DeepWriting: Making Digital Ink Editable via Deep Generative Modeling

Emre Aksan Fabrizio Pece Otmar Hilliges
Advanced Interactive Technologies Lab, ETH Zürich
{eaksan, pecef, otmarh}@inf.ethz.ch

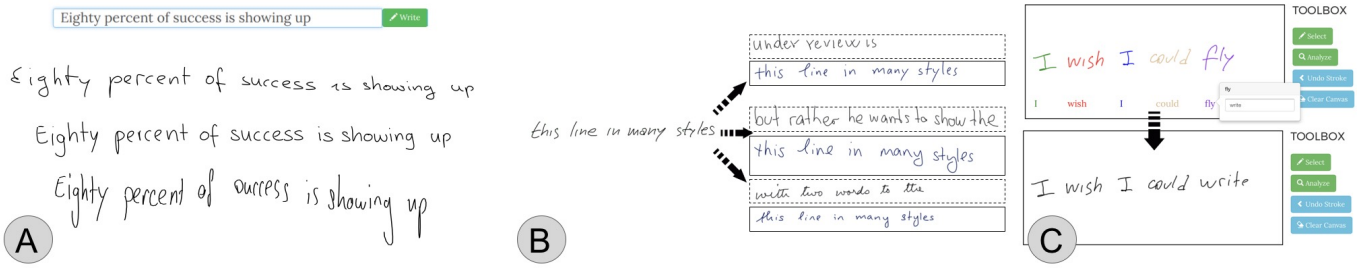

Figure 1: We propose a novel generative neural network architecture that is capable of disentangling style from content and thus makes digital ink editable. Our model can synthesize handwriting from typed text while giving users control over the visual appearance (A), transfer style across handwriting samples (B, solid line box synthesized stroke, dotted line box reference style), and even edit handwritten samples at the word level (C).

ABSTRACT

Digital ink promises to combine the flexibility and aesthetics of handwriting and the ability to process, search and edit digital text. Character recognition converts handwritten text into a digital representation, albeit at the cost of losing personalized appearance due to the technical difficulties of separating the interwoven components of content and style. In this paper, we propose a novel generative neural network architecture that is capable of disentangling style from content and thus making digital ink editable. Our model can synthesize arbitrary text, while giving users control over the visual appearance (style). For example, allowing for style transfer without changing the content, editing of digital ink at the word level and other application scenarios such as spell-checking and correction of handwritten text. We furthermore contribute a new dataset of handwritten text with fine-grained annotations at the character level and report results from an initial user evaluation.

ACM Classification Keywords

H.5.2 User Interfaces: Input Devices and Strategies; I.2.6 Learning

Author Keywords

Handwriting; Digital Ink; Stylus-based Interfaces; Deep Learning; Recurrent Neural Networks

CHI 2018, April 21–26, 2018, Montreal, QC, Canada

ISBN 978-1-4503-5620-6/18/04…$15.00

DOI: https://doi.org/10.1145/3173574.3173779

INTRODUCTION

Handwritten text has served for centuries as our primary mean of communication and cornerstone of our education and culture, and often is considered a form of art [45]. It has been shown to be beneficial in tasks such as note-taking [35], reading in conjunction with writing [47] and may have a positive impact on short- and long-term memory [4]. However, despite progress in character recognition [32], fully digital text remains easier to process, search and manipulate than handwritten text which has lead to a dominance of typed text.

In this paper we explore novel ways to combine the benefits of digital ink with the versatility and efficiency of digital text, by making it editable via disentanglement of style and content. Digital ink and stylus-based interfaces have been of continued interest to HCI research (e.g., [22, 23, 44, 56, 61]). However, to process digital ink one has typically to resort to optical character recognition (OCR) techniques (e.g., [17]) thus invariably losing the personalized aspect of written text. In contrast, our approach is capable of maintaining the author's original style, thus allowing for a seamless transition between handwritten and digital text. Our approach is capable of synthesizing handwritten text, taking either a sequence of digital ink or ASCII characters as input. This is a challenging problem: while each user has a unique handwriting style [49, 60], the parameters that determine style are not well defined. Moreover, handwriting style is not fixed but changes temporally based on context, writing speed and other factors [49]. Hence so far it has been elusive to algorithmically recreate style faithfully, while being able to control content. A comprehensive approach to handwriting synthesis must be able to maintain *global* style while preserving *local* variability and context (e.g., many users mix cursive and disconnected styles dynamically).

Embracing this challenge we contribute a novel generative deep neural network architecture for the conditional synthesis of digital ink. The model is capable of capturing and reproducing local variability of handwriting and can mimic different user styles with high-fidelity. Importantly the model provides full control over the content of the synthetic sequences, enabling processing and editing of digital ink at the word level. The main technical contribution stems from the ability to disentangle latent factors that influence visual appearance and content from each other. This is achieved via an architecture that combines autoregressive models with a learned latent-space representation, modeling temporal and time-invariant categorical aspects (i.e., character information).

More precisely we propose an architecture comprising of a recurrent variational autoencoder [12, 28] in combination with two latent distributions that allow for *conditional* synthesis (i.e., provide control over style and content) alongside a novel training and sampling algorithm. The system has been trained on segmented samples of handwritten text collected from 294 authors, and takes into account both temporal and stylistic aspects of handwriting. Further, it is capable of synthesizing novel sequences from typed-text, transferring styles from one user to another, editing digital ink at the word level and thus enables compelling application scenarios including spell-checking and auto-correction of digital ink.

We characterize the performance of the architecture via a thorough technical evaluation, and assess its efficacy and utility via a preliminary experimental evaluation of the interactive scenarios we implemented. Further, we contribute a new dataset that enhances the IAM On-Line Handwriting Database (IAM-OnDB) [31] and includes handwritten text collected from 294 authors with character level annotations. Finally, we plan to release an open-source implementation of our model.

RELATED WORK

Our work touches upon various subjects including HCI (e.g., [22, 39, 44, 56]), handwriting analysis [43] and machine learning (e.g., [15, 18, 28, 42]).

Understanding Handwriting

Research into the recognition of handwritten text has led to drastic accuracy improvements [15, 42] and such technology can now be found in mainstream UIs (e.g., Windows, Android, iOS). However, converting digital ink into ASCII characters removes individual style. Understanding what exactly constitutes style has been the subject of much research to inform font design [8, 14, 38] and the related understanding of human reading has served as a source of inspiration for the modern parametric-font systems [25, 29, 48]. Nonetheless no equivalent parametric model of handwritten style exists and analysis and description of style remains an inexact science [38, 43]. We propose to learn a latent representation of style and to leverage it in a generative model of user specific text.

Pen-based interaction

Given the naturalness of the medium [47, 11], pen-based interfaces have seen enduring interest in both the graphics and HCI literature [50]. Ever since Ivan Sutherland's Sketchpad [51] researchers have explored sensing and input techniques for small screens [27, 57], tablets [23, 40] and whiteboards [36, 39, 56] and have proposed ways of integrating paper with digital media [21, 6]. Furthermore many domain specific applications have been proposed. For instance, manipulation of hand-drawn diagrams [3] and geometric shapes [2], note-taking (e.g., NiCEBook [6]), sharing of notes (e.g., NotePals [13]), browsing and annotation of multimedia content [54], including digital documents [58] using a stylus. Others have explored creation, management and annotation of handwritten notes on large screen displays [39, 56]. Typically such approaches do not convert ink into characters to preserve individual style. Our work enables new interactive possibilities by making digital ink editable and interchangeable with a character representation, allowing for advanced processing, searching and editing.

Handwriting Beautification

Zitnick [61] proposes a method for beautification of digital ink by exploiting the smoothing effect of geometric averaging of multiple instances of the same stroke. While generating convincing results, this method requires several samples of the same text for a single user. A supervised machine learning method to remove slope and slant from handwritten text and to normalize it's size [16] has been proposed. Zanibbi et al [59] introduce a tool to improve legibility of handwritten equations by applying style-preserving morphs on user-drawn symbols. Lu et al. [33] propose to learn style features from trained artist, and to subsequently transfer the strokes of a different writer to the learnt style and therefore inherently remove the original style. Text beautification is one of the potential applications for our work, however the proposed model only requires a single sequence as input, retains the global style of the author when beautifying and can generate novel (i.e., from ASCII characters) sequences in that style.

Handwriting Synthesis

A large body of work is dedicated to the synthesis of handwritten text (for a comprehensive survey see [15]). Attempts have been made to formalize plausible biological models of the processes underlying handwriting [24] or by learning sequences of motion primitives [55]. Such approaches primarily validate bio-inspired models but do not produce convincing sequences. In [5, 41] sigma-lognormal models are proposed to synthesize handwriting samples by parameterizing rapid human movements and hence reflecting writer's fine motor control capability. The model can naturally synthesize variances of a given sample, but it lacks control of the content.

