DATA INCUBATION — SYNTHESIZING MISSING DATA FOR HANDWRITING RECOGNITION

Jen-Hao Rick Chang, Martin Bresler, Youssouf Chherawala, Adrien Delaye, Thomas Deselaers, Ryan Dixon, Oncel Tuzel

Apple

ABSTRACT

In this paper, we demonstrate how a generative model can be used to build a better recognizer through the control of content and style. We are building an online handwriting recognizer from a modest amount of training samples. By training our controllable handwriting synthesizer on *the same* data, we can synthesize handwriting with previously underrepresented content (*e.g.*, URLs and email addresses) and style (*e.g.*, cursive and slanted). Moreover, we propose a framework to analyze a recognizer that is trained with a mixture of real and synthetic training data. We use the framework to optimize data synthesis and demonstrate significant improvement on handwriting recognition over a model trained on real data only. Overall, we achieve a 66% reduction in Character Error Rate.

Index Terms— Handwriting synthesis, handwriting recognition, synthetic data, controllable generative models

1. INTRODUCTION

Data is the most valuable asset for machine learning applications, but are we using it efficiently? The standard approach is relatively simple: (i) collect and annotate data, and (ii) train a recognizer using the data. While this approach has achieved success in many standard machine learning tasks, such as image classification [1], we will show that it is sub-optimal for handwriting recognition.

Handwriting data can be described (a) by its content (*i.e.*, the text that was written, a sequence of characters) and (b) by its style (*i.e.*, all other information about a handwriting sample except content — printed or cursive, neat or messy, round or square corners, tight or loose spacing, *etc.*). The content space is combinatorial and some handwriting styles are rare compared to others. Therefore, it is difficult to collect sufficient data to cover the entire content and style space well.

In this paper, we propose *data incubation* that grows the collected data and improve the generalization of recognizer training. This capability is enabled by a controllable generative model that is used to augment training with synthesized data covering underrepresented content and rare styles (Fig. 1).

How does a controllable generative model improve data efficiency for a downstream handwriting recognizer? After all, we learn the generative model with the same dataset that we use to train the recognizer. Our approach is based on the following observation: Since the total number of collected samples is limited, the diversity of contents and styles in the collected samples is also limited (Fig. 1a). By utilizing our controllable synthesizer that models the data distribution as a factored distribution of style and content, we can combine information across collected samples, complement the real training set with synthesized handwriting, and fill both the content and the style gaps in the collected training data (Fig. 1b-d).

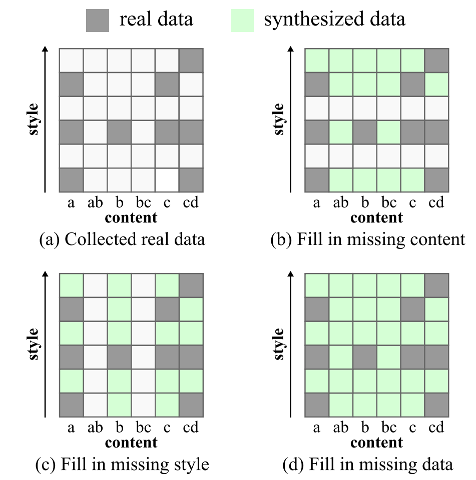


Fig. 1: Data incubation starts with (a) collected data sparsely covering the content and the style space. We then use a controllable generative model that can (b) synthesize samples with unseen content, and (c) mix new style from existing training samples. (d) Finally, we synthesize new samples that close gaps in content and style, leading to a better coverage of the real data distribution for handwriting recognition.

Effectively, we introduce prior knowledge about content and style being factorizable into the creation of synthetic training data. This inductive bias allows us to strategically increase the size of our training set to target a more balanced distribution of writing styles and content as well as training recognizers with better generalization.

