The Visual Computer
https://doi.org/10.1007/s00371-024-03534-9

REVIEW

Generative adversarial networks for handwriting image generation: a review

Randa Elanwar[1] · **Margrit Betke**[2]

Accepted: 31 May 2024

Abstract
Handwriting synthesis, the task of automatically generating realistic images of handwritten text, has gained increasing attention in recent years, both as a challenge in itself, as well as a task that supports handwriting recognition research. The latter task is to synthesize large image datasets that can then be used to train deep learning models to recognize handwritten text without the need for human-provided annotations. While early attempts at developing handwriting generators yielded limited results [1], more recent works involving generative models of deep neural network architectures have been shown able to produce realistic imitations of human handwriting [2–19]. In this review, we focus on one of the most prevalent and successful architectures in the field of handwriting synthesis, the generative adversarial network (GAN). We describe the capabilities, architecture specifics, and performance of the GAN-based models that have been introduced to the literature since 2019 [2–14]. These models can generate random handwriting styles, imitate reference styles, and produce realistic images of arbitrary text that was not in the training lexicon. The generated images have been shown to contribute to improving handwriting recognition results when augmenting the training samples of recognition models with synthetic images. The synthetic images were often hard to expose as non-real, even by human examiners, but also could be implausible or style-limited. The review includes a discussion of the characteristics of the GAN architecture in comparison with other paradigms in the image-generation domain and highlights the remaining challenges for handwriting synthesis.

Keywords Handwriting synthesis · Generative adversarial networks · Text-image generation

1 Introduction

In supervised machine learning, the availability of labeled datasets is crucial to the training and validation of models. Unfortunately, there are still many domains without sufficiently large labeled datasets to develop robust supervised learning solutions, making the resulting models less general and susceptible to over-fitting. Since data collection and annotation processes are typically costly, time-consuming, tedious, and error-prone (because they necessarily depend on humans), creating training data without human labeling help has become an important goal. The artificial generation of data samples, also termed data synthesis, has been widely adopted as a data augmentation approach [20–26]. Data synthesis also has a place in semi-supervised learning, which depends on a small set of labeled samples to automatically label a larger set of unlabeled data or to amend the behavior of a model self-trained on some unlabeled data.

In this review, we focus on the specific problem of data synthesis that requires the generation of images of text that looks handwritten. Being able to synthesize realistic-looking images of handwriting can then support the development of supervised or semi-supervised machine learning models that can recognize handwritten text. Handwriting recognition is an important technology for digitizing and understanding daily (and historic) data production. Research on handwriting recognition has been active for decades, addressing the lack of large human-annotated training data by developing approaches to handwriting synthesis that can provide automatically labeled training data. We reviewed the works

✉ Randa Elanwar
randa.elanwar@eri.sci.eg

Margrit Betke
betke@bu.edu

[1] Computers and Systems Department, Electronics Research Institute, Cairo, Egypt

[2] Department of Computer Science, Boston University, Boston, USA

Published online: 02 July 2024

presenting these attempts in publications between 2003 and 2013 [1].

Previous attempts to generate handwritten text were based on a collection of samples for a few characters, either human-written or built, using templates [1]. To generate words in cursive script, these methods perturb the character templates and then concatenate them. However, since these templates are specific to a given data corpus, they cannot represent arbitrary handwriting styles or go beyond the textual content of the corpus.

These early attempts at handwriting generation, previously reviewed by Elanwar [1], were based on algorithms that inject some noise into real samples of glyphs, i.e., symbols of characters. Perturbation models or affine transformations usually succeed in giving the generated glyphs a realistic look but they are not able to achieve smooth connectivity between glyphs for cursive handwriting. Few of the early systems were based on machine learning to suggest one-out-of-many learned styles for glyph shapes and then connect them using transitional strokes. Such systems dedicated a model to learn the connection shape between adjacent characters. There are newer publications following this methodology, working with limited-size data and depending on probabilistic models or other algorithms, for example, Lian et al. in 2018 [27] and Souibgui et al. in 2022 [28]. Such methods are known as "few-shot generation modeling" or "few-shot compositional generation." As argued by Souibgui et al. [28], these solutions might make the best choice for low-resource languages, for which publicly available handwritten text datasets are scarce. In recent years, the development of deep neural networks has produced flexible and generic alternatives to the previous algorithms, enabling the synthesis of a variety of handwriting styles and words that were not in the training lexicon of images, also called out-of-vocabulary (OOV) words.

In this article, we review the solutions recently published on handwriting synthesis that use a generative adversarial network (GAN) architecture. It is notable that the work by Goodfellow et al. [29], which introduced the GAN framework in 2014, included the synthesis of handwritten digits as a use case for GANs. In 2018, isolated handwritten Chinese characters images were rendered from a printed font [30] using the GAN-variant "CycleGAN" [31]. We here go beyond the single-digit task and focus on models that can generate images of words or even sentences. These models started to emerge in 2019 with the seminal work by Alonso et al. [2], which we describe first. We then follow with reviews of the GAN-based handwriting synthesis models published in 2020 by Fogel et al. [3], Kang et al. [6] and Davis et al. [8], in 2021 by Zdenek and Nakayama [4], Liu et al. [5], and Gan and Wang [7], and in 2022 by Gan et al. [9] and Luo et al. [10]. To the best of our knowledge, these are the important pioneering works on GAN-based handwriting synthesis to date. The GAN-based models published after 2022 [11–14] are extensions to these nine models that use different datasets or refine one or more of their model components but do not introduce any major changes to them.

Beyond the scope of our paper is a detailed review of other generative deep learning approaches to handwriting synthesis that use transformers [15–17] or diffusion models [18, 19]. However, we discuss these paradigms of image generation in Sect. 4 with respect to their differences and similarities to GAN-based models and derive some insights for the future of generative learning approaches to handwriting.

The task we have discussed so far, also called "offline handwriting generation," requires a model to synthesize images that look like handwriting, without having to provide information about how a human hand could have written the text with a pen. On the other hand, a model for "online handwriting generation" not only synthesizes an image of text but also computes a sequence of "digital ink stamps" that explain how a human hand would have guided a pen spatially and temporally to write the text. Discussing the recent works on online handwriting synthesis [32–37] is also beyond the scope of our review.

An important motivation to developing handwriting synthesis models, as we outlined above, is their use in supporting the development of handwriting recognition models, offering an endless source of training data at minimum cost. However, we also want to stress that learning how to create realistic handwriting styles might be a goal in itself. Use cases may evolve with the growth of personalization technology. Handwriting synthesis may have a role in marketing personalized gifts and luxury products. A motivation to using the GAN architecture in particular is that it involves the comparison of real and synthesized handwriting and thus can be used to detect the forgery of signatures, preventing illegal access to financial assets, blackmailing, or planting of false evidence in crimes.

This review paper is structured as follows. We describe the seminal GAN-based model architecture for handwriting synthesis in Sect. 2. We then devise a categorization scheme of text-image-generating GANs in the introduction of Sect. 3. Section 3.1 describes the datasets used for model evaluation, and Sect. 3.2 describes the evaluation mechanisms. Next, we review the architecture specifics of the nine models under consideration (Sects. 3.3–3.11) and conclude with a comparison of their capabilities and features (Sect. 3.12).

We briefly introduce the other text-image-generation models which preceded and followed the appearance of GANs and discuss their similarities with and differences to GANs in Sect. 4 and conclude with a discussion in Sect. 5.

2 GAN-based handwriting synthesis: the seminal model

This section describes the seminal model for GAN-based handwriting synthesis, proposed by Alonso et al. [2]. It is a variant of the original GAN architecture proposed by Goodfellow et al. [29], which functions as follows (see Fig. 1): A generator network G maps a random (latent) noise vector z to a sample in the image space to fool a discriminator network D, which attempts to classify this sample image as a real or generated (fake) image. The adversarial loss computed by the discriminator is used to optimize both the generator and the discriminator networks' weights. During training, G learns to generate more realistic images that D fails to discriminate correctly.

The original GAN architecture [29] suffers from the drawback of not having control over the generated image content—in our case, which words are being synthesized by the network. The conditional GAN model [38] was therefore adopted to let the user specify which words the GAN should generate. The input text t is encoded by an embedding network into the vector y (see Fig. 1). This network is also referred to as a "content encoder" or "text encoder." Its role is to embed the target text into a fixed length vector y that is used as a condition input to the generator. With this mechanism, it is possible to generate specific handwritten word images $G(z, y)$ by pairing a latent space vector z with the desired text t.

For the use case of creating training datasets for handwriting recognition systems, it is important that the synthesized text is legible by such systems. Alonso et al. [2] therefore proposed to augment the original GAN architecture with an additional module, the recognition network R (Fig. 1), with the goal to ensure that the output of the synthesis model is recognizable, i.e., legible. During the training of the GAN architecture, the recognition error of this network R is added to the training loss to guide network G to generate legible words.

To enable follow-up with the different modifications of the base model in Fig. 1 introduced by the reviewed generative models, we summarize the notation of the base model in Table 1. We now provide the definitions of the relevant loss functions.

The goal of the discriminator D is to label real images x as true (1) and generated images $G(z)$ as false (0). The loss function of the discriminator D can thus be defined as

$$l_{\mathrm{D}} = \mathrm{Error}(D(x), 1) + \mathrm{Error}(D(G(z)), 0). \tag{1}$$

The goal of the generator G is to confuse the discriminator to mislabel generated images $G(z)$ as being true. Therefore, the generator loss is

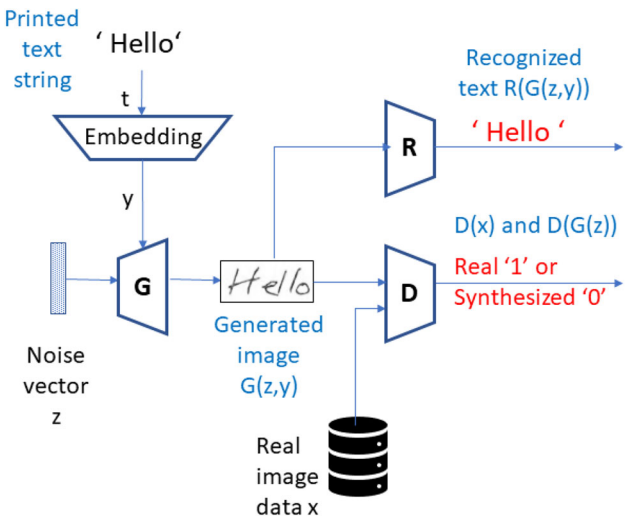

Fig. 1 The main GAN-based architecture used for handwriting generation, first proposed by Alonso et al. [2], and also adopted by Fogel et al. [3] and Zedenek and Nakamaya [4]. In addition to the generator–discriminator network pair (G, D), used in all GANs, a text embedding network helps condition the GAN on text embedding y, which represents the target string t, and a recognition network R guides the generator G to synthesize text images $G(z, y)$. The discriminator network D is trained by alternating generated $G(z, y)$ and real image samples x. The discriminative decisions $D(x)$ and $D(G(z))$ contribute to calculating the adversarial loss l_{D} needed to update the weights of both G and D. The recognition result $R(G(z, y))$ of the generated image contributes to calculating the recognition loss l_R needed to update G. This base architecture will be represented by a dashed rectangle in the following figures to highlight the modifications introduced by other works

Table 1 Notation in reviewed equations

Symbol	Meaning
t	Condition (target) text string
$R(G(z))$	Recognized text string from recognizer R
y	Embedding vector of t
z	Latent noise vector
x	Real image from dataset
$G(z)$	Synthesized image from generator G
l_{D}	Discriminator loss function
l_{G}	Generator loss function
l_{adv}	Adversarial loss function
l_{R}	Recognizer loss function

$$l_{\mathrm{G}} = \mathrm{Error}(D(G(z)), 1). \tag{2}$$

Since the task of the discriminator D is a binary classification problem, applying the binary cross-entropy function will measure the difference between the distributions of x and z for image space X and latent space ζ. This yields the equations

$$l_{\mathrm{D}} = -\sum_{x\in X, z\in\zeta} (\log(D(x)) + \log(1 - D(G(z)))), \tag{3}$$

and

$$l_{\mathrm{G}} = -\sum_{z\in\zeta} \log(D(G(z))). \tag{4}$$

The losses l_{D} and l_{G} can be combined into one loss function

$$l_{D,G} = \mathbb{E}_x[\log(D(x))] + \mathbb{E}_z[\log(1 - D(G(z)))], \tag{5}$$

where $\mathbb{E}_x$ and $\mathbb{E}_z$ are the expected values over the distributions of x and z, respectively. Training the discriminator D aims at maximizing Eq. 5 (i.e., to tell apart real and fake images), while training the generator G aims at minimizing Eq. 5 (i.e., to minimize the distance between the distributions of x and z by generating realistic images $G(z)$). Furthermore, to condition the GAN to generate images of specific text t, Eq. 5 needs to be updated by replacing the distributions for x and z with distributions conditioned on the embedding y of t:

$$l_{D,G} = \mathbb{E}_x[\log(D(x|y))] + \mathbb{E}_z[\log(1 - D(G(z|y)))]. \tag{6}$$

Finally, the loss function for the recognition network R is defined as

$$l_{\mathrm{R}} = \mathbb{E}_{(z,t)}[\mathrm{CTC}(t, R(G(z, y)))], \tag{7}$$

which is based on the connectionist temporal classification (CTC) algorithm [39] for training neural networks to recognize words as sequences of letters without explicit segmentation of these words into letters. It is a dynamic programming algorithm that maximizes the log probability over all possible text segmentations.