Realistic handwritten characters such as Japanese Kanji or individual digits can be synthesized from learned statistical models of stroke similarity [9] or control point positions (requiring characters to be converted to splines) [53]. Follow-up work has proposed methods that connect such synthetic characters [10, 52] using a ligature model. Haines et al. [20] take character-segmented images of a single author's writing and attempts to replicate the style via dynamic programming. These approaches either ignore style entirely or learn to imitate a single reference style from a large corpus of data. In contrast, our method learns first how to separate content and style and

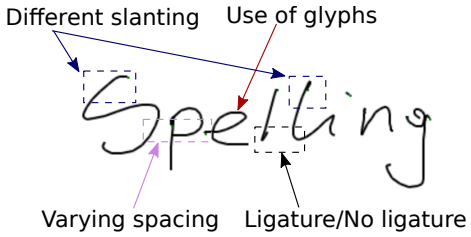

Figure 2: Example of intra-author variation that leads to entanglement of style and content, making conditional synthesis of realistic digital ink very challenging.

then is capable of transferring style from a single sample to arbitrary text, providing much more flexibility.

Kingma and Welling [28] propose a variational auto-encoder architecture for manifold learning and generative modelling. The authors demonstrate synthesis of single character digits via manipulation of two, abstract latent variables but the method can not be used for conditional synthesis. The work most closely related to ours proposes a long short-term memory recurrent (LSTM) neural network to generate complex sequences with long-range structure such as handwritten text [18]. The work demonstrates synthesis of handwritten text in specific styles limited to samples in the training set, and its model has no notion of disentangling content from style.

METHOD

To make digital ink fully editable, one has to overcome a number of technical problems such as character recognition and synthesis of realistic handwriting. None is more important though than the disentanglement of *style* and *content*. Each author has a unique style of handwriting [49], but at the same time, they also display a lot of intra-variability, such as mixing connected and disconnected styles, variance in usage of glyphs, character spacing and slanting (see Figure 2). Hence, it is hard to define or predict the appearance of a character, as often its appearance is strongly influenced by its content.

In this paper, we propose a data-driven model capable of disentangling handwritten text into their content and style components, necessary to enable editing and synthesis of novel handwritten samples in a user-specified style. The key idea underlying our approach is to treat style and content as two separate latent random variables (Figure 3-a). While the *content* component is defined as the set of alphanumeric characters and punctuation marks, the *style* term is an abstraction of the factors defining appearance. It is learned by the model and projected into a continues-valued latent space. One can make use of content and style variables to *edit* either style, content or both, or one can generate entirely new samples (Figure 3-b).

We treat digital ink as a sequence of temporarily ordered strokes where each stroke consists of (u, v) pen-coordinates on the device screen and corresponding *pen-up* events. While the pen-coordinates are integer values bounded by screen resolution of the device, *pen-up* takes value 1 when the pen is lifted off the screen and 0, otherwise. A handwriting sample is formally defined as $\mathbf{x} = \{x_t\}_{t=1}^{T}$ where x_t is a stroke and T

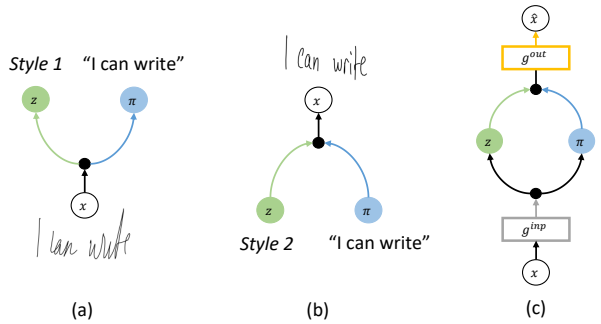

Figure 3: High-level representation of our approach. $\mathbf{x}, \mathbf{z}$ and $\boldsymbol{\pi}$ are random variables corresponding to handwritten text, *style* and *content*, respectively. (a) A given handwritten sample can be decomposed into *style* and *content* components. (b) Similarly, a sample can be synthesized using *style* and *content* components. (c) Our model learns inferring and using latent variables by reconstructing handwriting samples. g^{inp} and g^{out} are feed-forward networks projecting the input into an intermediate representation and predicting outputs, respectively.

is total number of strokes. Moreover, we label each stroke x_t with character y_t, end-of-character eoc_t and beginning-of-word bow_t labels. y_t specifies which character a stroke x_t belongs to. Both eoc_t and bow_t are binary-valued and set to 1 if x_t correspond to the last stroke of a character sequence or the first stroke of a new word, respectively (Figure 4).

We propose a novel autoregressive neural network (NN) architecture that contains continuous and categorical latent random variables. Here the continuous latent variable which captures the appearance properties is modeled by an isotropic Normal distribution (Figure 5 (green)). Whereas the content information is captured via a Gaussian Mixture Model (GMM), where each character in the dataset is represented by an isotropic Gaussian (shown in Figure 5 (blue)). We train the model by reconstructing a given handwritten sample $\mathbf{x}$ (Figure 3-c). Handwriting is inherently a temporal domain and require exploiting long-range dependencies. Hence, we leverage recurrent neural network (RNN) cells and operate in the stroke level x_t. Moreover, we make use of and predict y_t, eoc_t, bow_t in auxiliary tasks such as controlling the word spacing, character segmentation and recognition. (Figure 5).

The proposed architecture which we call conditional variational recurrent neural network (C-VRNN) builds on prior work on variational autoencoders (VAE) [28] and its recurrent variant, variational recurrent neural networks (VRNN) [12]. While VAEs only work with non-temporal data, VRNN can reconstruct and synthesize timeseries, albeit without conditioning, providing no control over the generated content. In contrast, our model synthesizes realistic handwriting with natural variation and conditioned on a user-specified content, enabling a number of compelling applications (Figure 1).

Background

Multi-layer recurrent neural networks (RNN) [18] and variational RNN (VRNN) [12] are most related to our work. We briefly recap these and highlight differences. In our notation

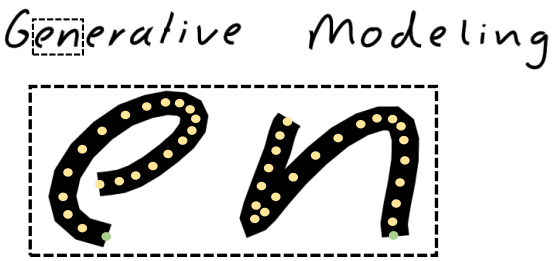

Figure 4: Discretization of digital handwriting. A handwriting sample (top) is represented by a sequence of temporarily ordered strokes. Yellow and green nodes illustrate sampled strokes. The green nodes correspond to *pen-up* events.

superscripts correspond to layer information such as *input*, *latent* or *output* while subscripts denote the time-step t. Moreover, we drop parametrization for the sake of brevity, and therefore readers should assume that all probability distributions are modeled using neural networks.

Recurrent Neural Networks

RNNs model variable length input sequences $\mathbf{x} = (x_1, x_2, \cdots, x_T)$ by predicting the next time step x_{t+1} given the current x_t. The probability of a sequence $\mathbf{x}$ is given by

$$\begin{aligned} p(\mathbf{x}) &= \prod_{t=1}^{T} p(x_{t+1}|x_t), \\ p(x_{t+1}|x_t) &= g^{out}(h_t) \\ h_t &= \tau(x_t, h_{t-1}), \end{aligned} \tag{1}$$

where τ is a *deterministic* transition function of an RNN cell, updating the internal cell state h. Note that x_{t+1} implicitly depends on all inputs until step $t+1$ through the cell state h.

The function g^{out} maps the hidden state to a probability distribution. In vanilla RNNs (e.g., LSTM, GRU) g^{out} is the only source of variability. To express the natural randomness in the data, the output function g^{out} typically parametrizes a statistical distribution (e.g., Bernoulli, Normal, GMM). The output is then calculated by sampling from this distribution. Both functions τ and g^{out} are approximated by optimizing neural network parameters via maximizing the log-likelihood:

$$\mathscr{L}_{rnn}(\mathbf{x}) = logp(\mathbf{x}) = \sum_{t=1}^{T} logp(x_{t+1}|x_t) \tag{2}$$

Multi-layered LSTMs with a GMM output distribution have been used for handwriting modeling [18]. While capable of conditional synthesis, they can not disentangle style from content due to the lack of latent random variables.

Variational Recurrent Neural Networks

VRNNs [12] modify the deterministic τ transition function by introducing a latent random variable $\mathbf{z} = (z_1, z_2, \cdots, z_T)$ increasing the expressive power of the model and to better capture variability in the data by modeling

$$\begin{aligned} p(\mathbf{x}, \mathbf{z}) &= p(\mathbf{x}|\mathbf{z})p(\mathbf{z}), \\ p(\mathbf{x}, \mathbf{z}) &= \prod_{t=1}^{T} p(x_t|z_t)p(z_t), \\ p(x_t|z_t) &= g^{out}(z_t, h_{t-1}), \\ p(z_t) &= g^{p,z}(h_{t-1}), \\ h_t &= \tau(x_t, z_t, h_{t-1}), \end{aligned} \tag{3}$$

where $g^{p,z}$ is a multi-layer feed forward network parameterizing the prior distribution $p(z_t)$ and the latent variable $\mathbf{z}$ enforces the model to project the data variability on the prior distribution $p(\mathbf{z})$. Note that x_t still depends on the previous steps, albeit implicitly through the internal state h_{t-1}.