Training with synthetic data, however, comes with its own challenges. Synthetic data might introduce a shift from the *empirical distribution* that is formed by the collected real data. While this shift can be due to the artifacts produced by the generative model, it can also come from generating content and styles that are underrepresented or missing in the collected data, *i.e.*, shifting the empirical distribution shown in Fig. 1a to that in Fig. 1d. In this aspect, this distribution shift can be seen both as a problem but also as a feature. It is critical to make careful and deliberate choices about which data to synthesize. If we only synthesize handwriting from well-formed English sentences with highly legible, printed style, a recognizer trained on this data will end up performing poorly on messy, cursive styles and content including slang, uncommon words, or abbreviations. On the other hand, we can carefully choose which data to synthesize and with that we will be able to close the gaps in the empirical distribution.

In this paper, we demonstrate a systematic recipe to train a handwriting recognizer with data synthesized by a controllable generative model. This recipe enables us to examine and understand the problems associated with the synthetic data. We build on the recently-proposed controllable generative model [2], which separately controls the content and the handwriting style. As we will demonstrate, utilizing data incubation significantly improves our multilingual handwriting recognizer, achieving a 66% reduction in Character Error Rate when compared to a recognizer trained with only original data.

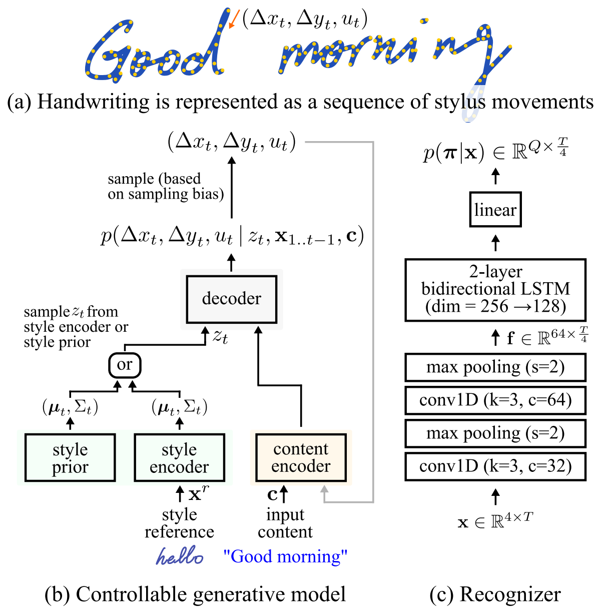
2. RELATED WORK

Online handwriting recognition is a field of high relevance in mobile computing because (a) an increasing number of devices are equipped with styluses making handwritten input a natural interface, (b) many languages are harder to enter with a keyboard than Latin scripts because the size of their alphabet or the use of grapheme clusters makes building intuitive keyboards difficult. While early work in online handwriting recognition used segment-and-decode methods [3, 4, 5, 6], recent work trends toward end-to-end approaches utilizing Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), coupled with Connectionist Temporal Classification (CTC) decoding [7, 8, 9, 10, 11].

Online handwriting synthesis models [2, 12, 13, 14, 15, 16] are sequence-to-sequence models that take input text (a sequence of characters) and output handwriting (a sequence of strokes). To improve downstream recognizers, it is important for the synthesized handwriting to contain few artifacts and a wide range of handwriting styles — from highly legible, printed-style handwriting to less legible, cursive handwriting.

Synthetic data augmentation for training machine learning models is not new in the literature. Most of the reference papers focus on image [17, 18] and speech [19, 20, 21, 22, 23] synthesis, and obtain modest improvements. To the best of our knowledge, this paper is the first to use a controllable generative model to synthesize online handwriting and explicitly control the content, style, and variation of the generation process to significantly improve the accuracy of downstream handwriting recognizers.

The missing data problem has been studied in [24, 25]. These works focus on model selection whereas we use a generative model to fill in the missing data.



