Generating Handwriting via Decoupled Style Descriptors

Atsunobu Kotani[0000−0001−6117−6630], Stefanie Tellex[0000−0002−2905−4075], and James Tompkin[0000−0003−2218−2899]

Brown University

Abstract. Representing a space of handwriting stroke styles includes the challenge of representing both the style of each character and the overall style of the human writer. Existing VRNN approaches to representing handwriting often do not distinguish between these different style components, which can reduce model capability. Instead, we introduce the Decoupled Style Descriptor (DSD) model for handwriting, which factors both character- and writer-level styles and allows our model to represent an overall greater space of styles. This approach also increases flexibility: given a few examples, we can generate handwriting in new writer styles, and also now generate handwriting of new characters across writer styles. In experiments, our generated results were preferred over a state of the art baseline method 88% of the time, and in a writer identification task on 20 held-out writers, our DSDs achieved 89.38% accuracy from a single sample word. Overall, DSDs allows us to improve both the quality and flexibility over existing handwriting stroke generation approaches.

1 Introduction

Producing computational models of handwriting is a deeply *human* and *personal* topic—most people can write, and each writer has a unique style to their script. Capturing these styles flexibly and accurately is important as it determines the space of descriptive expression of the model; in turn, these models define the usefulness of our recognition and generation applications. For deep-learning-based models, our concern is how to architecture the neural network such that we can represent the underlying stroke characteristics of the styles of writing.

Challenges in handwriting representation include reproducing fine detail, generating unseen characters, enabling style interpolation and transfer, and using human-labeled training data efficiently. Across these, one foundational problem is how to succinctly represent both the style variation of each character and the overall style of the human writer—to capture both the variation within an 'h' letterform and the overall consistency with other letterform for each writer.

As handwriting strokes can be modeled as a sequence of points over time, supervised deep learning methods to handwriting representation can use recurrent neural networks (RNNs) [17,2]. This allows consistent capture of style features that are distant in time and, with the use of variational RNNs (VRNNs), allows the diverse generation of handwriting by drawing from modeled distributions.

However, the approach of treating handwriting style as a 'unified' property of a sequence can limit the representation of both character- and writer-level features. This includes specific character details being averaged out to maintain overall writer style, and an reduced representation space of writing styles.

Instead, we explicitly represent 1) writer-, 2) character- and 3) writer-character-level style variations within an RNN model. We introduce a method of Decoupled Style Descriptors (DSD) that models style variations such that character style can still depend on writer style. Given a database of handwriting strokes as timestamped sequences of points with character string labels [32], we learn a representation that encodes three key factors: writer-independent character representations ($\mathbf{C}_h$ for character *h*, $\mathbf{C}_{his}$ for the word *his*), writer-dependent character-string style descriptors ($\mathbf{w}_h$ for character *h*, $\mathbf{w}_{his}$ for the word *his*), and writer-dependent global style descriptors ($\mathbf{w}$ per writer). This allows new sequence generation for existing writers (via new $\mathbf{w}_{she}$), new writer generation via style transfer and interpolation (via new $\mathbf{w}$), and new character generation in the style of existing writers (via new $\mathbf{C}_2$, from only a few samples of character *2* from *any* writer). Further, our method helps to improve generation quality as more samples are provided for projection, rather than tending towards average letterforms in existing VRNN models.

In a qualitative user study, our model's generations were preferred 88% of the time over an existing baseline [2]. For writer classification tasks on a held-out 20-way test, our model achieves accuracy of 89.38% from a single word sample, and 99.70% from 50 word-level samples. In summary, we contribute:

- Decoupled Style Descriptors as a way to represent latent style information;
- An architecture with DSDs to model handwriting, with demonstration applications in generation, recognition, and new character adaptation; and
- A new database—BRUSH (BRown University Stylus Handwriting)—of handwritten digital strokes in the Latin alphabet, which includes 170 writers, 86 characters, 488 common words written by all writers, and 3668 rarer words written across writers.

Our dataset, code, and model will be open source at http://dsd.cs.brown.edu.

2 Related Work

Handwriting modeling methods either handle images, which capture writing appearance, or handle the underlying strokes collected via digital pens. Each may be online, where observation happens along with writing, or offline. Offline methods support historical document analysis, but cannot capture the motion of writing. We consider an online stroke-based approach, which avoids the stroke extraction problem and allows us to focus on modelling style variation. Work also exists in the separate problem of typeface generation [12,5,31,24,41].

General style transfer methods. Current state-of-the-art style transfer works use a part of the encoded reference sample as a style component, e.g., the output of a CNN encoder for 2D images [26,29], or the last output of an LSTM for

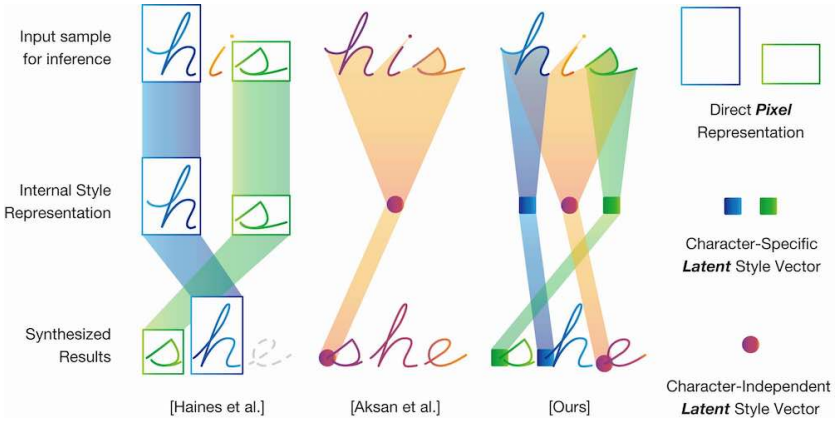


Fig. 1: Illustrating synthesis approaches. Given test sample *his* for reference, we wish to generate *she* in the same style. Left: Pixels of h and s are copied from input with a slight modification [19]; however, this fails to synthesize e as it is missing in the reference. Middle: A global latent writer style is inferred from *his* and used as the initial state for LSTM generation [2]. Right: Our approach infers both character and writer style vectors to improve quality and flexibility.

speech audio [34]. These can be mixed to allocate parts of a conditioning style vector to disentangled variation [25]. Common style representations often cannot capture small details, with neural networks implicitly filtering out this content information, because the representations fail to structurally decouple style from content in the style reference source. Other approaches [27,38] tackle this problem by making neural networks predict parameters for a transformation model (an idea that originates from neuroevolution [36,18]); our $\mathbf{C}$ prediction is related.