3 Review of specific GAN-based handwriting generation systems

We categorize GAN-based models for handwriting synthesis according to the input used to generate images. There are two main categories, style transfer GANs and conditioned GANs. Style transfer GANs are uni-modal architectures (img2img) that take a two-dimensional (2D) image as input and generate a 2D output image. Conditioned GANs, on the other hand, take as input not only a 2D image but also attribute vectors that represent various types of information, for example, a class label, a style vector, a text embedding, etc. Conditioned GANs are therefore multi-modal architectures that take a 2D image and other conditions as input and generate an output image obeying these conditions.

GAN-based models for handwriting synthesis are predominantly conditioned GANs, which can be conditioned on text input to generate a random-style handwritten word corresponding to the input text [2–4, 12]. The text could be as short as one character or a single word, or as long as a complete sentence. GANs can additionally be conditioned using a style vector to control the style of the generated handwritten word in terms of skew, character size, line thickness, cursiveness, etc. The style vector could be explicitly fed to the GAN [10] or learned by the GAN from an input image, i.e., a reference style [5–9, 11, 13, 14]. Accordingly, handwriting generation GANs can be categorized according to the generation process as GANs generating random styles and GANs reproducing input styles. Furthermore, GANs can be categorized according to the generated image size or content as GANs generating variable-size output images [3, 4, 7–12, 14], GANs generating arbitrary-length words [3–5, 7–10, 12, 14], and GANs generating unconstrained or out-of-vocabulary (OOV) text [2, 4–11, 13]. The next section describes the instantiations of these categories.

In this section, we first describe the datasets used to train and evaluate the reviewed models (Sect. 3.1), and then, we explain the different qualitative and quantitative evaluation methods (Sect. 3.2). Next, we review nine GAN-based architectures generating images of handwritten text (Sects. 3.3–3.11). Finally, in Sect. 3.12, we compare the performance of the nine architectures based on the evaluation methods previously explained in Sect. 3.2.

3.1 Datasets for handwriting generation

The reviewed models have been trained and evaluated on publicly available datasets of handwritten text, as shown in Table 2, which facilitates comparing their results. The datasets used are:

- *The IAM dataset by Marti and Bunke (2002)* [40] This dataset contains about 100k images of words from the English language. It is divided into training, test, and two validation sets. The dataset is divided into words written by 657 different authors. The train, test, and validation sets contain words written by mutually exclusive authors. In other words, all words written by each author only appear in one of the four sets. This dataset was used by all nine of the reviewed works.
- *The CVL dataset by Kleber and Sablatnig (2013)* [41] This dataset consists of seven handwritten documents (one German and six English texts) with about 83K words, written by 310 writers. It is divided into train and test sets. The English part of this dataset was used by four of the reviewed works [3, 4, 7, 10].
- *The REMIS dataset by Grosicki and ElAbed (2009)* [42] This dataset is composed of made-up mail and fax let-

Table 2 Datasets used by different GAN-based architectures for offline handwriting generation

Dataset	Size	Language	No. of Authors	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
IAM	100k	English	657	✓	✓	✓	✓	✓	✓	✓	✓	✓
CVL	83K	German/English	310	×	✓	✓	×	×	✓	×	×	✓
REMIS	13K	French	1300	✓	✓	×	✓	×	×	✓	×	✓
OpenHaRT	43K	Arabic	unknown	✓	×	×	×	×	×	×	×	×

The GAN-based models are: Alonso et al. [2], ScrabbleGAN [3], JokerGAN [4], HTG-GAN [5], GANwriting [6], HiGAN [7], Davis et al. [8], HiGAN+ [9], and SLOGAN [10]

ters written in French. 12,723 pages written by 1,300 volunteers have been collected and scanned. More than 250k snippets of words have been extracted from the letters. The dataset is divided into training (43k), validation (70k+), and test (7,464) subsets. This dataset was used by five of the reviewed works [2], [3], [5], [8], [10].

- *The OpenHaRT dataset by Tong et al. (2010 and 2013)* [43] This dataset offers Arabic handwritten text, obtained at the document level, and includes a large vocabulary. It was collected in three phases (2008–2011). The handwriting of native Arab speakers, who copied news lines in their handwriting, was scanned. The dataset is divided into training (approx. 42k pages), validation (approx. 500 pages), and test (approx. 600 pages) sets. This dataset was used by Alonso et al. [2].

Fig. 2 Examples of successful latent sampling, resulting in variable-size images with single words or arbitrary-length text

3.2 Assessment of generated handwriting images

To evaluate the performance of their architectures and be able to compare their results, researchers adopted different methods for expressing and assessing their findings, They displayed the generated images in different fashions to show their models' capabilities and also computed image similarity metrics to quantify these capabilities. Aside from assessing whether the generated results looked artificial or were masterful imitations of handwriting, they subjected their generated images to the recognition test by handwritten text recognition systems.

The visualization techniques used as qualitative assessment methods are: Latent-guided synthesis, style interpolation, word ladder, out-of-vocabulary (OOV), and long text synthesis. The quantitative assessment methods used are:

1. Handwritten text recognition (HTR) using evaluation metrics such as word error rate (WER), character error rate (CER), and normalized edit distance (NED).
2. Human (user) assessment using evaluation metrics such as accuracy (ACC), precision (P), recall (R), false-positive rate (FPR) and false omission rate (FOR), and user preference.
3. Quality and similarity measures using evaluation metrics such as geometry score (GS), Fréchet inception distance (FID), and multi-scale structural similarity image score (MS-SSIM).

3.2.1 Assessment by visualizing results

Latent-vector-guided synthesis One of the qualitative evaluation methods of the robustness of a generative process is presenting the GAN architecture with different randomly sampled noise vectors and different word conditions (Fig. 2). A realistic appearance of the resulting generated images in terms of fewer artifacts, a homogeneous background, and coherent character sizes and orientations would then indicate a robust GAN performance. Also, the legibility of the handwritten words and the matching of the word condition are other signs of good performance. Reference-guided synthesis might also be considered when the objective of the model is to imitate a reference handwriting style. Displaying pairs of original and generated images visualizes the ability of the model to disentangle the original style, map it to a latent space, achieve style diversity, and reproduce more text in the same desired style.

Style interpolation Another method of visualizing the results of a GAN for handwriting synthesis is to show examples of sampling interpolation between two different styles defined by two latent vectors, respectively. This is achieved by generating images using interpolated latent values and

Fig. 3 Examples of generating images using the interpolation of two different styles (styles A and B)

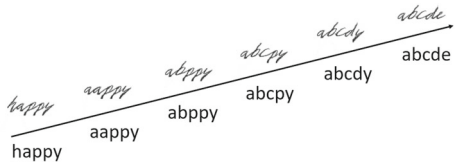

Fig. 4 Visualization of generated images of handwriting using a word ladder

showing that the synthesized handwriting gradually changes from one style to another style (Fig. 3). This evaluates the ability of the GAN to generalize, i.e., generate continuously changing, diverse styles, and cluster them in the latent space.

Word ladder The word ladder is a method to evaluate the robustness of a generative process when observing the images it generates with a fixed latent vector. Each word image, displayed on the ladder (Fig. 4), is a new word generated based on the same latent vector but a different input text. The word ladder can be used to observe qualitatively whether the handwriting style of the generated images is preserved with changing words. Such preservation indicates that the GAN architecture has indeed learned to map a latent space vector to one writing style.

Out-of-vocabulary and long text synthesis One last method of evaluating the capabilities of a GAN model is conditioning it with a relatively long text or words out of the vocabulary of the training data. Stable models should not obtain degraded results since they are supposed to be learning individual character styles and transitions between characters. Models that show degraded results in such cases are lexicon-based or depend on sequential models that cannot keep track of long sequences [2, 3, 6].

3.2.2 Assessment using handwritten text recognition (HTR)

According to Dilipkumar [20], the handwriting recognition problem cannot be considered a solved problem since the state-of-the-art (SOTA) models trained on specific datasets perform poorly on real-world samples. The suggested reason is that SOTA deep learning models are impacted by having been trained mostly on synthesized data (for which there is an endless supply) and not on sufficiently large real datasets. This leads to outstanding performance when testing on synthetic data (representing most of the training samples) but this does not guarantee the generalization of good performance on real data.

Researchers working on generating handwritten text use handwriting recognition (HTR) to judge the quality of a model for handwriting synthesis. They compare the performance of a HTR system when trained with real training samples only to that with a mix of real and synthetic training samples, where the synthetic samples are produced by the model for handwriting synthesis under investigation. Whenever a performance improvement of the HTR system occurs due to the augmentation of the training samples using synthetic data, researchers consider this as a clue of high-quality handwriting synthesis.

3.2.3 Assessment with user studies

To evaluate the quality of generated images, some researchers add human assessment and preference studies. Participants of the experiments are usually shown a mix of real and generated images, and asked to spot the generated images. A confusion matrix of the human preference indicates the population plausibility through accuracy (ACC). Other metrics such as precision (P), recall (R), false-positive rate (FPR), and false omission rate (FOR) values can be used as well. The population accuracy weighs the quality of the generative process.

3.2.4 Assessment of the generated image quality

The method of assessing the images generated by the nine reviewed models depends on the objective of the generative process. For architectures that sample a random-style vector for image generation, the focus is on image fidelity, while for architectures that imitate a reference style of an image, the goal is high similarity between the reference and the generated image.

The geometry score (GS) [44] compares the topology of the underlying real and generated manifolds and provides a way to measure potential mode collapse (the lower GS value the better). Mode collapse is the phenomenon that, after a long phase of generations, the model starts to generate new samples that are very similar to each other (or, in the extreme case, the same).

The Fréchet inception distance (FID) [45] measures visual quality and sample diversity. It gives a distance between real and generated data distributions, so the lower its value the better. Although it was not designed for handwriting image data, it can fairly serve as an indication of similarity between real and generated handwritten text. However, some researchers [4] claim it cannot assess style transfer quality since it was introduced for unconditional image generation and cannot tell how well the results match the conditions.

The multi-scale structural similarity image score (MS-SSIM) [46] is a multi-scale variant of a perceptual similarity metric. This type of metric attempts to predict human perceptual similarity judgments and discard irrelevant aspects. MS-

SSIM values range between 0.0 and 1.0; higher MS-SSIM values correspond to perceptually more similar images.
The GAN-train and GAN-test metrics [47] evaluate conditional image generation via the image recognition task (here HTR), the case for which FID is not the best metric. For GAN-train, a recognition model is trained on a training set of generated images and tested on a test set of real images. GAN-train is an indicator of the diversity of generated images. Conversely, for GAN-test, real images are used to train a model, which is then tested on generated data. GAN-test is a measure of the fidelity of generated images with respect to the original data.

Recently, Gan et al. [9] proposed the use of three additional metrics to evaluate the quality of synthesized handwritten text images, the inception score (IS) [48], which measures the realism and diversity of generated images, the kernel inception distance (KID) [49], which, similar to FID, measures the distance between distributions of the generated and real samples, and the peak signal-to-noise ratio (PSNR), which measures the reconstruction error.

In the subsections below, we review the capabilities and architecture of nine handwriting generation models, which, as mentioned above, are the only GAN-based architectures for the task of word synthesis that we could find in the literature (at the time of writing). Due to the chronological dependency of the reviewed architectures, we present them roughly in the order in which they were published, making exceptions for two works [6, 8] for ease of developing of the relevant concepts.

3.3 Alonso et al., A2iA, France, 2019

Motivation The seminal contribution of Alonso et al. [2] for the task of handwriting synthesis, i.e., augmenting the conditional GAN architecture with a recognition network R that is trained using the CTC loss function, was motivated by their goal to create legible images of words.

Method An overview of the method proposed by Alonso et al. [2] was outlined in the previous section. We here give some details on the architecture and loss functions used. In their design, the embedding network consisted of recurrent layers of bidirectional long short-term memory (Bi-LSTM) [50] to encode the input character string (word) t. The recognition network R is a gated convolutional recurrent network (CRNN), originally for scene text recognition by Shi et al. [51], consisting of an encoder of five layers, with tanh activations and convolutional gates, followed by a max pooling layer, and a decoder made up of two stacked bidirectional LSTM layers.

The generator network G uses up-sampling ResBlocks [52], conditional batch normalization (CBN) layers [53], and a self-attention layer [54]. The discriminator D consists of down-sampling ResBlocks, a self-attention layer, and a global sum pooling layer.

