Adversarial Generation of Handwritten Text Images Conditioned on Sequences

Eloi Alonso
A2iA SA, Paris, France
École des Ponts ParisTech

Bastien Moysset, Ronaldo Messina
A2iA SA, Paris, France

***Abstract*—We propose a system based on Generative Adversarial Networks (GAN) to produce synthetic images of handwritten words. We use bidirectional LSTM recurrent layers to get an embedding of the word to be rendered, and we feed it to the generator network. We also modify the standard GAN by adding an auxiliary network for text recognition. The system is then trained with a balanced combination of an adversarial loss and a CTC loss. Together, these extensions to GAN enable to control the textual content of the generated word images yielding realistic-looking images on both French and Arabic languages. State-of-the-art offline handwriting text recognition systems tend to use neural networks and therefore require a large amount of annotated data to be trained. In order to partially satisfy this requirement, we could use those synthetic images to increase the amount of training data. We show that integrating generated images into the existing training data of a text recognition system can slightly enhance its performance.**

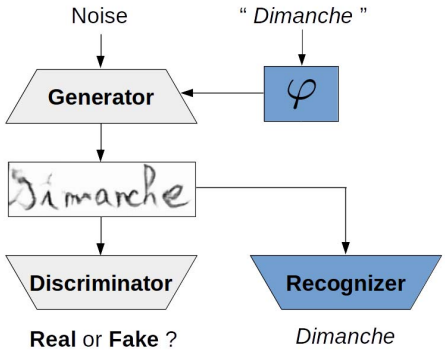


Figure 1. Adversarial generation of an image of text, conditioned on the textual content ("Dimanche"). The differences with a standard GAN are shown in blue.

I. Introduction

Rendering images of cursive handwritten text is an interesting problem that can have different applications. Recently, state-of-the-art solutions for cursive text recognition [1], [2] employ deep neural networks. The supervised training of these neural networks requires large amounts of annotated data; namely images of handwritten text with their corresponding transcripts. However, annotating images of text is a costly, time-consuming task. Automatic generation of cursive handwritten text images can reverse the annotation process: starting from a given word, we generate a corresponding image of cursive text.

We tackle here the challenge of generating realistic-looking data with arbitrary content, and then assess the use of such synthetic data to train neural networks in order to improve the performance of handwritten text recognition. Data augmentation techniques based on distortion and additive noise do not allow to enlarge the textual contents of the training data. Moreover, having control of the generated text enables the creation of training material that covers even rare sequences of characters, which can be expected to improve the recognition performance.

The problem of generating images of handwritten text has already been addressed in the past. Many techniques [3] are based on a collection of templates of a few characters, either human-written or built using Bezier curves. These templates are possibly perturbed and finally concatenated. However, this class of solutions, that simply concatenates character models, cannot faithfully reproduce the distribution of real-world images. It is also complex to have templates that are generic enough to result in truly cursive text. Alternatively, following the online approach, we can consider handwriting as a trajectory, typically recorded with pen computing. In this setting, the model aims at producing a sequence of pen positions to generate a given word. Graves et al. [4] use a Long Short-Term Memory (LSTM) [5], [6] recurrent neural network to predict such a sequence, and let the network condition its prediction on the target string to synthesize handwriting. However, this method does not allow to deal with some features useful for offline recognition, such as background texture or line thickness variations.

Generative Adversarial Networks (GAN) [7] offer a powerful framework for generative modeling. This architecture enables the generation of highly realistic and diverse images. The original GAN does not allow any control over the content of generated images, but many works [8]–[10] proposed a modified GAN for class-conditional image generation. However, we want to condition our generation on the sequence of characters to render, not on a single class. Closer to our goal, Reed et al. [11] conditions the generation on a textual description of the image to be produced. In addition to a random vector, their generator receives an embedding of the description text, and their discriminator is trained to classify as fake a real image with a non-matching description, to enforce the generator to produce description-matching images.

To the best of our knowledge, there is only one work [12]

2379-2140/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDAR.2019.00083

on a GAN for text image synthesis. While our generation process is directly conditioned on a sequence of characters, this method follows a style transfer approach, resorting to a CycleGAN [13] to render images of isolated handwritten Chinese characters from a printed font.

