Text and Style Conditioned GAN for Generation of Offline Handwriting Lines

Brian Davis[1]
briandavis@byu.net
Chris Tensmeyer[2]
tensmeye@adobe.com
Brian Price[2]
bprice@adobe.com
Curtis Wigington[2]
wigingto@adobe.com
Bryan Morse[1]
morse@cs.byu.edu
Rajiv Jain[2]
rajijain@adobe.com

[1] Brigham Young University
Provo, Utah, USA
[2] Adobe Research
San Jose, California, USA

Abstract

This paper presents a GAN for generating images of handwritten lines conditioned on arbitrary text and latent style vectors. Unlike prior work, which produce stroke points or single-word images, this model generates entire lines of offline handwriting. The model produces variable-sized images by using style vectors to determine character widths. A generator network is trained with GAN and autoencoder techniques to learn style, and uses a pre-trained handwriting recognition network to induce legibility. A study using human evaluators demonstrates that the model produces images that appear to be written by a human. After training, the encoder network can extract a style vector from an image, allowing images in a similar style to be generated, but with arbitrary text.

1 Introduction

In this work, we generate images of lines of handwriting, conditioned on the desired text and a latent style vector. Handwriting is an expressive and unique form of communication that is often considered more intimate than typed text. Generating images that mimic an author's style would allow people to generate their own handwriting from typed text. While a convenience for many, this is particularly valuable to those with physical disabilities that hinder or prevent them from writing. Our results achieve human plausibility and begin to approximate the style of example handwriting images, as seen in Fig. 1.

Handwritten image generation can also provide additional data to train more accurate general handwriting recognition models [2]. Generating a large number of images in the

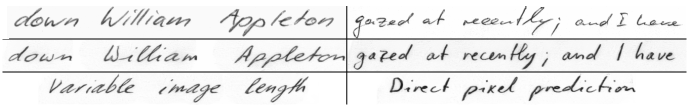

Figure 1: Examples of our model mimicking two authors' style. Top: original authors' writing. Middle: reconstructions using our model. Bottom: novel text using the same style.

style of each user of an application (e.g., scanner) could allow us to train personalized single-author recognition models. Personalized models tend to be more accurate for their target author's writing than a general recognition model.

Several previous approaches have framed handwriting generation as stroke prediction [8, 27, 29], i.e., modeling how a pen moves on paper. Training these methods often requires *online* handwriting data, captured by writing on a digital device. Without post-processing, such methods do not model ink textures that result from writing on physical media. In contrast, our method uses widely available *offline* handwriting data, i.e., images of the physical media. We frame handwriting generation as conditional image generation, directly learning from and predicting pixels [2, 6, 19].

Our approach achieves realism by combining Generative Adversarial Networks (GAN) [7] and autoencoder methods [23] with an auxiliary loss to achieve legibility [2]. Our encoder can map an example image into a latent style space, and then the model can produce images in a similar style, either reconstructing the original or using arbitrary text.

The width of an image of real handwriting depends on the text and writing style. Instead of fixing our model's output size or heuristically determining the output width from the input text, we use a deep network to predict the horizontal layout of the characters. Our model achieves state-of-the-art visual quality for offline handwriting generation and, unlike prior methods [2, 19], generates entire lines of handwriting conditioned on arbitrarily long text.

It is challenging to train a network using many (sometimes competing) loss functions that yield gradients that differ in size by orders of magnitude. To overcome this, we propose an improvement upon the gradient balancing technique of [2].

There are potential ethical concerns due to nefarious uses of this technology, e.g., low-skill convincing forgery. However, we believe this concern is minor as the method is not targeted at imitating signatures and can only produce digital images, not physical documents.

Our primary contributions are: (1) a method combining GANs and autoencoders to train a handwriting generator on offline handwriting images to produce realistic handwritten images that mimic example image styles; (2) a model that generates variable-length handwritten line images from arbitrary length text and style; and (3) improved multi-loss training through gradient balancing, allowing disparate losses to be used together more easily.

2 Prior Work

Conditional image generation methods take as input a description of the desired output image. Descriptions used in prior work include semantic layout masks [3, 14, 30, 32], sketches [4, 14, 32], image and desired attribute [5], image classes [28], key-words [18] and natural language descriptions [35, 36]. Many recent image generation approaches employ GANs [7], where the generator produces samples to fool a discriminator that attempts to classify images as real or generated. In our work, we employ a GAN and condition the output on both a target text and a latent style vector.