Recent image-based offline methods. Haines et al. produced a system to synthesize new sentences in a specific style inferred from source images [19]. Some human intervention is needed during character segmentation, and the model can only recreate characters that were in the source images. Alonso et al. addressed the labeling issue with a GAN-based approach [16,3]; however, their model presents an image quality trade-off and struggles to generate new characters. There are also studies on typeface generation from few reference data [4,37]: Baluja generates typefaces for Latin alphabets [6], and Lian et al. for Chinese [30]. Our method does not capture writing implement appearance, but does provides underlying stroke generation and synthesizes new characters from few examples.

Stroke-based online methods. Deep learning methods, such as Graves' work, train RNN models conditioned on target characters [17,13,40]. The intra-variance of a writer's style was achieved with Mixture Density Networks (MDN) as the final synthesis layer [10]. Berio et al. use recurrent-MDN for graffiti style transfer [9].

Table 1: Property comparison of state-of-the-art handwriting generation models.

Method	Style transfer?	No human segmentation?	Infinite variations?	Synthesize missing samples?	Benefit from more samples?	Smooth interpolation?	Learn new characters?
Graves (2013)	No	Yes	Yes	No	No	No	No
Berio et al. (2017)	Yes	Yes	Yes	No	No	Sort of	No
Haines et al. (2017)	Yes	No	Sort of	No	Yes	No	No
Aksan et al. (2018)	Yes	Sort of	Yes	Yes	No	Sort of	No
Ours	Yes	Yes	Yes	Yes	Yes	Yes	Yes

However, these methods cannot learn to represent handwriting styles per writer, and so cannot perform writer style transfer.

State-of-the-art models can generate characters in an inferred style [2]. Aksan et al.'s DeepWriting model uses Variational Recurrent Neural Networks (VRNN) [15] and assumes a latent vector z that controls writer handwriting style. Across writers, this method tends to average out specific styles and so reduces detail. Further, while sample efficient, VRNN models have trouble exploiting an abundance of inference samples because the style representation is only the last hidden state of an LSTM. We avoid this limitation by extracting character-dependent style vectors from samples and querying them as needed in generation.

Sequence methods beyond handwriting. Learning-based architectures for sequences were popularized in machine translation [14], where the core idea is to encode sequential inputs into a fixed-length latent representation. Likewise, text-to-speech processing has been improved by sequence models [33,35], with extensions to style representation for speech-related tasks like speaker verification and voice conversion. Again, one approach is to use the (converted) last output of an LSTM network as a style representation [21]. Other approaches [22,23] models multiple stylistic latent variables in a hierarchical manner and introduces an approach to transfer styles within a standard VAE setting [28].

Broadly, variational RNN approaches [2,15,22] have the drawback that they are incapability of improving generation performance with more inference samples. While VRNNs are sample efficient when only a few samples are available for style inference, a system should also generate better results as more inference samples are provided (as in [19]). Our method attempts to be scalable and sample efficient through learning decoupled underlying generation factors.

We compare properties of four state of the art handwriting synthesis models (Tab. 1), and illustrate two of their different approaches (Fig. 1).

3 Method

Input, preprocess, and output. A stroke sequence $\mathbf{x} = (p_1, \ldots, p_N)$ has each p_t store the change in x- and y-axis from the previous timestep ($\Delta x_t = x_t - x_{t-1}$, $\Delta y_t = y_t - y_{t-1}$), and a binary termination flag for the 'end of stroke' ($eos = \{0, 1\}$). This creates an $(N, 3)$ matrix. A character sequence $\mathbf{s} = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$

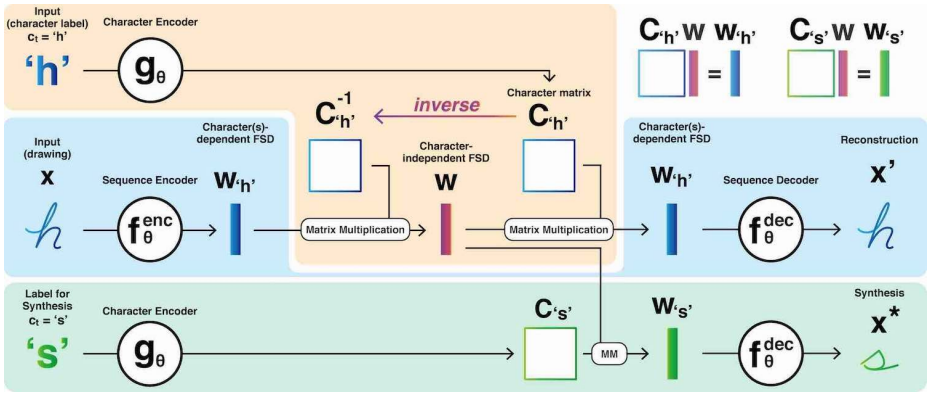


Fig. 2: High-level architecture. Circles are parametrized function approximators, and rectangles/squares are variables. *Blue region*: Encoder-decoder architecture. *Orange region*: Character-conditioned layers. *Green region*: Synthesis procedure.

contains character vectors $\mathbf{c}_t$ where each is a one-hot vector of length equal to the total number of characters considered. This similarly is an (M, Q) matrix.

The IAM dataset [32] and our stroke dataset were collected by asking participants to naturally write character sequences or words, which often produces cursive writing. As such, we must solve a segmentation problem to attribute stroke points to specific characters in $\mathbf{s}$. This is complex; we defer explanation to our supplemental. For now, it is sufficient to say that we use unsupervised learning to train a segmentation network $k_\theta(\mathbf{x}, \mathbf{s})$ to map regions in $\mathbf{x}$ to characters, and to demark 'end of character' labels ($eoc = \{0, 1\}$) for each point.

As output, we wish to predict $\mathbf{x}'$ comprised of $\mathbf{p}'_t$ with: 1) coefficients for Mixture Density Networks [10] $(\pi_t, \mu_x, \mu_y, \sigma_x, \sigma_y, \rho)$, which provide variation in output by sampling Δx_t and Δy_t from these distributions at runtime; 2) 'end of stroke' *eos* probability; and 3) 'end of character' *eoc* probability. This lets us generate cursive writing when *eos* probability is low and *eoc* probability is high.