In this paper, we make the following contributions:

- An adversarial architecture, schematically represented in Fig. 1, to generate images of handwritten words, with arbitrary content.
  - The use of bidirectional LSTM recurrent layers to encode the sequence of characters to be produced.
  - Introduce an auxiliary network for text recognition, in order to control the textual content of the generated images.
- We obtain realistic-looking images on both French and Arabic datasets.
- Finally, we slightly improve text recognition performance on the RIMES dataset [14], using a neural network trained on a dataset extended with synthetic images.

II. Proposed adversarial model

We introduce here our adversarial model for handwritten word generation. Section II-A gives the general idea and defines the training objectives of the different parts. We detail the network architectures in Section II-B and describe our optimization settings in Section II-C.

A. Auxiliary Recognizer Generative Adversarial Networks

A standard GAN [7] comprises a generator (G) and a discriminator (D) network, shown in gray in Fig. 1. G maps a random noise $\boldsymbol{z}$ to a sample in the image space. D is trained to discriminate between real and generated (fake) images. Adversarially, G is trained to produce images that D fails to discriminate correctly. These networks hence have competing objectives.

In order to control the textual content of the generated images, we modify the standard GAN as follows. First, we use a recurrent network (φ) to encode $\boldsymbol{s}$, the sequence of characters to be rendered in an image. G takes this embedding $\varphi(\boldsymbol{s})$ as a second input. Then, in the vein of [9], the generator is asked to carry out a secondary task. To this end, we introduce an auxiliary network for text recognition (R). We then train G to produce images that R is able to recognize correctly, thereby completing its original adversarial objective with a "collaboration" constraint with R. We use the hinge version of the adversarial loss [15] and the CTC loss [16] to train this system. Formally, D, R, G and φ are trained to minimize the following objectives:

$$
\begin{aligned}
L_D &= -\,\mathbb{E}_{(\boldsymbol{x},\boldsymbol{s})\sim p_{data}}\Big[\min\big(0,-1+D(\boldsymbol{x})\big)\Big] \\
&\quad -\,\mathbb{E}_{\boldsymbol{z}\sim p_z,\boldsymbol{s}\sim p_w}\Big[\min\Big(0,-1-D\big(G(\boldsymbol{z},\varphi(\boldsymbol{s}))\big)\Big)\Big] \\
L_R &= +\,\mathbb{E}_{(\boldsymbol{x},\boldsymbol{s})\sim p_{data}}\Big[\mathrm{CTC}\big(\boldsymbol{s},R(\boldsymbol{x})\big)\Big] \\
L_{(G,\varphi)} &= -\,\mathbb{E}_{\boldsymbol{z}\sim p_z,\boldsymbol{s}\sim p_w}\Big[D\big(G(\boldsymbol{z},\varphi(\boldsymbol{s}))\big)\Big] \\
&\quad +\,\mathbb{E}_{\boldsymbol{z}\sim p_z,\boldsymbol{s}\sim p_w}\Big[\mathrm{CTC}\Big(\boldsymbol{s},R\big(G(\boldsymbol{z},\varphi(\boldsymbol{s}))\big)\Big)\Big]
\end{aligned}
$$

with p_{data} the joint distribution of real `[image, word]` pairs, p_z a prior distribution on input noise and p_w a prior distribution of words, potentially different from the word distribution of the real dataset.

B. Networks architecture

Fig. 2 and the text below describe the architecture of networks φ, G, D and R. The residual blocks (ResBlocks) we used are detailed in Fig. 3.