Recent GANs model the content and style of an image. StyleGAN [20] and the improved StyleGAN2 [21] learn a mapping from random noise vectors to style vectors that influence style by controlling the mean and magnitude of the generator network feature maps via AdaIN layers. MUNIT [13] translates images from one domain to another by learning autoencoders that encode the input image as separate, latent content and style vectors. FUNIT [24] builds on MUNIT to allow the target class to be unseen during training and instead be specified by a handful of images at test time. We similarly employ an autoencoder that reconstructs images from style and content, but our encoder only needs to extract the style vector. In contrast to both these methods, our content description is a variable-length text string instead of a fixed-sized latent vector, and we produce variable-width images based on the combination of the style vector and content.

Graves's well-known *online* handwriting generation LSTMs [8] predict future stroke points from previous stroke points and an input text. In contrast, we directly generate an image of *offline* handwriting that includes realistic ink textures. The authors of [1] use RNNs to perform online generation and explicitly model content and style separately. In [15], a GAN framework is used to train the generator of [8].

Alonso et al. [2] proposed an offline handwriting generator for isolated, fixed-size word images. It is trained using offline handwriting images. Their generator conditions on a fixed-size RNN-learned text embedding and random style vector. They train the generator in GAN fashion produce word images that fool a discriminator, but also include a loss to encourage legible text according to a jointly-trained handwriting recognition model. Specifically, the generator is updated to minimize the CTC loss [10] between the output of the recognition model and the input text. We similarly employ a recognition model to improve legibility, but we instead use a pre-trained feedforward network. In contrast, we produce higher quality handwritten line images from arbitrarily long text (instead of just words), and we can extract styles from existing images to generate similarly styled images.

Contemporary work [19] improves upon [2] by extracting style from a set of 15 word images from a single author. Generated images are fed to a writer classifier, and to learn writer styles, the generator is updated to fool this classifier. The generated word images are realistic, but don't recreate style perfectly. Recently, ScrabbleGAN [6] improved upon [2] by making the generated image width proportional to the input text length. In contrast, our model learns the output width based on the provided style and the input text.

3 Method

We view the handwriting generation process as having three inputs: content, style, and noise. Content is the desired text. Style is the unique way a writer forms characters using a particular physical medium and writing instrument. Noise is the natural variation of individual handwriting, even when writing the same content in the same style.

We train our model with GAN, reconstruction, perceptual, and text recognition losses. Fig. 2 shows an overview of our training process, including six networks: (1) a generator network G to produce images from spaced text, a style vector, and noise. (2) a style extractor network S, that produces a style vector from an image and the recognition predictions; (3) a spacing network C, which predicts the horizontal text spacing based on the style vector; (4) a patch-based convolutional discriminator D; (5) a pretrained handwriting recognition network R to encourage image legibility and correct content; and (6) a pretrained encoder E, to compute a perceptual loss.

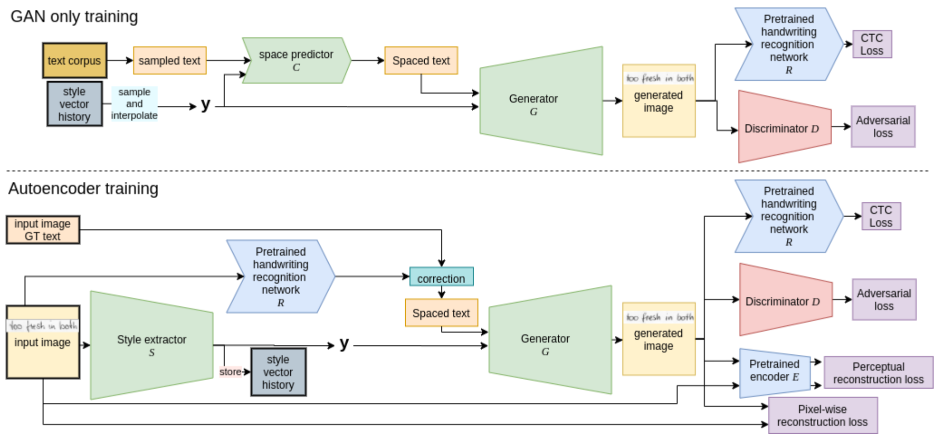

Figure 2: Overview of our method. To generate an image, we take input text and style vector $\mathbf{y}$. Text is spaced by spacer C using $\mathbf{y}$. This spaced text and $\mathbf{y}$ are passed to the generator G. We use both fully GAN training steps (top) and autoencoder based training steps (bottom).