Decoupled Style Descriptors (DSD). We begin with the encoder-decoder architecture proposed by Cho et al. [14] (Fig. 2, blue region). Given a supervised database $\mathbf{x}, \mathbf{s}$ and a target string c_t, to represent handwriting style we train a parameterized encoder function f^{enc}_θ to learn writer-dependent character-dependent latent vectors $\mathbf{w}_{c_t}$. Then, given $\mathbf{w}_{c_t}$, we simultaneously train a parameterized decoder function f^{dec}_θ to predict the next point p'_t given all past points $p'_{1:t-1}$. Both encoder and decoder f_θ are RNNs such as LSTM models:

$$p'_t = f^{\text{dec}}_\theta(p'_{1:t-1} | \mathbf{w}_{c_t}). \tag{1}$$

This method does not factor character-independent writer style; yet, we have no way of explicitly describing this property via supervision and so we must devise a construction to learn it implicitly. Thus, we add a layer of abstraction (Fig. 2, orange region) with three assumptions:

1. If two stroke sequences $\mathbf{x}_1$ and $\mathbf{x}_2$ are written by the same writer, then consistency in their writing style is manifested by a character-independent writer-dependent latent vector $\mathbf{w}$.
2. If two character sequences $\mathbf{s}_1$ and $\mathbf{s}_2$ are written by different writers, then consistency in their stroke sequences is manifested by a character-dependent writer-independent latent matrix $\mathbf{C}$. $\mathbf{C}$ can be estimated via a parameterized encoder function g_θ, which is also an RNN such as an LSTM:

$$\mathbf{C}_{c_t} = g_\theta(\mathbf{s}, c_t). \tag{2}$$

3. $\mathbf{C}_{c_t}$ instantiates a writer's style $\mathbf{w}$ to draw a character via $\mathbf{w}_{c_t}$, such that $\mathbf{C}_{c_t}$ and $\mathbf{w}$ are latent factors:

$$\mathbf{w}_{c_t} = \mathbf{C}_{c_t}\mathbf{w}, \tag{3}$$

$$\mathbf{w} = \mathbf{C}_{c_t}^{-1}\mathbf{w}_{c_t}. \tag{4}$$

This method assumes that $\mathbf{C}_{c_t}$ is invertible, which we will demonstrate in Sec. 4. Intuitively, the multiplication of writer-dependent character vectors $\mathbf{w}_{c_t}$ with the inverse of character-DSD $\mathbf{C}_{c_t}^{-1}$ (Eq. 4) factors out character-dependent information from writer-dependent information in $\mathbf{w}_{c_t}$ to extract a writer style representation $\mathbf{w}$. Likewise, Eq. 3 restores writer-dependent character $\mathbf{w}_{c_t}$ by multiplying the writer-specific style $\mathbf{w}$ with a relevant character-DSD $\mathbf{C}_{c_t}$.

We use this property in synthesis (Fig. 2, green region). Given a target character c_t, we use encoder g_θ to generate a $\mathbf{C}$ matrix. Then, we multiply $\mathbf{C}_{c_t}$ by a desired writer style $\mathbf{w}$ to generate $\mathbf{w}_{c_t}$. Finally, we use trained decoder f_θ^{dec} to create a new point p'_t given previous points $p'_{1:t-1}$:

$$p'_t = f_\theta^{\text{dec}}(p'_{1:t-1}|\mathbf{w}_{c_t}), \text{ where } \mathbf{w}_{c_t} = \mathbf{C}_{c_t}\mathbf{w}. \tag{5}$$

Interpreting the linear factors. Eq. 3 states a linear relationship between $\mathbf{C}_{c_t}$ and $\mathbf{w}$. This exists at the latent representation level: $\mathbf{w}_{c_t}$ and $\mathbf{C}_{c_t}$ are separately approximated by independent neural networks f_θ^{enc} and g_θ, which themselves are nonlinear function approximators [27,38]. As $\mathbf{C}_{c_t}$ maps a vector $\mathbf{w}$ to another vector $\mathbf{w}_{c_t}$, we can consider $\mathbf{C}_{c_t}$ to be a fully-connected neural network layer (without bias). However, unlike standard layers, $\mathbf{C}_{c_t}$'s weights are not implicitly learned through backpropagation but are predicted by a neural network g_θ in Eq. 2. A further interpretation of $\mathbf{C}_{c_t}$ and $\mathbf{C}_{c_t}^{-1}$ as two layers of a network is that they respectively share a set of weights and their inverse. Explicitly forming $\mathbf{C}_{c_t}$ in this linear way makes it simple to estimate $\mathbf{C}_{c_t}$ for *new* characters that are not in the training dataset, given few sample pairs of $\mathbf{w}_{c_t}$ and $\mathbf{w}$, using standard linear least squares methods (Sec. 4).

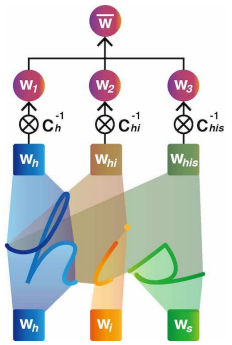
Mapping character and stroke sequences with f_θ and g_θ. Next, we turn our attention to how we map sequences of characters and strokes within our function approximators. Consider the LSTM f_θ^{enc}: Given a character sequence $\mathbf{s}$ as size of (M, Q) where M is the number of characters, and a stroke sequence $\mathbf{x}$ of size

$(N, 3)$ where N is the number of points, our goal is to obtain a style vector for each character $\mathbf{w}_{c_t}$ in that sequence. The output of our segmentation network k_θ preprocess defines 'end of character' bits, and so we know at which point in $\mathbf{x}$ that a character switch occurs, e.g., from h to e in *hello*.

First, we encode $\mathbf{x}$ using f_θ^{enc} to obtain a $\mathbf{x}^*$ of size (N, L), where L is the latent feature dimension size (we use 256). Then, from $\mathbf{x}^*$, we extract M vectors at these switch indices—these are our writer-dependent character-dependent DSDs $\mathbf{w}_{c_t}$. As f_θ^{enc} is an LSTM, the historical sequence data up to that index is encoded within the vector at that index (Fig. 3, top). For instance, for *his*, $\mathbf{x}^*$ at switch index 2 represents how the writer writes the first two characters *hi*, i.e., $\mathbf{w}_{hi}$. We refer to these $\mathbf{w}_{c_t}$ as 'writer-character-DSDs'.

Likewise, LSTM g_θ takes a character sequence $\mathbf{s}$ of size (M, Q) and outputs an array of $\mathbf{C}$ matrices that forms a tensor of size (M, L, L) and preserves sequential dependencies between characters: The i-th element of the tensor $\mathbf{C}_{c_i}$ is a matrix of size (L, L)—that is, it includes information about previous characters up to and including the i-th character. Similar to $\mathbf{x}^*$, for *his*, the second character matrix $\mathbf{C}_{c_2}$ contains information about the first two characters *hi*—$\mathbf{C}$ is really a character sequence matrix. Multiplying character information $\mathbf{C}_{c_t}$ with writer style vector $\mathbf{w}$ creates a writer-character-DSD $\mathbf{w}_{c_t}$.

