Study on Generative Adversarial Network for Handwritten Text

Bo Ji

Zhejiang University, China

Keywords: GANs, Handwritten Text, HWGANs.

Abstract: Generative adversarial networks (GANs) has proven hugely successful in variety of applications of image processing. However, generative adversarial networks for handwriting is relatively rare somehow because of difficulty of handling sequential handwriting data by Convolutional Neural Network (CNN). In this paper, we propose a handwriting generative adversarial network framework (HWGANs) for synthesizing handwritten stroke data. The main features of the new framework include: (i) A discriminator consists of an integrated CNN-Long-Short-Term-Memory (LSTM) based feature extraction and a Feedforward Neural Network (FNN) based binary classifier; (ii) A recurrent latent variable model as generator for synthesizing sequential handwritten data. The numerical experiments show the effectivity of the new model. Moreover, comparing with sole handwriting generator, the HWGANs synthesize more natural and realistic handwritten text.

1. Introduction

Learning generative sequential models is a long-standing machine learning challenge and historically in the domain of dynamic Bayesian networks (DBNs) such as Hidden Markov Model (HMMs), then further handled by Recurrent Neural Network. However, it has been observed that the synthesized texts of the previous models are not as natural as human-being's normal written text.

Besides, generative adversarial networks (GANs) are a merging technique proposed in 2014 [4] to learn deep representation without numerous annotated training data. This architecture achieves the generation of high reality. Therefore, a common analogy is to think of bringing the idea of GANs into handwritten text generation of digital ink in order to achieve more realistic handwritten text than handwritten generator alone.

In this paper, we propose a generative adversarial architecture to generate realistic handwritten strokes. The following bulleted points summarize our key contributions.

We design a discriminator consisting of a compact CNN-LSTM based feature extraction, followed by an auxiliary feedforward neural network to distinguish between genuine and forgery hand-writing.

We construct an adversarial architecture by combining the designed discriminator and the handwritten generator proposed by Alex Grave [1], referred as Alex's Generator throughout the whole remaining paper.

We use HWGANs to obtain more realistic English handwritten text than Alex's Generator.

2. Related Work

Variational RNN (VRNN) Handwritten Generator: VRNN [8] explores the inclusion of latent random variables into the hidden state of an RNN by combining a recurrent variational autoencoder (VAE), which is effective modeling paradigm to recover complex multimodal distributions over the non-sequential data space [9]. Whereas their synthesized handwritten data are not realistic enough.

Alex Grave's Handwritten Generator: Alex Grave proposed an RNN based generator model to mimic handwriting data [1]. For each timestamp, Alex's Generator encodes prefix sampled path to produce a set of parameters of a probability distribution of next stroke point, then sample the next stroke point given this distribution. There are two variants of Alex's Generator, i.e., handwritten

predictor and handwritten synthesizer, where the later one has the capability to synthesize handwritten text for given text.

3. Proposed Adversarial Approach

In this section, we present our proposed adversarial approach for digital ink handwritten text generation. As standard architectures of GANs, HWGANs comprise a discriminator D and a generator G. For a more rigorous description, let us introduce the notations used throughout the whole remaining paper. Denote $H = \{S_1, S_2, \dots, S_n\}$ as a handwritten text instance, which consists of a sequence of strokes, where S_i refers as the i-th stroke of current handwritten instance. One stroke S comprises a sequential stroke points, i.e., $S = \{(x_1, y_1, s_1); (x_2, y_2, s_2); \dots; (x_m, y_m, s_m)\}$, where (x_i, y_i, s_i) is i-th stroke point with (x_i, y_i) as its coordinate axis and $s_i \in \{0, 1\}$ as pen up/down status respectively to indicate the starting or ending stroke point.

3.1 Discriminator

Discriminator D is essentially a binary classifier to predict handwritten text of digital ink as a forgery or not. The CNN-LSTM model consists of a CNN and an LSTM, whose configurations are displayed in Table 1. CNN serves to encode extracted path signature feature into a matrix, since the input 3D tensor possesses variable widths, consequently results in that the output matrix of CNN possesses variable columns. Then each column of this encoded matrix is further fed into an LSTM cell sequentially, an FNN is assembled on the top of last output of the LSTM to calculate the probability of current instance as genuine written text. Thus, loss function of discriminator is naturally selected as binary cross entropy loss. The whole procedure is shown in Figure 1.