The adversarial loss function of the discriminator D (Eq. 1) was implemented as a hinge function $l_{\mathrm{D}} = -\mathbb{E}_{(x,t)}[\min(0, -1 + D(x))] - \mathbb{E}_{(z,t)}[\min(0, -1 - D(G(z, y)))]$. The CTC loss term was not only used to define the recognition loss l_{R} (Eq. 7), but also added to the adversarial loss of the generator:

$$l_{\mathrm{G}} = -\mathbb{E}_{(z,t)}[D(G(z, y))] + \mathbb{E}_{(z,t)}[\mathrm{CTC}(t, R(G(z, y)))], \tag{8}$$

which we simplify to

$$l_{\mathrm{G}} = l_{adv} + \lambda\ l_{\mathrm{R}}, \tag{9}$$

including the regularization factor λ.

Alonso et al. noticed that, during training, the magnitudes of the gradients of the weights in R were much larger than in D. They, therefore, proposed the use of the above regularization factor λ, for which they tested three values in an ablation study. They found that the existing larger contribution from gradients in R was valuable, as it yielded the most legible synthesized images, and therefore recommended the use of $\lambda = 1$.

Furthermore, Alonso et al. proposed to train D with one batch of real images and one batch of generated images per training step, They trained R with real data only, to prevent the model from learning how to recognize generated images of text.

Results Alonso et al. tested their model using both French and Arabic datasets, producing variable-length words, sometimes not present in the training set (see details on the datasets in Sect. 3.1). The generated images were of fixed dimensions. The generated images were used to train a handwritten text recognition (HTR) engine to observe the effect of augmenting the training dataset with synthesized samples (see Table 6). The authors praised the overall visual quality of the images generated by their model, even though they reported the generation of a few instances with "style collapse" where the characters of the generated words lose coherence. Image similarity metrics are reported in Table 8

3.4 Fogel et al., ScrabbleGAN, Amazon Rekognition, Israel, and Cornell Tech, USA, 2020

Motivation Fogel et al. [3] were motivated by the goal to generate arbitrary long words without suffering the style collapse that they noticed in the work by Alonso et al. [2]. They wanted to be able to generate different handwriting styles by changing the latent factors of the noise vector z, i.e., generate both cursive and non-cursive text, with either a bold or

thin pen stroke. They also wanted to allow for variable-size output images.

Method In designing ScrabbleGAN, Fogel et al. avoided the use of recurrent layers as an embedding network to process the input text string. Instead, their embedding network is composed of a bank of filters, as large as the alphabet size. Individual filters, corresponding to each character, are applied to the input string to generate a text map of each character. These text maps (filter outputs) are multiplied by the noise vector z, which controls the handwriting style. The resulting maps are then concatenated horizontally into a wide text embedding vector y, used to condition the generator G to generate adjacent character images. The generator G can then be looked at as a concatenation of identical class-conditional generators, where each class is a character. For an input embedding y, each of these generators produces a patch containing one handwritten character image in parallel. Each convolutional-up-sampling layer in G widens the receptive field and achieves the overlap between every two neighboring characters. The overlap allows adjacent characters to connect smoothly giving a realistic cursive word. In order to generate the same style for the entire word, the noise vector z is kept constant throughout the generation of all the characters in the input text string.

ScrabbleGAN uses the following architectures for the networks G, D, and R: The generator network G consists of three fully convolutional residual blocks, which up-sample the spatial resolution, followed by conditional instance normalization layers. Finally, a convolutional layer with a tanh activation is used to output the final image. The discriminator D consists of four residual blocks (also fully convolutional to cope with varying-width generated images), followed by a linear layer with one output. The final prediction is the average of the patch predictions, which is fed into a GAN hinge-loss [55]. ScrabbleGAN uses a similar design as Alonso et al. [2] for the recognition network R. Its convolutional recurrent neural network (CRNN) architecture has six convolutional layers and five pooling layers, all with ReLU activation and a final linear layer to output class scores compared to the ground truth using the CTC loss.

During the training of ScrabbleGAN, same gradient balancing approach as proposed by Alonso et al. (Eq. 9) is used to avoid gradient explosion. Only the recognizer network R requires labeled data for its optimization, while the discriminator D only predicts whether or not an image of a handwritten word is realistic. Therefore, unlabeled data can be used to optimize it. This allows ScrabbleGAN to be trained in a semi-supervised fashion using partially labeled data.

Results ScrabbleGAN was evaluated using the same datasets as Alonso et al. and an additional dataset (see Sect. 3.1). Qualitatively inspecting their results, Fogel et al. mentioned that their generated images contain fewer artifacts when compared to the images generated by Alonso et al.'s model [2]. They reported better FID and GS values than Alonso et al. (see Table 8). They also reported some quantitative results in the form of WER and NED of an HTR evaluation (see Table 6).

The ScrabbleGAN architecture was used by Chang et al. [12] to generate handwritten text images in other languages in a cross-lingual fashion. The authors reported that their GAN model generates handwritten images of a target language without seeing any labeled handwritten data of that language (i.e., zero-shot). Their generator was trained on English images to generate handwritten images of a variety of other languages and scripts like Vietnamese, Arabic, and Cyrillic.

3.5 Zdenek and Nakayama, JokerGAN, The University of Tokyo, Japan, 2021

Motivation Zdenek and Nakayama [4] found the solutions based on a fixed size of characters set, like that of ScrabbleGAN, not suitable to be extended to some languages like Japanese or Chinese. The reason is that the memory requirements for the bank of base filters (embedding network) grow significantly as the size of the character set increases. They wanted to generate images of handwritten text of arbitrary words and variable length, but with fewer memory requirements. They also wanted to improve the character alignment in the generated word image by adding more conditional inputs to G related to the vertical properties of characters in the target word. They named this information "text line embedding" (TLE), marking characters that rise above the main body line (i.e., ascenders like h, b, and l) and characters that drop below (i.e., descenders like g, y, and j).

Method Zdenek and Nakayama [4] were inspired by the use of text maps in ScrabbleGAN. In their design, however, the target text map is a result of the concatenation of "embedding elements." Every embedding element represents one character and is the concatenation of three pieces of information per character: (1) character embedding, (2) the latent vector z, and (3) the text line (TLE) embedding. Each embedding element is passed through a base filter (rather than a bank of filters as in ScrabbleGAN), implemented as a linear neural network layer. All outputs are horizontally tiled next to each other to create a text base map. This modification allows the JokerGAN model to operate on huge alphabetic sets like that of the Asian languages.

The JokerGAN architecture of the networks G, D, and R is similar to the Alonso et al. [2] model shown in Fig. 1 but Zdenek and Nakayama replaced the conditional batch normalization layers of G by multi-class conditional batch normalization (MCCBN) layers. These layers operate on the text embedding feature maps. With multi-class-conditional batch normalization (MCCBN), multiple classes of characters can be used per image. During the generative process, the

feature maps are divided into k identical size regions, where k is the number of characters of the target word. Different gain and bias parameters are learned to compute the values of each region of the batch-normalized feature map for each character in the sequence of the k characters.

The latent vector z, sampled from a normal distribution, is also injected into the MCCBN layer of G to generate different handwriting styles, as well as the text line conditions to prevent misalignment and distortion of the generated word images. Similar to ScrabbleGAN, JokerGAN uses a semi-supervised fashion with partially labeled data to train its networks. The training losses are the adversarial loss and the CTC loss combined (Eq. 9).

Results JokerGAN was evaluated using two of the datasets used by ScrabbleGAN in addition to a Japanese dataset. It was tested to generate out-of-vocabulary words, and for language domain transfer, that is, training the model in one language and generating word images in another. JokerGAN showed agreeable results for both tasks. The visual results also showed JokerGAN's ability to generate multiple words at a time, despite being trained on single words, by introducing a symbol (or class) for white space, at the size of one character. The symbol was concatenated to the word condition to appear as white space in the generated image.

The images generated by JokerGAN were used to augment the training of a handwritten text recognition engine. The experimental results indicated an improvement in HTR performance when trained on images generated by JokerGAN compared to HTR trained without data augmentation. Zdenek and Nakayama reported their HTR performance augmented with generated images Table 6 and mentioned that they outperformed ScrabbleGAN in the human assessment of the fidelity of the generated images with better FID, GAN-train, and GAN-test measures values (Table 7).

3.6 Liu et al., HTG-GAN, Institute of Automation, China, 2021

Motivation All three models JokerGAN, ScrabbleGAN and the original model by Alonso et al. (Fig. 1), are not able to imitate the calligraphic style of an input image. The reason is that these models are conditioned on the desired text string and a latent noise vector z, but not on writing style attributes or an input image of a handwritten word. In other words, these models are not able to reproduce a writer's style in a reference text image to generate an image with new text. The generated style is obtained from the randomly sampled latent vector z instead. This motivated the approach by Liu et al. [5], who proposed to describe a particular writer's style by a latent vector s that represents a set of content-agnostic calligraphic attributes (text skew, slant, roundness, stroke width, ligatures, etc.) and is decoupled from the latent vector y that describes the "content," i.e., the desired text string. The model

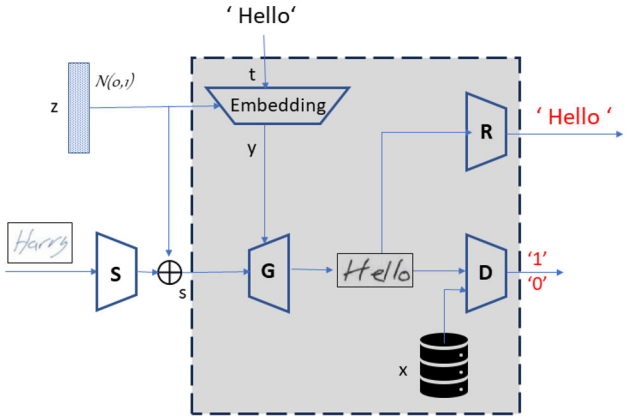

Fig. 5 HTG-GAN architecture: During training, the encoder network extracts a style vector from an image, allowing images in a similar style to be generated, but with arbitrary text. The noise vector z is usually added to the text embedding; however, during inference, a randomly sampled latent style vector from the training database of styles is used to generate the desired text

by Liu et al., called HTG-GAN, is designed to learn writers' calligraphic styles through input images of their handwriting samples, and then, during inference, mimic a selected style with only the desired text string t as an input.

Methods To compute the latent vector s from an input image of a sample of a writer's handwriting, Liu et al. added a block, the calligraphic style encoder S, to build the HTG-GAN architecture, see Fig. 5. Style encoder S consists of four residual blocks and two fully connected layers with ReLU activation and spectral normalization used in each block. The two fully connected layers are used to obtain the mean and variance for Gaussian sampling. The generator G has three residual blocks similar to those in S and uses nearest neighbor interpolation to perform up-sampling. One final convolutional layer outputs the generated image. The discriminator D consists of four residual blocks similar to S, followed by a final fully connected layer to output the binary signal "synthetic" or "real." The recognizer R uses the convolutional recurrent neural network (CRNN) architecture proposed by Alonso et al. [2].

During the training stage, in addition to the adversarial loss l_{adv} and the CTC loss l_{R} that guide G to generate realistic and legible handwriting images, HTG-GAN uses the Kullback–Leibler divergence loss l_{KL} [56] to guide G to generate diverse styles from different latent representations. Also, a reconstruction loss l_{rec} was added to the training losses to encourage G to generate visually pleasing images. The reconstruction loss evaluates the pixel-wise similarity between the generated image and the input image (L1 loss). Accordingly, the full objective function of HTG-GAN is:

$$l_{\text{S,G,D,R}} = \lambda_1 \, l_{\text{KL}} + \lambda_2 \, l_{\text{adv}} + \lambda_3 \, l_{\text{rec}} + \lambda_4 \, l_{\text{R}} \tag{10}$$

where λ_1, λ_2, λ_3 and λ_4 are balancing weights.

Results The authors compared the performance of HTG-GAN to the model by Alonso et al. and to ScrabbleGAN on the same datasets. Their results were comparable in regard to the image similarity metrics (see Table 8). It was reported that the images generated by HTG-GAN had better visual quality and fewer artifacts. Moreover, comparing results for the handwritten text recognition task, a slight improvement over ScrabbleGAN performance was reported (See Table 6).

3.7 Kang et al., GANwriting, Universitat Autonoma de Barcelona, Spain, 2020

Motivation The goal of Kang et al.'s work [6] was to create a handwriting generator, called GANwriting, that can imitate a reference handwriting style of a particular writer, provided by sample images of the writer's handwriting. The novel idea was to add a block to the model architecture, called the writer classifier W that can penalize the generated image if it does not hold the desired style and can guarantee that the provided calligraphic attributes characterizing a particular handwriting style were properly transferred to the generated word instances. Kang et al. also introduced the calligraphic style encoder S to the architecture by Alonso et al. [2] (Fig. 1), which was also used by Liu et al. [5] in HTG-GAN, as described above.

Method The GANwriting architecture includes a text embedding network and the networks G, D, and R, as suggested by Alonso et al. (Fig. 1), but Kang et al. made some changes: The embedding network consists of three fully connected layers with ReLU activation functions and batch normalization, and its output y includes two types of encodings: (1) low-level encodings of different characters that form a word and their spatial position within the string and (2) global string encodings aiming for consistency of the whole word. These two feature encodings are concatenated and fed to the generator G, together with the style features s, as a single feature map $F = [F_s||y]$.

The style features s are computed by the encoder S, which uses a VGG-19 backbone network with batch normalization (VGG-19-BN) [57], and additive noise z. The input to the generator G is thus $F = [F_s||y] + z$.

