Handwriting Styles: Benchmarks and Evaluation Metrics

Omar Mohammed, Gérard Bailly, Damien Pellier

Handwriting Styles: Benchmarks and Evaluation Metrics

Omar MOHAMMED[1,2], Gerard BAILLY[1], Damien PELLIER[2]
[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab*, 38000 Grenoble, France
[2] LIG - Laboratoire d'Informatique de Grenoble, 38000 Grenoble, France

***Abstract*—Extracting styles of handwriting is a challenging problem, since the style themselves are not well defined. It is a key component to develop systems with more personalized experiences for humans. In this paper, we propose baseline benchmarks, in order to set anchors to estimate the relative quality of different handwriting style methods. This will be done using deep learning techniques, which have shown remarkable results in different machine learning tasks, learning classification, regression, and most relevant to our work, generating temporal sequences. We discuss the challenges associated with evaluating our methods, which is related to evaluation of generative models in general. We then propose evaluation metrics, which we find relevant to this problem, and we discuss how we evaluate the performance metrics. In this study, we use IRON-OFF dataset [1]. To the best of our knowledge, no existing benchmarks or evaluation metrics for this task exit yet, and this dataset has not been used before in the context of handwriting synthesis.**

I. INTRODUCTION

The characterization and the extraction of human style profile, given some human activity (like speech, handwriting, human interactions, etc), is an open research problem. Usually, there is no clear definition of styles, making style extraction an ill-posed problem. In case of generative models, taking styles into account allows a more personalized output.

In this paper, we look at the problem in the case of handwritten letters. Ideally, given a letter from a writer, we would like to have information about the letter symbol and the factors characterizing the letter style (which, be default, characterize the writer). By doing so, we can: *i*) better study what constitutes the human profile in handwriting, and *ii*) produce more human-acceptable samples.

Our contributions in this paper are:

- We propose 2 evaluation metrics to measure the quality of handwriting synthesis.
- We compare 4 different methods that can capture the handwriting styles, and we propose two of them to be benchmarks for future work.
- We then propose a simple approach to ground our evaluation metrics, given a prior knowledge about the cardinal power of our previously proposed methods. We expect the metrics on those methods to match our prior knowledge.

*Institute of Engineering Univ. Grenoble Alpes
**Author email: omar-Samir.Mohammed AT grenoble-inp DOT fr

II. RELATED WORK

Our work is based on three areas of research, which we discuss in the following sections:

- Generative models for temporal data.
- Representation of the data (continuous vs discrete).
- Evaluation metrics for generated data.

A. Generative models

Some of the remarkable recent advances in deep learning [2] happened in the area of generative models. For generating static data, such as images, the work done using *Variational Autoencoders* [3] and *Generative Adversarial Networks* [4] has shown remarkable results.

In contrast, generating temporal data, such as caption for image caption, tracings of a letter, is more challenging: the data is generated sequentially, and it is difficult to keep the coherence for long sequences. Advances in recurrent neural networks architectures, like *Long-Short Term Memory* (LSTM) [5] and *Gated Recurrent Units* (GRU) [6], [7], showed impressive results on handling long term dependencies in temporal sequences.

These advances later allowed the development of state-of-the-art neural networks architectures for generating biased temporal sequences are showing impressive results: generating text and image captions, [8]–[11], music generation [12] and speech synthesis [13].

B. Data representation

The generation of continuous data has always been tricky. Bishop [14] studied the limitations of the traditional output activation functions used for continuous data, showing that such simple functions can not model rich distributions. He proposed *Mixture Density Networks* (MDN) activation, which is *Gaussian Mixture Model* (GMM) working as the output of a neural network. The network learns the parameters of this GMM. Graves [15] combined LSTM networks with MDN, to generate continuous handwritten characters, using *IAM Handwriting Database* [16]. While the results are impressive, the MDN approach are quite difficult to train. Another possible approach for generating continuous tracings is *Gaussian Scale Mixtures* (GSM) [17].

In order to simplify the procedure, and focus on our investigation of styles, we quantize the tracings using *Freeman codes* for direction, and speed - see Section III-B more details -, and apply the *SoftMax* function to the output of the last

layer, instead of MDN. This was inspired by the work done in [13], [18], where they show impressive results for data that is originally continuous, given a good quantization policy. Having a categorical distribution is more flexible and generic that a continuous distribution, and requires no assumption about the data distribution shape.