Table 1: Architecture Configuration of CNN-LSTM Model, where input shape, kernel shape and output shape are in the manners of (width, height, channel), (# of kernels, mask, stride) and (width, height, channel) respectively. The stride can either be a single number for both width and height stride, or a tuple (sW, sH) representing width and height stride separately.

	Input Shape	Kernel Shape	Output Shape	Hidden Size
Conv1	(W_I , 128, 7)	(32, 3, 1)	(W_I , 128, 32)	-
AvgPool	(W_I , 128, 32)	(-, 2, 2)	(W_I =2, 64, 32)	-
Conv2	(W_I =2, 64, 32)	(64, 3, 1)	(W_I =2, 64, 64)	-
AvgPool	(W_I =2, 64, 64)	(-, 2, 2)	(W_I =4, 32, 64)	-
Conv3	(W_I =4, 32, 64)	(128, 3, 1)	(W_I =4, 32, 128)	-
Conv4	(W_I =4, 32, 128)	(256, 3, (1, 2))	(W_I =4, 16, 256)	-
AvgPool	(W_I =4, 16, 256)	(-, 2, 2)	(W_I =8, 8, 256)	-
Conv5	(W_I =8, 8, 256)	(128, 3, (1, 2))	(W_I =8, 4, 128)	-
Conv6	(W_I =8, 4, 128)	(256, 3, (1, 2))	(W_I =8, 2, 256)	-
AvgPool	(W_I =8, 2, 256)	(-, 2, 2)	(W_I =16, 1, 256)	-
LSTM	256	-	-	256
FC1	256	-	-	128
FC2	128	-	-	1

Figure 1: Diagram of CNN-LSTM model.

3.2 Generator

Generator G is basically stacked LSTMs with sampling layers for generating strokes points from a certain initial stroke point. In this paper, we consider two architectures of generator designed for prediction and synthesis tasks, denoted as G_p and G_s proposed by Alex [1], illustrated as Figure 2. The prediction generator G_p only requires the network to generate random handwritten strokes, while the synthesis G_s generator further asks the model to produce handwriting texts given prescribed text of user.

Although these two generators are used in different scenarios, they share numerous things in common. First of all, both of them are stacked LSTMs with outputs as parameters of an appropriate probability distribution to predict the status of next stroke point given current generated stroke path. For this probability distribution, Alex [1] proposed to use a mixture of bivariate Gaussian distribution associated with a Bernoulli distribution to represent the distribution of pen up/down. Our model incorporates the same idea. Secondly, both generators update their parameters by maximizing log-likelihood of generated stroke sequences. During the adversarial training, they utilize the signal from the discriminator described in Section 3.1 to adjust network parameters. Thirdly, both G_p and G_s use biased sampling [1] to achieve higher quality of handwriting generation.

Different from the prediction generator G_p, the inputs of synthesis generator G_s requires not only the stroke point data but also the text information, since the synthesis network needs to know what texts to generate. The text is a one-hot vector processed from the input string, and input to LSTM layers with window parameters to constrain the synthesis network.