During training the model learns to mimic style as S and G act as an encoder and decoder in an autoencoder, with E helping supply the reconstruction loss. G requires the input text to have spacing information (spaced text), which is extracted from the target image using R and the ground truth text (Sec. 3.3). G and D act as a GAN, supplying the model an adversarial loss for realism. R allows a handwriting recognition loss to supervise the legibility of generated images. Sampling a style vector and predicting spaced text using C allows the model produce novel images. C is supervised using styles predicted by S (not pictured in Fig 2).

We now present details for each part of our model and its training. Full architectural diagrams for G, S, C, D, E, and R are provided in supplementary material.

3.1 Generator *G* and Discriminator *D*

G is based on StyleGAN [20] but differs in architecture and receives the 1D spaced text as input with the style vector concatenated at each spatial position. Spaced text (Sec. 3.3) is a one-hot encoding of the target text with additional blank characters and repeated characters, which encode the spacing information. This informs horizontal character placement and was key to training the model successfully. The network blocks consist of a convolutional layer, additive noise, ReLU activation, and AdaIN [20], which uses the style vector to determine feature map statistics. To increase resolution, we use nearest-neighbor upsampling followed by a convolution and blurring operation. Most upsampling is in only the vertical dimension because the spaced text input is already wide.

Our discriminator D must be able handle variable sized inputs, so it is a fully convolutional, multi-resolution patch-based discriminator that we train with a hinge loss.

3.2 Style Extractor *S*

S inputs the image and the output of R on the image to produce a style vector (Fig 3). First, it uses a convolutional network to extract a 1D (horizontal) sequence of features. Then the recognition result (Fig. 4) is used to roughly localize each recognized character in the feature sequence. For each predicted character, we crop the feature sequence with a window size of

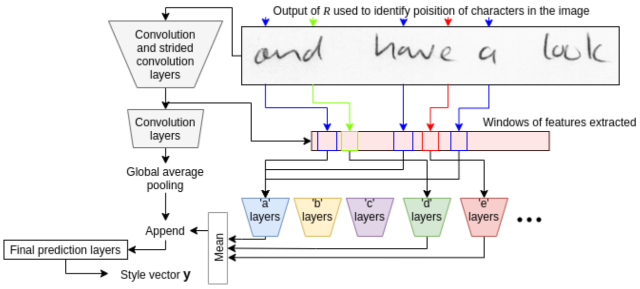

Figure 3: Style Extractor S architecture with character-specific heads to process feature windows from detected character locations.

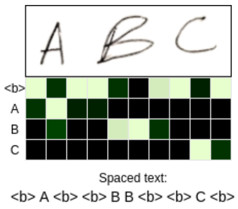

Figure 4: Example output of R (lighter is higher prob.) and corresponding spaced text.

5 (roughly 40 pixels, an area slightly larger than most characters in the IAM dataset) centered on the character and then pass each window through character-specific layers to extract character features. Features from all instances of all characters are averaged, weighted by R's predicted confidence for each instance, giving a final character feature vector.

To obtain global style features, we pass the entire feature sequence through 1D convolutional layers and perform global average pooling. This is appended to the character feature vector, from which fully connected layers predict the final style vector of dimension 128.

3.3 Spaced Text and Spacing Network C

We found spaced text essential for training with a reconstruction loss. Without it, G has difficulty achieving horizontal alignment with the input image and fails to train. Spaced text can be derived for a particular image from the output of R or predicted directly by C for a novel style.

Width and spacing are encoded using repeated characters and blank symbols <b> (Fig. 4). *Dataset spaced text* is obtained by taking the predicted character at each horizontal position from the output of R on a dataset image (Fig. 4), keeping blanks and repeated characters (artifacts typically removed when decoding the output of a CTC trained model). We correct recognition errors in the dataset spaced text using the ground truth text.