At each time step the latent random variable $\mathbf{z}$ is modeled as isotropic Normal distribution $z_t \sim \mathscr{N}(\mu_t, \sigma_t I)$. The transition function τ takes samples z_t as input, introducing a new source of variability.

Since we do not have access to the true distributions at training time, the posterior $p(\mathbf{z}|\mathbf{x})$ is intractable and hence makes the marginal likelihood, i.e., the objective, $p(\mathbf{x})$ also intractable. Instead, an approximate posterior distribution $q(\mathbf{z}|\mathbf{x})$ is employed, imitating the true posterior $p(\mathbf{z}|\mathbf{x})$ [28], where $q(\mathbf{z}|\mathbf{x})$ is an isotropic Normal distribution and parametrized by a neural network $g^{q,z}$ as follows:

$$q(z_t|x_t) = g^{q,z}(x_t, h_{t-1}) \tag{4}$$

The model parameters are optimized by jointly maximizing a variational lower bound:

$$logp(\mathbf{x}) \geq \mathbb{E}_{q(z_t|x_t)} \sum_{t=1}^{T} logp(x_t|z_t) - KL(q(z_t|x_t)||p(z_t)), \tag{5}$$

where $KL(q||p)$ is the Kullback-Leibler divergence (non-similarity) between distributions q and p. The first term in loss (5) ensures that the sample x_t is reconstructed given the latent sample $z_t \sim q(z_t|x_t)$ while the KL term minimizes the discrepancy between our approximate posterior and prior distributions so that we can use the prior $z_t \sim p(z_t)$ for synthesis later. Moreover, the $q(\mathbf{z}|\mathbf{x})$ network enables inferring latent properties of a given sample, providing interesting applications. For example, a handwriting sample can be projected into the latent space z and reconstructed with different slant.

A plain auto-encoder architecture learns to faithfully reconstruct input samples, the latent term $\mathbf{z}$ transforms the architecture into a fully generative model. Note that the KL-term never becomes 0 due to the different amount of input information to $g^{p,z}$ and $g^{q,z}$. Hence, the KL-term enforces the model to capture the common information in the latent space z.

Conditional Variational Recurrent Neural Network

While multi-layer RNNs and VRNNs have appealing properties, neither is directly capable of full conditional handwriting synthesis. For example, one can synthesize a given text in a given style by using RNNs but samples will lack natural variability. Or one can generate high quality novel samples with VRNNs. However, VRNNs lack control over *what* is

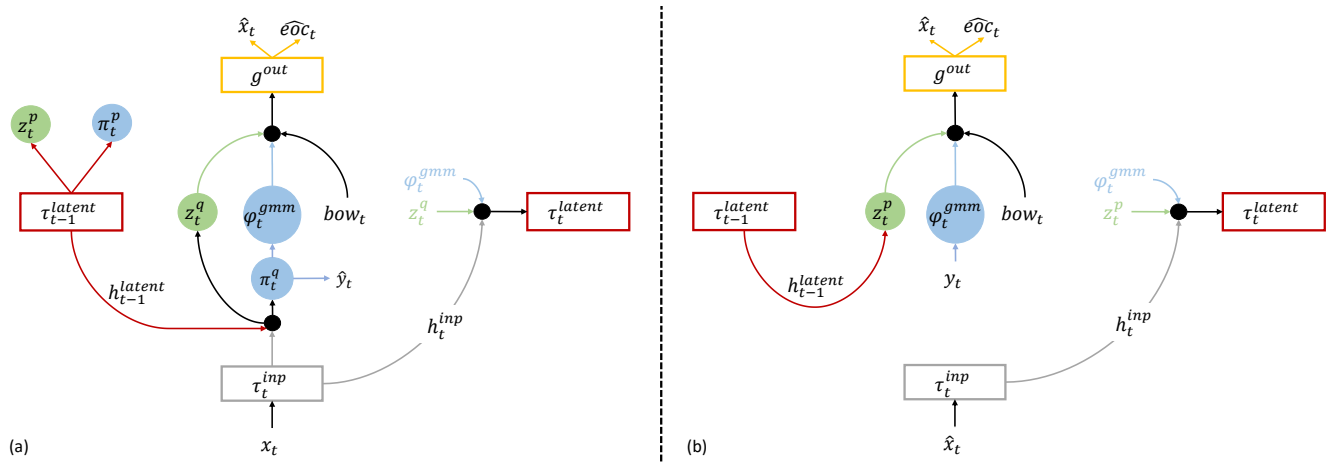

Figure 5: Schematic overview of our handwriting model in training (a) and sampling phases (b), operating at the stroke-level. Subscripts denote time-step t. Superscripts correspond to layer names such as *input*, *latent* and *output* layers or the distributions of the random variables such as $z_t^q \sim q(z_t|x_t)$ and $z_t^p \sim p(z_t|x_t)$. (τ and h) An RNN cell and its output. (g) A multi-layer feed-forward neural network. (Arrows) Information flow color-coded with respect to source. (Colored circles) Latent random variables. Outgoing arrows represent a sample of the random variable. (Green branch) Gaussian latent space capturing style related information along with latent RNN (τ_t^{latent}) cell at individual time-steps t. (Blue branch) Categorical and GMM random variables capturing content information. (Small black nodes) An auxiliary node for concatenation of incoming nodes.

written. Neither model have inference networks to decouple style and content, which lies at the core of our work.

We overcome this issue by introducing a new set of latent random variables, $\mathbf{z}, \pi$, capturing style and content of handwriting samples. More precisely our new model describes the data as being generated by two latent variables $\mathbf{z}$ and π (Figure 3) such that

$$\begin{aligned} p(\mathbf{x},\mathbf{z},\pi) &= p(\mathbf{x}|\mathbf{z},\pi)p(\mathbf{z})p(\pi), \\ p(\mathbf{x},\mathbf{z},\pi) &= \prod_{t=1}^{T} p(x_t|z_t)p(z_t)p(\pi_t), \\ p(x_t|z_t,\pi_t) &= g^{out}(z_t,\pi_t), \\ p(z_t) &= g^{p,z}(h_{t-1}^{latent}), \\ p(\pi_t) &= g^{p,\pi}(h_{t-1}^{latent}), \\ h_t^{latent} &= \tau^{latent}(x_t,z_t,\pi_t,h_{t-1}^{latent}), \\ q(z_t|x_t) &= g^{q,z}(x_t,h_{t-1}^{latent}), \end{aligned} \tag{6}$$

where $p(\pi_t)$ is a K-dimensional multinomial distribution specifying the characters that are synthesized.

Similar to VRNNs, we introduce an approximate inference distribution $q(\pi|\mathbf{x})$ for the categorical latent variable:

$$q(\pi_t|x_t) = g^{q,\pi}(x_t,h_{t-1}^{latent}) \tag{7}$$

Since we aim to decouple style and content in handwriting, we assume that the approximate distribution has a factorized form $q(z_t,\pi_t|x_t) = q(z_t|x_t)q(\pi_t|x_t)$. Both $q(\pi|\mathbf{x})$ and $q(\mathbf{z}|\mathbf{x})$ are used to infer content and style components of a given sample $\mathbf{x}$ as described earlier.

We optimize the following variational lower bound:

$$\begin{aligned} log p(\mathbf{x}) \geq \mathscr{L}_{lb}(\cdot) = \mathbb{E}_{q(z_t,\pi_t|x_t)} \sum_{t=1}^{T} log p(x_t|z_t,\pi_t) \\ -KL(q(z_t|x_t)||p(z_t)) - KL(q(\pi_t|x_t)||p(\pi_t)), \end{aligned} \tag{8}$$

where the first term ensures that the input stroke is reconstructed by using its latent samples. We model the output by using bivariate Gaussian and Bernoulli distributions for $2D$-pixel coordinates and binary *pen-up* events, respectively.

Note that our output function g^{out} does not employ the internal cell state h. By using only the latent variables z and π for synthesis, we aim to enforce the model to capture the patterns only in the latent variables z and π.

High Quality Digital Ink Synthesis

The C-VRNN architecture as discussed so far enables the crucial component of separating continuous components from categorical aspects (i.e., characters) which potentially would be sufficient to conditionally synthesize individual characters. However, to fully address the entire handwriting task several extension to control important aspects such as word-spacing and to improve quality of the predictions are necessary.

Character classification loss

Although we assume that the latent random variables $\mathbf{z}$ and π capture style and content information, respectively, and make a conditional independence assumption, in practice full disentanglement is an ambiguous task. Since we essentially ask the model to learn by itself what style and what content are we found further guidance at training time to be necessary.