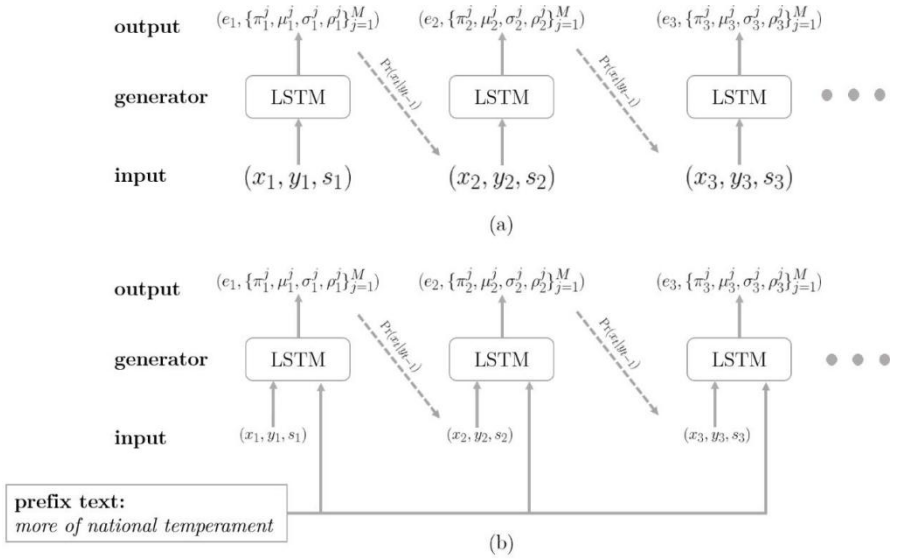

Figure 2: (a) Diagram of handwritten generator for prediction G_p; (b) Diagram of handwritten generator for synthesis G_s; where M is the number of Gaussian distributions with parameters as $e, \pi, \mu, \sigma, \rho$. .

4. Experiments

In this section, we numerically demonstrate the superiority of the adversarial network we proposed than Alex's Generator.

4.1 Experimental Settings

Both of the prediction network G_p and synthesis network G_s are trained on IAM dataset [10], which contains 80 distinct characters. Each stroke point of data possesses x-axis and y-axis offset from the previous stroke point and a binary end-of-stroke indicator. Architecture configuration details of discriminator D and two generators G_p, G_s are shown in Table 1 to 3 respectively.

It is widely acknowledged that training GANs is relatively difficult [11, 6], and we employ multiple tricks to train GANs more effectively. Firstly, we normalize input features between -1 and 1. Secondly, label smoothing technique is applied to both generator and discriminator during adversarial training [6]. Besides these, discriminator D keeps trained with positive samples from IAM dataset and negative samples from generators G_s and G_p supervisedly during adversarial training.

Table 2: Architecture Configuration of LSTM-based Generator for Prediction.

	Input Shape	Hidden Size
LSTM1	3	512
LSTM2	515	256
LSTM3	259	512
FC	1280	121

Table 3: Architecture Configuration of LSTM-based Generator for Synthesis.

	Input Shape	Hidden Size
LSTM1	57	512
LSTM2	569	512
FC1	512	15
FC2	512	121

4.2 Numerical Results

In this section, experimental results are present to further demonstrate the advantages of our HWGANs to Alex's Generator. In general, the HWGANs can produce competitive handwritten samples in prediction task, more promisingly, it can generate handwritten samples of which stroke points tend to distribute more spatially uniformly than those sampled from Alex's Generator, and the font styles of handwritten text from HWGANs are more neat and sometimes more diverse than those of Alex's Generator.

We first describe the performance of our model on prediction tasks compared with Alex's Generator. The handwriting instances are sampled with bias as 3.0. As shown in Figure 3, handwriting samples produced by HWGANs are very competitive to that of Alex, which demonstrates the excellent abilities of learning handwriting structure of GANs. The samples from two networks are slightly different in font styles.