The generator G consists of two residual blocks, using AdaIN [58] as the normalization layer, four convolutional modules with nearest neighbor up-sampling, and a final tanh activation layer to generate the output image. The discriminator D starts with a convolutional layer, followed by six residual blocks with LeakyReLU activations and average pooling, and a final binary classification layer.

Quite different from the Alonso et al. base model, the recognizer R of the GANwriting architecture consists of an encoder and a decoder, coupled with an attention mechanism. A VGG-19-BN followed by a two-layered bidirectional gated recurrent unit (B-GRU) is used as the encoder network, and the decoder network is a one-directional RNN that outputs character-by-character predictions at each time step. The attention mechanism dynamically aligns the context features from each time step of the decoder with high-level features from the encoder, hopefully corresponding to the next character to decode.

The writer classifier W of GANwriting follows the same architecture as the discriminator D, but with a final classification by a multilayer perceptron with a number of nodes equal to the number of writers $|\mathcal{W}|$ in the training dataset.

The optimization process of GANwriting is based on three loss functions: the discrimination loss l_{D}, which is implemented as a binary cross-entropy loss (Eq. 1), the writer classifier loss l_{W}, which is implemented as a multi-class cross-entropy loss with the number of classes being the number of writers $|\mathcal{W}|$, and the recognizer loss l_{R} as the Kullback–Leibler divergence loss [56]. The whole GANwriting architecture was trained end to end with the combination of the three proposed loss functions:

$$l(H; D; W; R) = l_{\mathrm{D}}(H; D) + l_{\mathrm{W}}(\mathrm{H}; \mathrm{W}) + l_{\mathrm{R}}(\mathrm{H}; \mathrm{R}), \quad (11)$$

where H stands for the combination of the G, S, and embedding networks. Kang et al. [6] did not mention any gradient balancing attempts during training.

Results Kang et al. did not provide comparisons to previous work on handwriting generation. However, later works [9, 10] have run experiments using the GANwriting model to obtain results for the sake of comparison (see Tables 6 and 8). Alternatively, Kang et al. reported that their results outperform FUNIT [59], an image-to-image translation architecture for natural scene text. Furthermore, human examiners reportedly found various synthesis results produced by GANwriting to be satisfactory. By design, GANwriting requires multiple reference images per writer to extract a reliable style feature for each synthetic sample during training (i.e., a few-shot setup). Thus, a slight degradation has been found to occur when either the input text is an out-of-vocabulary (OOV) word or has a style never seen during training. Additionally, GANwriting cannot generate long handwritten words (longer than ten letters) and can only imitate a given input handwriting style, i.e., it cannot generate random-style text.

Kang et al. extended their work [11] to generate handwritten text lines by a periodic padding module inside the S block. This method was able to generate handwriting samples of any length irrespective of the length of the style input by replacing the Seq2Seq-based recognizer with a Transformer-based recognizer. The authors did not compare the results of their original and extended models.

The GANwriting architecture was also extended by Wang et al. [13] to generate multi-scale and more complex writing

styles by introducing attentional feature fusion (AFF) to the GANwriting model. The style VGG-19-based encoder was modified to obtain multi-scale features including global and local features. The resulting model was named AFFGanWriting and reportedly generates images of better visual quality than those generated by GANwriting or a previous model by Wang et al. [16] that was based on a Transformer.

3.8 Gan and Wang, HiGAN, The University of the Chinese Academy of Sciences, China, 2021

Motivation The goal of Gan and Wang [7] was to design a model that can generate diverse handwriting conditioned on arbitrary-length texts and disentangled styles, extending the work of Kang et al. [6], GANwriting, so that longer texts and arbitrary styles can be produced. Gan and Wang proposed the Handwriting imitation GAN (HiGAN) model, which offers two options for the latent representation of the style s: (1) a randomly sampled style from a prior distribution, or (2) a style disentangled from a reference image through the pre-trained style encoder S.

Method HiGAN uses the same model blocks G, D, S, W, and R as GANwriting (Fig. 6), and details of the internal design of the blocks can be found in the implementation code that the authors shared.

HiGAN expands on the loss functions used for training. Two types of adversarial losses are used that guide the training of the generator G: (1) For an arbitrary text string embedding y and a style feature s, randomly sampled from a prior normal distribution $N(0; 1)$, the generator G synthesizes image $G(y, s)$ using the loss function

$$l_{\text{adv1}} = \mathbb{E}_X[\log(D(X))] + \mathbb{E}_{y,s}[\log(1 - D(G(y, s)))]. \quad (12)$$

(2) For a real input image X, the generator synthesizes a realistic image conditioned on the disentangled style $S(X)$, using the loss function:

$$l_{\text{adv2}} = \mathbb{E}_X[\log(D(X))] + \mathbb{E}_{y,X}[\log(1 - D(G(y, S(X))))]. \quad (13)$$

Combining the two losses, the overall adversarial loss during training is

$$l_{\text{adv}} = l_{\text{adv1}} + l_{\text{adv2}}. \quad (14)$$

The full objective of HiGAN can be summarized as follows: (1) When maximizing the adversarial loss l_{adv}, the discriminator D, recognizer R, and writer identifier W are optimized, and (2) when minimizing the adversarial loss, the generator G and style encoder S are jointly optimized:

$$l_{\text{D}} = -l_{\text{adv}}, \quad (15)$$

$$l_{\text{G,S}} = l_{\text{adv}} + \lambda_1 l_{\text{R}} + \lambda_2 l_{\text{W}} + \lambda_3 l_{\text{S}} + \lambda_4 l_{\text{KL}}, \quad (16)$$

where λ_1, λ_2, λ_3, and λ_4 are balancing weights. Here, loss terms l_{w} and l_{KL} are computed by the writer classifier W, which offers two options: Styles from known writers, defined with writer IDs, e.g., w_1, w_2, etc., can be disentangled or trained using data from unseen writers, who do not have a corresponding identifier. Consequently, two versions of losses are available to guide G to reproduce the input style. Loss l_{W} is implemented as a cross-entropy function and l_{KL} is the Kullback–Leibler divergence loss. The recognizer R is first optimized by minimizing the CTC loss for each (image X, ground-truth text t) pair in the training set:

$$\text{CTC loss} = \mathbb{E}_{X,t}[-t \log(R(X))]. \quad (17)$$

Then, the parameters of R are kept fixed when minimizing the adversarial loss. The trained R can guide G to synthesize a legible handwriting image $G(y; s)$ through the loss term l_{R} in Eq. 16:

$$l_{\text{R}} = \mathbb{E}_{y,s}[-y \log(R(G(y, S(X))))]. \quad (18)$$

Similarly, for the style encoder S, first, the latent style reconstruction loss is employed. Then, the model is forced to reconstruct the style s of any synthetic image $G(y; s)$ through the loss term l_{S} in Eq. 16:

$$l_{\text{S}} = \mathbb{E}_{y,s}[\|s - S(G(y, s))\|_1]. \quad (19)$$

Results The performance of the HiGAN architecture was compared to the performance of GANwriting and ScrabbleGAN on the same datasets. HiGAN showed better performance regarding the visual quality of the generated images, the quantitative evaluation of image similarities, and the handwritten text recognition error rates (see Tables 6 and 8). The experiments showed that HiGAN could synthesize even long texts of similar styles. However, spaces between words were omitted, making the entire sentence a single very long word. It should also be noted that HiGAN sometimes produced a low visual quality of synthetic images due to blurred and distorted characters.

Inspired by the HiGAN architecture, Zdenek and Nakamaya [14] proposed JokerGAN++ to support the imitation of style from reference images, a feature that is not provided by JokerGAN. They introduced a style encoder block to their architecture that is based on a Vision Transformer (ViT) [60]. The authors report that JokerGAN++ produces better images than ScrabbleGAN, JokerGAN, and HiGAN with regard to qualitative and quantitative HTR assessment.

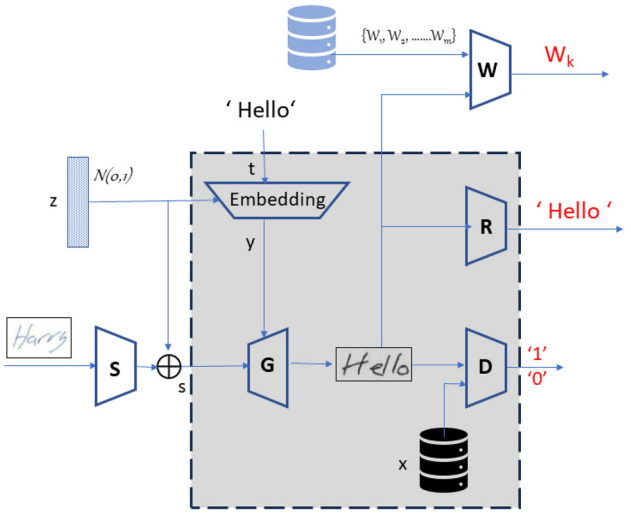

Fig. 6 The GANwriting architecture: Novel modifications are the additions of a writer classifier network W and a style encoder network S. A writer's style is provided to W by $m = 15$ image samples of the writer's handwriting for training (few-shot training). After training, S can extract a style vector from an image, allowing images in a similar style to be generated, but with arbitrary text. Additive noise z is added to the text embedding as usual, and some noise is added to the disentangled style vector s. The design shown here was also used for HiGAN architecture [7]

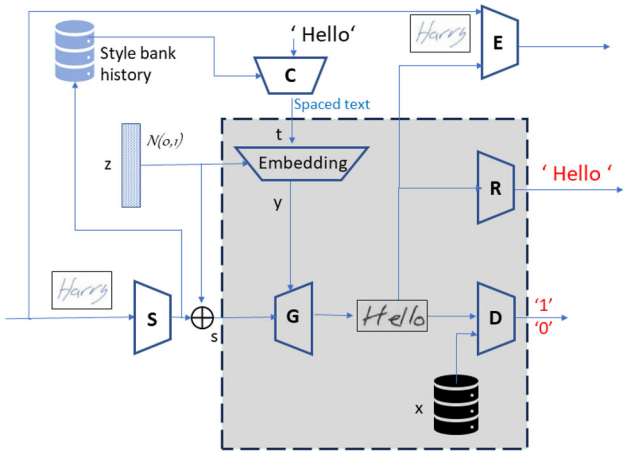

Fig. 7 The architecture by Davis et al.: The style encoder S disentangles a style vector s from the reference handwriting image and uses this vector to (1) help the spacing network C estimate the proper character sizes and inter-word spaces and (2) update the style bank to enhance the future estimates of the spacing network C. The text embedding makes use of the spacing network information and the latent noise to guide G to convey the desired text string in diverse styles. The networks D and R function as usual. Network E computes the perceptual reconstruction loss between the styles in both the reference and the generated image to urge G to transfer the same input style

3.9 Davis et al., Brigham Young University and Adobe Research, USA, 2020

Motivation Davis et al. [8] wanted to generate images with a full line of text with spacing between words and the possibility to reproduce a writer's style for a given input text and new arbitrary text. They modified the architecture proposed by Alonso et al. in Fig. 1 such that their GAN was conditioned on both an arbitrary text string, and a latent style vector extracted from a reference image of real handwriting. They combined variational auto-encoders with GANs to generate variable-size images of handwritten lines. The generated image size is predicted using their deep architecture that estimates the characters' sizes and the inter-word spacing. Those estimates are based on the input writing style, disentangled from the reference image, and the target/conditioned text.

Method To accomplish their goals, Davis et al. introduced two remarkable functional networks in their architecture, see Fig. 7. The first network is the spacing network C that predicts the horizontal text spacing from the extracted style vector. The second network is a pre-trained encoder E that computes a perceptual loss [61]. Perceptual losses encourage natural and pleasing generation results. These losses measure image similarities more robustly than per-pixel losses. The perceptual loss forces G to generate a handwriting style that mimics the input image style. In other words, while G learns to reconstruct images from style and content, the encoder E only needs to extract the style vector.

The architecture proposed by Davis et al. can be trained in two modes: GAN training and auto-encoder training. In the GAN-only training, the adversarial losses, including CTC from network R, are computed and used to update G and D. In the auto-encoder training, the reconstruction losses (pixel and perceptual) are computed to update G and S. The mean square error (MSE) loss is used to train the network C. The network E is trained both as an auto-encoder with a decoder and $L1$ reconstruction loss when the objective is to copy the reference style on a new text, and as a handwriting recognition network with CTC loss when the objective is to reproduce both the reference style and text.

The architecture functions as follows: (1) A generator network G produces images from spaced text, a style vector, and noise, (2) a style extractor network S computes a style vector from an image and the recognition predictions, (3) a spacing network C predicts the horizontal text spacing based on the style vector, (4) a patch-based convolutional discriminator D to detect real versus synthesized images, (5) a pre-trained handwriting recognition network R to encourage image legibility and correct content, and (6) a pre-trained encoder E to compute a perceptual loss.
Davis et al. explained the details of the internal design of the six networks in their supplementary material [8].