Estimating $\mathbf{w}$. When we encode a stroke sequence $\mathbf{x}$ that draws $\mathbf{s}$ characters via f_θ^{enc}, we extract M character(s)-dependent DSDs $\mathbf{w}_{c_t}$ (e.g., $\mathbf{w}_h$, $\mathbf{w}_{hi}$ and $\mathbf{w}_{his}$, *right*). Via Eq. 4, we obtain M distinct candidates for writer-DSDs $\mathbf{w}$. To overcome this, for each sample, we simply take the mean to form $\mathbf{w}$:



$$\overline{\mathbf{w}} = \frac{1}{M} \sum_{t=1}^{M} \mathbf{C}_{c_t}^{-1} \mathbf{w}_{c_t}. \tag{6}$$

Generation approaches via $\mathbf{w}_{c_t}$. Consider the synthesis task in Fig. 1: given our trained model, generate how a new writer would write *she* given a reference sample of them writing *his*. From the *his* sample, we can extract 1) segment-level writer-character-DSDs ($\mathbf{w}_h$, $\mathbf{w}_i$, $\mathbf{w}_s$), and 2) the global $\overline{\mathbf{w}}$. To synthesize *she*, our model must predict three writer-character-DSDs $(\mathbf{w}_s, \mathbf{w}_{sh}, \mathbf{w}_{she})$ as input to the decoder f_θ^{dec}. We introduce two methods to estimate $\mathbf{w}_{c_t}$:

$$\text{Method } \alpha : \mathbf{w}_{c_t}^{\alpha} = \mathbf{C}_{c_t} \overline{\mathbf{w}} \tag{7a}$$

$$\text{Method } \beta : \mathbf{w}_{c_t}^{\beta} = h_\theta([\mathbf{w}_{c_1}, \ldots, \mathbf{w}_{c_t}]) \tag{7b}$$

where h_θ is an LSTM that restore dependencies between temporally-separated writer-character-DSDs as illustrated in Fig. 3, green rectangle. We train our model to reconstruct $\mathbf{w}_{c_t}$ both ways. This allows us to use method α when test reference samples do not include target characters, e.g., *his* is missing an e for *she*, and so we can reconstruct $\mathbf{w}_e$ via $\mathbf{w}$ and $\mathbf{C}_e$ (Fig. 3, right). It also allows us to use Method β when test reference samples include relevant characters that,

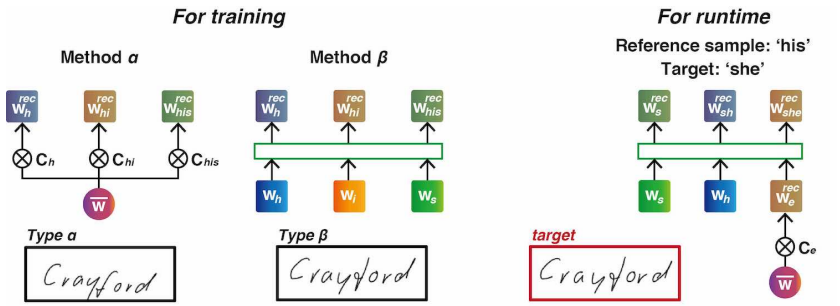


Fig. 3: Reconstruction methods to produce writer-character-DSD $\mathbf{w}_{c_t}$, with training sample $\mathbf{s}, \mathbf{x}$ of *his* and test sample $\mathbf{s}$ of *she*. Green rectangle is h_θ as defined in Equation 7b. *Training*: Method α multiplies writer style $\overline{\mathbf{w}}$ with each character string matrix $\mathbf{C}_{c_t}$. Method β restore temporal dependencies of segment-level writer-character-DSDs ($\mathbf{w}_h$, $\mathbf{w}_i$, $\mathbf{w}_s$) via an LSTM , which produces higher-quality results that are preferred by users (Sec. 4). Target test image is in red. *Runtime*: Both prediction model Method α and β are combined to synthesize a new sample given contents within the reference sample.

via f_θ^{enc}, provide writer-character-DSDs, e.g., *his* contains *s* and *h* in *she* and so we can estimate $\mathbf{w}_s$ and $\mathbf{w}_h$. As these characters could come from any place in the reference samples, h_θ restores the missing sequence dependencies.

3.1 Training losses

We defer full architecture details for our supplemental material, and here explain our losses. We begin with a point location loss $\mathscr{L}^{loc}$ on predicted shifts in x, y coordinates, $(\Delta x, \Delta y)$. As we employ mixture density networks as a final prediction layer in f_θ^{dec}, we try to maximize the probability for the target shifts $(\Delta x^*, \Delta y^*)$ as explained by Graves et al. [17]:

$$\mathscr{L}^{loc} = -\sum_t \log\Big(\sum_j \pi_t^j \mathcal{N}(\Delta x_t^*, \Delta y_t^* | \mu_{x_t}^j, \mu_{y_t}^j, \sigma_{x_t}^j, \sigma_{y_t}^j, \rho_t^j)\Big).$$

Further, we consider 'end of sequence' flags *eos* and 'end of character' flags *eoc* by computing binary cross-entropy losses $\mathscr{L}^{eos}, \mathscr{L}^{eoc}$ for each.

Next, we consider consistency in predicting writer-DSD $\mathbf{w}$ from different writer-character-DSDs $\mathbf{w}_{c_t}$. We penalize a loss $\mathscr{L}^{\mathbf{w}}$ that minimizes the variance in $\mathbf{w}_t$ in Equation 6:

$$\mathscr{L}^{\mathbf{w}} = \sum_t (\overline{\mathbf{w}} - \mathbf{w}_t)^2 \tag{8}$$

Further, we penalize the reconstruction of each writer-character-DSD. We compare the writer-character-DSD retrieved by f_θ^{enc} from inference samples as $\mathbf{w}_{c_t}$ to their reconstructions ($\mathbf{w}_{c_t}^\alpha$, $\mathbf{w}_{c_t}^\beta$) via generation Methods α and β:

$$\mathscr{L}_{A\in(\alpha,\beta)}^{\mathbf{w}_{c_t}} = \sum_t (\mathbf{w}_{c_t} - \mathbf{w}_{c_t}^A)^2 \tag{9}$$

When $t = 1$, $\mathscr{L}_{\beta}^{\mathbf{w}_{c_1}} = (\mathbf{w}_{c_1} - h_\theta(\mathbf{w}_{c_1}))^2$. As such, minimizing this loss prevents h_θ in generation Method β from diluting the style representation $\mathbf{w}_{c_1}$ generated by f_θ^{enc} because h_θ is induced to output $\mathbf{w}_{c_1}$.

Each loss can be computed for three types of writer-character-DSD $\mathbf{w}_{c_t}$: those predicted by f_θ^{enc}, Method α, and Method β. These losses can also be computed at character, word, and sentence levels, e.g., for words:

$$\mathscr{L}_{word} = \sum_{A \in (f_\theta^{\text{enc}}, \alpha, \beta)} \left(\mathscr{L}_A^{loc} + \mathscr{L}_A^{eos} + \mathscr{L}_A^{eoc} + \mathscr{L}_A^{\mathbf{w}} + \mathscr{L}_A^{\mathbf{w}_{c_t}} \right). \tag{10}$$

Thus, the total loss is: $\mathscr{L}_{total} = \mathscr{L}_{char} + \mathscr{L}_{word} + \mathscr{L}_{sentence}$.
$\mathscr{L}_{f_\theta^{\text{enc}}}^{\mathbf{w}_{c_t}} = 0$ by construction from Equation 9; we include it here for completeness.