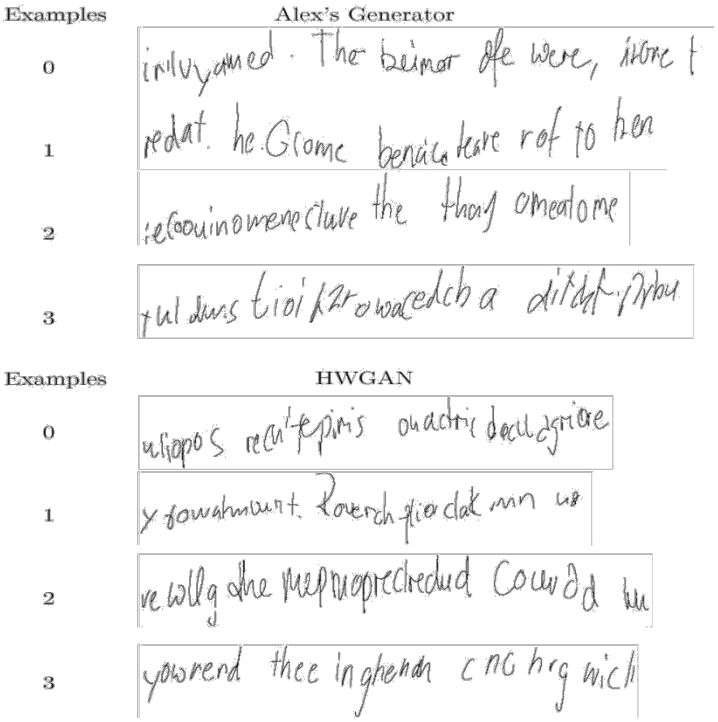

Figure 3: Prediction handwriting samples randomly generated by Alex's Generator and HWGANs. The top line in each block is generated by Alex's Generator, while the others are from HWGANs.

We then compare our model with Alex's Generator in synthesis task. As an overview, the results of both models are shown in Figure 4. Generally speaking, we can see that HWGANs may produce more realistic, more natural and more neat handwritten texts than Alex's Generator. More detailed comparisons are followed in the remaining section.

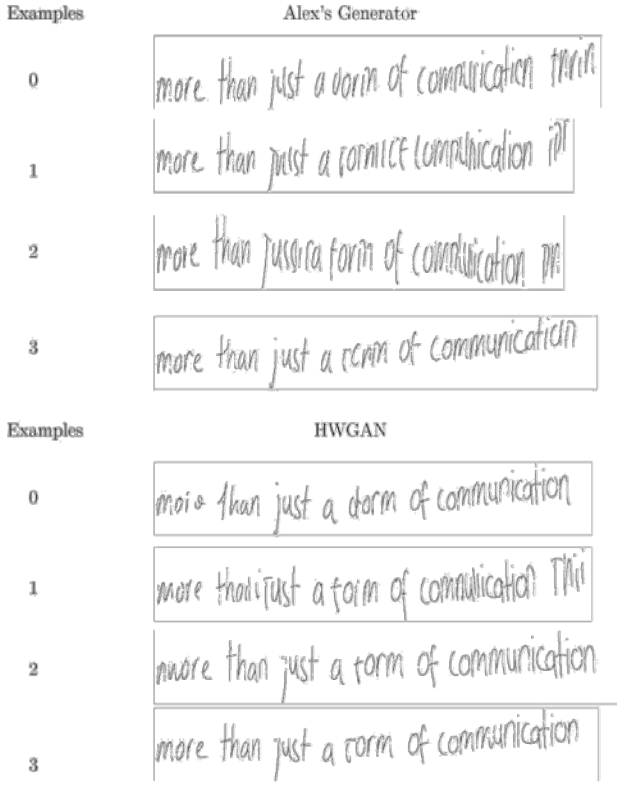

Figure 4: Generated handwritten texts by Alex's Generator and HWGANs. The top block is from Alex's Generator, the bottom one is from HWGANs.

Given that the value of bias used in biased sampling has a large impact on the results, we next test both models under different sampling biases. In Figure 5, samples are from two networks under sampling biases as 4, 5, 7 and 10 respectively. The results of HWGANs distribute almost more spatially uniformly, and neat on every bias. On the other hand, it seems that HWGANs may produce more diverse handwritten text styles, like the “A”.