They also modified the gradient balancing technique, previously introduced by Alonso et al. [2]. In the previous works, the balancing terms were all learned during training and updated on each epoch. To reduce memory requirements, they forced some training steps to only store the gradients

(for later balancing), and other steps to update the parameter values. The weights in the balancing formula were chosen heuristically so as to emphasize the parts they discovered the model has struggled with.

Results Davis et al. provided many ablation study details and visual representations of the results of their experimental work. They studied the effect of the different losses they used on the output image legibility and quality. They showed evidence that the network S extracted styles accurately at the author level and clustered style vectors for the same writer without intentional training. Commenting on their reconstruction results, the authors noted that their model is able to mimic aspects of a writer's global style, but failed to copy character shape styles. Nonetheless, they describe the generated images to be convincing, based on a human assessment experiment conducted via Amazon Mechanical Turk. The participants were fooled by the synthesized images, voting them to be real most of the time.

The authors used the same datasets as were used for the model by Alonso et al. [2] and for ScrabbleGAN [3]. They described their results to be similar in quality to those by ScrabbleGAN based on two image similarity metrics FID and GS (see Table 8).

3.10 Gan et al., HiGAN+, University of Posts & Telecommunications and University of the Chinese Academy of Sciences, China, 2022

Motivation According to Gan et al. [9], the architecture proposed by Davis et al., which learns to extract styles from images based on the pixel-to-pixel reconstruction loss, cannot correctly imitate styles of reference samples in most cases. They attributed the reasons to the spatial misalignment of image pairs, and the texture existence limiting the efficiency of pixel-based methods. To enhance the visual quality of the generated images and also achieve a more accurate handwriting style transfer, Gan et al. [9] proposed a modified version called HiGAN+ of their previous work HiGAN [7]. With HiGAN+, they aimed to reproduce the same style as a reference image on a new input text string.

To address the blurriness of characters, which was degrading the generated image quality, and to better transfer the reference style, Gan et al. were motivated to add terms to the loss function used by HiGAN. They also wanted a compacter model and thus redesigned the writer identifier network W such that the style encoding was conducted in the earlier layers.

Method Gan et al. made use of the comment by Davis et al. about the problem of generating character styles versus a global word style. The new design of the generator converts the text into individual character embeddings, rather than an entire text embedding, and then concatenates those local characters patches together into words. With convolutions, the overlaps and transitions among characters are learned. This is similar to the feature map creation of ScrabbleGAN.

Gan et al. added a patch discriminator network to decide whether a given patch was cropped from real or synthetic images. That was intended to improve the local texture details of synthetic images, since, instead of grading the whole image, it verified the patch fidelity. Details of the internal design of the blocks of HiGAN+ were not explained in the paper and might be found in the implementation code that the authors have shared.

Gan et al. modified the objective function they developed for HiGAN, Eq. 16, by adding additional loss terms to guide the generator:

$$l_{\mathrm{G,S}} = l_{\mathrm{adv}} + \lambda_1 l_{\mathrm{patch}} + \lambda_2 l_{\mathrm{R}} + \lambda_3 l_{\mathrm{W}} + \lambda_4 l_{\mathrm{ctx}} + \lambda_5 l_{\mathrm{S}} + \lambda_6 l_{\mathrm{recn}} + \lambda_7 l_{\mathrm{KL}}, \tag{20}$$

where λ_1, λ_2,…, λ_7 are balancing weights. Some of these weights were empirically set, and others were dynamically adjusted during training with the gradient balancing strategy. Loss terms l_{adv}, l_{R}, l_{W}, l_{S}, and l_{KL} are the same as in HiGAN. The local patch loss l_{patch} penalizes the local structures to help achieve good local consistency, especially when the input text is long.

The contextual loss l_{ctx} measures the similarity of two handwriting images, requiring no spatial alignment and allowing slight deformations as it focuses on the high-level style features. The content reconstruction loss l_{recn} improves the content and style consistency since it regularizes the generative model to achieve a more robust handwriting style transfer.

The training of HiGAN+ was done in three stages, (1) pre-training the writer identifier W and text recognizer R, (2) reusing writer identifier W as style encoder S, and (3) GAN optimization with gradient balancing.

Results Gan et al. tested HiGAN+ using several qualitative and quantitative metrics. In particular, they used image similarity metrics to evaluate the visual quality of synthesized images and HTR to check on readability of the results. They introduced a writer identification error metric to evaluate handwriting style transferability. Gan et al. compared HiGAN+ to the related works discussed in this survey [3, 6–8], and the transformer-based architecture [15]. The comparisons favor HiGAN+ over the other structures visually and quantitatively (see Table 8) even with one-shot handwriting style transfer. The assessment study showed that humans were fooled by the images generated by HiGAN+ and preferred its imitated styles over the images generated by other architectures (see Table 7). However, the error analysis showed that HiGAN+ failed to generate plausible images for scribbled handwriting and for punctuation marks and digits.

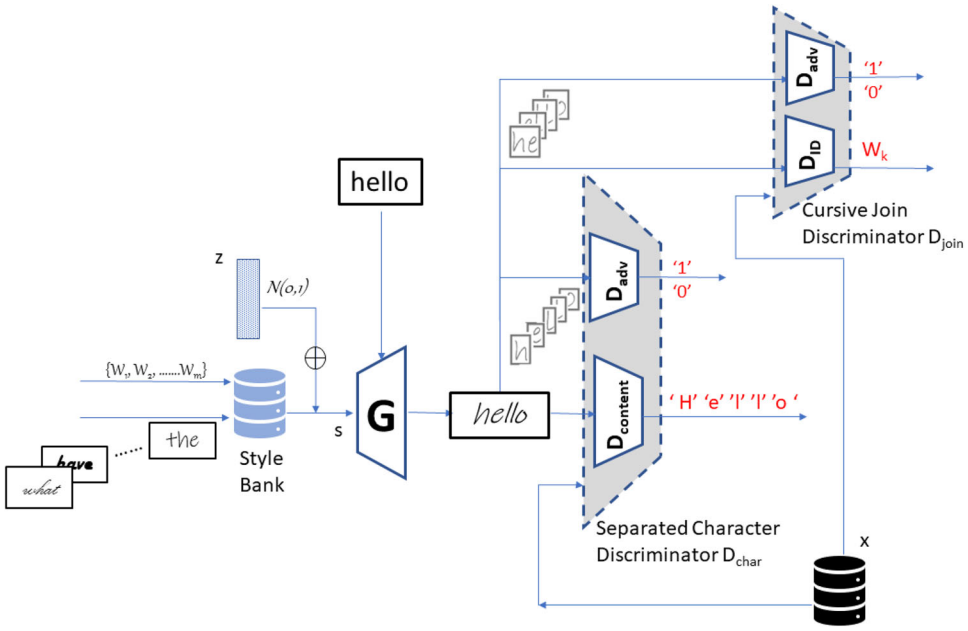

Fig. 8 SLOGAN architecture: The style encoder S is replaced by a lookup table of handwriting samples associated with their writer ID. The input is an image of machine-printed text rather than an embedding of a text string. In the inference stage, the writer ID is input to the bank to obtain its corresponding style vector. Noise is added to parameterize this style (i.e., create a new unknown style) if needed. The discriminator D_{char} checks for the character shape legibility, while discriminator D_{join} checks for the character transition legibility

3.11 Luo et al., SLOGAN, South China University of Technology, China, 2022

Motivation Luo et al. [10] provided some interesting reasoning about the previous works on handwriting generation, suggesting that the latent vectors are not sufficient for representing variance in handwriting styles and thus limit the ability of these vectors to represent style diversity. They declared that there is an imbalance in the IAM dataset, used by the previously discussed models, regarding the frequency of contributions of individual writers. They suggested that gaps in the style space cannot be filled by the previous solutions since no new styles can be invented, and collecting more data with new styles is infeasible. In their architecture, called SLOGAN, Luo et al. proposed a solution to this problem by using a style bank to store vectors of parameterized handwriting styles. The idea is for styles, identified by IDs, to be acquired by the generator to guide the synthetic images toward specific styles. New styles can then be synthesized by controlling the latent style parameters.

Luo et al. also noted that the previously discussed solutions are not sufficiently flexible to embed text contents, especially out-of-vocabulary and long texts. The reason, in their opinion, is the failure of previous architectures to accurately detect transition locations of adjacent characters or learn their shapes. To solve this, they suggested that the conditioned text should be fed to the GAN as a machine-printed style image. In such a way, various contents could be generated by changing the string characters and realigning their positions on the input image.

Method The SLOGAN architecture, shown in Fig. 8, is significantly different than the previously discussed works. Luo et al. gave up the recognition network R, first introduced by Alonso et al. [2]. Nonetheless, SLOGAN is able to generate legible text images. Luo et al. dealt with the problem of legibility as an image style transfer problem as in CycleGAN. The input is an image of printed text rather than a text string (i.e., conditioned text), so SLOGAN also does not include an embedding network. SLOGAN consists of a style bank, a generator, and two discriminators, each with dual heads.

The style bank is a simple lookup table that stores m handwriting styles as latent vectors, each having a writer ID. The style bank is randomly initialized and jointly updated with the generator under the supervision of writer IDs. The generator G is an encoder-decoder architecture (i.e., identical mapping of both the same handwriting style and content). It takes a white-background machine-printed-style image as input and generates a version of that image with the printed text converted to handwriting.

The separated character discriminator D_{char} supervises the generator at the character level. It comprises an attention mechanism to overcome the need for character-level annotation and localizes characters using a text string. The discriminator D_{char} has two heads, namely $D_{\text{char,adv}}$ and $D_{\text{char,context}}$. After localizing characters in the input image, adversarial training and content (character class) training for every character follows. The cursive join discriminator D_{join} is a global discriminator that models the relationship between adjacent characters. It works on patches segmented from the feature map with overlapping receptive fields to focus on the regions between adjacent characters. Discriminator D_{join} also has two heads, namely $D_{\text{join,adv}}$ and $D_{\text{join,ID}}$. They undergo adversarial training and handwriting style super-

vision (i.e., writer style identification) on the segmented patches.

The designers of SLOGAN gave up network R but did not give up the need for text recognition loss to train G. In the previously reviewed models, R was a separate network to recognize the text in the generated images. In SLOGAN, one of the two discriminators D_{char}, performs the recognition internally on the character level using its $D_{\text{char,context}}$ head, so the recognition loss is implicitly added to the adversarial loss for training networks G and D.

The generator and discriminators are updated alternatively during training. To parameterize handwriting styles the style bank is updated jointly with the generator. At the inference stage, the latent style vector z is parameterized by individually manipulating each element to take value within the min-max range for any of the learned n parameters per style. The input printed image, i.e., conditioned text, can be manipulated to achieve different alignment effects such as curved text or text of arbitrary length.

Results SLOGAN was evaluated for the visual quality of the generated image, and the diversity in both style and content (Table 8), HTR evaluation (Table 6), and human assessment were used as well (Table 7). Volunteers were confused to tell the real from the generated images and voted for subtle imitation of input styles. Luo et al. compared their results to the here-discussed GAN-based works ScrabbleGAN [3], Alonso et al. [2], and GANwriting [6], as well as transformer-based works [15] and sequential model-based works as well. The quantitative evaluation indicates that SLOGAN outperforms them all.

Luo et al. did not provide an error analysis of SLOGAN, but one thing to note about their work is that SLOGAN can successfully generate new styles to fill the gap inside the style latent space. However, this will always be limited to the space defined by the training population only.

3.12 Comparison of model capabilities and architectures

Up to the time of writing, the nine reviewed handwriting generation systems were all the systems based on GANs architectures that we could find in the literature. In this section, we summarize their capabilities and architecture designs. As can be seen in Table 3, most works employ generator, discriminator, recognition, and embedding networks, trained with adversarial and CTC loss functions, and can handle handwritten text images, conditioned text, and latent noise. Table 3 also visualizes less common architecture components, loss functions, and input information, such as the use of writer identification networks, style banks, cross-entropy and contextual loss functions, and text line and spacing information as input.

A comparison of the reviewed systems for offline handwriting generation based on their capabilities and their provided features is given in Table 4. Eight of nine models can generate images by randomly sampling styles from a prior distribution (random-style generation) and generate words outside the lexicon or the corpus of words used to train the GAN architecture (unconstrained and out-of-vocabulary text generation). The generated images from seven of the models may contain very long words, multiple-spaced words, or even an entire line of text (arbitrary-length words). Six models ensure that the generated image width varies with the number of characters in the word to avoid distortion (variable size output image), and five models can imitate the handwriting styles of reference images (reproducing input style).

Under the row header "Code Availability," Table 4 lists for which works we were able to find implementation code that is shared publicly with the community on GitHub at the time of writing of this review paper. Unfortunately, only five of the nine works made their code available to the research community. We hope that more code will become available in the future, as it enables reproducibility of the results, comparisons between models, and furthers future research.

A comparison of the reviewed systems based on the quantitative methods used to report results is given in Table 5. Seven of nine models used HTR to evaluate the quality of the generated images. Any performance improvement in the recognition results was deemed to be due to the augmentation of the training samples using synthetic data, indicating high-quality handwriting synthesis.