C is a 1D convolutional network that consumes one-hot encoded target text with the style vector concatenated to each position. For each character c_i, C predicts how many blanks precedes c_i and how many times c_i is repeated in the spaced text. Multiple blanks are then appended to the output. C is trained to imitate the dataset spaced text using a MSE loss.

3.4 Handwriting Recognition Network R

R is a pretrained handwriting recognition network that encourages generated images to (legibly) contain the specified text by applying the CTC loss [10]. R's weights are frozen so the gradient merely flows through R to supervise G.

While state-of-the-art handwriting recognition methods [9, 31, 34] use CNN-RNNs, we obtained better results with R as a fully convolutional network based on [34]. RNNs have arbitrarily large context windows and may predict characters based on linguistic context instead of visual character shapes. In contrast, R only uses local visual features for character predictions and therefore provides better feedback for generating characters.

R is pretrained with CTC loss and warp grid data augmentation [33], which results in overall better generated images, but training the model on warped images causes some artifacts in the absence of the reconstruction loss (e.g., the first ablated model in Fig. 9).

3.5 Encoder Network *E*

E provides features for our perceptual loss [17]. Traditional methods for computing perceptual loss use features from pretrained image classification networks [16]. However, images of handwriting are different from natural images, so we choose a different approach. *E* is a fully convolutional network that collapses the image to a one-dimensional feature series capturing visual and semantic features. *E* is trained both as an autoencoder with a decoder and L1 reconstruction loss, and as a handwriting recognizing network with CTC loss.

3.6 Training/Losses

Our generation has three objectives: legible handwriting matching the target text, realistic handwriting that appears to be a human's, and handwriting style that mimics an example image. Each of these is achieved primarily through the respective losses: CTC loss backpropagating through *R*, adversarial loss, and reconstruction losses (pixel and perceptual). Additionally, MSE is used to train the spacing network *C*, and hinge loss is used to train *D*.

When using multiple loss terms, balancing them is crucial. We do so by improving the gradient-balancing method of [2]. Without balancing, training failed due to exploding gradients or failed to converge. Stable hyper-parameters possibly exist, but gradient balancing easily solved the problem. We balance gradients from CTC, adversarial, and reconstruction losses. The two reconstruction losses have equal weight and are summed.

In [2] the CTC loss gradient is normalized to have the same mean and standard deviation as the adversarial loss gradient. However, this does not preserve the sign of the CTC gradient, so we instead normalize the gradients to have the same mean magnitude (per layer). This additionally allows balancing multiple gradients. Totally equal contributions may not be desirable and can be adjusted by multiplicative weights on each gradient after normalization. We always use the gradient magnitude of the reconstruction loss for gradient normalization.

To reduce memory requirements, some training steps store only gradients (for later balancing) and others update the parameters. Our curriculum uses the following steps:

1. **Spacing:** This is skipped on every other round through the curriculum. A style is extracted from two dataset images by the same author and *C* predicts the spacing. The MSE loss between the prediction and dataset spaced text updates both *C* and *S*.
2. **Discriminator:** To update *D* we sample novel styles by interpolating/extrapolating styles sampled from a running window history of the 100 most recently extracted styles (during Spacing or Autoencoder steps). Extrapolation is kept within 0.5 of the distance between the two styles and is sampled uniformly from that range.
3. **GAN-only:** This follows standard GAN training while including the handwriting recognition supervision. It does not update the model but saves the gradient information. It samples styles like the Discriminator step. See top of Fig. 2.
4. **Autoencoder:** Pairs of images by the same author are concatenated width-wise, and a single style vector is extracted for both of them. Then each image is individually reconstructed using that style. We compute the reconstruction, adversarial and handwriting recognition losses with the reconstructed images. The gradients from this step and the GAN-only step are balanced. Both *S* and *G* are updated. See bottom of Fig. 2.