Sentence-level losses help to make the model predict spacing between words. While our model could train just with character- and word-level losses, this would cause a problem if we ask the model to generate a sentence from a reference sample of a single word. Training with $\mathscr{L}_{sentence}$ lets our model predict how a writer would space words based on their writer-DSD $\mathbf{w}$.

Implicit $\mathbf{C}$ *inverse constraint.* Finally, we discuss how $\mathscr{L}^{\mathbf{w}_{c_t}}$ at the character level implicitly constrains character-DSD $\mathbf{C}$ to be invertible. If we consider a single character sample, then mean $\overline{\mathbf{w}}$ in Equation 6 is equal to $\mathbf{C}_{c_1}^{-1}\mathbf{w}_{c_1}$. In this case, as $\mathbf{w}_{c_t}^{\alpha} = \mathbf{C}_{c_t}\overline{\mathbf{w}}$ (Eq. 7a), $\mathscr{L}_{\alpha}^{\mathbf{w}_{c_t}}$ becomes:

$$\mathscr{L}_{\alpha}^{\mathbf{w}_{c_t}} = (\mathbf{w}_{c_1} - \mathbf{C}_{c_1}\mathbf{C}_{c_1}^{-1}\mathbf{w}_{c_1})^2 \tag{11}$$

This value becomes nonzero when $\mathbf{C}$ is singular ($\mathbf{C}\mathbf{C}^{-1} \neq \mathbf{I}$), and so our model avoids non-invertible $\mathbf{C}$s.

Training through inverses. As we train our network end-to-end, our model must backpropagate through $\mathbf{C}_{c_t}$ and $\mathbf{C}_{c_t}^{-1}$. As derivative of matrix inverses can be obtained with $\frac{d\mathbf{C}^{-1}}{dx} = -\mathbf{C}^{-1}\frac{d\mathbf{C}}{dx}\mathbf{C}^{-1}$, our model can train.

4 Experiments

Dataset. Our new dataset—BRUSH—provides characteristics that other online English handwriting datasets do not, including the typical online English handwriting dataset IAM [32]. First, we explicitly display a baseline in every drawing box during data collection. This enables us to create handwriting samples whose initial action is the x, y shift from the baseline to the starting point. This additional information might also help improve performance in recognition tasks.

Second, our 170 individuals wrote 488 words *in common* across 192 sentences. This helps to evaluate handwriting models and observe whether $\mathbf{w}$ and $\mathbf{C}$ are decoupled: given a sample that failed to generate, we can compare the generated results of the same word across writers. If writer A failed but B succeeded, then it is likely that the problem is not with $\mathbf{C}$ representations but with either $\mathbf{w}$ or

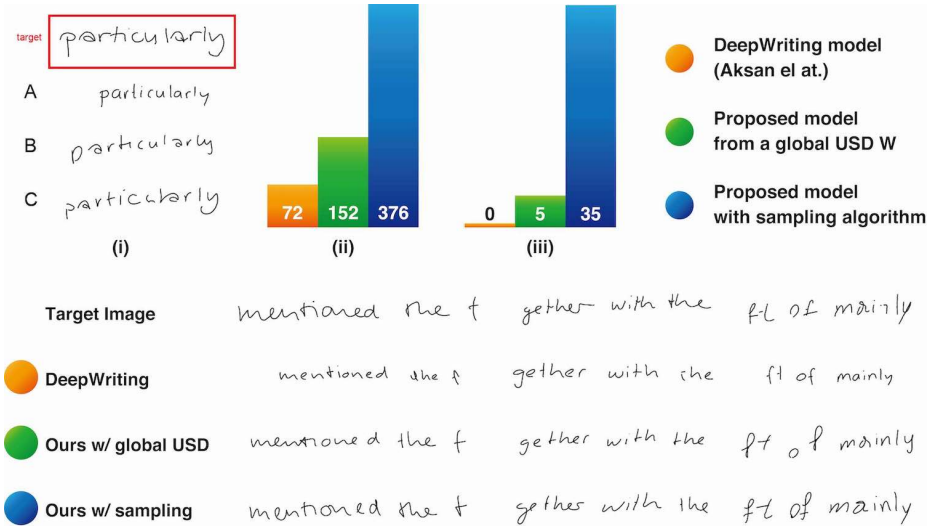


Fig. 4: Comparison of our proposed model vs. the state-of-the-art model [2]. *Top:* (i) Example writing similarity ordering task assigned to MTurk workers. (ii) Counts of most similar results with the target image. (iii) Sample-level vote. *Bottom:* Three examples of task orderings; see supplemental for all 40. The model of Aksan et al. [2] typically over-smooths the style and loses key details.

$\mathbf{w}_{c_t}$. If both A and B failed to draw the word but succeeding in generating other words, it is likely that $\mathbf{C}$ or $\mathbf{w}_{c_t}$ representations are to blame. We provide further details about our dataset and collection process in our supplemental material.

Third, for DeepWriting [2] comparisons, we use their training and test splits on IAM that mix writer identities—i.e., in training, we see some data from every writer. For all other experiments, we use our dataset, where we split between writers—our 20 test writers have never been seen writing *anything* in training.

Invertibility of $\mathbf{C}$. To compute $\mathbf{w}$ in Equation 4, we must invert the character-DSD $\mathbf{C}$. Our network is designed to make $\mathbf{C}$ invertible as training proceeds by penalizing a reconstruction loss for $\mathbf{w}_{c_t}$ and $\mathbf{C}_{c_t}\mathbf{C}_{c_t}^{-1}\mathbf{w}_{c_t}$ (Sec. 3.1). To test its success, we compute $\mathbf{C}$s from our model for all single characters (86 characters) and character pairs ($86^2 = 7,396$ cases), and found $\mathbf{C}$ to have full rank in each case. Next, we test all possible 3-character-strings ($86^3 = 636,056$ cases). Here, there were 37 rare cases with non-invertible $\mathbf{C}$s, such as *1Zb* and *6ak*. In these cases, we can still extract two candidate $\mathbf{w}$ from the first two characters (e.g., *1* and *1Z* in the *1Zb* sample) to complete generation tasks.