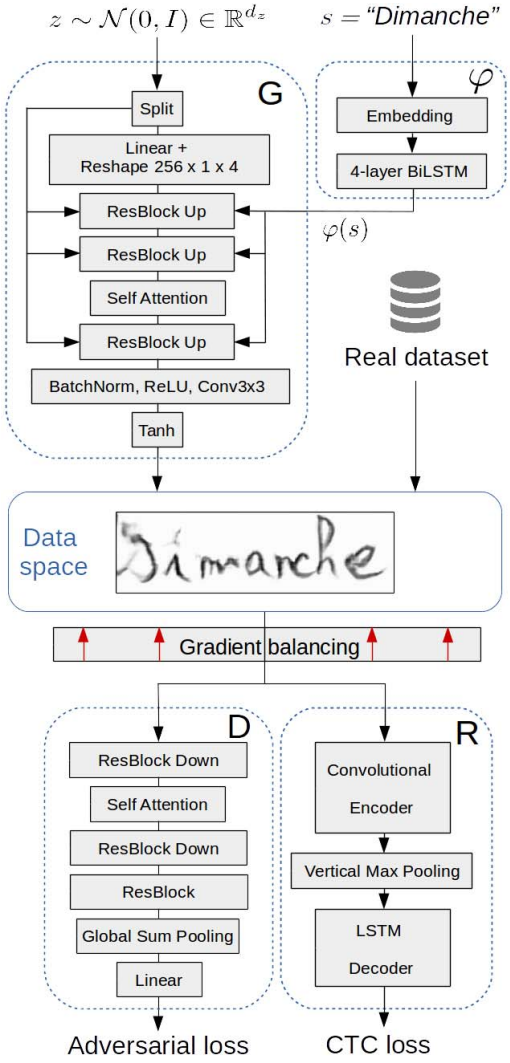


Figure 2. Architecture of the networks φ, G, D and R. For ease of reading, not all layers are represented (refer to the text for exact details). G receives a chunk of noise and $\varphi(\boldsymbol{s})$ in each ResBlock (details in Fig. 3). Both G and D include a self-attention layer. R follows the architecture of [1] and is trained with only real data using the CTC loss. We resort to the hinge version of the adversarial loss for D. When training G, we balance the gradients coming from D and R (details in Section II-C).

The network φ first embeds each character of the sequence $\boldsymbol{s}$ in $\mathbb{R}^{128}$, then encodes it with a four-layer bidirectional LSTM [5], [6] recurrent network (with a hidden state of size 128). $\varphi(\boldsymbol{s})$ is the output of the last bidirectional LSTM layer.

The network G is derived from [10]. The input noise, of dimension 128, is split into eight equal-sized chunks. The first one is passed to a fully connected layer of dimension 1024, whose output is reshaped to $256 \times 1 \times 4$ (with the convention $\text{depth} \times \text{height} \times \text{width}$). Each of the seven remaining chunks is concatenated with the embedding $\varphi(\boldsymbol{s})$, and fed to an up-sampling ResBlock through Conditional Batch Normalization (CBN) [17] layers (see Fig. 3). The consecutive ResBlocks have the following number of filters: 256, 128, 128, 64, 32, 16, 16. A self-attention layer [18] is used between the fourth and the fifth ResBlocks. We add a final convolutional layer and a tanh activation in order to obtain a $1 \times 128 \times 512$ image.

The network D is made up of seven down-sampling ResBlocks (with the following number of filters: 16, 16, 32, 64, 128, 128, 256), a self-attention layer between the third and the fourth ResBlocks, and a normal ResBlock (with 256 filters). We then sum the output along horizontal and vertical dimensions and project it on $\mathbb{R}$.

The auxiliary network R is a Gated Convolutional Network, introduced in [1] (we used the "big architecture"). This network consists in an encoder of five convolutional layers, with Tanh activations and convolutional gates, followed by a max pooling on the vertical dimension and a decoder made up of two stacked bidirectional LSTM layers.

C. Optimization settings

We used spectral normalization [22] in G and D, following recent works [10], [18], [22] that found that it stabilizes the training. We optimized our system with the Adam algorithm [23] (for all networks: $lr = 2 \times 10^{-4}, \beta_1 = 0, \beta_2 = 0.999$) and we used gradient clipping in D and R. We trained our model for several hundred thousand iterations with mini-batches of 64 images of the same type, either real or generated.