To prevent divergence during training we make use of character labels at training time and add an additional cross-entropy classification loss $\mathscr{L}_{classification}$ on the content component $q(\pi_t|x_t)$.

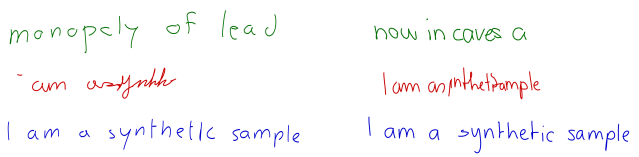

Figure 6: (top, green) Input samples used to infer style. (middle, red) Synthetic samples of a model with π only. They are generated using one-hot-encoded character labels, causing problems with *pen-up* events and with character placement. (bottom, blue) Synthetic samples of our model *with* GMM latent space.

GMM latent space

Conditioning generative models is typically done via one-hot encoded labels. While we could directly use samples from $q(\pi_t|x_t)$, we prefer using a continuous representation. We hypothesize and experimentally validate (see Figure 6) that the synthesis model can shape the latent space with respect to the loss caused by the content aspect.

For this purpose we use a Gaussian mixture model where each character in K is represented by an isotropic Gaussian

$$p(\varphi_t) = \sum_{k=1}^{K} \pi_{t,k} \mathcal{N}(\varphi_t|\mu_k, \sigma_k), \tag{9}$$

where $\mathcal{N}(\varphi_t|\mu_k, \sigma_k)$ is the probability of sampling from the corresponding mixture component k. π corresponds to the content variable in Eq. (7) which is here interpreted as weight of the mixture components. This means that we use $q(\pi_t|x_t)$ to select a particular Gaussian component for a given stroke sample x_t. We then sample φ_t from the k-th Gaussian component and apply the "re-parametrization trick" [28, 19] so that the gradients can flow through the random variables, enabling the learning of GMM parameters via standard backpropagation.

$$\varphi_t = \mu_k + \sigma_k \varepsilon, \tag{10}$$

where $\varepsilon \sim \mathcal{N}(0,1)$. Our continuous content representation results in similar letters being located closer in the latent space while dissimilar letters or infrequent symbols being pushed away. This effect is visualized in Figure 7.

Importantly the GMM parameters are sampled from a time-invariant distribution. That is they remain the same for all data samples and across time steps of a given input $\mathbf{x}$, whereas z_t is dynamic and employs new parameters per time step. For each Gaussian component in φ, we initialize μ_k, $1 \leq k \leq K$, randomly by using a uniform distribution $\mathcal{U}(-1,1)$ and σ_k with a fixed value of 1. The GMM components are trained alongside the other network parameters.

In order to increase model convergence speed and to improve results, we use ground truth character labels during training. More precisely, the GMM components are selected by using the real labels y instead of predictions of the inference network $q(\pi_t|x_t)$. Instead $q(\pi_t|x_t)$ is trained only by using the classification loss $\mathcal{L}_{classification}$ and not affected by the gradients of GMM with respect to π_t.

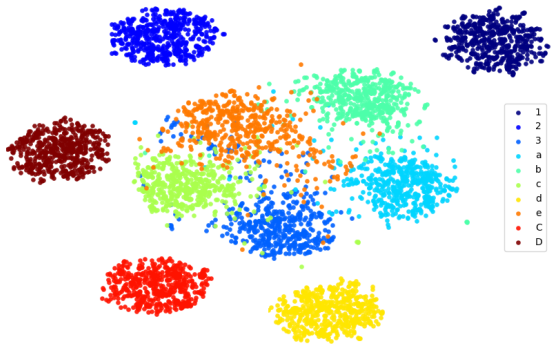

Figure 7: Illustration of our GMM latent space φ^{gmm}. We select a small subset of our alphabet and draw 500 samples from corresponding GMM components. We use the t-sne algorithm [34] to visualize 32-dimensional samples in 2D space. Note that the t-sne algorithm finds an arbitrary placement and hence the positioning does not reflect the true latent space. Nevertheless, letters form separate clusters.

Word spacing and character limits

At sampling time the model needs to automatically infer word spacing and which character to synthesize (these are a priori unknown). In order to control when to leave a space between words or when to start synthesizing the next character, we introduce two additional signals during training, namely *eoc* and *bow* signaling the *end* of a character and *beginning* of a word respectively. These labels are attained from ground truth character level segmentations (see Dataset section).

The *bow* signal is fed as input to the output function g^{out} and the output distribution of our handwriting synthesis takes the following form:

$$p(x_t|z_t, \pi_t) = g^{out}(z_t, \varphi_t, bow_t), \tag{11}$$

forcing the model to learn when to leave empty space at training and sampling time.

The *eoc* signal, on the other hand, is provided to the model at training so that it can predict when to stop synthesizing a given character. It is included in the loss function in the form of Bernoulli log-likelihood $\mathcal{L}_{eoc}$. Along with the reconstruction of the input stroke x_t, the eoc_t label is predicted.

Input RNN cell

Our model consists of two LSTM cells in the latent and at the input layers. Note that the latent cell is originally contributing via the transition function τ^{latent} in Eq. (6). Using an additional cell at the input layer increases model capacity (similar to multi-layered RNNs) and adds a new transition function τ^{inp}. Thus the synthesis model can capture and modulate temporal patterns at the input levels. Intuitively this is motivated by the strong temporal consistency in handwriting where the previous letter influences the appearance of the current (cf. Figure 2).

Algorithm 1 Training
The flow of information through the model at training time, from inputs to outputs. The model is presented in Figure 5-b color coded components in comments.

Inputs:
Strokes $\mathbf{x} = \{x_t\}_{t=1}^{T}$
Labels $(\mathbf{y}, \mathbf{eoc}, \mathbf{bow}) = \{(y_t, eoc_t, bow_t)\}_{t=1}^{T}$
$h_0^{inp} = h_0^{latent} = \mathbf{0}$
Outputs:
Reconstructed strokes $\hat{\mathbf{x}}$, predicted $\hat{\mathbf{y}}$ and $\hat{\mathbf{eoc}}$ labels.

$t \leftarrow 1$
while $t =\leq T$ **do**
  Update $h_t^{inp} \leftarrow \tau^{inp}(x_t, h_{t-1}^{inp})$ ▷ grey
  Sample $\pi_t^q \sim q(\pi_t|x_t)$ using Eq. (17) (equivalent to the character prediction $\hat{y}_t$), ▷ blue
  Select the corresponding GMM component and draw a content sample φ_t using Eq. (10), ▷ blue
  Estimate parameters of the isotropic Gaussian $q(z_t|x_t)$ using Eq. (16) and sample $z_t^q \sim q(z_t|x_t)$, ▷ green
  Using φ_t, z_t^q and bow_t, reconstruct the stroke $\hat{x}_t$ and predict $\widehat{eoc}_t$, ▷ yellow
  Estimate isotropic Gaussian and Multinomial distribution parameters of prior latent variables z_t^p and π_t^p by using Eq. (13-14), respectively,
  Update $h_t^{latent} \leftarrow \tau^{latent}(h_t^{inp}, z_t^q, \varphi_t, h_{t-1}^{latent})$,
  $t \leftarrow t+1$
end while
Evaluate Eq. (19) and update model parameters.

We now use a temporal representation h_t^{inp} of the input strokes x_t. With the cumulative modifications, our C-VRNN architecture becomes

$$p(x_t|z_t, \pi_t) = g^{out}(z_t, \varphi_t, bow_t), \tag{12}$$
$$z_t^p \sim p(z_t) = g^{p,z}(h_{t-1}^{latent}), \tag{13}$$
$$\pi_t^p \sim p(\pi_t) = g^{p,\pi}(h_{t-1}^{latent}), \tag{14}$$
$$h_t^{inp} = \tau^{inp}(x_t, h_{t-1}^{inp}), \tag{15}$$
$$z_t^q \sim q(z_t|x_t) = g^{q,z}(h_t^{inp}, h_{t-1}^{latent}), \tag{16}$$
$$\pi_t^q \sim q(\pi_t|x_t) = g^{q,\pi}(h_t^{inp}, h_{t-1}^{latent}), \tag{17}$$
$$h_t^{latent} = \tau^{latent}(h_t^{inp}, z_t, \varphi_t, h_{t-1}^{latent}). \tag{18}$$

Finally we train our handwriting model by using algorithm (1) and optimizing the following loss:

$$\mathscr{L}(\cdot) = \mathscr{L}_{lb} + \mathscr{L}_{classification} + \mathscr{L}_{eoc}. \tag{19}$$

In our style transfer applications, by following the steps $1-11$ of algorithm (1), we first feed the model with a reference sample and get the internal state of the latent LSTM cell h^{latent} carrying style information. The sampling algorithm (2) to generate new samples is then initialized with this h^{latent}.

We implement our model in Tensorflow [1]. The model and training details are provided in the Appendix. Code and dataset for learning and reproducing our results can be found at https://ait.ethz.ch/projects/2018/deepwriting/.