Qualitative evaluation with users. We use Amazon Mechanical Turk to asked 25 participants to rank generated handwriting similarity to a target handwriting (Fig. 4 (i)). We randomly selected 40 sentence-level target handwriting samples

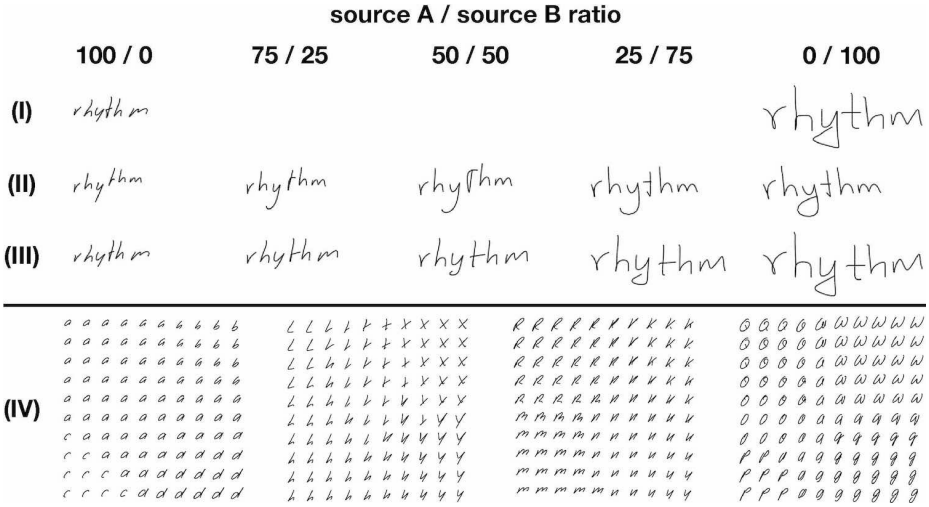


Fig. 5: Interpolation at different levels. (I) Original samples by two writers. (II) At the writer-DSD $\mathbf{w}$ level. (III) At the writer-character-DSD $\mathbf{w}_{c_t}$ level. (IV) At $\mathbf{C}$ level. *Left to right:* Characters used are *abcd*, *Lxhy*, *Rkmy*, *QWPg*.

from the validation set of IAM dataset [32]. Each participant saw randomly-shuffled samples; in total, 600 assessments were made. We compared the abilities of three models to generate the same handwriting style without seeing the actual target sample. We compare to the state-of-the-art DeepWriting model [2], which uses a sample from the same writer (but of a different character sequence) for style inference. We test both Methods α and β from our model. Method α uses the same sample to predict $\mathbf{w}$ and to generated a new sample. Method β randomly samples 10 sentence-level drawing by the target writer and creates a sample with the algorithm discussed in Sec. 3. DeepWriting cannot take advantage of any additional character samples at inference time because it estimates only a single character-independent style vector.

Figure 4 (ii) displays how often each model was chosen as the most similar to the target handwriting; our model with sampling algorithm was selected 5.22× as often as Aksan et al.'s model. Figure 4 (iii) displays which model was preferred across the 40 cases: of the 15 assessments per case, we count the number of times each model was the most popular. We show all cases in supplemental material.

Interpolation of $\mathbf{w}$*,* $\mathbf{w}_{c_t}$*, and* $\mathbf{C}$*.* Figure 5 demonstrates that our method can interpolate (II) at the writer-DSD $\mathbf{w}$ level, (III) at the writer-character-DSD $\mathbf{w}_{c_t}$ level, and (IV) at the character-DSD $\mathbf{C}$ level. Given two samples of the same word by two writers $\mathbf{x}^A$ and $\mathbf{x}^B$, we first extract writer-character-DSDs from each sample (e.g., $\mathbf{w}^A_{rhy}$, $\mathbf{w}^B_{rhythm}$), then we derive writer-DSDs $\overline{\mathbf{w}^A}$ and $\overline{\mathbf{w}^B}$ as in Sec. 3. To interpolate by γ between two writers, we compute the weighted average $\overline{\mathbf{w}^C} = \gamma\overline{\mathbf{w}^A} + (1-\gamma)\overline{\mathbf{w}^B}$. Finally, we reconstruct writer-character-DSDs from

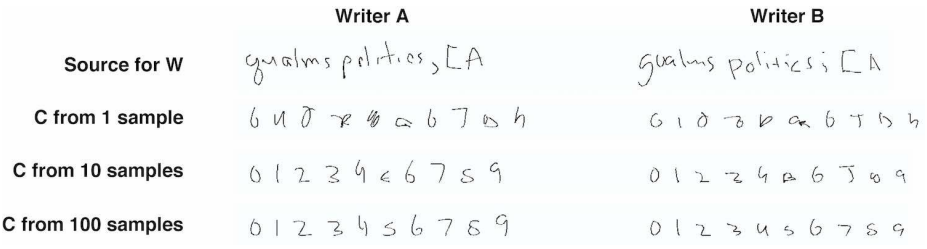


Fig. 6: Predicting $\mathbf{C}$ from new character samples, given a version of our model that is not trained on numbers. As we increase the number of samples used to estimate $\mathbf{C}$, the better the stylistic differences are preserved when multiplying with $\mathbf{w}$s from different writers A and B. *Note:* neither writers provided numeral samples; by our construction, samples can come from any writer.

$\overline{\mathbf{w}^C}$ (e.g., $\mathbf{w}^C_{rhy} = \mathbf{C}_{rhy}\overline{\mathbf{w}^C}$) and feed this into f^{dec}_θ to generate a new sample. For (III), we simply interpolate at the sampled character-level (e.g., $\mathbf{w}^A_{rhy}$ and $\mathbf{w}^B_{rhy}$). For (IV), we bilinearly interpolate four character-DSDs $\mathbf{C}_{c_t}$ placed at the corners of each image: $\overline{\mathbf{C}} = (r_A \times \mathbf{C}_A + r_B \times \mathbf{C}_B + r_C \times \mathbf{C}_C + r_D \times \mathbf{C}_D)$, where all r sum to 1. From $\overline{\mathbf{C}}$, we compute a writer-character-DSD as $\mathbf{w}_{\overline{c}} = \overline{\mathbf{C}}W$ and synthesize a new sample. In each case (II-IV), our representations are smooth.

Synthesis of new characters. Our approach allows us to generate handwriting for new characters from a few samples from any writer. Let us assume that writer A produces a new character sample *3* that is not in our dataset. To make *3* available for generation in other writer styles, we need to recover the character-DSD $\mathbf{C}_{\mathit{3}}$ that represents the shape of the character *3*. Given $\mathbf{x}$ for newly drawn character *3*, encoder f^{enc}_θ first extracts the writer-character-DSD $\mathbf{w}_{\mathit{3}}$. Assuming that writer A provided other non-*3* samples in our dataset, we can compute multiple writer-DSD $\mathbf{w}$ for A. This lets us solve for $\mathbf{C}_{\mathit{3}}$ using least squares methods. We form matrices $\mathbf{Q}, \mathbf{P}_{\mathit{3}}$ where each column of $\mathbf{Q}$ is one specific instance of $\mathbf{w}$, and where each column of $\mathbf{P}_{\mathit{3}}$ is one specific instance of $\mathbf{w}_{\mathit{3}}$. Then, we minimize the sum of the squared error, which is the Frobenius norm $\|\mathbf{C}_{\mathit{3}}\mathbf{Q} - \mathbf{P}_{\mathit{3}}\|^2_F$, e.g., via $\mathbf{C}_{\mathit{3}} = \mathbf{P}_{\mathit{3}}\mathbf{Q}^+$.