While D processes one real batch and one generated batch per training step, R is trained with real data only, to prevent it from learning how to recognize generated (and potentially false) images of text. To train the networks G and φ, we first produce a batch of "fake" images $\boldsymbol{x}_{\mathbf{fake}} := G(\boldsymbol{z}, \varphi(\boldsymbol{s}))$, and then pass it through D and R. (G, φ) learn from the gradients $\boldsymbol{\nabla_D} := -\frac{\partial D(\boldsymbol{x}_{\mathbf{fake}})}{\partial \boldsymbol{x}_{\mathbf{fake}}}$ and $\boldsymbol{\nabla_R} := \frac{\partial \text{CTC}(\boldsymbol{s}, R(\boldsymbol{x}_{\mathbf{fake}}))}{\partial \boldsymbol{x}_{\mathbf{fake}}}$ coming from these two networks. Since R and D have

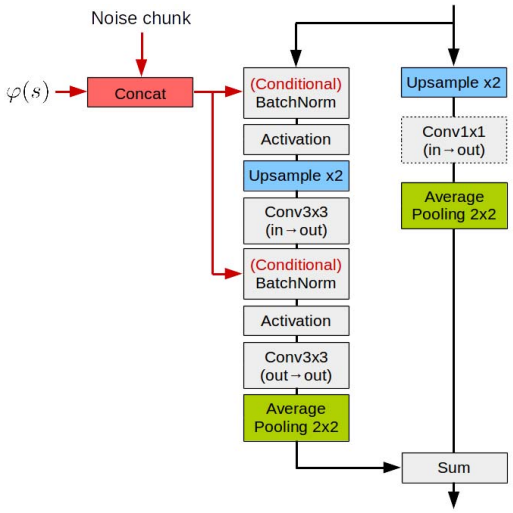


Figure 3. Detail of a ResBlock. The base components are shown in gray. In the ResBlocks of G, we concatenate a noise chunk with $\varphi(\boldsymbol{s})$ and feed it to CBN [17] layers (red). The unique hidden layer in CBN has 512 units. We also perform up-sampling (blue) with nearest neighbor interpolation. The ResBlocks of D resort to standard Batch Normalization [19] and operate down-sampling (green) with an average pooling. The activation is ReLU [20] in G and LeakyReLU [21] in D. The 1x1 convolution is only used when input (in) and output (out) numbers of channels are different. In the two 3x3 convolutions, the padding and stride are set to 1.

different architectures and losses, the norms of $\boldsymbol{\nabla_D}$ and $\boldsymbol{\nabla_R}$ can differ by several orders of magnitudes (we observed that $||\boldsymbol{\nabla_R}||_2$ is typically 10^2 to 10^3 times greater than $||\boldsymbol{\nabla_D}||_2$). To have (G, φ) learn from both D and R, we found it useful to balance the two gradients before propagating them to G. Therefore, we apply the following affine transformation to $\boldsymbol{\nabla_R}$:

$$\boldsymbol{\nabla_R} \leftarrow \alpha \times \left(\frac{\sigma_D}{\sigma_R}(\boldsymbol{\nabla_R} - \mu_R) + \mu_D\right)$$

With $\mu_\bullet$ and $\sigma_\bullet$ being the mean and the standard deviation of $\boldsymbol{\nabla_\bullet}, \bullet \in \{D, R\}$. α controls the relative importance of R with respect to D and is set to 1 in our model. The concrete impact of this transformation is discussed in Section III-B1.

III. Results

A. Experimental setup

In our experiments, we use 128×512 images of handwritten words obtained with the following preprocessing: we isometrically resize the images to a height of 128 pixels, then remove the images of width greater than 512 pixels and finally, pad them with white to obtain a width of 512 pixels for all the images (right-padding for French, left-padding for Arabic). Table I summarizes the meaningful characteristics of the two datasets we work with, namely the RIMES [14] and the OpenHaRT [24] datasets, while Fig. 4 shows some images from these two datasets.

To reflect the distribution found in natural language, the words to be generated are sampled from a large list of words

Table I: Characteristics of the subsets of RIMES and OpenHaRT.

Dataset	Language	Images	Words	Characters
RIMES	French	129414	6780	86
OpenHaRT	Arabic	710892	65575	77

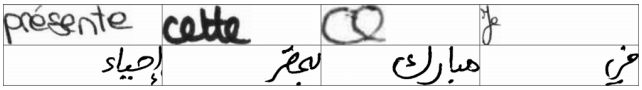


Figure 4. Images after preprocessing. First line: RIMES. Second line: OpenHaRT.

(French Wikipedia for French, OpenHaRT for Arabic). For the text recognition experiments on the RIMES dataset (Section III-D), we use a separate validation dataset of 6060 images.