Algorithm 2 Sampling
The procedure to *synthesize* novel handwriting sequences by conditioning on content and style. The model is presented in Figure 5-c and color coded components in comments.

Inputs:
Sequence of characters $\mathbf{y} = \{y_n\}_{n=1}^{N}$ to be synthesized.
h_0^{latent} is zero-initialized or inferred from another sample.
Probability threshold ε for switching to the next character.
Outputs:
Generated strokes $\hat{\mathbf{x}}$ of the given text $\mathbf{y}$.

$t \leftarrow 1$, $n \leftarrow 1$.
while $n \leq N$ **do**
  Estimate parameters of style prior $p(z_t)$ using Eq. (13) and sample z_t^p. ▷ green
  $\pi_t^p \leftarrow y_n$, select the corresponding GMM component and draw a content sample. φ_t using Eq. (10), ▷ blue
  $bow_t \leftarrow 1$ if it is the first stroke of a new word, 0, otherwise.
  Using φ_t, z_t^p and bow_t, synthesize a new stroke $\hat{x}_t$ and predict $\widehat{eoc}_t$, ▷ yellow
  $n \leftarrow n+1$ if $\widehat{eoc}_t > \varepsilon$, ▷ next character
  $t \leftarrow t+1$
end while

Character recognition

Disentanglement of content requires character recognition of handwritten input. Our model uses the latent variable π to infer character labels which is fed by the input LSTM cell. In our experiments we observe that bidirectional recurrent neural networks (BiRNN) [46] perform significantly better than standard LSTM models in content recognition task (60% against 96% validation accuracy). BiRNNs have access to future steps by processing the input sequence from both directions. However, they also require that the entire input sequence must be available at once. This inherent constraint makes it difficult to fully integrate them into our training and sampling architecture where the input data is available one step at a time.

Due to their desirable performance we train a separate BiRNN model to classify input samples. At sampling time we use its predictions to guide disentanglement. Note that technically $q(\pi|\mathbf{x})$ also infers character labels but the BiRNN's accuracy improves the quality of synthesis. We leave full integration of high accuracy character recognition for future work.

APPLICATION SCENARIOS

By disentangling content from style, our approach makes digital ink truly editable. This allows the generation of novel writing in user-defined styles and, similarly to typed text, of seamless editing of handwritten text. Further, it enables a wide range of exciting application scenarios, of which we discuss proof-of-concept implementations.

Conditional Handwriting Generation

To illustrate the capability to synthesize *novel* text in a user-specific style we have implemented an interface that allows user to type text, browse a database of handwritten samples from different authors and to generate novel handwriting. The novel sequence takes the content from the typed text and matches the style to a single input sample (cf. video). This could be directly embedded in existing note-taking applications to generate personalized handwritten notes from typed text or email clients could turn typed text into handwritten, personalized letters. For demonstration we have synthesized extracts of this paper's abstract in three different styles (see Figure 8).

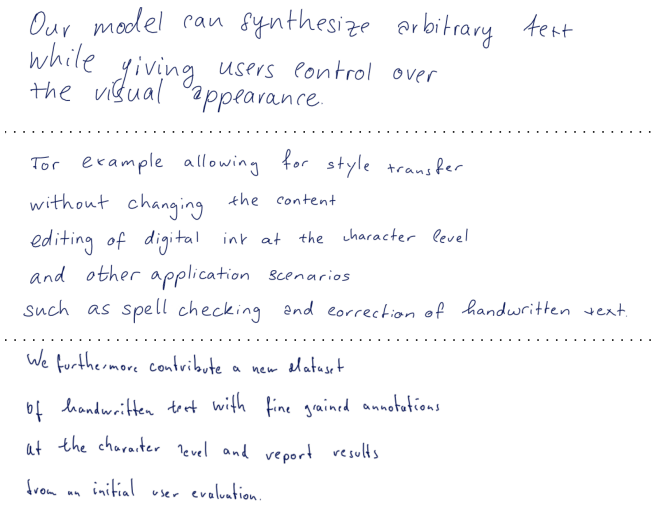

Figure 8: Handwritten text synthesized from the paper abstract. Each sentence is "written" in the style of a different author. For full abstract, see Appendix.

Content Preserving Style Transfer

Our model can furthermore transfer *existing* handwritten samples to *novel* styles, thus preserving their content while changing their appearance. We implemented an interactive tablet application that allows users to recast their own handwriting into a selected style (see Figure 9 for results and video figure for interface walk through). After scribbling on the canvas and selecting an author's handwriting sample, users see their strokes morphed to that style in real-time. Such solution could be beneficial for a variety of domains. For example, artist and comic authors could include specific handwritten lettering in their work, or preserving style during localization to a foreign language.

Beautification

When using the users own input style as target style, our model re-generates smoother versions of the original strokes, while maintaining natural variability and diversity. Thus obtaining an averaging effect that suppresses local noise and preserves global style features. The resulting strokes are then beautified (see Figure 12 and video), in line with previous work that solely relied on token averaging for beautification (e.g., [61]) or denoising (e.g., [7]).

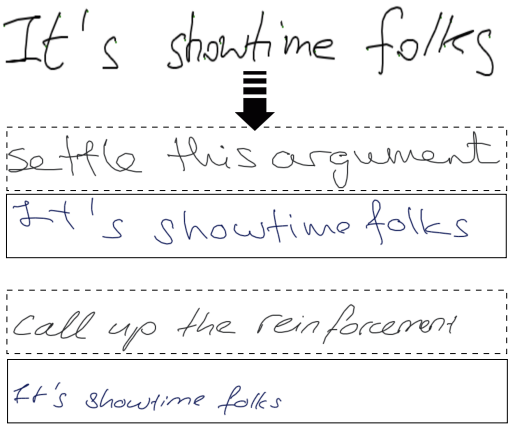

Figure 9: Style transfer. The input sequence (top) is transferred to a selected reference style (black ink, dotted outlines). The results (blue ink, solid outline) preserve the input content, and its appearance matches the reference style.

Word-level Editing

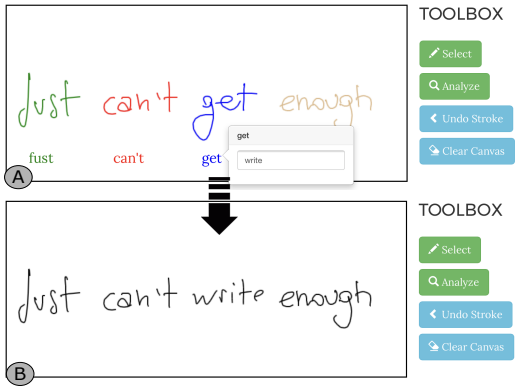

Figure 10: Our model allows editing of handwritten text at the word level. A) Handwriting is recognized, with each individual word fully editable. B) Edited words are synthesized and embedded in the original text, preserving the style.

At the core of our technique lies the ability to edit digital ink at the same level of fidelity as typed text, allowing users to change, delete or replace individual words. Figure 10 illustrates a simple prototype allowing users to edit handwritten content, while preserving the original style when re-synthesizing it. Our model recognizes individual words and characters and renders them as (editable) overlays. The user may select individual words, change the content, and regenerate the digital ink reflecting the edits while maintaining a coherent visual appearance. We see many applications, for example note taking apps, which require frequent edits but currently do not allow for this without loosing visual appearance.

Handwriting Spell-checking and Correction

A further application of the ability to edit digital ink at the word level is the possibility to spell-check and correct handwritten text. As a proof of concept, we implemented a functional handwriting spell-checker that can analyze digital ink,

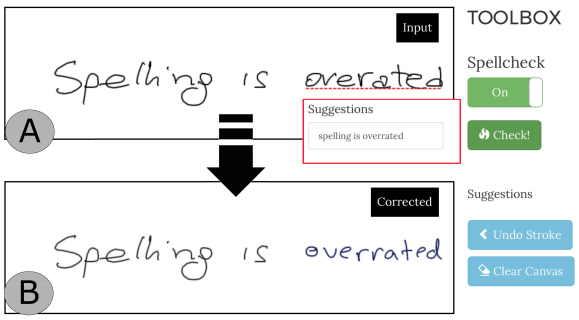

Figure 11: Our spell-checking interface. A) Spelling and grammar mistakes are detected and highlighted directly on the handwriting. Alternative spelling is offered (red box). B) Corrected words are synthesized and embedded in the original text (blue ink), preserving the writer style.

detect spelling mistakes, and correct the written samples by synthesizing the corrected sentence in the original style (see Figure 11 and video figure). For the implementation we rely on existing spell-checking APIs, feeding recognized characters into it and re-rendering the retrieved corrections.

PRELIMINARY USER EVALUATION

So far we have introduced our neural network architecture and have evaluated its capability to synthesize digital ink. We now shift our focus on initially evaluating users' perception and the usability of our method. To this end, we conducted a preliminary user study gathering quantitative and qualitative data on two separate tasks. Throughout the experiment, 10 subjects ($M = 27.9$; $SD = 3.34$; 3 female) from our institution evaluated our model using an iPad Pro and Apple Pencil.

