DOI: 10.1111/cgf.142621
EUROGRAPHICS 2021 / N. Mitra and I. Viola
(Guest Editors)
Volume 40 (2021), Number 2

Write Like You: Synthesizing Your Cursive Online Chinese Handwriting via Metric-based Meta Learning

Shusen Tang, Zhouhui Lian†

Wangxuan Institute of Computer Technology, Peking University, Beijing, P.R. China
Center For Chinese Font Design and Research, Peking University, Beijing, P.R. China
{tangshusen, lianzhouhui}@pku.edu.cn

Abstract

In this paper, we propose a novel Sequence-to-Sequence model based on metric-based meta learning for the arbitrary style transfer of online Chinese handwritings. Unlike most existing methods that treat Chinese handwritings as images and are unable to reflect the human writing process, the proposed model directly handles sequential online Chinese handwritings. Generally, our model consists of three sub-models: a content encoder, a style encoder and a decoder, which are all Recurrent Neural Networks. In order to adaptively obtain the style information, we introduce an attention-based adaptive style block which has been experimentally proven to bring considerable improvement to our model. In addition, to disentangle the latent style information from characters written by any writers effectively, we adopt metric-based meta learning and pre-train the style encoder using a carefully-designed discriminative loss function. Then, our entire model is trained in an end-to-end manner and the decoder adaptively receives the style information from the style encoder and the content information from the content encoder to synthesize the target output. Finally, by feeding the trained model with a content character and several characters written by a given user, our model can write that Chinese character in the user's handwriting style by drawing strokes one by one like humans. That is to say, as long as you write several Chinese character samples, our model can imitate your handwriting style when writing. In addition, after fine-tuning the model with a few samples, it can generate more realistic handwritings that are difficult to be distinguished from the real ones. Both qualitative and quantitative experiments demonstrate the effectiveness and superiority of our method.

CCS Concepts

• ***Computing methodologies*** → *Computer vision tasks; Computer graphics; Neural networks; Learning latent representations;*

1. Introduction

As we know, reading and writing play extremely important roles in human life, which correspond to inputting information from the world and outputting information to the world, respectively. So the question of how to empower machine reading (i.e., character recognition) and writing (i.e., character generation) skills has attracted intensive attention in the literature. In contrast to character recognition which has been studied by a large number of researchers [ZBL17, CBX*17, LJS19], learning to write like humans still requires further investigation because of its complexity and diversity.

Generally speaking, there are two different ways to represent a handwritten character. One is to regard it as aligned pixels (i.e., an image) and the other is to denote it as a sequence of strokes (i.e., a writing trajectory, see Figure 3), corresponding to offline and online handwritings, respectively. The latter one usually contains more information (e.g., timing) which can be converted into the former one easily. What's more, human beings typically write a character by drawing strokes one by one in the pre-defined order instead of "generating" an image at once. Most previous models (e.g., [KX17, CZPM18]) for handwriting generation (especially for Chinese handwritings) are based on images because of the popularity of Convolutional Neural Networks (CNNs) which have been shown to be effective in many image related tasks. Recurrent Neural Networks (RNNs) are widely used in sequence modeling, so utilizing RNN is a promising solution to model online handwritings.

Similar to the physiological characteristics (such as the fingerprint, face and iris), the handwriting also represents a human characteristic. In other words, the characteristic information contained in the handwritings of different writers is different. Motivated by the works of style transfer, we call this characteristic information "style". Our goal is to extract this "style" automatically from several characters written by a given writer and then imitate the

† Corresponding author.

Figure 1: *Some failure cases of using CNN-based models to handle offline cursive Chinese handwritings. The generated results are synthesized by zi2zi[†] [Tia17].*

writer's handwriting style to write. Almost all state-of-the-art models (e.g., [HB17, LFY*17]) for image style transfer extract the style information from some layers of a pre-trained CNN (e.g., VGG [SZ14]). However, this scheme is unsuitable for our purpose since what we intend to process are not images but sequential trajectories. In this paper, we adopt a metric-based meta learning strategy and pre-train a RNN as our style encoder using a carefully designed loss function. What's more, an attention-based adaptive style block (ASB) is introduced to enable the decoder to adaptively obtain style information instead of a fixed style embedding during the decoding process.

The main contributions of this paper are summarized as follows:

- We introduce a novel method to generate online Chinese handwritings with arbitrary styles. The user only needs to write a few Chinese character samples, our model can imitate the user's writing style and human writing process to write, instead of synthesizing a glyph image at once like other models. The source code of our method is available at `https://github.com/ShusenTang/WriteLikeYou`.
- To improve the generalization of our model on new writers, we propose to use metric-based meta learning and pre-train our style encoder using a well-designed loss function. In addition, unlike many existing models whose style encoder outputs a fixed vector, we introduce an attention-based adaptive style block which allows the decoder to adaptively receive information from the style encoder.
- Experiments demonstrate that our method performs better compared to other existing approaches and is capable of adapting to any new writers. Moreover, the user study verifies that our synthesized Chinese handwriting is difficult to be distinguished from the real one.

2. Related Work

2.1. Chinese Handwriting Generation

In recent years, lots of methods have been proposed for the handwriting generation of alphabetic languages (e.g., English), including methods using RNNs [Gra13, APH18, KTT20] and other approaches [AH19, LLZY18, HAB16, SIHU19, FAEC*20]. Compared to alphabetic languages, Chinese has a much larger charset (e.g., even the most commonly used Chinese charset GB2312 consists of 6763 characters) and Chinese characters have more complex shapes and topological structures. Making machines learn to write Chinese characters is thus more interesting and challenging.

Some previous methods (e.g., [XJJL09, LHC*15, LZCX18]) have been reported on Chinese handwriting generation by assembling components of characters. They first decompose the sample characters into reusable components and then adopt the best-suited way to compose the target character. These models inevitably require prior knowledge such as elaborate preceding parsing, and thus fail to satisfactorily handle characters with connected and cursive strokes.

Recently, lots of CNN-based models for offline Chinese glyph synthesis have emerged [RMC15, SRL*17, ZPIE17, Tia17, ZZC18, GGL*19, WGL20]. However, these methods fail to reflect the process of human writing, and cannot handle scribbled handwritings (see Figure 1). The generated results of these methods inevitably have problems such as inconsistent strokes, wrong topologies and blurs.

Up to now, just a few works have been reported that aim to deal with online Chinese handwritings. [Ha15] modifies and extends Graves' approach [Gra13] to use LSTM to generate fake (i.e., unreadable) Chinese characters. [ZYZ*16] proposes a online Chinese handwriting generation model, which is mainly for generating characters as augmentation data for their recognition network. However, their model does not involve style information thus it is style-agnostic. FontRNN [TXL*19] utilizes a similar transfer learning strategy to generate Chinese character skeletons via RNNs. But they focus on font generation and each trained model can only synthesize one font (same as the training set). In contrast, our model focuses on Chinese handwriting generation, and can synthesize results with arbitrary styles once it is trained without retraining. Recently, DeepImitator [ZTY*20] uses a CNN to extract the style information from several handwritten character images, which is integrated with an attention module and a RNN to generate personalized online handwritings. However, the loss they used for style encoder is the simple cross-entropy softmax, which fails to encourage discriminative learning of features, and DeepImitator is not able to generalize to new character class because their character embeddings are jointly trained with the generative model.