We evaluate the performance with Fréchet Inception Distance (FID) [25] and Geometry Score (GS) [26]. FID is widely used and gives a distance between real and generated data. GS compares the topology of the underlying real and generated manifolds and provides a way to measure the mode collapse. For these two indicators, lower is better. In general, we observed that FID correlates with visual impression better than GS. For each experiment, we computed FID (with 25k real and 25k generated images) and GS (with 5k real and 5k generated images, 100 repetitions and default settings for the other parameters) every 10000 iterations and trained the system with different random seeds. We then chose independently the best FID and the best GS among the different runs. To verify the textual content, we relied on visual inspection. To measure the impact of data augmentation on the text recognition performance, we used Levenshtein distance at the character level (Edit Distance) and Word Error Rate.

B. Ablation study

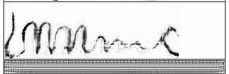
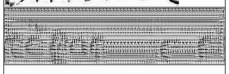
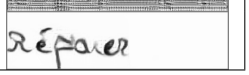
For all the experiments in this section, we used the RIMES database described in Section III-A.

1) Gradient balancing: When training the networks (G, φ), the norms of the gradients coming from D and R may differ by several orders of magnitudes. As mentioned in Section II, we found it useful to balance these two gradients. Table II reports FID, GS and a generated image for different gradient balancing settings.

Without gradient balancing, we observed that $||\boldsymbol{\nabla_R}||_2$ was typically 10^2 to 10^3 times greater than $||\boldsymbol{\nabla_D}||_2$, meaning that the learning signal for (G, φ) is biased toward satisfying R. As a result, the word "réparer" is clearly readable, but the FID is high (141.35) and the generated image is not realistic (the background is noisy, the letters are too far apart).

With $\alpha = 0.1$, $||\boldsymbol{\nabla_R}||_2$ is much smaller than $||\boldsymbol{\nabla_D}||_2$, meaning that G and φ take little account of the auxiliary recognition task. As illustrated by the second image in Table II, we lose control of the textual content of the generated image. FID is better than before, but still high (72.93).

Table II: FID, GS and a generated image of the word "réparer", for four settings: no gradient balancing, $\alpha = 0.1$, $\alpha = 1$ (our model) and $\alpha = 10$.

α	FID	GS	Images
None	141.35	2.44×10^{-3}	
0.1	72.93	4.23×10^{-2}	
10	222.47	2.92×10^{-3}	
1	**23.94**	$\mathbf{8.58 \times 10^{-4}}$	

In a way, the generated image is quite realistic, since the background is whiter and the writing more cursive.

On the contrary, when setting α to 10, G and φ mostly learn from the feedback of R and the generation is thus successfully conditioned on the textual content. In fact, we can distinguish the letters of "réparer" in the third generated image in Table II. However, as we are focusing on optimizing the generation process to have a minimal CTC cost, we observe strong visual artifacts that remind of the one obtained by Deep Dream generators [27]. FID is much higher (222.47) and the resulting images are very noisy, as demonstrated by the third image in Table II.

The best compromise corresponds to $\alpha = 1$. We obtain the best FID of 23.94 and GS of 8.58×10^{-4}, while the generated image is both readable and realistic. For all other experiments, we set α to 1.

2) Adversarial loss: Using the network architecture described in Section II, we test three different adversarial training procedures: the "vanilla" GAN [7] (GAN), the Least Squares GAN [28] (LSGAN) and the Geometric GAN [10], [15], [18], used in our model. FID and GS are reported in Table III.

Table III: FID and GS for different adversarial losses.

Adversarial Loss	FID	GS
GAN	36.32	5.29×10^{-3}
LSGAN	116.09	3.78×10^{-3}
Geometric GAN	**23.94**	$\mathbf{8.58 \times 10^{-4}}$

As shown in Table III, Geometric GAN leads to the best performance in terms of FID and GS. LSGAN fails to produce text-like outputs in three out of five trials. The low FID for vanilla GAN indicates that it produces realistic images. The high GS in Table III shows that both GAN and LSGAN suffer from a style collapse, and we observed that the textual content was not controlled. The trends given by FID and GS have been successfully confirmed by visual inspection of the generated samples.