Handwriting Beautification

The first part of our experiment evaluates text beautification. Users were asked to compare their original handwriting with its beautified counterpart. Specifically, we asked our subjects to repeatedly write extracts from the LOB corpus [26], for a total of 12 trials each. In each trial the participant copied down the sample and we beautified the strokes with the results being shown side-by-side (see Figure 12, top). Users were then asked to rate the aesthetics of their own script (Q: *I find my own handwriting aesthetically pleasing*) and the beautified version (Q: *I find the beautified handwriting aesthetically pleasing*), using a 5-point Likert scale. Importantly these were treated as independent questions (i.e., users were allowed to like both).

Handwriting Spell-Checking

In the second task we evaluate the spell-checking utility (see Figure 11). We randomly sampled from the LOB corpus and perturbed individual words such that they contained spelling mistakes. Participants then used our tool to correct the written text (while maintaining it's style), and subsequently were asked to fill in a standard System Usability Scale (SUS) questionnaire and take part in an exit interview.

Results

Our results, summarized in Figure 12 (bottom), indicate that users' reception of our technique is overall positive. The beautified strokes were on average rated higher ($M = 3.65$, 95% CI [3.33-3.97]) with non overlapping confidence intervals. The SUS results further supports this trend, with our system scoring positively ($SUS = 85$). Following the analysis technique suggested in [30], our system can be classified as Rank A, indicating that users are likely to recommend it to others.

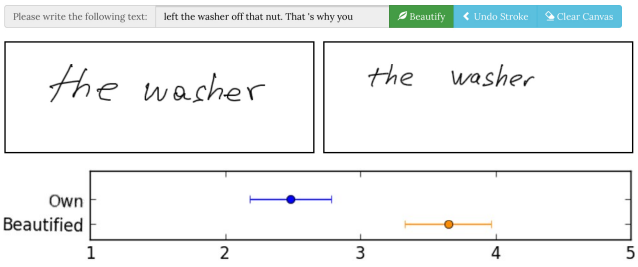

Figure 12: Task 1. Top: Experimental Interface. Participants input on the left; beautified version shown on the right. Bottom: Confidence interval plot on a 5-point Likert scale.

The above results are also echoed by participants' comments during the exit interviews (e.g., *I have never seen anything like this*, and *Finally others can read my notes.*). Furthermore, some suggested additional applications that would naturally fit our model capabilities (e.g., *This would be very useful to correct bad or illegible handwriting*, *I can see this used a lot in education, especially when teaching how to write to kids* and *This would be perfect for note taking, as one could go back in their notes and remove mistakes, abbreviations and so on*). Interestingly, the ability to preserve style while editing content were mentioned frequently as the most valued feature of our approach (e.g., *Having a spell-checker for my own handwriting feels like writing personalized text messages!*).

THE HANDWRITING DATASET

Machine learning models such as ours rely on the availability of annotated training data. Prior work mostly relied on the IAM On-Line Handwriting Database (IAM-OnDB) [31]. However, this was captured with a low-resolution and low-accuracy digital whiteboard and only contains annotations at the sequence level. To disentangle content and style, more fine-grained annotations are needed. Hence, we contribute a novel dataset of handwritten text with character level annotations.

The proposed dataset accumulates the IAM-OnDB dataset with newly collected samples. At time of writing, the unified dataset contains data from 294 unique authors, for a total of 85181 word instances (writer median 292 words) and 406956 handwritten characters (writer median 1349 characters), with further statistics reported in Table 1. The data is stored using the JSON data-interchange format, which makes it easy to distribute and use in a variety of programming environments.

The large number of subjects contained in our dataset allows to capture substantially large variation in styles, crucial to perform any handwriting-related learning task, since handwriting typically exhibits large variation both inter and intra subjects. Furthermore, the dataset contains samples from different digi-

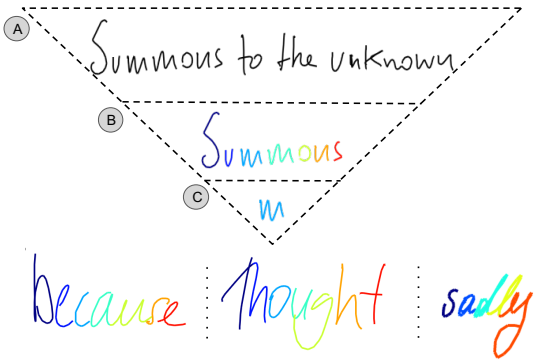

Figure 13: (Top) Our dataset offers different level of annotation, including sentence-wide annotation (A), as well as fine-grained segmentation at word (B) and character (C) level. (Bottom) Samples from our dataset, with colour-coded character segmentation. Different styles are available, including challenging styles to segment (e.g., joined-up cursive, right).

	IAM-OnDB	Ours	**Unified**
Avg. Age (SD)	24.84 (± 6.2)	23.55 (± 5.7)	**24.85 (± 6.19)**
Females %	34.00	32.63	**33.55**
Right-handed %	91.50	96.84	**93.22**
# sentences	11242	63182	**17560**
# unique words	11059	6418	**12718**
# word instances	59141	26040	**85181**
# characters	262981	143975	**406956**

Table 1: Data statistics. In bold, the final *unified* dataset.

tization mechanisms and should hence be useful in learning models that are robust to the exact type of input.

We developed a web tool to collect samples from 94 authors (see Figure 14). Inline with IAM-OnDB we asked each subject to write extracts of the Lancaster-Oslo-Bergen (LOB) text corpus [26] using an iPad Pro. Besides stylus information, we recorded age, gender, handedness and native language of each author. The data is again segmented at the character level and misspelled or unreadable samples have been removed.

Samples from 200 authors in the dataset stem from the IAM-OnDB dataset and were acquired using a smart whiteboard. The data provide stylus information (i.e., x-y coordinates, pen events and timestamps) as well as transcription of the written text, on a per-line basis (Figure 13, A). We purged 21 authors from the original data due to low-quality samples or missing annotations. Furthermore, to improve the granularity of annotations we process the remaining samples, segmenting them down to the character level, obtaining ASCII labels for each character (Figure 13, B and C). For segmentation we used a commercial tool [37] and manually cleaned-up the results.

CONCLUSION AND FUTURE WORK

We have proposed a novel approach for making digital ink editable, which we call conditional variational recurrent neural networks (C-VRNN). At the core of our method lies a deep NN architecture that disentangles content from style. The key idea underlying our approach is to treat style and content as

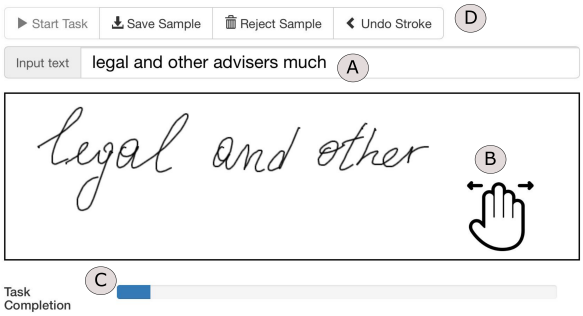

Figure 14: Data collection interface. Users are presented with text to write (A) in the scrollable collection canvas (B). A progress bar (C) informs the user on the status. A writer can save, reject or correct samples using the toolbox buttons (D).

two separate latent random variables with distributions learned during training. At sampling time one can then draw from the content and style component to *edit* either style, content or both, or one can *generate* entirely new samples. These learned statistical distributions make traditional approaches to NN training intractable, due to the lack of access to the true distributions. Moreover, to produce realistic samples of digital ink the model needs to perform auxiliary tasks, such as controlling the spacing in between words, character segmentation and recognition. Furthermore, we have build a variety of proof-of-concept applications, including conditional synthesis and editing of digital ink at the word level. Initial user feedback, while preliminary, indicates that users are largely positive about the capability to edit digital ink - in one's own handwriting or in the style of another author.

To enable the community to build on our work we release our implementation as open-source. Finally, we have contributed a compound dataset, consolidating existing and newly collected handwritten text into a single corpus, annotated to the character level. Data and code are publicly available [1].

While our model can create both disconnected and connected (cursive) styles, its performance is currently better for the former, simpler case. This also applies to most state-of-the-art character recognition techniques, and we leave extending our method to fully support cursive script for future work. Further, we are planning to integrate our currently auxiliary character recognition network into the proposed architecture. One interesting direction in this respect would be the inclusion of a full language model. Finally, and in part inspired by initial feedback, we believe that the underlying technology bears a lot of potential for research in other application domains dealing with time-series, such as motion data (e.g., animation, graphics) or sketches and drawings (e.g., arts and education).

ACKNOWLEDGEMENTS

This work was supported in parts by the ERC grant OPTINT (StG-2016-717054). We thank all the participants for their time and efforts in taking part in our experiments.