The HTR system used by researchers developing GAN-based models for handwriting synthesis is typically the recognizer network R. ScrabbleGAN, JokerGAN, and HTG-GAN, for example, use the same architecture for R as suggested by Alonso et al. [2]. The other works reviewed here proposed different architectures for R. The HTR performance is based on two main metrics: the word error rate (WER), which indicates the percentage of mistakenly recognized words in the test set, and the normalized edit distance (NED), which is the edit distance between the predicted word and the ground-truth (GT) word normalized by the length of the GT word (see Table 6). The lower the values of WER and NED, the better is the recognition result.

For the IAM dataset, the performance of ScrabbleGAN, JokeGAN, and HTG-GAN is relatively close. SLOGAN outperforms them all. However, for the RIMES dataset, the performance of ScrabbleGAN, SLOGAN, and Alonso et al.'s model is almost the same. HTG-GAN has a slight advantage over them. For the CVL dataset, later models could not outperform the reference results by ScrabbleGAN.

From Table 5, we note that for only five of nine models user studies were reported to assess the quality of the generated images. Some studies observed the users' preferences in selecting the most visually convincing generated images.

Table 3 Comparison between GAN-based architectures designs, inputs, and training losses

Feature type	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
Architecture components of the GAN:									
Generator network G	✓	✓	✓	✓	✓	✓	✓	✓	✓
Discriminator network D	✓	✓	✓	✓	✓	✓	✓	✓	✓
Recognition network R	✓	✓	✓	✓	✓	✓	✓	✓	x
Embedding network	✓	✓	✓	✓	✓	✓	✓	✓	x
Style encoder network S	x	x	x	✓	✓	✓	✓	✓	x
Writer identification network W	x	x	x	x	✓	✓	x	✓	x
Spacing network C	x	x	x	x	x	x	✓	x	x
Pretrained encoder network E	x	x	x	x	x	x	✓	x	x
Style bank	x	x	x	x	x	x	✓	x	✓
Input data of the GAN:									
Conditioned text	✓	✓	✓	✓	✓	✓	✓	✓	✓
Latent noise	✓	✓	✓	✓	✓	✓	✓	✓	✓
Handwritten text image	x	x	x	✓	✓	✓	✓	✓	✓
Latent style	x	x	x	x	✓	✓	✓	✓	✓
Printed text image, writer ID	x	x	x	x	x	x	x	x	✓
Text line information	x	x	✓	x	x	x	x	x	x
Text spacing information	x	x	x	x	x	x	✓	x	x
Loss functions of the GAN:									
Adversarial loss	✓	✓	✓	✓	✓	✓	✓	✓	✓
CTC loss	✓	✓	✓	✓	x	✓	✓	✓	x
Kullback-Leibler loss	x	x	x	✓	✓	✓	x	✓	x
Cross-entropy loss	x	x	x	x	✓	✓	x	✓	x
Reconstruction losses	x	x	x	✓	x	✓	✓	✓	x
Perceptual losses	x	x	x	x	x	x	✓	x	x
Mean-square error loss	x	x	x	x	x	x	✓	x	x
Contextual loss	x	x	x	x	x	x	x	✓	x

The GAN-based models are: Alonso et al. [2], ScrabbleGAN [3], JokerGAN [4], HTG-GAN [5], GANwriting [6], HiGAN [7], Davis et al. [8], HiGAN+ [9], and SLOGAN [10]

Table 4 Features of GAN-based architectures for handwriting generation

Feature type	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
Generation of random style	✓	✓	✓	✓	×	✓	✓	✓	✓
Reproduction of input style	×	×	×	×	✓	✓	✓	✓	✓
Variable-size output image	×	✓	✓	×	×	✓	✓	✓	✓
Arbitrary-length words	×	✓	✓	✓	×	✓	✓	✓	✓
Unconstrained and OOV text generation	✓	×	✓	✓	✓	✓	✓	✓	✓
Code availability	×	✓	×	×	✓	✓	✓	✓	×

The GAN-based models are: Alonso et al. [2], ScrabbleGAN [3], JokerGAN [4], HTG-GAN [5], GANwriting [6], HiGAN [7], Davis et al. [8], HiGAN+ [9], and SLOGAN [10]

Table 5 Assessment strategies used for the reviewed models

Feature type	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
Evaluation using HTR performance	✓	✓	✓	✓	×	✓	×	✓	✓
Evaluation using human assessment	×	×	✓	×	✓	×	✓	✓	✓
Evaluation using image similarity metrics									
→ FID	✓	✓	✓	✓	✓	✓	✓	✓	✓
→ GS	✓	✓	×	✓	×	×	✓	×	✓
→ MS-SSIM	×	×	×	✓	×	×	×	✓	×
→ IS, KID, PSNR	×	×	×	×	×	××	×	✓	×
→ GAN-train, GAN-test	×	×	✓	×	×	×	×	×	×

The GAN-based models are: Alonso et al. [2], ScrabbleGAN [3], JokerGAN [4], HTG-GAN [5], GANwriting [6], HiGAN [7], Davis et al. [8], HiGAN+ [9], and SLOGAN [10]

Table 6 Model performance on three commonly used datasets, measured with the handwritten text recognition (HTR) metrics

Dataset: IAM									
Metric	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
Word error rate (%)	–	23.98	36.3	20.5	17.26*	–	–	1.86**	**14.97**
Normalized edit distance	–	13.57	13.04	**7.35**	–	–	–	–	–
Dataset: RIMES									
Metric	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
Word error rate (%)	11.9	11.68	–	**10.15**	–	–	–	–	11.5
Normalized edit distance	4.03	3.74	–	**2.7**	–	–	–	–	–
Dataset: CVL									
Metric	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
Word error rate (%)	–	**22.9**	62.33	–	–	28.91	–	–	34.98***
Normalized edit distance	–	**15.62**	28.42	–	–	–	–	–	–

Bold values indicate the best value

Only the self-reported results of the authors' best performing model that uses both real and synthesized data are listed. The GAN-based models are: Alonso et al. [2], ScrabbleGAN [3], JokerGAN [4], HTG-GAN [5], GANwriting [6], HiGAN [7], Davis et al. [8], HiGAN+ [9], and SLOGAN [10]. * The result for GANwriting, (17.26), was not reported by Kang et al. [6], but by Luo et al. [10]. ** Gan et al. [9] may have used a different training setting that yielded a low WER for HiGAN+ on IAM. They also reported a different WER of 25.24% for ScrabbleGAN [3]. *** Instead of training and testing on CVL, as was done by others [3, 4, 7], Luo et al. [10] trained on IAM plus synthetic data and tested on CVL only. This explains a somewhat higher word error rate

Table 7 Model performance according to human evaluation and user preference studies

Metric	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
TP (%)	–	–	–	–	27.01	–	34.2	31.72	31.5
TN (%)	–	–	–	–	22.3	–	18	18.32	19.45
FP (%)	–	–	–	–	27.69	–	31.9	18.28	28.55
FN (%)	–	–	–	–	22.99	–	15.8	31.68	20.5
Precision (%)	–	–	–	–	49.4	–	–	–	52.46
Recall (%)	–	–	–	–	54.1	–	–	–	60.58
FPR (%)	–	–	–	–	55.4	–	–	–	59.48
FOR (%)	–	–	–	–	50.8	–	–	–	51.31
Accuracy (%)	–	–	–	–	49.3	–	52.2	**50.04**	**50.95**
*User preference rate for realistic look (%)	–	(26.5)	**43.9**	–	–	–	–	–	–
**User preference rate of visual quality (%)	–	(9.18)	–	–	(5.42)	(21.02)	(10.27)	**37.25**	–

Bold values indicate the best value

The GAN-based models are: Alonso et al. [2], ScrabbleGAN [3], JokerGAN [4], HTG-GAN [5], GANwriting [6], HiGAN [7], Davis et al. [8], HiGAN+ [9], and SLOGAN [10]. *In the experiment conducted by Zdenek and Nakayama [4], the user picks 1-out-of-3 generated images as the most realistic. The numbers in the table show the percentage of people who find the generated images more realistic. The result for ScrabbleGAN (26.5) was not reported by Fogel et al. [3], but by Zdenek and Nakayama [4]. **In the experiment conducted by Gan et al. [9], the user picks 1-out-of-5 generated images from different GAN models as the most visually convincing. The numbers in the table show the percentage of preferred images picked by the users. The results in parentheses (.) were not originally reported by the respective model developers but by Gan et al. for the purpose of comparison

The reported results show the percentage of preferred images (like the study led by Gan et al. to compare HiGAN+ to five previous works). In some cases, the reported results show the percentage of users voting for the images generated by some model (like the study led by Zdenek and Nakayama to compare the quality of images generated by JokerGAN vs. ScrabbleGAN). The higher percentages indicate a stronger preference.

Other studies were concerned about the rates of the user classification of images as real or fake images, computing metrics such as accuracy (ACC), precision (P), recall (R), false-positive rate (FPR), and false omission rate (FOR) and constructing a confusion matrix. The classification accuracies closer to 50% suggest random classification. In such cases, human experts cannot tell which images are fake. The reported results are shown in Table 7. In that context, we note

Table 8 Model performance according to image similarity metrics

Dataset: IAM + RIMES									
Metric	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
FID	23.94	23.78	–	22.85	–	–	23.72	–	**12.06**
GS	8.58×10^{-4}	7.6×10^{-4}	–	7.41×10^{-4}	–	–	7.19×10^{-1}	–	$\mathbf{5.59} \times 10^{-4}$
Dataset: IAM only									
Metric	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
FID	–	(14.31*)	9.18	12.18	120.07	18.31**	20.65	**5.95**	–
GS	–	–	–	$\mathbf{2.23} \times 10^{-3}$	–	–	4.88×10^{-2}	–	–
GAN-train	–	(51.84)	**49.14**	–	–	–	–	–	–
GAN-test	–	(28.41)	**10.9**	–	–	–	–	–	–
IS	–	(1.33)	–	–	(1.33)	(1.33)	(1.24)	**1.41**	–
KID	–	(2.95)	–	–	(1.39)	(1.67)	(3.13)	**0.37**	–
PSNR	–	(11.26)	–	–	(10.8)	(11.76)	(12.03)	**12.34**	–
MSSIM	–	(0.19)	–	–	0.20	(0.25)	(0.18)	**0.33**	–

Bold values indicate the best value

The GAN-based models are: Alonso et al. [2], ScrabbleGAN [3], JokerGAN [4], HTG-GAN [5], GANwriting [6], HiGAN [7], Davis et al. [8], HiGAN+ [9], and SLOGAN [10]. The results in parentheses (.) were not originally reported by the respective model developers but were presented later by developers of competing architectures for the purpose of comparison. *This result may not be accurate, as it was reported with different values in different works. **Gan et al. [7] reported two FID results for latent-vector-guided and reference-guided synthesis by HIGAN, respectively. Following Gan et al. [9], we here report the average value

that generated images by both SLOGAN and HiGAN+ are the most perplexing to human experts.

Table 5 also shows that for all nine models image similarity measurements were used to assess the quality of the generated images, although they vary in the metrics used and the dataset they were generate from (see Table 8).

The geometry score (GS) measures the potential mode collapse after a long phase of generations. The lower GS value is the better. The Fréchet inception distance (FID) measures the distance between real and generated data distributions, so the lower its value is the better. The multi-scale structural similarity image score (MS-SSIM) predicts human perceptual similarity judgments with values ranging between 0.0 and 1.0. Higher MS-SSIM values correspond to perceptually more similar images. The GAN-train and GAN-test metrics evaluate conditional image generation via the image recognition task (here HTR). GAN-train is an indicator of the diversity of generated images. Conversely, GAN-test measures the fidelity of generated images with respect to the original data. The word error rate (WER) is used as the measurement of performance in both methods. The lower the values are the better.

The inception score (IS) measures the diversity of generated images. The higher IS value is the better. The kernel inception distance (KID) measures the distance between distributions of the generated and real samples. The lower KID value is the better. The peak signal-to-noise ratio (PSNR) measures the reconstruction error. The higher PSNR value is the better.

For models trained with a combination of samples from IAM and RIMES datasets, we note that FID and GS values are very similar except for the SLOGAN model which has remarkable improvement over them.

For models trained with the IAM dataset only, HiGAN+ has the best performance regarding all metrics except for GS where HTG-GAN is better, and for GAN-train/test metrics where JokeGAN has the best performance.

4 GANs versus other generative models

One of the earliest categories of models used for image generation is the auto-encoder (AE). The AE paradigm takes the raw input image and performs data encoding by learning a mapping of the input image x to a low-dimensional latent space z through a series of CNN layers (encoder). Vector z can summarize (or compress) the most important features of the high-dimensional image x. The decoder (usually some de-convolutional layers) can then use z to reconstruct an image very similar to the original image x. However, the compression made by the AE might lead to lower-quality reconstruction as the dimension of the latent vector becomes smaller.