We now define the loss functions used in training our model and formalize the gradient balancing. Let I be a dataset image, t_I its corresponding text, and c_I its corresponding dataset spaced text. Let I' be the concatenation of I and another image by the same author. Let y_s be a sampled style, obtained by sampling two stored style vectors from the running window history and interpolating/extrapolating a point on the line between them. Let t_s be text sampled from a text corpus. MSE and $L1$ are mean squared error and L1 loss. CTC is connectionist temporal classification loss [10]. S, G, R, E, C, and D are the networks described in Sec. 3.

$$\text{Spacing network loss } l_c = MSE(C(t_I, S(I')), c_I) \tag{1}$$

$$\text{Discriminator loss } l_d = max(1 - D(I), 0) + max(1 + D(G(C(t_s, y_s), y_s)), 0) \tag{2}$$

$$\text{Generated image adversarial loss } l_{adv,g} = -D(G(C(t_s, y_s), y_s)) \tag{3}$$

$$\text{Generated image recognition loss } l_{rec,g} = \text{CTC}(R(G(C(t_s, y_s), y_s), t_s) \tag{4}$$

$$\text{Reconstructed image adversarial loss } l_{adv,r} = -D(G(c_I, S(I'))) \tag{5}$$

$$\text{Reconstructed image recognition loss } l_{rec,r} = \text{CTC}(R(G(c_I, S(I')), t_I)) \tag{6}$$

$$\text{Combined reconstruction loss } l_{auto,r} = L1(G(c_I, S(I')), I) + L1(E(G(c_I, S(I'))), E(I)) \tag{7}$$

∇l_c is used to updated C and S, and ∇l_d is used to update D. The remaining gradients are balanced. Let $m^i_{\nabla l_x}$ be the mean absolute gradient of loss l_x for layer i in the model. The gradient of each loss $l_x \in \{l_{adv,g}, l_{rec,g}, l_{adv,r}, l_{rec,r}\}$ is normalized by multiplying each layer i's gradient by $m^i_{\nabla l_{auto,r}} / m^i_{\nabla l_x}$. After normalization, the weighted sum $\nabla l_{auto,r} + 0.5(\nabla l_{adv,g}) + 0.6(\nabla l_{rec,g}) + 0.4(\nabla l_{adv,r}) + 0.75(\nabla l_{rec,r})$ is used to updated G and S. The weights were chosen heuristically to emphasize the parts we found the model struggled with.

We use a batch size of four, being two pairs of images by the same author for Autoencoder and Spacing steps. We train our model for 175,000 steps of the curriculum. The stopping point was based on subjective evaluation of the validation set. We use two Adam optimizers in training; one for the discriminator, and one for the rest of the model (except the pretrained R and E). Both optimizers use a learning rate of 0.0002 and betas of (0.5,0.999).

4 Experiments

We first discuss the data used. Then we compare to prior methods. We then show exploration into our method with an ablation study (Sec. 4.1) and an examination of the style space (Sec. 4.2). We finally discuss a user study we performed (Sec. 4.3).

We use the IAM handwriting dataset [25] and the RIMES dataset [11], which contain segmented images of handwriting words and lines with accompanying transcriptions. We developed our method exclusively with the IAM training (6,161 lines) and validation (1,840 lines) sets, and reserved the test sets for experiments (FID/GS scores use training images). Note that IAM consists of many authors, but authors are disjoint across train/val/test splits. We resize all images to a fixed height of 64 pixels, maintaining aspect ratio. We apply a random affine slant transformation to each image in training (-45°, 45° uniform).

Fig. 5 compares our results to those from Alonso et al. [2] and ScrabbleGAN [6]. Our results appear to have similar quality as [6]. It can be seen in Fig. 6 that [6] (left) lacks diversity in horizontal spacing; despite the style changing, the images are always the same length. This is due to their architectural choice to have the length dependant on content, not style. Our method takes both content and style into consideration for spacing, leading to variable length images for the same text. We report Fréchet Inception Distance (FID) [12]

	Dataset	FID	GS
Alonso et al. [2]	RIMES words	23.94	8.58×10^{-4}
ScrabbleGAN [6]	RIMES words	23.78	7.60×10^{-4}
Ours (trained on RIMES lines)	RIMES words	37.60	1.70×10^{-1}
Ours (trained on RIMES lines)	RIMES lines	23.72	8.29×10^{-1}
Ours (trained on IAM lines)	IAM lines	20.65	1.10×10^{-2}

Table 1: FID and GS scores in comparison to prior methods.