As architectured (and detailed in supplemental), g_θ actually has two parts: an LSTM encoder g^{LSTM}_θ that generates a 256×1 character representation vector $\mathbf{c}^{\text{raw}}_{c_t}$ for a substring c_t, and a fully-connected layer g^{FC2}_θ that expands $\mathbf{c}^{\text{raw}}_{c_t}$ and reshapes it into a 256×256 matrix $\mathbf{C}_{c_t} = g^{\text{FC2}}_\theta(\mathbf{c}^{\text{raw}}_{c_t})$. Further, as the output of an LSTM, we know that $\mathbf{c}^{\text{raw}}_{c_t}$ should be constrained to values $[-1, +1]$. Thus, for this architecture, we directly optimize the (smaller set of) parameters of the latent vector $\mathbf{c}^{\text{raw}}_{c_t}$ to create $\mathbf{C}_{c_t}$ given the pre-trained fully-connected layer weights, using a constrained non-linear optimization algorithm (L-BFGS-B) via the objective $f(\mathbf{c}^{\text{raw}}_{c_t}) = \|\mathbf{P}_{\mathit{3}} - g^{\text{FC2}}_\theta(\mathbf{c}^{\text{raw}}_{c_t})\mathbf{Q}\|^2_F$.

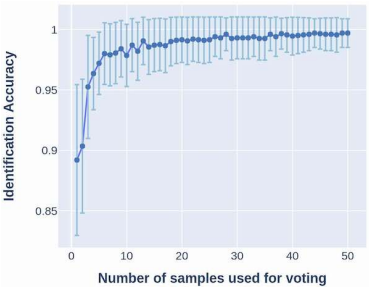
To examine this capability of our approach, we retrained our model with a modified dataset that *excluded* numbers. In Figure 6, we see generation using our

estimate of new Cs from different sample counts. We can generate numerals in the style of a particular writer even though they never drew them, using relatively few drawing samples of the new characters from *different* writers.

Writer recognition task. Writer recognition systems try to assign test samples (e.g., a page of handwriting) to a particular writer given an existing database. Many methods use codebook approaches [8,11,20,7] to catalogue characteristic patterns such as graphemes, stroke junctions, and key-points from offline handwriting images and compare them to test samples. Zhang et al. [39] extend this idea to online handwriting, and Adak et al. study idiosyncratic character style per person and extract characteristic patches to identify the writer [1].

To examine how well our model might represent the latent distribution of handwriting styles, we perform a writer recognition task on our trained model on the randomly-selected 20-writer hold out set from our dataset. First, we compute 20 writer DSDs $\overline{\mathbf{w}}_i^{writer}$ from 10 sentence-level samples—this is our offline 'codebook' representing the style of each writer. Then, for testing, we sample from 1–50 new word-level stroke sequences per writer (using words with at least 5 characters), and calculate the corresponding writer DSDs ($N = 1,000$ in total). With the vector L of true writer labels, we compute prediction accuracy:

$$A = \frac{1}{N}\sum_{i=1}^{N} I(L_i, \arg\min_j (\overline{\mathbf{w}}_i^{word} - \overline{\mathbf{w}}_j^{writer})^2),\ I(x,y) = \begin{cases} 1, & \text{if } x = y \\ 0, & \text{otherwise} \end{cases} \quad (12)$$



We repeated the random sampling of 1–50 words over 100 trials and compute mean accuracy and standard error. When multiple test samples are provided, we predict writer identity for each word and average their predictions. Random accuracy performance is 5%. Our test prediction accuracy rises from $89.20\% \pm 6.23$ for one word sample, to $97.85\% \pm 2.57$ for ten word samples, to $99.70\% \pm 1.18$ for 50 word samples. Increasing the number of test samples per writer increases accuracy because some words may not be as characteristic as others (e.g., 'tick' vs. 'anon'). Overall, while our model was not trained to perform this classification task, we can still achieved promising accuracy results from few samples—this is an indication that our latent space is usefully descriptive.

Additional experiments. In our supplemental material, along with more architecture, model training procedure, and sampling algorithm details, we also: 1) compare to two style extraction pipelines, a stacked FC+ReLU layers and AdaIN, and find our approach more capable; 2) demonstrate the importance of learning style and content of character-DSD $\mathbf{C}$ by comparing with a randomly-initialized version; 3) ablate parts of our loss function, and illustrate key components; 4) experimentally show that our model is more efficient than DeepWriting by comparing generation given the same number of model parameters.

5 Discussion

While users preferred our model in our study, it still sometimes fails to generate readable letters or join cursive letters. One issue here is the underlying inconsistency in human writers, which we only partially capture in our data and represent in our model (e.g., cursive inconsistency). Another issue is collecting high-quality data with digital pens in a crowdsourced setting, which can still be a challenge and requires careful cleaning (see supplemental for more details).

Decoupling additional styles. Our model could potentially scale to more styles. For instance, we might create an age matrix $\mathbf{A}$ from a numerical age value a as $\mathbf{C}$ is constructed from c_t, and extract character-independent age-independent style descriptor as $\mathbf{w}^* = \mathbf{A}^{-1}\mathbf{C}_{c_t}^{-1}\mathbf{w}_{c_t}$. Introducing a new age operator $\mathbf{A}$ invites our model to find latent-style similarities across different age categories (e.g., between a child and a mature writer). Changing the age value and thus $\mathbf{A}$ may predict how a child's handwriting changes as s/he becomes older. However, training multiple additional factors in this way is likely to be challenging.

Alternatives to linear $\mathbf{C}$ *multiplication operator.* Our model can generate new characters by approximating a new $\mathbf{C}$ matrix from few pairs of $\mathbf{w}$ and $\mathbf{w}_{c_t}$ thanks to their linear relationship. However, one might consider replacing our matrix multiplication 'operator' on $\mathbf{C}$ with parametrized nonlinear function approximators, such as autoencoders. Multiplication by $\mathbf{C}^{-1}$ would become an encoder, with multiplication by $\mathbf{C}$ being a decoder; in this way, g_θ would be tasked with predicting encoder weights given some predefined architecture. Here, consistency with $\mathbf{w}$ must still be retained. We leave this for future work.

6 Conclusion

We introduce an approach to online handwriting stroke representation via the Decoupled Style Descriptor (DSD) model. DSD succeeds in generating drawing samples which are preferred more often in a user study than the state-of-the-art model. Further, we demonstrate the capabilities of our model in interpolating samples at different representation levels, recovering representations for new characters, and achieving a high writer-identification accuracy, despite not being trained explicitly to perform these tasks. Online handwriting synthesis is still challenging, particularly when we infer a stylistic representation from few numbers of samples and try to generate new samples. However, we show that decoupling style factors has potential, and believe it could also apply to style-related tasks like transfer and interpolation in other sequential data domains, such as in speech synthesis, dance movement prediction, and musical understanding.