A variant of the AE, which generates new data that is not strictly similar to the input data, is known as the variational auto-encoder (VAE). A VAE replaces the deterministic bottleneck representation z for a random sampling operation. Instead of learning specific values for the latent variables in the compressed vector z, it learns a random distribution over

each latent variable in z parameterized by mean and standard deviation. VAEs represent a probabilistic twist over AEs where they can sample from the mean and standard deviation to compute different latent variables (i.e., different z vectors) and generate new data. The rise and rapid evolution of GAN architectures caught the attention of handwriting generation researchers by 2018. The reason was the ability of GANs to generate high-fidelity images compared to those generated by auto-encoders and variational auto-encoders that were so popular before. For several years, GANs have remained the preferred type of image-generation models, with researchers proposing different architectures and optimization methods, even though GANs can be challenging to train. The GAN training process is inherently unstable, in particular, the simultaneous dynamic training of the two competing networks G and D. When training a GAN, one may face two problems, namely mode collapse (Sect. 3.2.4) and divergence (or non-convergence) of the model. Model collapse can lead to a lack of novelty in image generation—the generated images are not radically new or different from the images in the training data domain, and the GAN does not generalize well and scale.

Although stable training of GANs remains an open problem, many empirical tips and tricks have been proposed [62] that result in the reliable training of a stable GAN model. The recommendations involve (1) modifying the design of the GAN architecture, (2) selecting an appropriate optimization algorithm, and (3) proposing a loss function that reduces the divergence between the distribution of the training image data and the distribution of the generated image data. The notable work by Saxena and Cao [62] reviews the divergence of these distributions and describes regularization schemes across 24 GAN models. The work discusses the concerns raised by the authors of each model, the approaches used to handle these concerns, and the strengths and limitations of each proposed solution. Similarly, one by one, we have detailed the motivation, architecture modifications, loss functions, training procedure (all use the Adam optimizer [63]), and results for the nine pioneer GAN models for handwriting generation.

Alongside continued research on GANs, there has been a search for new paradigms for general image generation in order to find models that achieve training stability and efficiency, as well as quality and novelty of image generation. The most popular paradigm is the diffusion model [64–66], which started out as a model [67] that reportedly could generate images of animals (cats, horses) and scenes (bedrooms) with higher average FID scores than the StyleGAN model [68].

Diffusion models are now used in commercial products (e.g., DALLE-3 [69] and Stable Diffusion [70]) to create both photorealistic and non-photorealistic imagery. Only recently, the diffusion paradigm has been applied to the task of generating handwriting in fixed-sized images [18, 19] with results on datasets IAM and RIMES that reportedly have lower error scores than GANs [18] and also do much better on the task of writer retrieval [19].

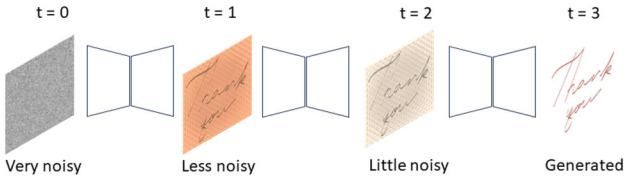

Fig. 9 Image generation using iterative noise cancellation via multiple diffusions

Unlike VAE or GAN models that generate samples in "one shot," guided by the vector of latent variables, diffusion models gradually de-noise an input sample by capturing the most important information and alleviating noise until a noise-free sample is generated (Fig. 9). A white noise image can be thought of as the representation of all possible images, including desired images of handwriting. Generating a desired image can then be done by a de-noising process that starts with a white noise image and iteratively cancels noise until a handwriting image emerges.

The training process of a diffusion model starts with "forward noising," where the information in the original image is gradually wiped out by an incremental amount of noise until the image contains pure noise. Then, the network is trained to estimate and gradually subtract the noise until it recovers the original image. To generate new images, the diffusion model performs the same iterative method of noise cancellation (de-noising), using a trained auto-encoder with skip connections, which estimates the amount of noise added to the input image of pure noise, then subtracts the noise from the image, and repeats the process multiple times.

A drawback of the original diffusion model [66] is that it works in the high-dimensional image space rather than the much lower-dimensional latent space, and is therefore slow to train. This motivated research work on latent diffusion models (LDMs) [70]. A LDM is very similar to an auto-encoder with an encoder–decoder structure. The difference is that the encoder network outputs the latent representation of the input image which is not directly decoded by the decoder network. Alternatively, a series of diffusion processes start on the latent representation rather than the original image input (i.e., the lower-dimensional latent space). Finally, after de-noising, the "clean" latent vector is decoded and projected back to the image space. The stable diffusion model [70] is the conditioned version of the latent diffusion model (LDM), where text is used as a conditional input to guide the de-noising process and generate a specific image content. The text must be encoded (embedded) before being concatenated to the latent representation that undergoes the diffusion process.

With a transformer network, the attention weights adapt dynamically to the input and are not static as in the convolution weights of a trained GAN generator. Therefore, high-quality zero-shot image generation is possible and visually satisfying. Transformers assume minimal prior knowledge about the structure of the problem, in contrast to convolutional blocks. Transformers make few or no assumption on the input data for the model design thus transformers have weak inductive bias. The model size of transformers-based architecture is a major drawback. The lack of specific problem assumptions leads to large models with many weights and requires large training datasets, or, alternatively, a pre-trained model on a large dataset from a different domain. The huge number of parameters for global spatial attention to the entire input image makes the computation of attention maps very expensive.

The existing paradigms for image generation, GANs, AEs, VAEs, and diffusion models, all share the concept of encoding the image information into a latent representation and decoding this representation back to a generated image under specific embedded conditions. There are similarities in the model architectures proposed under each of these paradigms and their performance for fixed-size images. The ability to generate variable-size images, however, is important for the generation of arbitrary-length words (Fig. 2). For variable-size images, GANs can generate the highest quality images without notable mode collapse, as we have seen for six out of the reviewed nine pioneer models.

5 Conclusions

Handwriting synthesis can be helpful to forensic examiners, people with disabilities, and researchers working on handwriting recognition systems, especially for low-resource languages. Handwritten text images have diverse writing styles and difficult-to-segment cursive joins [18]. Recent work on handwritten text generation shows that augmentation of training data using synthetic text images has improved the performance of handwritten text recognition systems.

In our previous work [1], we reviewed a decade of published works on handwriting generation and discussed their limitations, particularly in producing truly cursive text. As we described in this review, handwritten text synthesis faces many difficulties: the need to generate variable-sized images, as short as one word and as long as an entire text line, to generate arbitrary-length words out of the training vocabulary, and to imitate a reference writer's style. The conditional GAN architectures that have emerged since we published our last review article [1] have shown remarkable capabilities to transfer style and generate images of realistic handwritten text. As we detailed in this review article, they produce styles that are based on latent vectors sampled from a given distribution or disentangled from reference images.

In this article, we reviewed the 2019 seminal GAN model by Alonso et al. [2] and eight additional pioneer GAN-based handwriting generation models in detail, as well as works that used or adapted these models, with publication dates to the end of 2023. The range of dates shows that the research area is new and active.

We noticed that the handwritten datasets used were mostly in English with very few exceptions in Arabic, French, German, and Japanese. Notably, the other most spoken languages in the world, Chinese, Hindi, and Spanish, or languages with low resources have had less attention. We need more research involving other languages than English to investigate the challenges that these languages bring to handwriting generation. Such research could give rise to generative models that can be used to create large-size image datasets of synthesized handwriting, starting from a given word and generating a corresponding image of the handwritten text. These datasets could then be used to support the development of handwriting recognition models, providing researchers with images and ground-truth labels for training these models, without ensuing the costs of human annotation experiments.

As we detailed, the researchers' goal was to explore the best designs for the embedding, generator, and discriminator networks. They investigated the introduction of auxiliary networks to the seminal model for various assistive roles like recognition, encoding, and style extraction. They conducted numerous ablation studies to find out what kind of loss functions could aid the generator in obtaining what they see as the best realistic and meaningful image. They evaluated their systems qualitatively and quantitatively, using metrics from other domains, to demonstrate the superiority of their work. When we gathered the results they reported in tables to enable comparisons of performance, we noticed some mismatches between the numbers in the comparison tables reported in the individual papers. It is difficult to clearly point out weaknesses in the reviewed architectures, as most of the papers claim superiority and do not provide sufficient quantitative and qualitative error analyses (e.g., figures of failure cases). The numerical results look comparable in most cases, which makes us indecisive about a preferable model. The authors of the reviewed models have reported occasional flaws in the generation process in the form of visually degraded instances of generated words (Fig. 10). The authors did not report any issues with mode collapse. The low quality of some images may have been due to low-probability latent vectors. The flaws may also be attributed to a difficulty in capturing and imitating some complex writing styles, not seen before in the training data. This lack of generalization is an inevitable data-related issue that was initially the motive for the image-generation research.

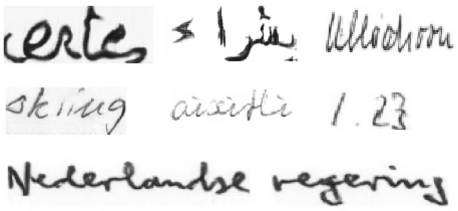

Fig. 10 Examples of unclear handwritten text images generated by the reviewed GAN-based architectures

A potential ethical concern is the illegal use of handwriting synthesis in forgery. Some researchers believe that such concern is overstated [7, 8]—as long as the work does not target imitating signatures and can only produce digital images, rather than physical documents, no ethical concerns should arise. Gan and Wang [7] also declared that the published works are still not strong enough to fool handwriting identification experts.

The reviewed articles make exceptional contributions. The efforts of the authors are undeniable. However, it is worth noting that human handwriting is very arbitrary, and thus, all the reviewed works indeed have limits for synthesizing meaningful handwriting images (Fig. 10). Despite such impressive efforts in developing models that imitate offline handwriting and their promising results, handwriting synthesis remains a challenging and unsolved problem. Future works on handwriting synthesis will likely continue to focus on ways to address style representation and content embedding by trying different encoder designs or different concepts of representation. Researchers should keep working on the evaluation methods. So far there is no clear relationship between the human assessment metrics and the success of style transfer, or text recognition results. Future works should explore different languages other than English especially low-resource languages and languages with large character sets. It is important that the researchers publish their generated datasets as well as their code. Many research areas are in dire need of labeled datasets, and regardless of the quality of the generated images, the privilege of having images of handwritten text with associated annotations will make a great difference for such research areas.

Acknowledgements Not applicable.

Author Contributions R.E. wrote the main manuscript text, and M.B. managed the manuscript structure and level of details in each section. All authors reviewed the manuscript.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Data Availability Statement Not applicable.

Declarations

Conflict of interest Not applicable.

Ethical Approval Not applicable.

References

1. Elanwar, R.I.: The state of the art in handwriting synthesis. In: 2nd International Conference on New Paradigms in Electronics & Information Technology. PEIT'013. ERI, Luxor, pp. 1–12 (2013)
2. Alonso, E., Moysset, B., Messina, R.: Adversarial generation of handwritten text images conditioned on sequences. In: International Conference on Document Analysis and Recognition, ICDAR, pp. 481–486. IEEE, Sydney (2019)
3. Fogel, S., Averbuch-Elor, H., Cohen, S., Mazor, S., Litman, R.: ScrabbleGAN: semi-supervised varying length handwritten text generation. In: IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 4324–4333. IEEE, New Jersey (2020)
4. Zdenek, J., Nakayama, H.: JokerGAN: memory-efficient model for handwritten text generation with text line awareness. In: 29th ACM International Conference on Multimedia. MM '21, pp. 5655–5663. ACM (2021)
5. Liu, X., Meng, G., Xiang, S., Pan, C.: Handwritten text generation via disentangled representations. IEEE Signal Process. Lett. **28**, 1838–1842 (2021)
6. Kang, L., Riba, P., Wang, Y., Rusiñol, M., Fornés, A., Villegas, M.: GANwriting: content-conditioned generation of styled handwritten word images. In: European Conference on Computer Vision. ECCV, vol. 12368, pp. 237–289. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58592-1_17
7. Gan, J., Wang, W.: HiGAN: Handwriting imitation conditioned on arbitrary-length texts and disentangled styles. In: AAAI Conference on Artificial Intelligence, vol. 35(9), pp. 7484–7492. MLR Press, Cambridge (2021). https://doi.org/10.1609/aaai.v35i9.16917
8. Davis, B., Tensmeyer, C., Price, B., Wigington, C., Morse, B., Jain, R.: Text and style conditioned GAN for generation of offline handwriting lines. arXiv:2009.00678 (2020)
9. Gan, J., Wang, W., Leng, J., Gao, X.: HiGAN+: handwriting imitation GAN with disentangled representations. ACM Trans. Graph. **42**(1), 1–17 (2022)
10. Luo, C., Zhu, Y., Jin, L., Li, Z., Peng, D.: SLOGAN: handwriting style synthesis for arbitrary-length and out-of-vocabulary text. IEEE Trans. Neural Netw. Learn. Syst. (2022). https://doi.org/10.1109/TNNLS.2022.3151477