Good morning
$(\Delta x_t, \Delta y_t, u_t)$
(a) Handwriting is represented as a sequence of stylus movements
$(\Delta x_t, \Delta y_t, u_t)$
sample (based on sampling bias)
$p(\Delta x_t, \Delta y_t, u_t \mid z_t, \mathbf{x}_{1..t-1}, \mathbf{c})$
decoder
z_t
sample z_t from style encoder or style prior
or
(μ_t, Σ_t)
(μ_t, Σ_t)
style prior
style encoder
content encoder
$\mathbf{x}^r$
style reference
hello
$\mathbf{c}$
input content
"Good morning"
(b) Controllable generative model
$p(\boldsymbol{\pi}|\mathbf{x}) \in \mathbb{R}^{Q\times\frac{T}{4}}$
linear
2-layer bidirectional LSTM (dim = 256 →128)
$\mathbf{f} \in \mathbb{R}^{64\times\frac{T}{4}}$
max pooling (s=2)
conv1D (k=3, c=64)
max pooling (s=2)
conv1D (k=3, c=32)
$\mathbf{x} \in \mathbb{R}^{4\times T}$
(c) Recognizer

Fig. 2: (a) Online handwriting is a sequence of stylus movements. (b) Our handwriting generative model takes an input content $\mathbf{c}$ and outputs handwriting $\mathbf{x}$. The handwriting style is controlled using a latent style variable z_t. The model can either mimic the style of an existing style reference handwriting or it can sample a new style from a prior distribution to generate unseen styles. (c) Our recognizer takes an input handwriting $\mathbf{x}$ and outputs the probabilities of all symbols at each time step, $p(\pi|\mathbf{x})$, which is converted to $p(\mathbf{c}|\mathbf{x})$ by CTC decoding.

3. GENERATIVE AND RECOGNIZER MODELS

3.1. Problem formulation and dataset

An online handwriting sample $\mathbf{x} = [(\Delta x_t, \Delta y_t, u_t)]_{t=1...T}$ is a sequence of stylus movements on a writing surface, where $\Delta x_t \in \mathbb{R}$ and $\Delta y_t \in \mathbb{R}$ are the movement of the stylys in x and y directions respectively at time t, and $u_t \in \{0, 1\}$ is a binary variable indicating whether the stylus touches the surface during the movement, as shown in Fig. 2a. The corresponding content of the sample $\mathbf{c} = [c_1, \dots c_M]$ is a sequence of characters, where c_m is represented as a one-hot vector of dimension Q (the total number of characters). Note that in general $\mathbf{x}$ and $\mathbf{c}$ have different lengths.

We use a subset of a proprietary dataset collected for research and development. The subset consists of 600k online handwriting samples written by 1,500 people. Half of the samples (300k) is in English, and the rest is in French, German, Italian, Spanish, and Portuguese.

3.2. Controllable generative model

We use the controllable generative model proposed in [2]. Given N real training samples, $\mathcal{X} = \{(\mathbf{x}^i, \mathbf{c}^i)\}_{i=1...N}$, the model learns a distribution of handwriting samples, $p(\mathbf{x}|\mathbf{c}, \mathbf{z})$, conditioned on the content $\mathbf{c}$ and the style $\mathbf{z}$. In other words, given $\mathbf{c}$ and $\mathbf{z}$, we can generate new handwriting by sampling from the distribution. Note that the style $\mathbf{z}$ is modeled as a latent variable with a prior distribution. Fitting a prior distribution for style is important, because it enables the model to create new styles (*i.e.*, $\mathbf{z}$ not associated with any training sample). In other words, we fill in the missing style via sampling the style $\mathbf{z}$ from the prior distribution during synthesis.

Fig. 2b shows an overview of the model architecture. We model the output distribution $p(\Delta x_t, \Delta y_t, u_t|z_t, \mathbf{x}_{1..t-1}, \mathbf{c})$ as $p(\Delta x_t, \Delta y_t|\cdot)p(u_t|\cdot)$, where $p(\Delta x_t, \Delta y_t|\cdot)$ is a mixture of 20 bivariate Gaussian distributions modeling the movement of the stylus, and $p(u_t|\cdot)$ is a Bernoulli distribution modeling whether the stylys touches the surface. The prior distribution of z_t, $p(z_t|\mathbf{x}_{1..t-1}, \mathbf{c})$, is a multivariate Gaussian distribution $\mathcal{N}(z_t|\boldsymbol{\mu}_t, \Sigma_t))$, where Σ_t is a diagonal covariance, and $\boldsymbol{\mu}_t$ and Σ_t are functions of $\mathbf{x}_{1..t-1}$ and $\mathbf{c}$ (generated by a linear layer). We refer to [2] for the details. We trained two generative models: an English model using the English training data and a multilingual model using training data from six languages.