2.2. Sequence-to-Sequence Model

The Sequence-to-Sequence (Seq2Seq) model was first introduced for neural machine translation [CMG*14] which consists of two RNNs: encoder and decoder. The encoder encodes the input sequence into a fixed-dimension vector, and the decoder decodes this vector into the output sequence. After that, [BCB14, LPM15] argue that the fixed-dimension vector is a bottleneck for improving the performance and propose the attention mechanism, which allows the encoder no longer try to encode the full input sequence into a fixed-dimension vector and lets the decoder "attend" to different parts of the input sequence at each decoding step. There exist many other successful applications of the Seq2Seq model, such as chatbot [QLW*17, XLG*17], speech recognition [CJLV16, BCS*16] and video representation [HHL*17, SMS15]. Our proposed model except the style encoder and the adaptive style block, broadly speaking, is a Seq2Seq model with the attention mechanism.

[†] The code of zi2zi is publicly available at `https://github.com/kaonashi-tyc/zi2zi`.

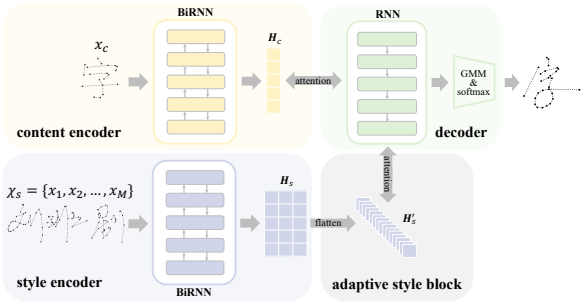


Figure 2: *The architecture of our proposed model.*

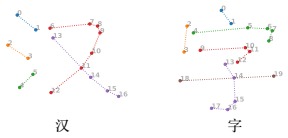


Figure 3: *Two Chinese characters in sequential format. Each color denotes one stroke and integers determine the writing order.*

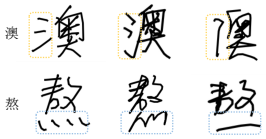


Figure 4: *The style information can be usually extracted from the local parts of handwritten Chinese characters.*

2.3. Metric-based Meta Learning

Meta learning, also known as "learning to learn", aims to deploy models that can rapidly adapt to new tasks which have never been encountered during training. In this case, writers (i.e., handwriting styles) correspond to the tasks and we try to enable our model to adapt to new writers rapidly. We adopt the key idea of metric-based meta learning: learning a metric function over objects, so we pre-train a style encoder to learn the prior knowledge. During adaptation, the model integrates this prior knowledge with new tasks to acquire new skills fast, imitating new writers' handwritings. As demonstrated in Section 5.4, we get the best results under the few-shot adaptation strategy, which requires us to fine-tune the pre-trained model. Fine-tuning a network [HR18, SRL*17] is a widely used and effective method for transfer learning [Ben12, ZQD*21]. In our case, we can fine-tune the model parameters on a few samples written by a new writer for the sake of more realistic synthesis results if the re-training dataset is available.

The crucial requirement is that the style information extracted by the style encoder should be discriminative between different writers and compact for the same writer. Although cross-entropy softmax is demonstrably one of the most commonly used loss function to pre-train feature extraction networks, it is more suitable for classification and does not explicitly encourage discriminative learning of features. Therefore, various losses have been proposed, such as the contrastive loss [HCL06], the triplet loss [SKP15], and the large-margin softmax loss [LWY*17, DGZ18, WCLL18, WWZ*18]. The contrastive loss and triplet loss require carefully designed pair/triplet training procedures, hence both of them are time-consuming and performance-sensitive. The large-margin softmax loss overcomes the above problems, which reduces the target activation (i.e., enforces a stricter decision criteria compared to the normal softmax loss) to learn discriminative features and has been successfully applied in face recognition. We discuss the large-margin loss in detail in Section 4.3.2.

3. Overview

Given several Chinese characters written by a writer, our goal is to imitate this writer to write realistic handwritings in the same style by drawing strokes one by one like humans. To achieve this goal, as depicted in Figure 2, we propose a model which mainly contains three RNNs: (1) a content encoder, which converts the reference input character x_c into the content information, (2) a style encoder, which extracts the style information from M style input characters $\mathcal{X}_s = \{x_1, x_2, ..., x_M\}$ written by a given writer $w \in \mathcal{W}$, where $\mathcal{W}$ is the training writer set, and (3) a decoder, which integrates the content and style information together and generates the output x'. More details of our model are described in Section 4. We train our model with supervision to force the output x' to contain the same style as $\mathcal{X}_s$ while guaranteeing the correct content from x_c.

After training, we apply two different adaptation strategies. First, because the characteristic of a certain writer to be used as the style information can be inferred by the style encoder, our model can be conditioned on samples from the writer w' who is outside the training set (i.e., $w' \notin \mathcal{W}$), achieving zero-shot adaptation to the new writer. Second, for the sake of better performance, we can also fine-tune the model parameters on a few samples written by a new writer if the retraining dataset is available, achieving few-shot adaptation to the new writer. Experiments show that both of these two strategies are capable of adapting our model to new writers.

4. Method Description

4.1. Data Representation

As shown in Figure 3, a handwritten Chinese character typically consists of several ordered strokes which can be simply represented by a sequence of key points $[P_1, P_2, ..., P_n]$. Like other Seq2Seq models, we also pad it out to a fixed length N in practice:

$$[P_1, P_2, ..., P_n, ..., P_N]. \quad (1)$$

Then the handwritten character can be drawn in vector format, which is more reasonable and natural than the bitmap image.

In [Gra13], P_i is a vector $(\Delta x, \Delta y, p)$, where $(\Delta x, \Delta y) \in \mathcal{R}^2$ denotes the pen offset from the previous point, and p has value 1 if this point ends a stroke and value 0 otherwise. After that, [Ha15] expands the binary value p into a one-hot vector (p_1, p_2, p_3) to let the model know exactly when to stop writing after it has finished writing a complete character. Now, p denotes three possible point categories: (1) $p_1 = 1$ when the pen is now touching the paper, a line will be drawn to connect this point with the next point, (2) $p_2 = 1$ means that the pen is to be lifted up (i.e., end of one stroke), and (3) $p_3 = 1$ indicates that the model has finished writing a complete character and should stop writing, so the p_3 of the point P_i is 1 if and only if $i > n$, where n is the actual sequence length.

We use both of these two representation methods described above and call the first representation format-3 and the second one

format-5. Since our content encoder and style encoder are only for extracting information instead the writing prediction, the inputs of them are constructed using format-3. For the opposite reason, we use format-5 on our decoder for better predictive capability.

4.2. Content Encoder

The role of the content encoder is to encode the reference character x_c into the content information that can be perceived by the decoder and then informs the decoder which Chinese character should be written. Therefore, the content encoder must have strong information extraction capability. The bidirectional recurrent neural network (BiRNN) [SP97] has been shown to be more suitable for information extraction than regular RNNs, so we adopt a BiRNN (BiLSTM to be specific) as our content encoder. The content encoder ENC_c encodes x_c as hidden states $H_c = \text{ENC}_c(x_c)$, where H_c consists of N hidden vectors and each hidden vector is the concatenation of forward and backward states.

As mentioned in Section 2.2, attention mechanisms are widely used and proved to be effective in Seq2Seq models, so we apply the attention mechanism between the content encoder and decoder. Consequently, at the decoding time step t, the content input of decoder is:

$$c_t = \text{attention}(H_c, h_{t-1}), \tag{2}$$

where h_{t-1} is the decoder hidden state at the previous time step, and $\text{attention}(\cdot)$ is an attention mechanism. As suggested by [TXL*19], here we use the monotonic attention [RLL*17] (refer to the supplementary materials for details) in our experiments.