11. Kang, L., Riba, P., Rusiñol, M., Fornés, A., Villegas, M.: Content and style aware generation of text-line images for handwriting recognition. IEEE Trans. Pattern Anal. Mach. Intell. **44**(12), 8846–8860 (2022)
12. Chang, C.C., Perera, L.P.G., Khudanpur, S.: Crosslingual handwritten text generation using GANs. In: International Conference on Document Analysis and Recognition. ICDAR, pp. 285–301. Springer, San Jose (2023)
13. Wang, H., Wang, Y., Wei, H.: Affganwriting: a handwriting image generation method based on multi-feature fusion. In: International Conference on Document Analysis and Recognition. ICDAR, pp. 302–312. Springer, San Jose (2023)
14. Zdenek, J., Nakayama, H.: Handwritten text generation with character-specific encoding for style imitation. In: International Conference on Document Analysis and Recognition. ICDAR, pp. 313–329. Springer, San Jose (2023)
15. Bhunia, A.K., Khan, S., Cholakkal, H., Anwer, R.M., Khan, F.S., Shah, M.: Handwriting transformers. In: IEEE International Conference on Computer Vision. ICCV, pp. 1086–1094. IEEE, Montreal (2021)
16. Wang, Y., Wang, H., Sun, S., Wei, H.: An approach based on transformer and deformable convolution for realistic handwriting samples generation. In: International Conference on Pattern Recognition. ICPR, pp. 1457–1463. IEEE, Montreal (2022)
17. Pippi, V., Cascianelli, S., Cucchiara, R.: Handwritten text generation from visual archetypes. In: IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 22458–22467. IEEE, Vancouver (2023)
18. Zhu, Y., Li, Z., Wang, T., He, M., Yao, C.: Conditional text image generation with diffusion models. In: IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 14235–14245. IEEE, Vancouver (2023)
19. Nikolaidou, K., Retsinas, G., Christlein, V., Seuret, M., Sfikas, G., Smith, E.B., Mokayed, H., Liwicki, M.: Wordstylist: styled verbatim handwritten text generation with latent diffusion models. In: International Conference on Document Analysis and Recognition. ICDAR, pp. 384–401. Springer, San Jose (2023)
20. Dilipkumar, D.: Generative Adversarial Image Refinement for Handwriting Recognition. Carnegie Mellon University, Pennsylvania (2017). http://www.ml.cmu.edu/research/dap-papers/F17/dap-dilipkumar-deepak.pdf
21. Waheed, A., Goyal, M., Gupta, D., Khanna, A., Al-Turjman, F., Pinheiro, P.R.: CovidGAN: data augmentation using auxiliary classifier GAN for improved COVID-19 detection. IEEE Access **8**, 91916–91923 (2020). https://doi.org/10.1109/ACCESS.2020.2994762
22. Jain, S., Seth, G., Paruthi, A., Soni, U., Kumar, G.: Synthetic data augmentation for surface defect detection and classification using deep learning. J. Intell. Manuf. **33**, 1007–1020 (2022). https://doi.org/10.1007/s10845-020-01710-x
23. Xu, M., Yoon, S., Fuentes, A., Park, D.S.: A comprehensive survey of image augmentation techniques for deep learning. Pattern Recognit. **137**, 109347 (2023). https://doi.org/10.1016/j.patcog.2023.109347
24. Yang, S., Xiao, W., Zhang, M., Guo, S., Zhao, J., Shen, F.: Image Data Augmentation for Deep Learning: A Survey (2023)
25. Chlap, P., Min, H., Vandenberg, N., Dowling, J., Holloway, L., Haworth, A.: A review of medical image data augmentation techniques for deep learning applications. J. Med. Imaging Radiat. Oncol. **65**(5), 545–563 (2021)
26. Liang, W., Liang, Y., Jia, J.: MiAMix: enhancing image classification through a multi-stage augmented mixed sample data augmentation method. Processes (2023). https://doi.org/10.3390/pr11123284
27. Lian, Z., Zhao, B., Chen, X., Xiao, J.: Easyfont: a style learning-based system to easily build your large-scale handwriting fonts. ACM Trans. Graph. **38**(1), 1–18 (2018)
28. Souibgui, M.A., Biten, A.F., Dey, S., Fornés, A., Kessentini, Y., Gómez, L., Karatzas, D., Lladós, J.: One-shot compositional data generation for low resource handwritten text recognition. In: IEEE Winter Conference on Applications of Computer Vision. WACV, pp. 935–943. IEEE, Waikola (2022)
29. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Commun. ACM **63**(11), 139–144 (2020)
30. Chang, B., Zhang, Q., Pan, S., Meng, L.: Generating handwritten Chinese characters using CycleGAN. In: IEEE Winter Conference on Applications of Computer Vision. WACV, pp. 199–207. IEEE, Lake Tahoe (2018)
31. Zhu, J.-Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: IEEE Conference on Computer Vision. ICCV, pp. 2223–2232. IEEE, Venice (2017)
32. Graves, A.: Generating Sequences With Recurrent Neural Networks. arXiv:1308.0850 (2013)
33. Aksan, E., Pece, F., Hilliges, O.: Deepwriting: making digital ink editable via deep generative modeling. In: SIGCHI Conference on Human Factors in Computing Systems. CHI-18, vol. 205, pp. 1–14. ACM, Montreal (2018)
34. Ji, B., Chen, T.: Generative adversarial network for handwritten text. arXiv:1907.11845 (2019)
35. Tolosana, R., Delgado-Santos, P., Perez-Uribe, A., Vera-Rodriguez, R., Fierrez, J., Morales, A.: DeepwriteSYN: on-line handwriting synthesis via deep short-term representations. In: AAAI Conference on Artificial Intelligence, vol. 35, pp. 600–608. MLR Press, Cambridge (2021)
36. Dai, G., Zhang, Y., Wang, Q., Du,: Q., Yu, Z., Liu, Z., Huang, S.: Disentangling writer and character styles for handwriting generation. In: IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 5977–5986. IEEE, Vancouver (2023)
37. Wen, C., Pan, Y., Chang, J., Zhang, Y., Chen, S., Wang, Y., Han, M., Tian, Q.: Handwritten Chinese font generation with collaborative stroke refinement. In: IEEE Conference on Computer Vision and Pattern Recognition. WACV, pp. 3882–3891. IEEE, Online (2021)
38. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv **1411**, 1784 (2014)
39. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural nets. In: 23rd International Conference on Machine Learning. ICML '06, pp. 369–376. ACM, Pennsylvania (2006)
40. Marti, U.-V., Bunke, H.: The IAM-database: an English sentence database for offline handwriting recognition. Int. J. Doc. Anal. Recognit. **5**(1), 39–46 (2002)
41. Kleber, F., Fiel, S., Diem, M., Sablatnig, R.: CVL-database: an off-line database for writer retrieval, writer identification and word spotting. In: 12th International Conference on Document Analysis and Recognition. ICDAR, pp. 560–564. IEEE, Washington (2013). https://doi.org/10.1109/ICDAR.2013.117
42. Grosicki, E., Abed, H.E.: ICDAR 2009 handwriting recognition competition. In: 10th International Conference on Document Analysis and Recognition. ICDAR, pp. 1398–1402. IEEE, Barcelona (2009). https://doi.org/10.1109/ICDAR.2009.184
43. Tong, A., Przybocki, M., Märgner, V., Abed, H.E.: Nist 2013 open handwriting recognition and translation (open hart'13) evaluation. In: 11th IAPR International Workshop on Document Analysis Systems, pp. 81–85. IEEE, Tours (2014). https://doi.org/10.1109/DAS.2014.43 . https://www.nist.gov/itl/iad/mig/openhart
44. Khrulkov, V., Oseledets, I.: Geometry score: a method for comparing generative adversarial networks. In: 35th International

Conference on Machine Learning. PMLR, vol. 80, pp. 2621–2629. MLR Press, Stockholm (2018)
45. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems. NIPS, vol. 30. Curran Associates, Inc., Long Beach (2017). https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf
46. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier GANs. In: 34th International Conference on Machine Learning. PMLR, vol. 70, pp. 2642–2651. MLR Press, Sydney (2017)
47. Shmelkov, K., Schmid, C., Alahari, K.: How good is my GAN? In: European Conference on Computer Vision. ECCV, pp. 213–229. Springer, Munich (2018)
48. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., Chen, X.: Improved techniques for training GANs. In: Advances in Neural Information Processing Systems. NIPS, vol. 29, pp. 2226–2234. Curran Associates, Inc., Barcelona (2016). https://proceedings.neurips.cc/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf
49. Bińkowski, M., Sutherland, D.J., Arbel, M., Gretton, A.: Demystifying MMD GANs. arXiv: 1801.01401 (2018)
50. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: continual prediction with LSTM. Neural Comput. **12**(10), 2451–2471 (2000)
51. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Trans. Pattern Anal. Mach. Intell. **39**(11), 2298–2304 (2016)
52. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 770–778. IEEE, Las Vegas (2016)
53. Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O., Courville, A.C.: Modulating early visual processing by language. In: Advances in Neural Information Processing Systems. NIPS, vol. 30. Curran Associates, Inc., Long Beach (2017). https://proceedings.neurips.cc/paper/2017/file/6fab6e3aa34248ec1e34a4aeedecddc8-Paper.pdf
54. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention generative adversarial networks. In: 36th International Conference on Machine Learning. PMLR, vol. 97, pp. 7354–7363. MLR Press, Long Beach (2019)
55. Lim, J.H., Ye, J.C.: Geometric GAN. arXiv:1705.02894 (2017)
56. Zhu, J.-Y., Zhang, R., Pathak, D., Darrell, T., Efros, A.A., Wang, O., Shechtman, E.: Toward multimodal image-to-image translation. In: Advances in Neural Information Processing Systems. NIPS, vol. 30. Curran Associates, Inc., Long Beach (2017). https://proceedings.neurips.cc/paper/2017/file/819f46e52c25763a55cc642422644317-Paper.pdf
57. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 (2014)
58. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: IEEE Conference on Computer Vision. ICCV, pp. 1501–1510. IEEE, Venice (2017)
59. Liu, M.-Y., Huang, X., Mallya, A., Karras, T., Aila, T., Lehtinen, J., Kautz, J.: Few-shot unsupervised image-to-image translation. In: IEEE/CVF International Conference on Computer Vision. ICCV, pp. 10551–10560. IEEE, Seoul (2019)
60. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv:2010.11929 (2020)
61. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision. ECCV, pp. 694–711. Springer, Amsterdam (2016). https://doi.org/10.1007/978-3-319-46475-6_43
62. Saxena, D., Cao, J.: Generative adversarial networks (GANs) challenges, solutions, and future directions. ACM Comput. Surv. (CSUR) **54**(3), 1–42 (2021)
63. Kingma, D.P., Welling, M.: Auto-encoding Variational Bayes. arXiv:1312.6114 (2013)
64. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Conference on Neural Information Processing Systems. NeurIPS, pp. 2256–2265, Online (2020)
65. Nichol, A.Q., Dhariwal, P.: Improved denoising diffusion probabilistic models. In: International Conference on Machine Learning. ICML, pp. 8162–8171, Online (2021)
66. Sohl-Dickstein, J., Weiss, E.A., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: International Conference on Machine Learning. ICML, pp. 2256–2265. Lille (2015)
67. Dhariwal, P., Nichol, A.Q.: Diffusion models beat GANs on image synthesis. In: Conference on Neural Information Processing Systems. NeurIPS, pp. 8780–8794, Online (2021)
68. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Conference on Computer Vision and Pattern Recognition, pp. 4401–4412. IEEE, Utah (2018)
69. Betker, J., Goh, G., Jing, L., Brooks, T., Wang, J., Li, L., Ouyang, L., Zhuang, J., Lee, J., Guo, Y., Manassra, W., Dhariwal, P., Chu, C., Jiao, Y., Ramesh, A.: Improving Image Generation with Better Captions. https://cdn.openai.com/papers/dall-e-3.pdf (2023)
70. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans. CVPR, pp. 10684–10695. IEEE, New Orleans (2022)

Randa Elanwar is an Egyptian researcher with expertise in machine learning for document analysis, handwriting recognition, and information retrieval. She earned her bachelor degree in Electronics and Communications at Faculty of Engineering - Cairo University in 2003 followed by Masters and Ph.D. in 2007 and 2012 at Cairo University as well. She is a former visiting researcher at Boston University in 2015. Currently, she is a senior researcher at the Computers and Systems Dept. at the Electronics Research Institute, Egypt. Her research interests include datasets collection and finding solutions to research problems of Arabic documents analysis and recognition.

Margrit Betke is a Professor of Computer Science at Boston University, where she co-leads the Artificial Intelligence and Image and Video Computing Research Groups. She also serves as a founding member of the BU Faculty of Computing and Data Sciences. She conducts research in computer vision, artificial intelligence, multimodal data science, deep learning, and medical image analysis. She has published over 150 original research papers. She earned her Ph.D. degree in Computer Science and Electrical Engineering at the Massachusetts Institute of Technology in 1995. Prof. Betke received the National Science Foundation Faculty Early Career Development Award in 2001 for developing "Video-based Interfaces for People with Severe Disabilities." She co-invented the "Camera Mouse," an assistive technology used worldwide by children and adults with severe motion impairments. While she was a Research Scientist at the Massachusetts General Hospital and Harvard Medical School, she co-developed the first patented algorithms for detecting and measuring pulmonary nodule growth in computed tomography. She is a Senior Member of the ACM and IEEE and an Associate Editor of the journal IEEE Transactions on Pattern Analysis and Machine Intelligence. She led large National Science Foundation research projects on developing visual tracking systems and on automated analysis of public communications.