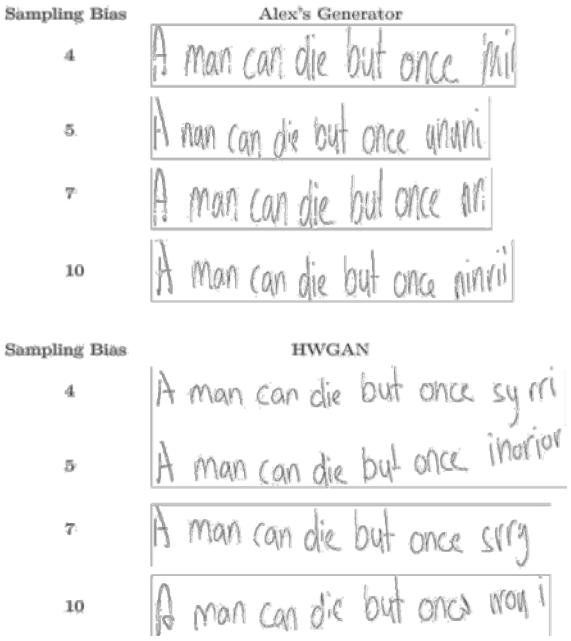

Figure 5: Synthesis handwriting data from Alex’s Generator and HWGANs using different sampling bias. The top block is sampled from Alex’s Generator, the bottom one is from HWGANs.

Besides the above, we also investigate both models based on different prescribed sentences with a fixed sampling bias as 5.0, and present the results in Figure 6, which further proves the advantages of our model observed on previous experiments are consistent.

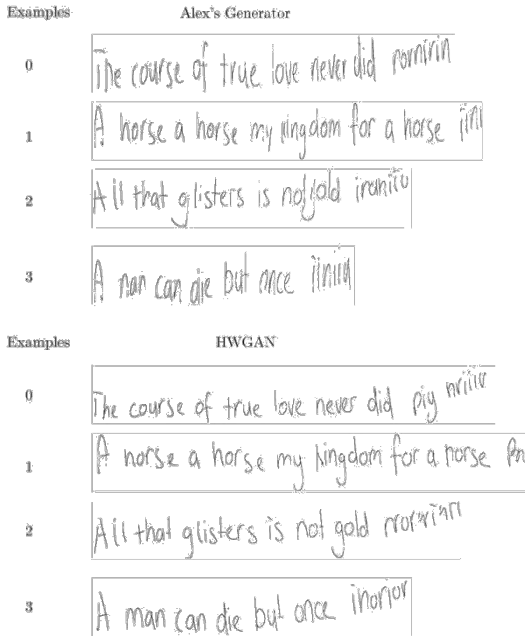

Figure 6: Synthesis handwriting data from Alex’s Generator and HWGANs given different characters. The top block is sampled from Alex’s Generator, and the bottom block is generated from HWGANs.

5. Conclusion

We present a new Generative Adversarial Network architecture HWGANs for synthesizing hand-written text of digital ink. The method comprises two main components, a discriminator consisting a PSF feature extractor, followed by a CNN-LSTM binary classifier to distinguish realistic and forgery handwritten data, and a generator following the architectures of Alex Grave's to produce either random handwritten characters or sentences with prefixed text. In particular, the numerical results show that HWGANs are generally better than Alex's Generator in terms of uniformness of spatial distribution of stroke points, neatness and diversity of font style. This phenomenon illustrates that adversarial training is beneficial for generating more realistic handwritten text data.

References

[1] Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv: 1308. 0850, 2013.

[2] Emre Aksan, Fabrizio Pece, and Otmar Hilliges. Deepwriting: Making digital ink editable via deep generative modeling. CoRR, abs/1801.08379, 2018.

[3] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. CoRR, abs/1506.02216, 2015.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672-2680, 2014.

[5] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil a Bharath. Generative adversarial networks: An overview. IEEE Signal Processing Magazine, 35 (1): 53-65, 2018.

[6] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In Advances in neural information processing systems, pages 2234-2242, 2016.

[7] Eloi Alonso, Bastien Moysset, and Ronaldo Messina. Adversarial generation of handwritten text images conditioned on sequences. arXiv preprint arXiv:1903.00277, 2019.

[8] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In Advances in neural information processing systems, pages 2980-2988, 2015.

[9] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv: 1312. 6114, 2013.

[10] U-V Marti and Horst Bunke. The iam-database: an English sentence database for offline hand-writing recognition. International Journal on Document Analysis and Recognition, 5(1):39-46, 2002.

[11] Martin Arjovsky and Leon Bottou. Towards principled methods for training generative adversarial networks. arXiv:1701.04862, 2017.