4.3. Style Encoder

The style encoder is the core of our model, which not only needs to effectively extract the style information of writers in the training set, but also be able to adapt to new writers rapidly.

4.3.1. Architecture

Like the content encoder, the style encoder is also a BiRNN, but the input is not a single character x_c but multiple ones (i.e., $\mathcal{X}_s = \{x_1, x_2, ..., x_M\}$). The style encoder ENC_s encodes $\mathcal{X}_s$ as hidden states $H_s = \text{ENC}_s(\mathcal{X}_s)$, where H_s is a M by N matrix of hidden vectors.

As shown in Figure 4, the style information of handwritten Chinese characters is often contained in local parts, e.g., the tips of writing brushes and connected strokes. Therefore, we introduce an attention-based adaptive style block (ASB) to allow the decoder to acquire style information adaptively instead of a fixed style vector (e.g., the mean of all $h_{s,i,j}$) during the decoding process. First, we flatten the H_s into a sequence of vectors:

$$H'_s = [h_{s,1,1}, h_{s,1,2}, ..., h_{s,M,N}], \tag{3}$$

now we can apply the attention mechanism like Equation (2):

$$s_t = \text{attention}(H'_s, h_{t-1}), \tag{4}$$

where h_{t-1} is the decoder hidden state at the previous time step. Here the $\text{attention}(\cdot)$ is computed as:

$$s_t = \sum_{i=1}^{M} \sum_{j=1}^{N} \alpha_{t,i,j} h_{s,i,j} \tag{5}$$

$$\alpha_{t,i,j} = \frac{\exp(\text{score}_{t,i,j})}{\sum_{m=1}^{M} \sum_{n=1}^{N} \exp(\text{score}_{t,m,n})} \tag{6}$$

$$\text{score}_{t,i,j} = h_{t-1} \mathbf{W} h_{s,i,j}, \tag{7}$$

where $\mathbf{W}$ is a trainable parameter matrix. The score calculation method in Equation (7) is proposed by [LPM15], another commonly used option is:

$$\text{score}_{t,i,j} = v^T \tanh(\mathbf{W} h_{t-1} + \mathbf{U} h_{s,i,j}), \tag{8}$$

which is proposed by [BCB14]. Experimental results presented in Section 5.3.1 shows that the first method is slightly better than the second one in our application.

4.3.2. Pre-training

Pre-training the style encoder is a key step to make our method work, it enables our model to adapt to new writers quickly. During pre-training, the style encoder converts each input character into the sequence of hidden states $[h_1, h_2, ..., h_N]$ and constructs the output feature x by:

$$x = \sum_{i=1}^{N} \alpha_i h_i. \tag{9}$$

There are two common strategies of calculating the above weight α_i: (1) $\alpha_i = 1$ if and only if $i = N$, this means x is the last hidden state h_N[‡]; (2) $\alpha_i = 1/N$ for all i, this means x is equal to the average of all hidden states. As we can see from Figure 4, as mentioned in Section 4.3.1, we argue that the style information contained in a handwritten Chinese character is not evenly distributed, the network should learn to let x focus on those style-plentiful hidden states. Therefore, we propose to calculate α_i as:

$$\alpha_i = \frac{\exp(\text{score}(\bar{h}, h_i))}{\sum_{j=1}^{N} \exp(\text{score}(\bar{h}, h_j))}, \text{ where } \text{score}(\bar{h}, h_i) = \bar{h} \mathbf{V} h_i, \tag{10}$$

$\bar{h} = \frac{1}{N} \sum_{i=1}^{N} h_i$ and $\mathbf{V}$ is a trainable parameter matrix. Here we use the same method of calculating $\text{score}(\cdot)$ as in Equation (7) for the consistency of pre-training and decoding inference. Experiments in Section 5.3.2 show that this strategy of getting α_i is much better than the two commonly used methods mentioned above.

Then, we can pre-train the style encoder to obtain the discriminative feature x, e.g., using the cross-entropy softmax loss for style classification (writer identification in this case). Recently, the large-margin softmax loss [LWY*17, DGZ18, WCLL18, WWZ*18] becomes popular for discriminative feature learning, which can be regarded as an improvement over the original softmax loss. Given the feature x_i and its label y_i, the original softmax loss is computed

[‡] In practice we take the actual sequence length n instead of N.

as

$$L = \frac{1}{B}\sum_{i=1}^{B} -\log\left(\frac{\exp(w_{y_i}^T x_i)}{\sum_{j=1}^{C}\exp(w_j^T x_i)}\right) = \frac{1}{B}\sum_{i=1}^{B} -\log\left(\frac{\exp(||w_{y_i}||\,||x_i||\cos\theta_{y_i,i})}{\sum_{j=1}^{C}\exp(||w_j||\,||x_i||\cos\theta_{j,i})}\right), \tag{11}$$

where B is the batch size, C is the number of categories (e.g., the number of writers in this case), w_j denotes the output layer weights corresponding to the writer j, $\theta_{j,i}$ is the angle between w_j and x_i, and the bias is set to zero. If we normalize the weight $||w_j|| = 1$ and $||x_i|| = 1$, then we get the modified softmax:

$$L_{\text{modified}} = \frac{1}{B}\sum_{i=1}^{B} -\log\left(\frac{\exp(s\cdot\cos\theta_{y_i,i})}{\sum_{j=1}^{C}\exp(s\cdot\cos\theta_{j,i})}\right), \tag{12}$$

where s is the scaling factor. At this point the loss is only determined by the angle between the weight w and the feature x, the smaller the angle between w_{y_i} and x_i is, the greater the probability that x_i will be correctly classified into y_i is, and the smaller the loss is. The key idea of the large-margin softmax loss is to force this angle to be rather small, so the penalty margins are introduced to the target logit ($\cos\theta_{y_i,i}$):

$$\psi(\theta_{y_i,i}) = \cos(m_1\theta_{y_i,i} + m_2) - m_3, \tag{13}$$

where m_1, m_2 and m_3 are penalty margins and often used separately, corresponding to angular softmax (A-Softmax) [LWY*17], additive angular margin softmax (Arc-Softmax) [DGZ18] and AM-Softmax [WCLL18, WWZ*18], respectively. Combining Equation (12) and (13) we formulate the large-margin softmax loss as:

$$L_{\text{LM}} = \frac{1}{B}\sum_{i=1}^{B} -\log\left(\frac{\exp(s\cdot\psi(\theta_{y_i,i}))}{\exp(s\cdot\psi(\theta_{y_i,i})) + \sum_{j=1,j\neq y_i}^{C}\exp(s\cdot\cos\theta_{j,i})}\right). \tag{14}$$

Experiments in Section 5.3.3 indicate that the large-margin softmax loss used in our application does not work as well as it does in face recognition. Possible reasons are as follows: (1) a person's handwriting is more diverse than his face pictures; (2) the numbers of individuals in face datasets (e.g., the relatively small face dataset CASIA-WebFace [YLLL14] contains 494,414 training faces belonging to 10,575 different individuals) are much larger than the Chinese handwriting dataset we use. Therefore, we propose to minimize the angle between the feature x_i and the average of features corresponding to the writer y_i instead of the weight w_{y_i}, and introduce the angular center loss (L_{AC}). For simplicity, we specify that one training batch contains B_w different writers, each writer consists of B_s samples, then we compute L_{AC} as:

$$L_{\text{AC}} = \frac{1}{B}\sum_{i=1}^{B} -\log\left(\frac{\exp(s\cdot\phi(\beta_{y_i,i}))}{\exp(s\cdot\phi(\beta_{y_i,i})) + \sum_{j\in\mathcal{C},j\neq y_i}\exp(s\cdot\cos\beta_{j,i})}\right) \tag{15}$$

$$\phi(\beta_{y_i,i}) = \cos(\beta_{y_i,i}) - m, \tag{16}$$

where $\mathcal{C}$ ($|\mathcal{C}| = B_w$) is the writer set in the *current* batch, and $\beta_{j,i}$ is the angle between the average of features of the corresponding writer j and the feature x_i:

$$\cos\beta_{j,i} = \frac{c_j^T x_i}{||c_j||\,||x_i||}, \text{ where } c_j = \frac{1}{B_s}\sum_{y_k=j} x_k. \tag{17}$$

As shown in Equation (16), we also introduce the penalty margin into L_{AC}. Here we only use one margin m because preliminary experiments show that the performance gain of combination use is relatively small and the hyper parameters are difficult to be tuned well.