One important property of the generative model is that we can control the variation of the synthesized data by scaling the standard deviations of the output distribution $p(\Delta x_t, \Delta y_t, u_t|z_t, \mathbf{x}_{1..t-1}, \mathbf{c})$. Intuitively, using a small standard deviation generates handwriting close to the mean of the distribution, which results in neat (average-style) handwriting; using a large standard deviation results in more variations in the synthesized handwriting. When synthesizing, we use *sampling bias* $b \geq 0$ to scale individual standard deviations and the categorical distribution in the mixture of the Gaussian distributions, following the rule in [14]. A small sampling bias produces large variations in handwriting, and a large sampling bias results in small variations and neat/typical handwriting.

3.3. Handwriting recognizer

A handwriting recognizer learns a distribution of the content, $p(\mathbf{c}|\mathbf{x})$, given an input handwriting sequence $\mathbf{x}$. As shown in Fig. 2c, our handwriting recognizer is a 2-layer convolution network followed by a 2-layer bidirectional LSTM and a linear layer. The input to the recognizer is a sequence containing $u_t, \sin(\theta_t), \cos(\theta_t), \ell_t$ for $t = 1, \ldots, T$, where θ and ℓ are the angle and length of the stylus movement at time t, *i.e.*, they are features computed from Δx_t and Δy_t. The network outputs the probabilities of each character (including a *blank* symbol) at every time t. We use CTC loss [7] to train the model.

4. SYNTHESIZING TRAINING DATA

If data is not synthesized carefully, the domain gap between real and synthetic data can deteriorate the performance of the recognizer on real data. Here, we present a systematic framework to identify problems with synthetic data and optimize parameters of the generative model to minimize the gap. Our recipe uses a similar procedure as [26] to evaluate the generation quality of a generative adversarial network.

Given a synthetic dataset (randomly split into training, validation, and test sets), we train three recognizers: one using the real training data, one using the synthetic training data, and one using both. Each model is evaluated using Character Error Rate (CER) on both the real and synthetic test sets. We use the notation $\text{CER}_{\text{t}\to\text{v}}$ to indicate the CER for a model trained on training set t and evaluated on test set v, where $\text{t}, \text{v} \in \{\text{r}, \text{s}, \text{b}\}$ indicating **r**eal, **s**ynthetic, and **b**oth data sets, respectively.

Note that the real training set (empirical distribution) is only a finite sample approximation to the true data distribution and might be biased due to the data collection procedure or lack certain content, style, or other variations. Here, we refer to the empirical distribution as $\mathcal{S}_r$, synthetic data distribution as $\mathcal{S}_s$ and the true data distribution as $\mathcal{S}_t$. We assume that the real test set is unbiased. The following are the *common* cases:

Case 1: $\mathcal{S}_s \not\subseteq \mathcal{S}_t$. Synthetic data contains artifacts or variations that do not exist in the true distribution. These artifacts are equivalent to introducing noisy samples to the training set. In an extreme case, the artifacts may correlate with the content; therefore, the recognizer can exploit these artifacts and will not generalize. The extreme case can be diagnosed by lower $\text{CER}_{\text{s}\to\text{s}}$, but high $\text{CER}_{\text{s}\to\text{r}}$ and $\text{CER}_{\text{r}\to\text{s}}$.

Case 2: $\mathcal{S}_s \subseteq \mathcal{S}_t$ & $\mathcal{S}_s \subset \mathcal{S}_r$. Synthetic data is a subset of the true distribution, but fails to capture all the variations that exist in the real dataset (empirical distribution). The model trained on synthetic data will fail to generalize well. For example, when both real and synthetic data share the same corpus but the synthetic data only contain printed-style handwriting or neat handwriting (when we use a high sampling bias), the recognizer trained on synthetic data will perform poorly on the real data. This case can be diagnosed by lower $\text{CER}_{\text{s}\to\text{s}}$ and $\text{CER}_{\text{r}\to\text{s}}$, but high $\text{CER}_{\text{s}\to\text{r}}$.