[1] https://ait.ethz.ch/projects/2018/deepwriting/

REFERENCES

1. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, and others. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).
2. James Arvo and Kevin Novins. 2000. Fluid sketches: continuous recognition and morphing of simple hand-drawn shapes. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*. ACM, 73–80.
3. James Arvo and Kevin Novins. 2005. Appearance-preserving manipulation of hand-drawn graphs. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*. ACM, 61–68.
4. Virginia Wise Berninger. 2012. Strengthening the mind's eye: The case for continued handwriting instruction in the 21st century. *Principal* 91 (2012), 28–31.
5. Ujjwal Bhattacharya, Réjean Plamondon, Souvik Dutta Chowdhury, Pankaj Goyal, and Swapan K Parui. 2017. A sigma-lognormal model-based approach to generating large synthetic online handwriting sample databases. *International Journal on Document Analysis and Recognition (IJDAR)* (2017), 1–17.
6. Peter Brandl, Christoph Richter, and Michael Haller. 2010. NiCEBook: Supporting Natural Note Taking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 599–608. DOI: http://dx.doi.org/10.1145/1753326.1753417
7. A. Buades, B. Coll, and J. M. Morel. 2005. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 2. 60–65. DOI: http://dx.doi.org/10.1109/CVPR.2005.38
8. Hans-Joachim Burgert. 2002. *The calligraphic line: thoughts on the art of writing*. H.-J. Burgert. Translated by Brody Neuenschwander.
9. Won-Du Chang and Jungpil Shin. 2012. A Statistical Handwriting Model for Style-preserving and Variable Character Synthesis. *Int. J. Doc. Anal. Recognit.* 15, 1 (March 2012), 1–19. DOI: http://dx.doi.org/10.1007/s10032-011-0147-7
10. Hsin-I Chen, Tse-Ju Lin, Xiao-Feng Jian, I-Chao Shen, and Bing-Yu Chen. 2015. Data-driven Handwriting Synthesis in a Conjoined Manner. *Computer Graphics Forum* 34, 7 (Oct. 2015), 235–244. DOI: http://dx.doi.org/10.1111/cgf.12762
11. Mauro Cherubini, Gina Venolia, Rob DeLine, and Andrew J. Ko. 2007. Let's Go to the Whiteboard: How and Why Software Developers Use Drawings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 557–566. DOI: http://dx.doi.org/10.1145/1240624.1240714
12. Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. 2015. A Recurrent Latent Variable Model for Sequential Data. *CoRR* abs/1506.02216 (2015). http://arxiv.org/abs/1506.02216
13. Richard C. Davis, James A. Landay, Victor Chen, Jonathan Huang, Rebecca B. Lee, Frances C. Li, James Lin, Charles B. Morrey, III, Ben Schleimer, Morgan N. Price, and Bill N. Schilit. 1999. NotePals: Lightweight Note Sharing by the Group, for the Group. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 338–345. DOI: http://dx.doi.org/10.1145/302979.303107
14. Johanna Drucker. 1995. *The Alphabetic Labyrinth: The Letters in History and Imagination*. Thames and Hudson.
15. Yousef Elarian, Radwan Abdel-Aal, Irfan Ahmad, Mohammad Tanvir Parvez, and Abdelmalek Zidouri. 2014. Handwriting Synthesis: Classifications and Techniques. *Int. J. Doc. Anal. Recognit.* 17, 4 (Dec. 2014), 455–469. DOI: http://dx.doi.org/10.1007/s10032-014-0231-x
16. Salvador Espana-Boquera, Maria Jose Castro-Bleda, Jorge Gorbe-Moya, and Francisco Zamora-Martinez. 2011. Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models. *Transactions on Pattern Recognition and Machine Intelligence* 33, 4 (April 2011), 767–779.
17. Evernote Corporation. 2017. How Evernotes image recognition works. http://blog.evernote.com/tech/2013/07/18/how-evernotes-image-recognition-works/. (2017). Accessed: 2017-08-10.
18. Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *CoRR* abs/1308.0850 (2013). http://arxiv.org/abs/1308.0850
19. Swaminathan Gurumurthy, Ravi Kiran Sarvadevabhatla, and Venkatesh Babu Radhakrishnan. 2017. DeLiGAN: Generative Adversarial Networks for Diverse and Limited Data. *arXiv preprint arXiv:1706.02071* (2017).
20. Tom S.F. Haines, Oisin Mac Aodha, and Gabriel J. Brostow. 2016. My Text in Your Handwriting. In *Transactions on Graphics*.
21. Michael Haller, Jakob Leitner, Thomas Seifried, James R. Wallace, Stacey D. Scott, Christoph Richter, Peter Brandl, Adam Gokcezade, and Seth Hunter. 2010. The NiCE Discussion Room: Integrating Paper and Digital Media to Support Co-Located Group Meetings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 609–618. DOI:http://dx.doi.org/10.1145/1753326.1753418

22. Tracy Hammond, Stephanie Valentine, Aaron Adler, and Mark Payton. 2015. *The Impact of Pen and Touch Technology on Education* (1st ed.). Springer Publishing Company, Incorporated.

23. Ken Hinckley, Michel Pahud, Hrvoje Benko, Pourang Irani, François Guimbretière, Marcel Gavriliu, Xiang 'Anthony' Chen, Fabrice Matulic, William Buxton, and Andrew Wilson. 2014. Sensing Techniques for Tablet+Stylus Interaction. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 605–614. DOI: http://dx.doi.org/10.1145/2642918.2647379

24. Geoffrey Hinton and Vinod Nair. 2005. Inferring Motor Programs from Images of Handwritten Digits. In *Proceedings of the 18th International Conference on Neural Information Processing Systems (NIPS'05)*. MIT Press, Cambridge, MA, USA, 515–522. http://dl.acm.org/citation.cfm?id=2976248.2976313

25. Fian Hussain and Borut Zalik. 1999. Towards a feature-based interactive system for intelligent font design. In *Information Visualization, 1999. Proceedings. 1999 IEEE International Conference on*. 378–383. DOI: http://dx.doi.org/10.1109/IV.1999.781585

26. Stig Johansson, Atwell Eric, Garside Roger, and Leech Geoffrey. 1986. *The Tagged LOB Corpus: User's Manual.* Norwegian Computing Centre for the Humanities, Bergen, Norway.

27. Wolf Kienzle and Ken Hinckley. 2013. Writing Handwritten Messages on a Small Touchscreen. In *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '13)*. ACM, New York, NY, USA, 179–182. DOI: http://dx.doi.org/10.1145/2493190.2493200

28. Diederik P Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*.

29. Donald E. Knuth. 1986. *The Metafont Book*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

30. James R. Lewis and Jeff Sauro. 2009. The Factor Structure of the System Usability Scale. In *Proceedings of the Human Centered Design: First International Conference (HCD 2009)*, Masaaki Kurosu (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 94–103. DOI: http://dx.doi.org/10.1007/978-3-642-02806-9_12

31. Marcus Liwicki and Horst Bunke. 2005. IAM-OnDB. An On-Line English Sentence Database Acquired from Handwritten Text on a Whiteboard. In *In Proc. 8th Int. Conf. on Document Analysis and Recognition*. 956–961.

32. Marcus Liwicki, Alex Graves, Horst Bunke, and Jürgen Schmidhuber. 2007. A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. In *In Proceedings of the 9th International Conference on Document Analysis and Recognition, ICDAR 2007*.

33. Jingwan Lu, Fisher Yu, Adam Finkelstein, and Stephen DiVerdi. 2012. HelpingHand: Example-based Stroke Stylization. *ACM Trans. Graph.* 31, 4, Article 46 (July 2012), 10 pages. DOI: http://dx.doi.org/10.1145/2185520.2185542

34. Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

35. Pam A. Mueller and Daniel M. Oppenheimer. 2014. The Pen Is Mightier Than the Keyboard: Advantages of Longhand Over Laptop Note Taking. *Psychological Science* (2014). DOI: http://dx.doi.org/10.1177/0956797614524581

36. Elizabeth D. Mynatt, Takeo Igarashi, W. Keith Edwards, and Anthony LaMarca. 1999. Flatland: New Dimensions in Office Whiteboards. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 346–353. DOI: http://dx.doi.org/10.1145/302979.303108

37. MyScript. 2016. MyScript: The Power of Handwriting. http://myscript.com/. (2016). Accessed: 2016-10-04.

38. Gerrit Noordzij. 2005. *The stroke : Theory of Writing*. London : Hyphen. Translated from the Dutch.

39. Florian Perteneder, Martin Bresler, Eva-Maria Grossauer, Joanne Leong, and Michael Haller. 2015. cLuster: Smart Clustering of Free-Hand Sketches on Large Interactive Surfaces. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology (UIST '15)*. ACM, New York, NY, USA, 37–46. DOI: http://dx.doi.org/10.1145/2807442.2807455

40. Ken Pfeuffer, Ken Hinckley, Michel Pahud, and Bill Buxton. 2017. Thumb + Pen Interaction on Tablets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 3254–3266. DOI: http://dx.doi.org/10.1145/3025453.3025567

41. Réjean Plamondon, Christian OâĂŹreilly, Javier Galbally, Abdullah Almaksour, and Éric Anquetil. 2014. Recent developments in the study of rapid human movements with the kinematic theory: Applications to handwriting and signature synthesis. *Pattern Recognition Letters* 35 (2014), 225–235.