It is worth mentioning that the proposed L_{AC} is different from the center loss [WZLQ16] which minimizes the Euclidean distance of the feature and the *global* centroid of features corresponding to its class. The center loss requires all class centers to be stored in memory and updated with gradient descent, hence it is computationally expensive and cannot converge in our case.

4.4. Decoder

The RNN-based decoder predicts the next point conditioned on the previous output and the current content and style information from the content encoder and style encoder, respectively.

At the current decoding time step t, the previous target point P_{t-1} (represented as format-5, and it is the previous output P'_{t-1} in testing) is concatenated with the content output c_t and the style output s_t as $a_t = [P_{t-1}; c_t; s_t]$ for decoding: $h_t = \text{DEC}(h_{t-1}, a_t)$,

where h_t is the decoder hidden state at time step t. Then h_t will be mapped into o_t for predicting the output point P'_t through a linear layer. As suggested by [Gra13], we model the point offset $(\Delta x, \Delta y)$ using the Gaussian mixture model (GMM) with R bivariate normal distributions, and point categories (p_1, p_2, p_3) using a three-category classifier (i.e., a softmax layer). Therefore, o_t is represented as:

$$o_t = [\{\pi^r, \mu_x^r, \mu_y^r, \delta_x^r, \delta_y^r, \rho_{xy}^r\}_{r=1}^{R}, q_1, q_2, q_3], \tag{18}$$

where the superscript r above stands for the r^{th} distribution in GMM. Then we can optimize model parameters by minimizing the negative log-likelihood:

$$L_o = -\frac{1}{n}\sum_{i=1}^{n}\log(p(\Delta x_i, \Delta y_i)), \tag{19}$$

$$p(\Delta x, \Delta y) = \sum_{r=1}^{R}\pi^r\mathcal{N}(\Delta x, \Delta y|\mu_x^r, \mu_y^r, \delta_x^r, \delta_y^r, \rho_{xy}^r), \tag{20}$$

where n is the number of target points and $\mathcal{N}(\cdot)$ is the bivariate normal distribution function.

With regard to the point category, we use the cross entropy loss between the target p and the predicted q:

$$L_c = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{3} p_{k,i}\log(q_{k,i}), \tag{21}$$

where N is the fixed maximum length described in Section 4.1.

In addition, we utilize the style cycle loss to explicitly optimize the style of output to be consistent with the style input $\mathcal{X}_s$:

$$L_{\text{cycle}} = ||x' - \bar{x}||_1, \tag{22}$$

where x' is the style feature of the decoder output extracted by the style encoder using Equation (9), while $\bar{x}$ is the average feature of the style input $\mathcal{X}_s$.

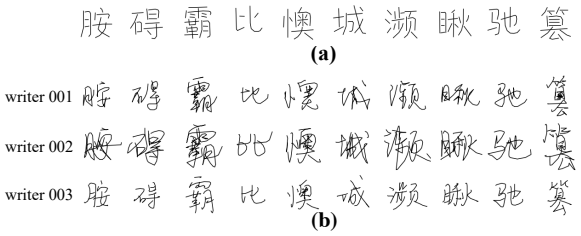


Figure 5: *Examples of datasets we use for (a) the reference input and (b) the style input.*

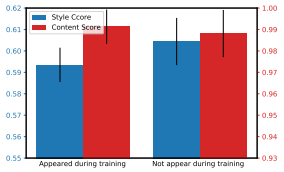


Figure 6: *The good generalization ability of our method in terms of content.*

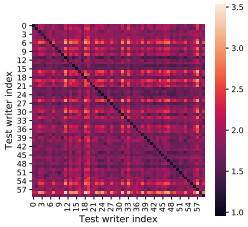


Figure 7: *The heatmap of the DTW matrix.*

Finally, the total loss function is defined as the weighted sum of L_o, L_c and L_{cycle}:

$$L = L_o + \lambda_c L_c + \lambda_{\text{cycle}} L_{\text{cycle}}, \tag{23}$$

where λ_c and λ_{cycle} are hyper parameters that control the weights of L_c and L_{cycle}, respectively.

For testing, unlike the training process, we obtain the current point P'_t by sampling from the GMM determined by the current output o_t and then feed the sampled P'_t as input for the next time step. We continue this sample process until $p'_{3,i} = 1$ or when $i = N$, so the sampled output $[P'_1, P'_2, ..., P'_n, ..., P'_N]$ is not deterministic but random. This is very similar to the human writing process: each time we write the same character, it looks slightly different.

5. Experiments

5.1. Experimental Setup

5.1.1. Datasets

As described in Section 3, the input of our model contains two parts: a reference character x_c and M style characters $\mathcal{X}_s$. The reference input only tells the model which character should be generated, so theoretically glyphs in any neat font can be used as the reference characters, and here we adopt the commonly used average Chinese font [JLTX19] after manual point-annotation (see Figure 5(a)). With regard to the style input $\mathcal{X}_s$, we use the CASIA Online Chinese Handwriting Databases [LYWW11] (see Figure 5(b)) including OLHWDB1.0, OLHWDB1.1 and OLHWDB1.2 for training and the Competition Test for testing[§]. In total, we have about 3.7 million Chinese characters written by 1020 writers for training and about 0.2 million characters written by 60 writers for testing. To explore the capability of generalization to new contents, 1/10 contents are left for evaluation rather than training. In addition, in order to avoid the negative impact of redundant points during learning and meanwhile reduce the number of points, as suggested by [HE17], we adopt the Ramer-Douglas-Peucker [DP73] algorithm (parameter ε is set to 4.0) on our data to remove redundant points. After simplification, more than 98% of the samples have less than 110 points, so we set the fixed maximum length N in Equation (1) to 110 and discard the points that exceed this maximum length.

[§] The details of these databases can be found on http://www.nlpr.ia.ac.cn/databases/handwriting/Home.html.

5.1.2. Implementation Details

In our experiments, the content encoder and decoder are both single-layer LSTMs with hidden sizes of 256 and 512 respectively, while the style encoder consists of three stacked LSTMs with the hidden size of 256. We set $\lambda_c = 2.0$ and $\lambda_{\text{cycle}} = 5.0$. We use the Adam [KB14] optimizer to train our model with the batch size of 128 (1024 for pre-training the style encoder), learning rate of 0.001 and gradient clipping of 1.0. For data augmentation, as suggested by [HE17], we multiply the offset $(\Delta x, \Delta y)$ by a random scale factor in the range $[0.90, 1.10]$ and dropping some points randomly with a probability of 0.10. Unless otherwise specified, we set the penalty margin m in Equation (16) to 0.2 and the number of style input characters M to 10.

