "My Text in Your Handwriting," ACM Trans. on Graphics, Vol.35, No.3, 19 pages, 2016.
[20] J. Lin, C. Hong, R. Chang, Y. Wang, and S. Lin: "Complete Font Generation of Chinese Characters in Personal Handwriting Style," in Proc. of 34th IEEE Int'l Performance Computing and Communications Conf., 5 pages, 2015.
[21] A. Zong and Y. Zhu: "StrokeBank: Automating Personalized Chinese Handwriting Generation," in Proc. of 28th AAAI Conf. on Artificial Intelligence, pp.3024–3029, 2014.
[22] L. Xing and Y. Qiao: "DeepWriter: A Multi-Stream Deep CNN for Text-Independent Writer Identification," in Proc. of 15th Int'l Conf. on Frontiers in Handwriting Recognition, pp.584–589, 2016.
[23] U. Marti and H. Bunke: "A Full English Sentence Database for Off-Line Handwriting Recognition", in Proc. of 5th Int'l Conf. on Document Analysis and Recognition, pp.705–708, 1999.
[24] M. Liwicki and H. Bunke: "IAM-OnDB — an On-Line English Sentence Database Acquired from Handwritten Text on a Whiteboard", in Proc. of 8th Int'l Conf. on Document Analysis and Recognition, pp.956–961, 2005.