3) Self-attention: We use a self-attention layer [18], in both the generator and the discriminator, as it may help to keep coherence across the full image. We trained our model with and without this module to measure its impact.

Table IV: Impact of self-attention.

	FID	GS
Without self-attention	67.86	4.51×10^{-3}
With self-attention	**23.94**	$\mathbf{8.58 \times 10^{-4}}$

Without self-attention, we still obtain realistic samples with correct textual content, but using self-attention improves performance both in terms of FID and GS, as shown in Table IV.

4) Conditional Batch Normalization: As described in Section II, G is provided a noise chunk and $\varphi(s)$ through each CBN layer. Another reasonable option, closer to [9], is to concatenate the whole noise $\boldsymbol{z}$ with $\varphi(\boldsymbol{s})$, and feed it to the first linear layer of G (in this scenario, CBN is replaced with standard Batch Normalization). Table V reports FID and GS for these two solutions.

Table V: Generator input via the first linear layer or via CBN layers.

	FID	GS
First linear layer	42.23	1.81×10^{-3}
CBN layers	**23.94**	$\mathbf{8.58 \times 10^{-4}}$

FID and GS in Table V indicates that feeding the generator inputs through CBN layers improves realism and reduces mode collapse. The visual inspection of the generated samples confirmed these trends and showed that the other solution prevents from correctly conditioning on the textual content.

C. Generation of handwritten text images

We trained the model detailed in Section II on the two datasets described in Section III-A, RIMES and OpenHaRT.

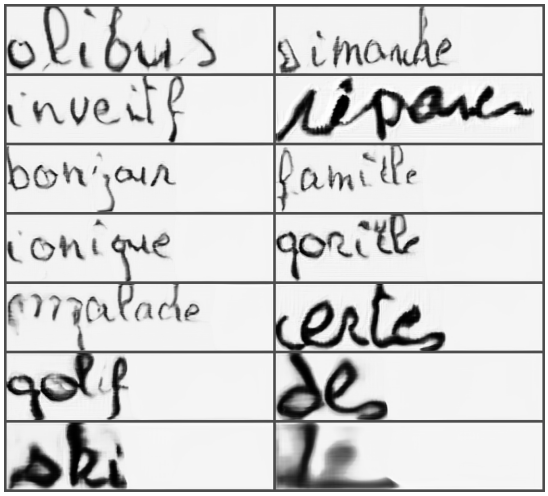


Figure 5. Images generated with our system trained on RIMES. Targets: *olibrius*, *Dimanche*, *inventif*, *réparer*, *bonjour*, *famille*, *ionique*, *gorille*, *malade*, *certes*, *golf*, *des*, *ski*, *le*.

Figure 6. Images generated with our system trained on OpenHaRT. Targets: كيان, نيجر, الحس, حبان, تبنت, اتخاذ, الردع, باشرت, حديثا, ترتكب, بشراء, ستثني.

Fig. 5 and Fig. 6 display some randomly generated (not cherry-picked) samples in French and Arabic respectively. For these two languages, we observe that our model is able to produce images of cursive handwriting, successfully conditioned on variable-length words (even if some words remain barely readable, e.g. *le* and *olibrius* in Fig. 5). The typography of the individual characters is varied, but we can detect a slight collapse of writing style among the images. For French, as we trained the generator to produce words from all Wikipedia, we are able to successfully synthesize words that are not present in the training dataset. In Fig. 5 for instance, the words *olibrius*, *inventif*, *ionique*, *gorille* and *ski* are not in RIMES, while *Dimanche*, *bonjour*, *malade* and *golf* appear in the corpus, but with a different case.

D. Data augmentation for handwritten text recognition

We aim at evaluating the benefits of generated data to train a model for handwritten text recognition. To this end, we trained from scratch a Gated Convolutional Network [1] (identical to the network R described in Section II-B) with the CTC loss, RMSprop optimizer [29] and a learning rate of 10^{-4}. We used the validation data described in III-A for early stopping.

Table VI: Extending the RIMES dataset with 100k generated images. Impact on the text recognition performance in terms of Edit Distance (ED) and Word Error Rate (WER) on the validation set.