5.1.3. Evaluation Metric

Dynamic time warping (DTW) [BC94] is used to calculate the distance between two sequences with different lengths and hence we use DTW to evaluate the similarity between the real and generated handwritings, lower DTW indicates higher similarity. As suggested by [TXL*19], the DTW distance will be normalized by the spatial scale and length of real handwritings.

In addition, to quantitatively evaluate our method in terms of content and style separately, we utilize two classifiers to score the generated handwriting. Specifically, for content evaluation, we train a character recognizer on the training set and use the recognition accuracy on generated handwriting as the *Content Score*, while for style evaluation, we similarly utilize a style classifier (i.e., writer identification) trained on the testing set (containing 60 writers) and regard its classification accuracy on the generated handwriting as the *Style Score*. Since the generated results are almost all readable and the Content Score is always extremely close to 1.0 (e.g., Table 4), we mainly show and discuss the Style Score. The details of these evaluation metrics can be found in the supplementary materials.

5.2. Exploratory Experiments

5.2.1. Generalization to New Contents

Although we mainly focus on style modeling, our model generalizes well to new contents (i.e., x_c) which have not appeared during training. As shown in Figure 6, we compute the average Style Score and Content Score of the generated results when inputting seen/unseen contents, respectively. We can see that the generated result has surprisingly higher Style Score and expectedly a bit lower

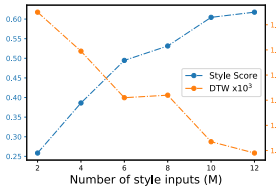


Figure 8: *Illustration of model performance with different numbers of input style characters.*

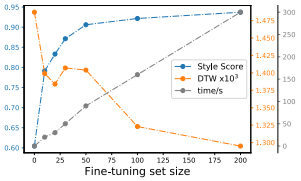


Figure 9: *Comparison of our methods fine-tuned on datasets with different sizes.*

Content Score if the corresponding content input x_c has not appeared during training. This demonstrates that our method is robust in content modeling. In all the following experiments, the test input contents have not appeared during training.

5.2.2. Different m

The hyper parameter m in L_{AC} controls the degree of penalty margin added in the target logit. Theoretically, the larger the m is, the more discriminative the characteristics of different writers learned by the style encoder are, and the higher the Style Score is. The experimental results, as listed in the bottom row of Table 3, show that a moderate penalty leads to better performance as expected, but if the penalty is too large, the model performance will decrease (i.e., $m = 0.3$) or even cannot converge (i.e., $m = 0.4$). This is because if we apply a too large penalty, L_{AC} will become too difficult to be optimized. Here we obtain the best result at $m = 0.2$ which is adopted by our model in other experiments.

5.2.3. Different M

The only source of style information is the style input characters $\mathcal{X}_s = \{x_1, x_2, ..., x_M\}$ whose size M obviously affects the model's performance. As shown in Figure 8, with more input style characters available, the Style Score improves as expected, meaning that the synthesized handwriting contains richer style information. But due to the limitation of memory size and the unnoticeable improvement of performance when M is too large, the style input size M is fixed as 10 in other experiments.

5.2.4. Consistency of Style

The style inputs $\mathcal{X}_s$ are sampled randomly from the handwriting set, so how consistent are the generated results when different $\mathcal{X}_s$ sampled from the same writer are used? We conduct two tests on a same trained model and generate 200 characters for each test writer in each test. As shown in Figure 7, we calculate the average DTW value (multiplied by 1000) between every two test writers and get a DTW square matrix. The dark diagonal in Figure 7 means that the generated results using different style inputs written by the same writer are very similar, and proves the effectiveness of the style encoder from the side.

5.3. Ablation Studies

In this section, we conduct several experiments to verify the effectiveness of each key module in our method.

	Train Style Score	Test Style Score
w/o ASB	0.812 ± 0.008	0.422 ± 0.015
Bahdanau [BCB14] ASB	0.840 ± 0.007	0.593 ± 0.009
Luong [LPM15] ASB	$\mathbf{0.841} \pm 0.008$	$\mathbf{0.604} \pm 0.011$

Table 1: The ablation study results of the attention-based adaptive style block (with 95% confidence interval, similarly hereinafter). The Train Style Score and Test Style Score represent the Style Score on the training and test sets, respectively.

	Last	Average	Ours
Style Score	0.580 ± 0.012	0.328 ± 0.012	$\mathbf{0.604} \pm 0.011$
DTW $\times 10^3$	1.513 ± 0.004	1.558 ± 0.006	$\mathbf{1.487} \pm 0.004$

Table 2: Comparison of three pre-training strategies.

5.3.1. Adaptive Style Block

The quantitative results to verify the effectiveness of the proposed ASB described in 4.3.1 are shown in Table 1. The Bahdanau and Luong denotes two different score calculation methods described in Equation (8) and (7), respectively. Table 1 shows that if we remove our ASB (i.e., the decoder receives a fixed style vector), the Train Style Score will not drop much while the Test Style Score will decline sharply. This demonstrates that the proposed ASB improves our model's generalization capability and thus enables the model to adapt to new writers effectively. In addition, the Luong ASB performs better than the Bahdanau ASB in our case.

5.3.2. Pre-training Strategy

Pre-training the style encoder is a key step in our method. Preliminary experiments show that our model will not fully converge if we train it from scratch. As mentioned in Section 4.3.2, during pre-training, there exist two common ways for calculating the output feature x in Equation (9): taking the last hidden state directly, or averaging all hidden states. Instead, we propose to calculate x using a method similar to the attention (see Equation (10)). Table 2 demonstrates that the averaging strategy works rather poorly and our method outperforms the other two.

5.3.3. Pre-training Loss

We conduct a series of experiments to summarize the performance of different pre-training losses. As we can see in Table 3, although the large-margin softmax loss with small penalty margin (e.g., $m_1 = 2, m_2 = 0.1$ or $m_3 = 0.1$) outperforms the modified softmax, once we add a slightly larger margin to the modified softmax (e.g., $m_1 = 4, m_2 = 0.2/0.3$ or $m_3 = 0.2/0.3$), the performance decreases and becomes worse than the original modified softmax. And the proposed L_{AC} outperforms all other losses with the wide range of the margin m. We already discussed the effects of different m in detail in Section 5.2.2.

5.3.4. Style Cycle Loss

We conduct an experiment to see how the model performs under different weights (λ_{cycle}) of the style cycle loss. As shown in Table

	Margin	Style Score	DTW $\times 10^3$
Softmax	/	0.503 ± 0.013	1.531 ± 0.005
Modified Softmax	/	0.541 ± 0.015	1.519 ± 0.004
A-Softmax [LWY*17]	$m_1 = 2$	0.559 ± 0.009	1.512 ± 0.004
	$m_1 = 4$	N/A	N/A
Arc-Softmax [DGZ18]	$m_2 = 0.1$	0.567 ± 0.013	1.507 ± 0.007
	$m_2 = 0.2$	0.520 ± 0.012	1.525 ± 0.005
	$m_2 = 0.3$	0.485 ± 0.014	1.533 ± 0.006
AM-Softmax [WCLL18, WWZ*18]	$m_3 = 0.1$	0.555 ± 0.010	1.499 ± 0.004
	$m_3 = 0.2$	0.484 ± 0.011	1.529 ± 0.006
	$m_3 = 0.3$	0.471 ± 0.014	1.527 ± 0.007
L_{AC} (Ours)	$m = 0$	0.538 ± 0.013	1.521 ± 0.004
	$m = 0.1$	0.569 ± 0.011	1.514 ± 0.006
	$m = 0.2$	$\mathbf{0.604} \pm 0.011$	$\mathbf{1.487} \pm 0.004$
	$m = 0.3$	0.582 ± 0.010	1.496 ± 0.004
	$m = 0.4$	N/A	N/A

Table 3: Comparison of different pre-training losses with various degrees of penalty margins. N/A indicates that the model has not converged.