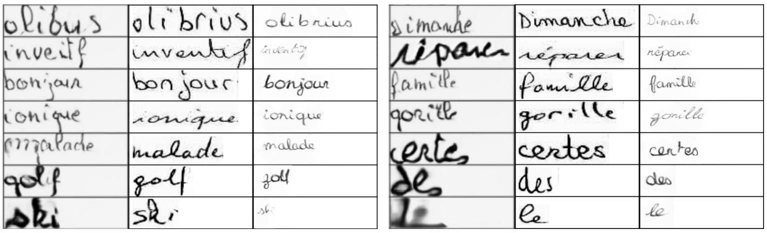
Figure 5: Comparing to prior methods on the RIMES dataset. Left: Alonso et al. [2], middle: ScrabbleGAN [6], right: ours. Our model was trained using full lines, whereas the other two used word images. Segmentation differences caused our model to produce smaller text.

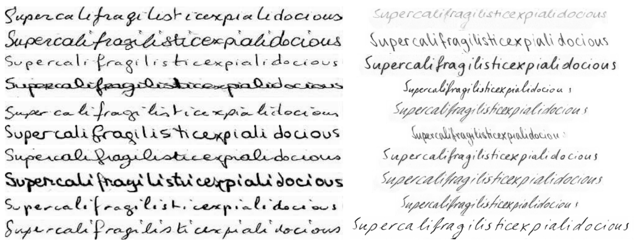
Figure 6: Contrasting variability of image length for ScrabbleGAN [6] (left) and our method (right) using a fixed word. ScrabbleGAN's horizontal spacing is mostly style agnostic, whereas the spacing in our model is style sensitive.

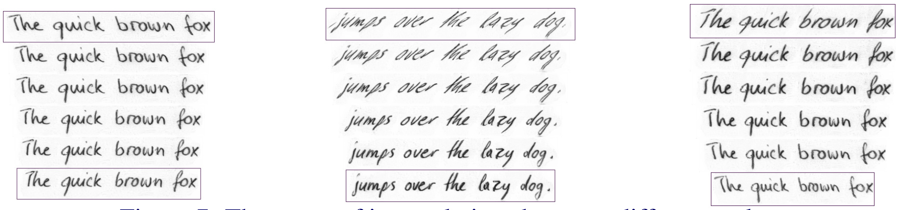
Figure 7: Three sets of interpolations between different styles.

and Geometry Score (GS) [22] in Table 1 using a setup similar to [6]. There exist some intricacies to the FID and GS calculation which are included in the Supplementary Materials.

Fig. 7 shows interpolation between three sets of two styles taken from test set images. These images look realistic, even on the interpolated styles. Notice the model even predicts faint background textures similar to dataset images. We note that while styles vary, it mostly varies in terms of global style elements (e.g., slant, ink thickness); the variation rarely comes from character shapes. Figs. 7 and 6 were generated with text not present in the training set; We notice no difference when generating with text from the dataset compared to other text. Fig. 8 shows reconstruction results of our model. The model mimics aspects of global style,

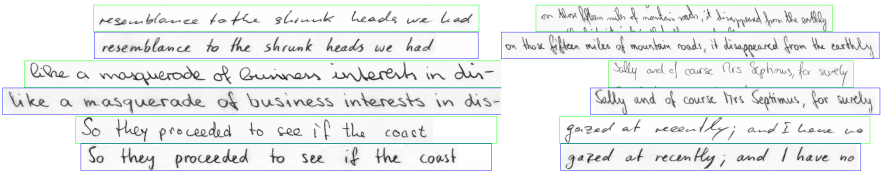
resemblance to the shrunk heads we had
resemblance to the shrunk heads we had
like a masquerade of business interests in dis-
like a masquerade of business interests in dis-
So they proceeded to see if the coast
So they proceeded to see if the coast
on those fifteen miles of mountain roads, it disappeared from the earthly
on those fifteen miles of mountain roads, it disappeared from the earthly
Sally and of course Mrs Septimus, for surely
Sally and of course Mrs Septimus, for surely
gazed at recently; and I have no
gazed at recently; and I have no

Figure 8: Reconstruction results. Green is original, blue is our model's reconstruction.