42. Réjean Plamondon and Sargur N. Srihari. 2000. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* 22, 1 (Jan. 2000), 63–84. DOI: http://dx.doi.org/10.1109/34.824821

43. Max Albert Eugen Pulver. 1972. *Symbolik der Handschrift (New ed.)*. Kindler, Munich.

44. Yann Riche, Nathalie Henry Riche, Ken Hinckley, Sheri Panabaker, Sarah Fuelling, and Sarah Williams. 2017. As We May Ink?: Learning from Everyday Analog Pen Use to Improve Digital Ink Experiences. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 3241–3253. DOI: http://dx.doi.org/10.1145/3025453.3025716

45. Andrew Robinson. 2007. *The Story of Writing*. Thames & Hudson, London, UK.

46. Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (1997), 2673–2681.

47. Abigail J. Sellen and Richard H.R. Harper. 2003. *The Myth of the Paperless Office*. MIT Press, Cambridge, MA, USA.

48. Ariel Shamir and Ari Rappoport. 1998. *Feature-based design of fonts using constraints*. Springer Berlin Heidelberg, Berlin, Heidelberg, 93–108. DOI: http://dx.doi.org/10.1007/BFb0053265

49. S Srihari, S Cha, H Arora, and S Lee. 2002. Individuality of Handwriting. *Journal of Forensic Sciences* 47, 4 (2002), 1–17. DOI:http://dx.doi.org/10.1520/JFS15447J

50. Craig J Sutherland, Andrew Luxton-Reilly, and Beryl Plimmer. 2016. Freeform Digital Ink Annotations in Electronic Documents: A Systematic Mapping Study. *Computer and Graphics* 55, C (Apr 2016), 1–20. DOI: http://dx.doi.org/10.1016/j.cag.2015.10.014

51. Ivan E. Sutherland. 1963. Sketchpad: A Man-machine Graphical Communication System. In *Proceedings of the May 21-23, 1963, Spring Joint Computer Conference (AFIPS '63 (Spring))*. ACM, New York, NY, USA, 329–346. DOI: http://dx.doi.org/10.1145/1461551.1461591

52. Jue Wang, Chenyu Wu, and Heung-Yeung Xu, Ying-Qing nd Shum. 2005. Combining shape and physical modelsfor online cursive handwriting synthesis. *International Journal of Document Analysis and Recognition (IJDAR)* 7, 4 (2005), 219–227. DOI: http://dx.doi.org/10.1007/s10032-004-0131-6

53. Jue Wang, Chenyu Wu, Ying-Qing Xu, Heung yeung Shum, and Liang Ji. 2002. Learning-based Cursive Handwriting Synthesis. In *in: Proc. Eighth International Workshop on Frontiers of Handwriting Recognition*. 157–162.

54. Nadir Weibel, Adam Fouse, Colleen Emmenegger, Whitney Friedman, Edwin Hutchins, and James Hollan. 2012. Digital Pen and Paper Practices in Observational Research. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 1331–1340. DOI: http://dx.doi.org/10.1145/2207676.2208590

55. Ben H Williams, Marc Toussaint, and Amos J Storkey. 2007. Modelling Motion Primitives and Their Timing in Biologically Executed Movements. In *Proceedings of the 20th International Conference on Neural Information Processing Systems (NIPS'07)*. Curran Associates Inc., USA, 1609–1616. http://dl.acm.org/citation.cfm?id=2981562.2981764

56. Haijun Xia, Ken Hinckley, Michel Pahud, Xiao Tu, and Bill Buxton. 2017. WritLarge: Ink Unleashed by Unified Scope, Action, & Zoom. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 3227–3240. DOI: http://dx.doi.org/10.1145/3025453.3025664

57. Dongwook Yoon, Nicholas Chen, and François Guimbretière. 2013. TextTearing: Opening White Space for Digital Ink Annotation. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY, USA, 107–112. DOI: http://dx.doi.org/10.1145/2501988.2502036

58. Dongwook Yoon, Nicholas Chen, François Guimbretière, and Abigail Sellen. 2014. RichReview: Blending Ink, Speech, and Gesture to Support Collaborative Document Review. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 481–490. DOI: http://dx.doi.org/10.1145/2642918.2647390

59. Richard Zanibbi, Kevin Novins, James Arvo, and Katherine Zanibbi. 2001. Aiding manipulation of handwritten mathematical expressions through style-preserving morphs. (2001).

60. Bin Zhang, Sargur N. Srihari, and Sangjik Lee. 2003. Individuality of Handwritten Characters. In *In Proc. 7th Int. Conf. on Document Analysis and Recognition*. 1086–1090.

61. C. Lawrence Zitnick. 2013. Handwriting Beautification Using Token Means. *ACM Trans. Graph.* 32, 4, Article 53 (July 2013), 8 pages. DOI: http://dx.doi.org/10.1145/2461912.2461985

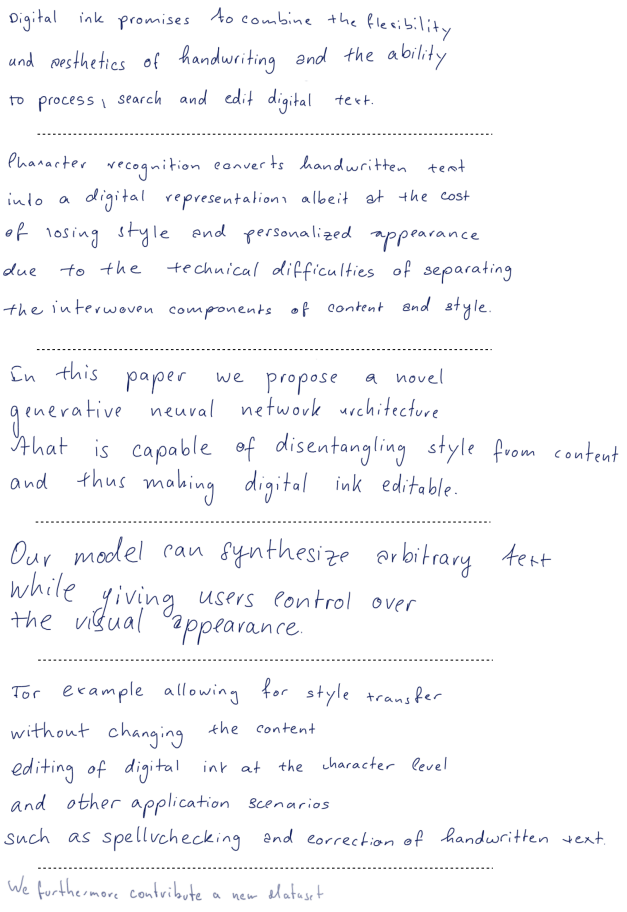

Figure 15: Handwritten text synthesized from the abstract of this paper. Each sentence is "written" in the style of a different author.

APPENDIX

Alphabet

We use the following numbers, letters and punctuation symbols in our alphabet:
0123456789*abcdefghijklmnopqrstuvwxyzABCDEFGHIJK LMNOPQRSTUVWXYZ*'.,-()/

Data Preprocessing

1. In order to speed up training we split handwriting samples that have more than 300 strokes into shorter sequences by using *eoc* labels so that the stroke sequences of letters remain undivided. We create 705 validation and 34577 training samples with average sequence length 261.012 ($\pm$ 85.1).
2. Pixel coordinates (u_0, v_0) of the first stroke is subtracted from the sequence such that each sample starts at the origin $(0,0)$.
3. By subtracting x_t from x_{t+1}, we calculate the changes in 2D-coordinate space and use relative values in training. Note that the $pen-up$ events remained intact.
4. Finally, we apply zero-mean unit-variance normalization on 2D-coordinates by calculating *mean* and *std* statistics on the whole training data.

Network Configuration

Both τ^{inp} and τ^{latent} are LSTM cells with 512 units. Similarly, feed-forward networks g^* consists of 1-layer with 512 units and *ReLu* activation function. We use 32-dimensional isotropic Gaussian distributions for latent variables z^p, z^q and for each GMM component.

Our BiRNN classifier consists of 3-layer bidirectional LSTM cells with 512 units. A 1-layer fully connected network with 256 units and *ReLu* activation function takes BiRNN representations and outputs class probabilities.

Training

We use ADAM optimizer with default parameters. Learning rate is initialized with 0.001 and decayed exponentially with a rate of 0.96 after every 1000 mini-batches. We train our model for 200 epochs by using a mini-batch size of 64.

Additional Results

As an additional result, we have synthesized the entire abstract from this paper using a different style per sentence. The result is shown in Figure 15, and illustrates how our model is able to generate a large variety of styles.