Case 3: $\mathcal{S}_s \subseteq \mathcal{S}_t$ & $\mathcal{S}_s \not\subseteq \mathcal{S}_r$. Synthetic data is a subset of the true distribution and contains variations (*e.g.* missing style and content) that are not covered in the real dataset. This is the scenario where the recognizer will benefit from the synthetic data. By filling in missing data, synthetic data improves the performance of the recognizer leading to lower $\text{CER}_{\text{s}\to\text{r}}$ (and $\text{CER}_{\text{s}\to\text{s}}$) than $\text{CER}_{\text{r}\to\text{r}}$ and $\text{CER}_{\text{r}\to\text{s}}$.

To mitigate the problems discussed in case 1 and case 2, either the generative model or the sampling process has to be improved accordingly. When the generative model exactly models the true distribution, $\text{CER}_{\text{s}\to\text{r}}$ and $\text{CER}_{\text{s}\to\text{s}}$ should be identical. In practice, this is difficult to achieve; however, choosing the generative model such that $\text{CER}_{\text{s}\to\text{r}}$ and $\text{CER}_{\text{s}\to\text{s}}$ are close is a good strategy to minimize the gap between synthetic and real data. Moreover, in general, neither case 2 nor case 3 are exact (synthetic and real data contains different variations of the true distribution); therefore, combining real data with synthetic data (CER$_{b\rightarrow r}$) improves over using synthetic data (CER$_{s\rightarrow r}$) or real data (CER$_{r\rightarrow r}$) alone. In the experiments section, we analyze these cases based on the sampling bias of the generative model.

5. EXPERIMENTS

We first use our proposed framework to evaluate the effect of style diversity. As discussed, increasing the sampling bias reduces the style diversity of synthetic data. Fig. 3 shows the CERs of the English recognizer when varying the sampling bias and the amount of synthetic training data. To avoid introducing new content, the data was synthesized using the same text corpus as the original training data. The results indicate that: (1) A large sampling bias (bias ≥ 0.4; *i.e.* neat training data) leads to low CER$_{r\rightarrow s}$ and high CER$_{s\rightarrow r}$, indicating a decrease in the diversity of the synthetic data (Fig. 3a), corresponding to Case 2. (2) A small sampling bias ($0.05 \leq$ bias < 0.4) increases the diversity of the data, covering a large distribution of styles, leads to lower CER$_{s\rightarrow r}$ and CER$_{b\rightarrow r}$ than CER$_{r\rightarrow r}$, corresponding to Case 3. Due to the increase in data diversity, we also observe an increase in CER$_{r\rightarrow s}$ and CER$_{s\rightarrow s}$. (3) Further reducing the sampling bias (bias < 0.05) leads to an increase in CER$_{s\rightarrow r}$ and CER$_{b\rightarrow r}$ due to the occasional generation of out-of-distribution samples, corresponding to Case 1. When we use a very small sampling bias, we sample from a mixture of Gaussian distributions with large standard deviations. Therefore, the larger standard deviations make it more likely to generate out-of-distribution samples. (4) Sampling style from the prior allows us to synthesize samples with style not in the collected training data, as shown by the comparison between *1M* and *1Ms* in Fig. 3d.

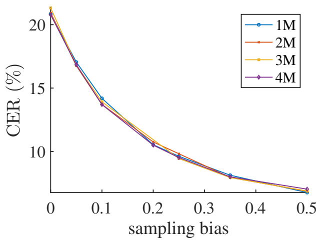
These results demonstrate the effectiveness of data incubation to improve recognizers and of the proposed recipe to analyze synthetic data and optimize synthesis. We reduce the CER of the English recognizer by 28% compared to the model trained only using real data (Table 1) because we are able to increase the diversity of writing styles.

For our multilingual recognizer, we show that using additional content helps improve the CER even further in a scenario where we don't have good coverage of all relevant content. When training our system with the same text corpus, we obtain an improvement of 46% on the normal multilingual test corpus. However, in this scenario, we only have 60k training samples for each of the non-English languages (compared to 300k for English). Therefore we increase the content diversity by synthesizing from a large multilingual text corpus. This leads to an improvement of 66% (Table 1).