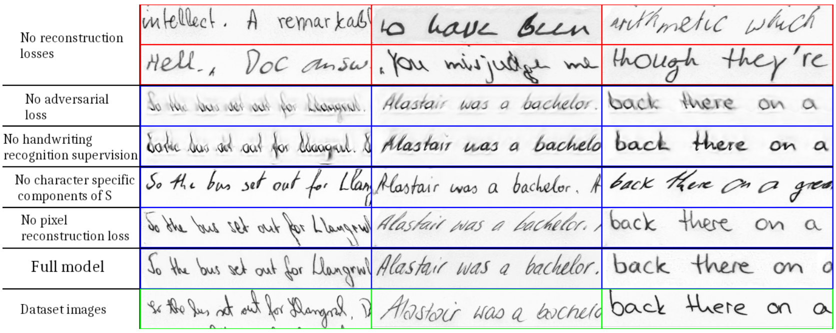
No reconstruction losses
No adversarial loss
No handwriting recognition supervision
No character specific components of S
No pixel reconstruction loss
Full model
Dataset images

Figure 9: Ablation study. Red images generated using randomly sampled styles. Blue images are attempting to reconstruct the bottom (green) images.

but often fails to copy character shape styles (e.g., whether the author loops the letter 'l'). Additional results are provided as supplementary material.

4.1 Ablation Study

We conducted an ablation study (see Fig. 9) by removing several key components of our model: the adversarial loss, the handwriting recognition loss, the autoencoding reconstruction losses, and the character specific heads in the style extractor. Without the reconstruction loss, the model still generates plausible images and has variety. However, the character shapes are not as well formed. Without the adversarial loss, the model produces blurry results. Curiously, the model produces legible images without the handwriting recognition loss, but with decreased realism. The reconstruction loss is likely responsible for legibility, but we are unsure why realism would suffer. Without the character specific components of S the model loses some ability to mimic styles. The pixel reconstruction loss only slightly improves image quality, and without the perceptual loss, the model was unable to converge.

We also were unable produce good results without using spaced text. We attempted a model without reconstruction and used 1D convolutions to allow the model to learn the spacing on its own. It failed to produce legible results. We attempted to train our model with the gradient balancing of [2], however the model failed to train. Earlier versions of our model successfully trained with that gradient balancing, but with decreased quality.

4.2 Latent Style Space

We are able to show evidence that S extracts styles meaningful at the author level. In Fig. 10 we show a UMAP [26] projection of style vectors extracted from the test set images. Styles extracted from the same author tend to be near each other. There is no specific loss to encourage this behavior; this clustering is learned as the model learns to reconstruct images.

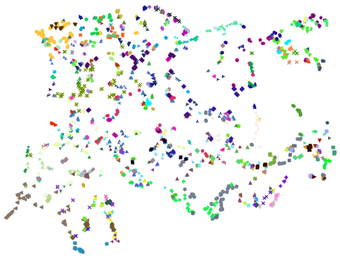

Figure 10: UMAP projection of the styles extracted from the test set images. Shape and color indicate author. Most styles from the same author cluster together, even though the model was not explicitly trained to do this.

Actually:	Guessed: Human	Guessed: Computer
Human	34.2%	15.8%
Generated	31.9%	18.0%
Poorly generated	10.5%	89.5%

Table 2: Top two rows are a confusion matrix of the human study results. Bottom row shows results on deliberately poor generated images as a measure of participant attention.

Taking the L2 distances between style vectors, the mean distance between styles taken from the same author is 0.916 with a standard deviation of 0.658, and the mean distance between styles taken from different authors is 2.264 with a standard deviation of 1.367.

4.3 Human Evaluation

We evaluated the realism of our generated images using Amazon Mechanical Turk. Participants viewed a single image at a time and were asked if the image was written by a human or a computer. Real images were sampled from the test set. Generated images used the same text. Styles were interpolated between styles extracted from the test set. After control measures to ensure participant reliability (described in supplementary material), 14,875 responses contributed to the final evaluation. Overall, the participants had an accuracy of 52.2% at determining whether an image was human or computer generated, indicating that our generated images are generally convincing. A confusion matrix of the results is presented in Table 2; there is a strong bias towards predicting the images to be human generated.