λ_{cycle}	0	0.1	1.0	5.0	10.0	20.0
Style Score	0.588	0.596	0.594	0.604	0.603	0.605
Content Score	0.994	0.994	0.993	0.988	0.971	0.946
L_{cycle}	/	0.139	0.102	0.099	0.096	0.083

Table 4: Evaluation of different weights of style cycle loss.

4, as the weight increases, the style cycle loss L_{cycle} gradually decreases, and the Style Score gradually increases, as expected. Nevertheless, too much attention to style information will cause the lose of some content information. Therefore, we set the weight of L_{cycle} to 5.0 for the trade-off between style and content consistency.

5.4. Few-shot Adaptation

The above experiments are conducted without fine-tuning on the specific testing writer, corresponding to zero-shot adaptation (i.e., the fine-tuning set size is equal to 0) as described in Section 3. In this section, we fine-tune the trained model on a few handwritten characters to synthesize more realistic results which are difficult to be distinguished from the real handwritings.

5.4.1. Fine-tuning Size

For each writer w' in the testing set, we fine-tune our model on a small fine-tuning set which contains a few characters written by the writer w'. Intuitively, the size of this set can affect the quality of synthesized results. As depicted in Figure 9, the quality improve quickly as the fine-tuning size gets larger, which becomes good enough (i.e., the Style Score is close to 1) when it exceeds 100. It is worth mentioning that it only takes a few hundred seconds for fine-tuning. This verifies that our model can be well adapted to new writers. Examples of synthesized results with different fine-tuning set sizes are shown in Firgure 10.

5.4.2. Mix the Spurious with the Genuine

Figure 10 depicts that the generated handwriting is very similar to the real one. To further verify how realistic the synthesized spurious handwriting is, we conduct a user study. Our questionnaire

Figure 10: *Visual comparison of few-shot adaptation with different fine-tuning set sizes.*

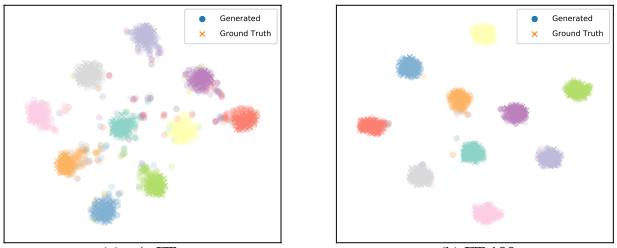
contains 50 questions. For each question (see the supplementary materials for details), given samples of a writer's handwriting, we ask participants to point out one character from the four candidates which they think is most likely written by that writer. The four randomly arranged candidates are the genuine handwriting, handwritings generated by our model without and with fine-tuning (with the fine-tuning size of 100, FT-100 for short), and the same character written by a random different writer. Finally, 101 individuals took part in our test, the statistical preference is shown in Table 6, which indicates that the participants are struggling to point out the difference between the genuine (real) and spurious (w/o FT and FT-100) handwritings. Namely, the synthesized handwriting is too realistic to be distinguished from the real handwriting. A large number of generated results can be found in the appendix of the supplementary materials.

5.4.3. Visualization of the Style Features

Figure 11 shows the t-SNE projection [MH08] of the style features (calculated by Equation (9)) using zero-shot adaptation (a) and few-shot adaptation (b). In this experiment, we randomly selected 10 test writers, for each writer, we randomly select 100 real handwritten characters, 100 characters synthesized by our model with/without fine-tuning. From Figure 11 we can see that regardless of using zero-shot adaptation or few-shot adaptation, there are obvious clusters for all writers, with a rather large inter-writer distance and a low intra-writer separation. Moreover, the projections of the style features of real and generated samples almost overlap in both (a) and (b). Therefore, Figure 11 demonstrates both an ease of correctly identifying the writer through a given generated handwriting and the difficulty in distinguishing real handwritings from synthesized ones.

5.4.4. Ablation Studies after Fine-tuning

The results of ablation studies conducted in Section 5.3 prove that our proposed methods are effective under the zero-shot adaptation configuration, which is pretty valuable because in most cases we don't have additional data to fine-tune the model. It is also interesting to figure out whether the effectiveness can preserve after fine-tuning or not. The experimental results are shown in Table 5. Although the gaps between different settings have been reduced after fine-tuning, our proposed designs are still quite competitive.



(a) w/o FT (b) FT-100

Figure 11: *T-SNE visualization of the style features of our synthesized handwritings and real ones.*

Pre-training Loss	ASB	Pre-training strategy	Style Score
L_{AC} (Ours, $m = 0.2$)	×	Ours	0.861 ± 0.012
	√	Last	0.929 ± 0.009
	√	Average	0.852 ± 0.010
	√	Ours	$\mathbf{0.937} \pm 0.009$
Softmax	√	Ours	0.902 ± 0.010
Modified Softmax	√	Ours	0.909 ± 0.009
A-Softmax ($m_1 = 2$)	√	Ours	0.926 ± 0.010
Arc-Softmax ($m_2 = 0.1$)	√	Ours	0.930 ± 0.010
AM-Softmax ($m_3 = 0.1$)	√	Ours	0.923 ± 0.010

Table 5: The results of ablation studies after fine-tuning (with the fine-tuning size of 100).

	real	w/o FT	FT-100	real-diff
Prefer. (%)	34.3	27.1	29.6	9.0

Table 6: Results of the user study described in Section 5.4.2.

5.4.5. Synthesis versus Retrieval

Given a writer in the test set, we retrieve the most similar writer from the training set which has the lowest DTW value and then compute the Style Score. Finally, we get a Style Score of 0.162 which is very low compared to our model (see Table 7). This verifies the style diversity between the training and test sets, and demonstrates that our model does not simply remember the training styles but extract the new the target style using its style encoder.

5.5. Comparison with the State of the Art

As mentioned in Section 2.1, FontRNN [TXL*19] was proposed to generate large-scale Chinese fonts via RNN, as well as to synthesize stylized online Chinese handwriting, and DeepImitator [ZTY*20], a multi-module framework, was introduced to address the problem of personal handwriting generation. In this section, we conduct qualitative and quantitative experiments to compare our model with FontRNN and DeepImitator to verify the superiority of our method.

5.5.1. Visual Comparison

We first visualize the results generated by different methods for qualitative comparison. As shown in Figure 12, although FontRNN and DeepImitator can synthesize readable cursive Chinese handwritten characters, it fails to generate samples consistent with the target style. What is worse, for different target writers, FontRNN needs to be re-trained, while our model is capable of adapting to new writers well with no time (zero-shot adaptation) or little time (few-shot adaptation) needed.