Data	ED	WER
RIMES only	4.34	12.1
RIMES + 100k	4.03	11.9

Table VI shows that extending the RIMES dataset with data generated with our adversarial model brings a slight improvement in terms of Edit Distance and Word Error Rate. Note that using only GAN-made synthetic images for training the text recognition model does not yield competitive results.

IV. Conclusion

We presented an adversarial model to produce synthetic images of handwritten word images, conditioned on the sequence of characters to render. Beyond the classical use of a generator and a discriminator to create plausible images, we employ recurrent layers to embed the word to condition on, and add an auxiliary recognition network in order to generate an image with legible text. Another crucial component of our model lies in balancing the gradients coming from the discriminator and from the recognizer when training the generator. Realistic word images could be obtained in both French and Arabic, using the same architecture and parameters, only the training data changes for each language.

Some experiments were performed to assess if the generated images could improve the recognition of a state-of-the-art architecture on the RIMES database (in French). Differently from other state-of-the-art results, aiming at lowest error rate, we did not use language models, pre-training and other tricks of the trade. Our aim was to measure the improvement brought by adding synthetic data to the training set. The results showed a slight reduction in error rate for the French model trained on combined data. It can be argued that RIMES has a somewhat simple linguistic content, the out of vocabulary ratio of test data is about 3%, and the images have no or very litle background noise. Thus there is little room for improvements by adding more synthetic data with varied linguistic content.

An immediate continuation of our experiments would be to train the described model on more challenging datasets, with textured background for instance. In these conditions, we expect that synthetic data would provide more significant performance gains. Furthermore, deeper investigation to reduce the observed phenomenon of style collapse would be a substantial improvement. Some artifacts are still visible in the generated images; we expect that some modifications to the architecture of the generator would help in alleviating this issue. Another important line of work is to extend this system to the generation of line images of varying size, which is quite challenging due to the large dimension (along the text) of the resulting images.

References

[1] T. Bluche and R. Messina, "Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition," *ICDAR*, 2017.

[2] J. A. Sanchez, V. Romero, A. H. Toselli, M. Villegas, and E. Vidal, "ICDAR2017 Competition on Handwritten Text Recognition on the READ Dataset," *ICDAR*, 2017.

[3] R. I. Elanwar, "The state of the art in handwriting synthesis," *Int. Conf. on New Paradigms in Electronics & information Technology*, 2013.

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

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 1997.

[6] F. A. Gers, J. A. Schmidhuber, and F. A. Cummins, "Learning to Forget: Continual Prediction with LSTM," *Neural Computation*, 2000.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *NIPS*, 2014.

[8] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[9] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *ICML*, 2017.

[10] A. Brock, J. Donahue, and K. Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis," in *ICLR*, 2019.

[11] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative Adversarial Text to Image Synthesis," in *ICML*, 2016.

[12] B. Chang, Q. Zhang, S. Pan, and L. Meng, "Generating Handwritten Chinese Characters Using CycleGAN," in *IEEE Winter Conf. on Applications of Computer Vision*, 2018.

[13] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," in *ICCV*, 2017.

[14] E. Grosicki and H. El Abed, "ICDAR 2009 handwriting recognition competition," in *ICDAR*, 2009.

[15] J. H. Lim and J. C. Ye, "Geometric GAN," *arXiv preprint arXiv:1705.02894*, 2017.

[16] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Nets," *ICML*, 2006.

[17] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville, "Modulating early visual processing by language," in *NIPS*, 2017.

[18] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-Attention Generative Adversarial Networks," in *ICML*, 2019.

[19] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *ICML*, 2015.

[20] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *ICML*, 2010.

[21] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML*, 2013.

[22] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," in *ICLR*, 2018.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

[24] "NIST 2010 Open Handwriting Recognition and Translation Evaluation Plan." [Online]. Available: https://www.nist.gov/sites/default/files/documents/itl/iad/mig/OpenHaRT2010_EvalPlan_v2-8.pdf

[25] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *NIPS*, 2017.

[26] V. Khrulkov and I. Oseledets, "Geometry Score: A Method For Comparing Generative Adversarial Networks," in *ICML*, 2018.

[27] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," *Google Research Blog*, 2015.

[28] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *ICCV*, 2017.

[29] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, 2012.