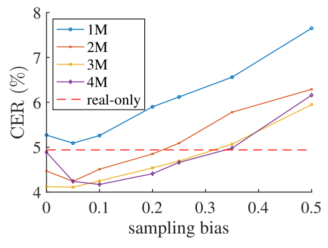
Finally, we show results on a special patterns (SP) evaluation set containing content like URLs, emails, addresses, and hashtags which are underrepresented in the collected training data. As shown in the last column of Table 1, expanding the synthesis corpus also enables our multilingual model to achieve 66% reduction in CER.

Table 1: CERs on real test data for the English and multi-language setup. *both* means real plus synthetic data; *training style* means the synthetic data mimic the style of real training samples; *extra style* means style is sampled from prior; *extra content* means synthetic data uses a corpus containing extra content. For the multi-language setup, we also report the results on a special patterns (SP) dataset containing URLs, made-up addresses, and made-up email addresses.

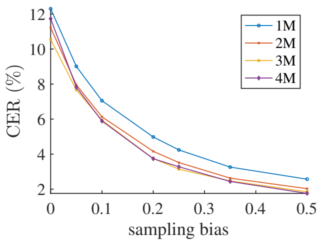
			Test		SP Test	
Setup	tr. data	|tr. data|	CER	impr.	CER	impr.
English	real	0.3	4.9	-	n/a	n/a
	both (training style)	1.3	4.7	↓ 4%	n/a	n/a
	both (extra style)	1.3	4.0	↓ 18%	n/a	n/a
	both (extra style)	4.3	**3.6**	↓ **28%**	n/a	n/a
Multi-Lang	real	0.6	8.2	-	18.5	-
	synthetic (extra style)	3.0	5.2	↓ 37%	18.9	↑ 2%
	both (extra style)	3.6	4.5	↓ 46%	16.9	↓ 9%
	both (extra style, content)	8.1	**2.8**	↓ **66%**	**6.3**	↓ **66%**



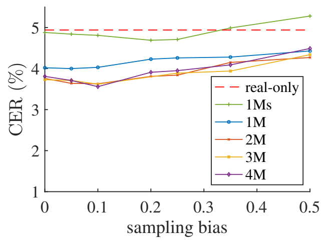
(a) CER$_{r\rightarrow s}$ on English



(b) CER$_{s\rightarrow r}$ on English



(c) CER$_{s\rightarrow s}$ on English



(d) CER$_{b\rightarrow r}$ on English

Fig. 3: Effect of sampling bias and synthetic dataset size on English recognizers. The synthetic data are generated via sampling the style prior distribution, *i.e.*, we fill in missing style, except for *1Ms* which mimics the style of real training samples. The text corpus is the same as the real training data.

6. CONCLUSION

We presented data incubation—a new framework to improve recognition by (i) training a controllable generative model to synthesize missing data, (ii) optimizing data synthesis, and (iii) training recognition models by strategically combining synthetic and real data. We applied this framework to handwriting recognition and achieved a 66% reduction in CER.

7. REFERENCES

[1] Mingxing Tan and Quoc Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019, pp. 6105–6114.

[2] Jen-Hao Rick Chang, Ashish Shrivastava, Hema Swetha Koppula, Xiaoshuai Zhang, and Oncel Tuzel, "Style equalization: Unsupervised learning of controllable generative sequence models," *arXiv preprint arXiv:2110.02891*, 2021.

[3] Larry S. Yaeger, Brandyn J. Webb, and Richard F. Lyon, "Combining neural networks and context-driven search for on-line, printed handwriting recognition in the newton," *AI Magazine*, 1998.

[4] James A. Pittman, "Handwriting recognition: Tablet PC text input," *IEEE Computer*, vol. 40, no. 9, pp. 49–54, 2007.

[5] Daniel Keysers, Thomas Deselaers, Henry A Rowley, Li-Lun Wang, and Victor Carbune, "Multi-language online handwriting recognition," *PAMI*, vol. 39, no. 6, pp. 1180–1194, 2016.