Acknowledgements. This work was supported by the Sloan Foundation and the National Science Foundation under award number IIS-1652561. We thank Kwang In Kim for fruitful discussions and for being our matrix authority. We thank Naveen Srinivasan and Purvi Goel for the ECCV deadline snack delivery service. Finally, we thank all anonymous writers who contributed to our dataset.

References

1. Adak, C., Chaudhuri, B.B., Lin, C.T., Blumenstein, M.: Intra-variable handwriting inspection reinforced with idiosyncrasy analysis (2019)
2. Aksan, E., Pece, F., Hilliges, O.: DeepWriting: Making Digital Ink Editable via Deep Generative Modeling. In: SIGCHI Conference on Human Factors in Computing Systems. CHI '18, ACM, New York, NY, USA (2018)
3. Alonso, E., Moysset, B., Messina, R.O.: Adversarial generation of handwritten text images conditioned on sequences. ArXiv **abs/1903.00277** (2019)
4. Azadi, S., Fisher, M., Kim, V.G., Wang, Z., Shechtman, E., Darrell, T.: Multi-content gan for few-shot font style transfer. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7564–7573 (2018)
5. Balashova, E., Bermano, A.H., Kim, V.G., DiVerdi, S., Hertzmann, A., Funkhouser, T.: Learning a stroke-based representation for fonts. Computer Graphics Forum **38**(1), 429–442 (2019). https://doi.org/10.1111/cgf.13540, `https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13540`
6. Baluja, S.: Learning typographic style: from discrimination to synthesis. Machine Vision and Applications **28**(5-6), 551–568 (2017)
7. Bennour, A., Djeddi, C., Gattal, A., Siddiqi, I., Mekhaznia, T.: Handwriting based writer recognition using implicit shape codebook. Forensic science international **301**, 91–100 (2019)
8. Bensefia, A., Nosary, A., Paquet, T., Heutte, L.: Writer identification by writer's invariants. In: Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition. pp. 274–279. IEEE (2002)
9. Berio, D., Akten, M., Leymarie, F.F., Grierson, M., Plamondon, R.: Calligraphic stylisation learning with a physiologically plausible model of movement and recurrent neural networks. In: Proceedings of the 4th International Conference on Movement Computing. pp. 1–8 (2017)
10. Bishop, C.M.: Mixture density networks (1994)
11. Bulacu, M., Schomaker, L.: Text-independent writer identification and verification using textural and allographic features. IEEE transactions on pattern analysis and machine intelligence **29**(4), 701–717 (2007)
12. Campbell, N.D., Kautz, J.: Learning a manifold of fonts. ACM Transactions on Graphics (TOG) **33**(4), 1–11 (2014)
13. Carter, S., Ha, D., Johnson, I., Olah, C.: Experiments in handwriting with a neural network. Distill (2016). https://doi.org/10.23915/distill.00004, `http://distill.pub/2016/handwriting`
14. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar (Oct 2014). https://doi.org/10.3115/v1/D14-1179, `https://www.aclweb.org/anthology/D14-1179`
15. Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A.C., Bengio, Y.: A recurrent latent variable model for sequential data. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 2980–2988. Curran Associates, Inc. (2015), `http://papers.nips.cc/paper/5653-a-recurrent-latent-variable-model-for-sequential-data.pdf`

16. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
17. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
18. Ha, D., Dai, A., Le, Q.V.: Hypernetworks. arXiv preprint arXiv:1609.09106 (2016)
19. Haines, T.S.F., Mac Aodha, O., Brostow, G.J.: My text in your handwriting. ACM Trans. Graph. **35**(3) (May 2016). https://doi.org/10.1145/2886099, `https://doi.org/10.1145/2886099`
20. He, S., Wiering, M., Schomaker, L.: Junction detection in handwritten documents and its application to writer identification. Pattern Recognition **48**(12), 4036–4048 (2015)
21. Heigold, G., Moreno, I., Bengio, S., Shazeer, N.: End-to-end text-dependent speaker verification. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 5115–5119. IEEE (2016)
22. Hsu, W.N., Glass, J.: Scalable factorized hierarchical variational autoencoder training. arXiv preprint arXiv:1804.03201 (2018)
23. Hsu, W.N., Zhang, Y., Glass, J.: Unsupervised learning of disentangled and interpretable representations from sequential data. In: Advances in Neural Information Processing Systems (2017)
24. Hu, C., Hersch, R.D.: Parameterizable fonts based on shape components. IEEE Computer Graphics and Applications **21**(3), 70–85 (2001)
25. Hu, Q., Szabó, A., Portenier, T., Favaro, P., Zwicker, M.: Disentangling factors of variation by mixing them. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
26. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (Oct 2017)
27. Jia, X., De Brabandere, B., Tuytelaars, T., Gool, L.V.: Dynamic filter networks. In: Advances in neural information processing systems. pp. 667–675 (2016)
28. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
29. Kotovenko, D., Sanakoyeu, A., Lang, S., Ommer, B.: Content and style disentanglement for artistic style transfer. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4422–4431 (2019)
30. Lian, Z., Zhao, B., Chen, X., Xiao, J.: Easyfont: a style learning-based system to easily build your large-scale handwriting fonts. ACM Transactions on Graphics (TOG) **38**(1), 1–18 (2018)
31. Lopes, R.G., Ha, D., Eck, D., Shlens, J.: A learned representation for scalable vector graphics. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
32. Marti, U.V., Bunke, H.: The iam-database: an english sentence database for offline handwriting recognition. International Journal on Document Analysis and Recognition **5**(1), 39–46 (2002)
33. Oord, A.v.d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499 (2016)
34. Qian, K., Zhang, Y., Chang, S., Yang, X., Hasegawa-Johnson, M.: Autovc: Zero-shot voice style transfer with only autoencoder loss. In: International Conference on Machine Learning. pp. 5210–5219 (2019)

35. Shen, J., Pang, R., Weiss, R.J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., et al.: Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 4779–4783. IEEE (2018)
36. Stanley, K.O., D'Ambrosio, D.B., Gauci, J.: A hypercube-based encoding for evolving large-scale neural networks. Artificial life **15**(2), 185–212 (2009)
37. Suveeranont, R., Igarashi, T.: Example-based automatic font generation. In: International Symposium on Smart Graphics. pp. 127–138. Springer (2010)
38. Wang, H., Liang, X., Zhang, H., Yeung, D.Y., Xing, E.P.: Zm-net: Real-time zero-shot image manipulation network. arXiv preprint arXiv:1703.07255 (2017)
39. Zhang, X.Y., Xie, G.S., Liu, C.L., Bengio, Y.: End-to-end online writer identification with recurrent neural network. IEEE Transactions on Human-Machine Systems **47**(2), 285–292 (2016)
40. 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**(4), 849–862 (2017)
41. Zongker, D.E., Wade, G., Salesin, D.H.: Example-based hinting of true type fonts. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques. pp. 411–416 (2000)