We also included deliberately poorly generated images for which we expected close to 100% accuracy for attentive participants. Our participants had 89.5% accuracy on these poorly generated images, indicating that the lack of distinguishability between real and generated images was not simply due to inattention. While our generated images fooled most participants, we note that the best performing participants (top 10%) had an average accuracy of 84.9%, indicating a wide range of participant performance. See Supplementary Materials for details about this experiment.

5 Conclusion

We have presented a system capable of directly generating the pixels of a handwriting image of arbitrary length. Our generation is conditioned both on text and style and relies on an spacing network to predict the space needed between text, enabling the generation of arbitrary length images. Our model is capable of extracting a style from example images and then generating handwriting in that style, but with arbitrary text. Our method does well at capturing the variations of global style in handwriting, such as slant and size.

References

[1] Emre Aksan, Fabrizio Pece, and Otmar Hilliges. DeepWriting: Making Digital Ink Editable via Deep Generative Modeling. In *SIGCHI Conference on Human Factors in Computing Systems*, CHI '18, 2018.

[2] Eloi Alonso, Bastien Moysset, and Ronaldo Messina. Adversarial generation of handwritten text images conditioned on sequences. In *15th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Sep 2019.

[3] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[4] Tao Chen, Ming-Ming Cheng, Ping Tan, Ariel Shamir, and Shi-Min Hu. Sketch2photo: Internet image montage. *ACM Trans. Graph.*, 28(5), 2009.

[5] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[6] Sharon Fogel, Hadar Averbuch-Elor, Sarel Cohen, Shai Mazor, and Roee Litman. ScrabbleGAN: Semi-supervised varying length handwritten text generation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[8] Alex Graves. Generating sequences with recurrent neural networks. *ArXiv*, abs/1308.0850, 2013.

[9] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[10] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 2006.

[11] E. Grosicki and H. E. Abed. ICDAR 2009 handwriting recognition competition. In *10th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2009.

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS) 30*. 2017.

[13] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[15] B. Ji and Tianyi Chen. Generative adversarial network for handwritten text. *ArXiv*, abs/1907.11845, 2019.

[16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[17] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Proceedings of the European Conference on Computer Vision (ECCV))*, 2016.

[18] Matthew Johnson, G. J. Brostow, J. Shotton, V. Kwatra, and R. Cipolla. Semantic photosynthesis. In Bernice E. Rogowitz, Thrasyvoulos N. Pappas, and Scott J. Daly, editors, *Human Vision and Electronic Imaging XII*, volume 6492. International Society for Optics and Photonics.

[19] Lei Kang, Pau Riba, Yaxing Wang, Marccal Rusinol, Alicia Fornés, and Mauricio Villegas. GANwriting: Content-conditioned generation of styled handwritten word images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, August 2020.

[20] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[21] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of styleGAN. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[22] Valentin Khrulkov and Ivan Oseledets. Geometry score: A method for comparing generative adversarial networks. In *Proceedings of the 35th International Conference on Machine Learning (PMLR)*, 2018.

[23] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of the 33rd International Conference on Machine Learning (PMLR)*, Jun 2016.

[24] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[25] U-V Marti and Horst Bunke. The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1), 2002.

[26] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[27] John F. J. Mellor, Eunbyung Park, Yaroslav Ganin, Igor Babuschkin, Tejas Kulkarni, Dan Rosenbaum, Andy Ballard, Théophane Weber, Oriol Vinyals, and S. M. Ali Eslami. Unsupervised doodling and painting with improved spiralq. *ArXiv*, abs/1910.01007, 2019.

[28] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[29] Reiichiro Nakano. Neural painters: A learned differentiable constraint for generating brushstroke paintings. In *Machine Learning for Creativity and Design (NeurIPS workshop*, Dec 2019.

[30] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[31] J. Puigcerver. Are multidimensional recurrent layers really necessary for handwritten text recognition? In *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Nov 2017.

[32] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[33] C. Wigington, S. Stewart, B. Davis, B. Barrett, B. Price, and S. Cohen. Data augmentation for recognition of handwritten words and lines using a cnn-lstm network. In *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[34] Curtis Wigington, Chris Tensmeyer, Brian Davis, William Barrett, Brian Price, and Scott Cohen. Start, follow, read: End-to-end full-page handwriting recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[35] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[36] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 41(8), 2018.