[6] Stefan Jaeger, Stefan Manke, Jürgen Reichert, and Alex Waibel, "Online handwriting recognition: the NPen++ recognizer," *IJDAR*, vol. 3, no. 3, pp. 169–180, 2001.

[7] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006, pp. 369–376.

[8] Réjean Plamondon and Sargur N Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *PAMI*, vol. 22, no. 1, pp. 63–84, 2000.

[9] Victor Carbune, Pedro Gonnet, Thomas Deselaers, Henry A Rowley, Alexander Daryin, Marcos Calvo, Li-Lun Wang, Daniel Keysers, Sandro Feuz, and Philippe Gervais, "Fast multi-language LSTM-based online handwriting recognition," *IJDAR*, vol. 23, no. 2, pp. 89–102, 2020.

[10] A Graves, M Liwicko, H Bunke, J Schmidhuber, and S Ferandez, "Unconstrained on-line handwriting recognition with recurrent neural networks," *NeurIPS*, 2008.

[11] Volkmar Frinken and Seiichi Uchida, "Deep BLSTM neural networks for unconstrained continuous handwritten text recognition," in *ICDAR*, 2015, pp. 911–915.

[12] Yexun Zhang, Ya Zhang, and Wenbin Cai, "Separating style and content for generalized style transfer," in *CVPR*, 2018, pp. 8447–8455.

[13] Ankan Kumar Bhunia, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Fahad Shahbaz Khan, and Mubarak Shah, "Handwriting transformers," *arXiv preprint arXiv:2104.03964*, 2021.

[14] Alex Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[15] Emre Aksan, Fabrizio Pece, and Otmar Hilliges, "Deepwriting: Making digital ink editable via deep generative modeling," in *CHI*, 2018, pp. 1–14.

[16] Atsunobu Kotani, Stefanie Tellex, and James Tompkin, "Generating handwriting via decoupled style descriptors," in *ECCV*. Springer, 2020, pp. 764–780.

[17] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le, "Adversarial examples improve image recognition," in *CVPR*, 2020, pp. 819–828.

[18] Hadi Pouransari, Mojan Javaheripi, Vinay Sharma, and Oncel Tuzel, "Extracurricular learning: Knowledge transfer beyond empirical distribution," in *CVPR*, 2021, pp. 3032–3042.

[19] Aleksandr Laptev, Roman Korostik, Aleksey Svischev, Andrei Andrusenko, Ivan Medennikov, and Sergey Rybin, "You do not need more data: Improving end-to-end speech recognition by text-to-speech data augmentation," in *CISP-BMEI*, 2020, pp. 439–444.

[20] Jason Li, Ravi Gadde, Boris Ginsburg, and Vitaly Lavrukhin, "Training neural speech recognition systems with synthetic speech augmentation," *arXiv preprint arXiv:1811.00707*, 2018.

[21] Andrew Rosenberg, Yu Zhang, Bhuvana Ramabhadran, Ye Jia, Pedro Moreno, Yonghui Wu, and Zelin Wu, "Speech recognition with augmented synthesized speech," in *ASRU*, 2019, pp. 996–1002.

[22] Xianrui Zheng, Yulan Liu, Deniz Gunceler, and Daniel Willett, "Using synthetic audio to improve the recognition of out-of-vocabulary words in end-to-end asr systems," in *ICASSP*, 2021, pp. 5674–5678.

[23] Masato Mimura, Sei Ueno, Hirofumi Inaguma, Shinsuke Sakai, and Tatsuya Kawahara, "Leveraging sequence-to-sequence speech synthesis for enhancing acoustic-to-word speech recognition," in *SLT*, 2018, pp. 477–484.

[24] Joseph G Ibrahim, Hongtu Zhu, and Niansheng Tang, "Model selection criteria for missing-data problems using the em algorithm," *Journal of the American Statistical Association*, vol. 103, no. 484, pp. 1648–1658, 2008.

[25] Fernando Llorente, Luca Martino, David Delgado, and Javier Lopez-Santiago, "Marginal likelihood computation for model selection and hypothesis testing: an extensive review," *arXiv preprint arXiv:2005.08334*, 2020.

[26] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari, "How good is my GAN?," in *ECCV*, 2018, pp. 213–229.