Metrics	DeepImitator	FontRNN	Ours	
			w/o FT	FT-100
Content Score	0.834 ± 0.008	0.875 ± 0.007	$\mathbf{0.988} \pm 0.005$	0.987 ± 0.006
Style Score	0.432 ± 0.014	0.233 ± 0.020	0.604 ± 0.011	$\mathbf{0.937} \pm 0.009$
DTW $\times 10^3$	1.604 ± 0.005	1.629 ± 0.012	1.487 ± 0.004	$\mathbf{1.323} \pm 0.004$
Prefer. (%)	12.35	9.64	36.44	**41.57**

Table 7: Quantitative comparisons with FontRNN [TXL*19] and DeepImitator [ZTY*20].

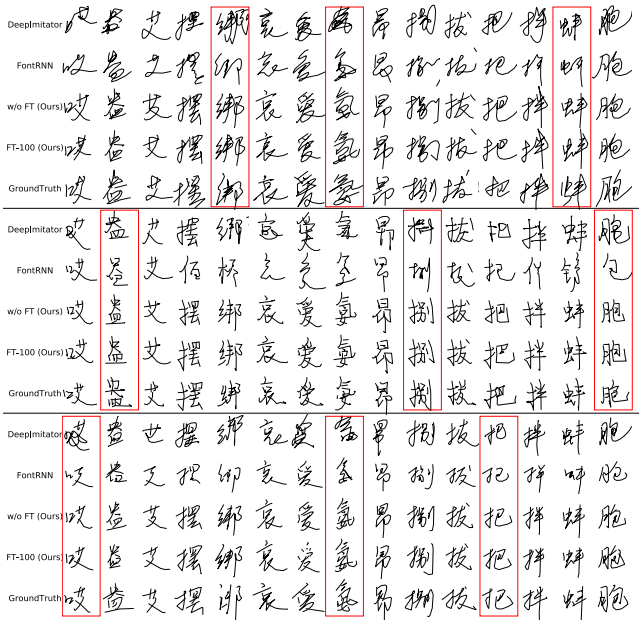


Figure 12: *Visual comparison with FontRNN [TXL*19] and DeepImitator [ZTY*20].*

5.5.2. Quantitative Comparison

We also perform quantitative comparisons. As we can see from Table 7, the Content Score values of DeepImitator and FontRNN are both relatively high while their Style Score and the DTW values are unsatisfactorily poor, indicating that it is difficult for them to handle the style of cursive Chinese handwriting. In addition, a user study is also designed to further compare our method with them (see the supplementary materials for details). Participants need to choose the one out of the four candidate fake handwritten characters that is most similar as the real one. We finally collect 101 valid questionnaire submissions, the preference of characters generated by different methods are listed in the last row of Table 7. We can see that participants consider that in most cases the results generated by our methods are more similar to real ones, while only 12.35% and

Figure 13: *Examples of the brush-writing characters that cannot be handled by our method directly.*

9.64% of the characters synthesized by DeepImitator and FontRNN are selected as preferred, respectively.

6. Limitations

Because our model is made up of RNNs whose training cannot be parallelized along the time dimension, both the operations of forward and back propagation are relatively slow. Specifically, the training time for one epoch takes about 10 hours on a single GeForce GTX 1080 Ti GPU. Besides, the Chinese handwriting discussed in this paper is limited to the writing trajectory without contour, so our model cannot directly handle brush-writing characters. Examples of brush-writing characters can be found in Figure 13. Appending an extra network to recover the contour shape for the trajectory generated by our method is a potential solution. At last, we only discuss the isolated Chinese handwritten characters in this paper. We are planing to explore the generation of the coherent handwritten text segments in our future work.

7. Conclusion

In this paper, we proposed a Seq2Seq model using metric-based meta learning to synthesize cursive Chinese characters written by any writers in sequential format which is more natural and valuable than the traditional image format. To enhance the capability of adapting to new writers rapidly, we introduced an attention-based adaptive style block and pre-trained the style encoder using an effectively designed strategy with our proposed angular center loss. The purpose of pre-training is not obtaining a fixed model but a learner that can quickly learn how to extract the style information from new writers' handwritings. After training, we introduced two adaptation strategies: zero-shot adaptation and few-shot adaptation. For zero-shot adaptation, our method can imitate any new writer's writing style without spending more time for adaptation. With few-shot adaptation, the synthesized handwritten characters are difficult to be distinguished from real samples as long as we fine-tune the model for only several hundred seconds. We conducted both qualitative and quantitative experiments to demonstrate the effectiveness of our method and its superiority compare to the state of the art.

Acknowledgements

This work was supported by Beijing Nova Program of Science and Technology (Grant No.: Z191100001119077) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology).

References

[AH19] AKSAN E., HILLIGES O.: Stcn: Stochastic temporal convolutional networks. *ArXiv abs/1902.06568* (2019). 2

[APH18] AKSAN E., PECE F., HILLIGES O.: Deepwriting: Making digital ink editable via deep generative modeling. In *CHI '18* (2018). 2

[BC94] BERNDT D. J., CLIFFORD J.: Using dynamic time warping to find patterns in time series. In *KDD workshop* (1994), vol. 10, Seattle, WA, pp. 359–370. 6

[BCB14] BAHDANAU D., CHO K., BENGIO Y.: Neural machine translation by jointly learning to align and translate. *CoRR abs/1409.0473* (2014). 2, 4, 7

[BCS*16] BAHDANAU D., CHOROWSKI J., SERDYUK D., BRAKEL P., BENGIO Y.: End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)* (2016), IEEE, pp. 4945–4949. 2

[Ben12] BENGIO Y.: Deep learning of representations for unsupervised and transfer learning. In *ICML Unsupervised and Transfer Learning* (2012). 3

[CBX*17] CHENG Z., BAI F., XU Y., ZHENG G., PU S., ZHOU S.: Focusing attention: Towards accurate text recognition in natural images. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), 5086–5094. 1

[CJLV16] CHAN W., JAITLY N., LE Q., VINYALS O.: Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016), IEEE, pp. 4960–4964. 2

[CMG*14] CHO K., MERRIENBOER B. V., GULCEHRE C., BAHDANAU D., BOUGARES F., SCHWENK H., BENGIO Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP* (2014). 2

[CZPM18] CHANG B., ZHANG Q., PAN S., MENG L.: Generating handwritten chinese characters using cyclegan. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2018), 199–207. 1

[DGZ18] DENG J., GUO J., ZAFEIRIOU S.: Arcface: Additive angular margin loss for deep face recognition. In *CVPR* (2018). 3, 4, 5, 8

[DP73] DOUGLAS D. H., PEUCKER T. K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. 6

[FAEC*20] FOGEL S., AVERBUCH-ELOR H., COHEN S., MAZOR S., LITMAN R.: Scrabblegan: Semi-supervised varying length handwritten text generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 4324–4333. 2

[GGL*19] GAO Y., GUO Y., LIAN Z., TANG Y., XIAO J.: Artistic glyph image synthesis via one-stage few-shot learning. *ACM Transactions on Graphics (TOG) 38*, 6 (2019), 1–12. 2

[Gra13] GRAVES A.: Generating sequences with recurrent neural networks. *ArXiv abs/1308.0850* (2013). 2, 3, 5

[Ha15] HA D.: Recurrent net dreams up fake chinese characters in vector format with tensorflow. *blog.otoro.net* (2015). URL: http://blog.otoro.net/2015/12/28/recurrent-net-dreams-up-fake-chinese-characters-in-vector-format-with-tensorflow/. 2, 3

[HAB16] HAINES T. S. F., AODHA O. M., BROSTOW G. J.: My text in your handwriting. *ACM Trans. Graph. 35* (2016), 26:1–26:18. 2

[HB17] HUANG X., BELONGIE S. J.: Arbitrary style transfer in real-time with adaptive instance normalization. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), 1510–1519. 2

[HCL06] HADSELL R., CHOPRA S., LECUN Y.: Dimensionality reduction by learning an invariant mapping. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06) 2* (2006), 1735–1742. 3

[HE17] HA D., ECK D.: A neural representation of sketch drawings. *ArXiv abs/1704.03477* (2017). 6

[HHL*17] HORI C., HORI T., LEE T.-Y., ZHANG Z., HARSHAM B., HERSHEY J. R., MARKS T. K., SUMI K.: Attention-based multimodal fusion for video description. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 4193–4202. 2

[HR18] HOWARD J., RUDER S.: Universal language model fine-tuning for text classification. In *ACL* (2018). 3

[JLTX19] JIANG Y., LIAN Z., TANG Y., XIAO J.: Scfont: Structure-guided chinese font generation via deep stacked networks. In *AAAI* (2019). 6

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014). 6

[KTT20] KOTANI A., TELLEX S., TOMPKIN J.: Generating handwriting via decouple style descriptors. *arXiv preprint arXiv:2008.11354* (2020). 2

[KX17] KONG W., XU B.: Handwritten chinese character generation via conditional neural generative models. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA* (2017), pp. 4–7. 1

[LFY*17] LI Y., FANG C., YANG J., WANG Z., LU X., YANG M.-H.: Universal style transfer via feature transforms. In *NIPS* (2017). 2

[LHC*15] LIN J.-W., HONG C.-Y., CHANG R.-I., WANG Y.-C., LIN S.-Y., HO J.-M.: Complete font generation of chinese characters in personal handwriting style. *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)* (2015), 1–5. 2

[LJS19] LUO C., JIN L., SUN Z.: A multi-object rectified attention network for scene text recognition. *Pattern Recognition 90* (2019), 109–118. 1

[LLZY18] LAI G., LI B., ZHENG G., YANG Y.: Stochastic wavenet: A generative latent variable model for sequential data. *ArXiv abs/1806.06116* (2018). 2

[LPM15] LUONG T., PHAM H., MANNING C. D.: Effective approaches to attention-based neural machine translation. In *EMNLP* (2015). 2, 4, 7

[LWY*17] LIU W., WEN Y., YU Z., LI M., RAJ B., SONG L.: Sphereface: Deep hypersphere embedding for face recognition. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 6738–6746. 3, 4, 5, 8

[LYWW11] LIU C.-L., YIN F., WANG D.-H., WANG Q.-F.: Casia online and offline chinese handwriting databases. *2011 International Conference on Document Analysis and Recognition* (2011), 37–41. 6

[LZCX18] LIAN Z., ZHAO B., CHEN X., XIAO J.: Easyfont: A style learning-based system to easily build your large-scale handwriting fonts. *ACM Trans. Graph. 38* (2018), 6:1–6:18. 2

[MH08] MAATEN L. V. D., HINTON G.: Visualizing data using t-sne. *Journal of machine learning research 9*, Nov (2008), 2579–2605. 8

[QLW*17] QIU M., LI F.-L., WANG S., GAO X., CHEN Y., ZHAO W., CHEN H., HUANG J., CHU W.: Alime chat: A sequence to sequence and rerank based chatbot engine. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (2017), pp. 498–503. 2

[RLL*17] RAFFEL C., LUONG M.-T., LIU P. J., WEISS R. J., ECK D.: Online and linear-time attention by enforcing monotonic alignments. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), JMLR. org, pp. 2837–2846. 4

[RMC15] RADFORD A., METZ L., CHINTALA S.: Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR abs/1511.06434* (2015). 2

[SIHU19] SUMI T., IWANA B. K., HAYASHI H., UCHIDA S.: Modality conversion of handwritten patterns by cross variational autoencoders. *ArXiv abs/1906.06142* (2019). 2

[SKP15] SCHROFF F., KALENICHENKO D., PHILBIN J.: Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), 815–823. 3

[SMS15] SRIVASTAVA N., MANSIMOV E., SALAKHUDINOV R.: Unsupervised learning of video representations using lstms. In *International conference on machine learning* (2015), pp. 843–852. 2

[SP97] SCHUSTER M., PALIWAL K. K.: Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing 45* (1997), 2673–2681. 4

[SRL*17] SUN D., REN T., LI C., ZHU J., SU H.: Learning to write stylized chinese characters by reading a handful of examples. *ArXiv abs/1712.06424* (2017). 2, 3

[SZ14] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014). 2

[Tia17] TIAN Y.: zi2zi: Master chinese calligraphy with conditional adversarial networks. https://kaonashi-tyc.github.io/2017/04/06/zi2zi.html, 2017. 2

[TXL*19] TANG S., XIA Z., LIAN Z., TANG Y., XIAO J.: Fontrnn: Generating large-scale chinese fonts via recurrent neural network. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 567–577. 2, 4, 6, 9

[WCLL18] WANG F., CHENG J., LIU W., LIU H.: Additive margin softmax for face verification. *IEEE Signal Processing Letters 25* (2018), 926–930. 3, 4, 5, 8

[WGL20] WANG Y., GAO Y., LIAN Z.: Attribute2font: Creating fonts you want from attributes. *arXiv preprint arXiv:2005.07865* (2020). 2

[WWZ*18] WANG H. J., WANG Y., ZHOU Z.-F., JI X., LI Z., GONG D., ZHOU J., LIU W.: Cosface: Large margin cosine loss for deep face recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), 5265–5274. 3, 4, 5, 8

[WZLQ16] WEN Y., ZHANG K., LI Z., QIAO Y.: A discriminative feature learning approach for deep face recognition. In *ECCV* (2016). 5

[XJJL09] XU S., JIN T., JIANG H., LAU F. C. M.: Automatic generation of personal chinese handwriting by capturing the characteristics of personal handwriting. In *IAAI* (2009). 2

[XLG*17] XU A., LIU Z., GUO Y., SINHA V., AKKIRAJU R.: A new chatbot for customer service on social media. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (2017), ACM, pp. 3506–3510. 2

[YLLL14] YI D., LEI Z., LIAO S., LI S. Z.: Learning face representation from scratch. *ArXiv abs/1411.7923* (2014). 5

[ZBL17] ZHANG X.-Y., BENGIO Y., LIU C.-L.: Online and offline handwritten chinese character recognition: A comprehensive study and new benchmark. *Pattern Recognition 61* (2017), 348–360. 1

[ZPIE17] ZHU J.-Y., PARK T., ISOLA P., EFROS A. A.: Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on* (2017). 2

[ZQD*21] ZHUANG F., QI Z., DUAN K., XI D., ZHU Y., ZHU H., XIONG H., HE Q.: A comprehensive survey on transfer learning. *Proceedings of the IEEE 109* (2021), 43–76. 3

[ZTY*20] ZHAO B., TAO J., YANG M., TIAN Z., FAN C., BAI Y.: Deep imitator: handwriting calligraphy imitation via deep attention networks. *Pattern Recognition* (2020), 107080. 2, 9

[ZYZ*16] ZHANG X.-Y., YIN F., ZHANG Y.-M., LIU C.-L., BENGIO Y.: Drawing and recognizing chinese characters with recurrent neural network. *IEEE Transactions on Pattern Analysis and Machine Intelligence 40* (2016), 849–862. 2

[ZZC18] ZHANG Y., ZHANG Y., CAI W.: Separating style and content for generalized style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2018). 2