C. Evaluation criteria

Traditionally, the evaluation of these kind of applications is subjective. But with the advance of machine and statistical learning, there was a need to develop metrics that are cheap to evaluate, yet have a good correlation with the human evaluation. There has been a lot of advancement in developing performance metrics for image captioning and machine translation [19]. Metrics like *BLEU* [20], *METEOR* [21] and *CIREr* [22] are being used extensively in image captioning and machine translation evaluation. However, to the best of our knowledge, no such metrics exists for handwriting synthesis.

III. Dataset and Pre-Processing

A. Dataset

The dataset we choose is *IRON-OFF* Cursive Handwriting Dataset [1]. While there are other well-known handwriting datasets already available, like *IAM Handwriting Database* [16], our dataset provides us with separated and labeled letters, instead of entire sentence, thus allowing us to focus more on the problem of styles. A quick summary of this dataset is given below:

- 700 writers total. We use 412 writers, who have written isolated letters.
- 10,685 isolated lower case letters.
- 10,679 isolated upper case letters.
- 410 euro signs.
- 4,086 isolated digits.
- Gender, handiness, age and nationality of all writers.
- For each letter, we have letter image - with size around 167x214 pixels, and a resolution of 300 dpi -, pen movement timed sequence comprising continuous X, Y and pen pressure, and also discrete pen state. This data is sampled at 100 points per seconds on a Wacom UltraPad A4.

One particular challenge in this dataset is that each writer wrote each letter only once. Since we are focusing on the styles, this makes it particularly challenging for us. We do not use the pressure or the pen state, in order to simplify the model.

B. Pre-processing

All images have been de-noised and cropped in order to focus on the letters, then down-scaled to 28x28 pixels.

We cleaned the selected motion captured isolated letters by removing frames related to false starts or corrections, extra strokes as well as removing entire tracings with lengths exceed 1 second, in particular due to lengthy pen-up duration. All tracings exceeding 99 time steps has been discarded from the dataset as well.

All the letter tracings are represented as two modalities: freeman code and speed. Each modality is quantized into 16 level, and then represented as one-hot encoded vectors.

Freeman codes [23] belongs to a family of compression algorithms called Chain codes. These algorithms are useful to encode an image when it has connected components inside it. They are considered compression algorithms as they can transform a sparse matrix, to just a small fraction of the size of the image, in the form of a sequence of codes. Original Freeman codes have 2 versions, 4-directional codes, and 8-directional codes. Both are fairly simple as they encode each direction with a unique number (from 0 to n-1, where n are the directions). A direction is defined in the image as the directed vector connecting two neighbouring pixels on the contour of a connected component.

In our work, we compute the direction angle between each two consequent points. Then, we convert each direction to its corresponding freeman code symbol, as shown in figure 1. Then, we perform one-hot encoding on the direction, and feed it to our network. In order to have a faithful reconstruction of the letters, we also quantize the speed of each displacement.

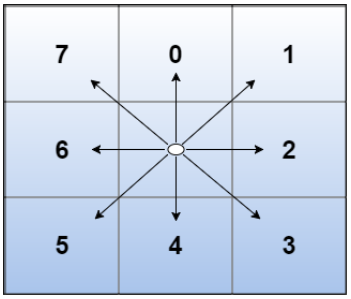


Fig. 1: Example for freeman code representation for 8 directions. Each direction will be given one number.

IV. Models

To build a model to generate handwriting traces, three steps are required:

- Hyper-parameters selection: like the number of layers, the learning rate, the size of each layer, etc.
- Training: including the training target for the model, and the loss function used.
- Inference: once the model is trained, we need to select a method to use it to generate the traces.

In the following subsections, we discuss these issues in detail.

A. Hyper-parameters selection

We ran random hyper-parameter tuning over a wide range of parameters, in order to select the final model. The selected model were a based GRU cell, with 3 hidden layers, each of size 256, and a dropout of 0.3. Adam optimizer [24] is selected, with a learning rate of $10^{-}3$. A *Multi-Layer Perceptron* (MLP) is applied to the output of the GRU at each time step, with an output size of 34. Two SoftMax operation are then applied, in order to extract the freeman code and the speed level.

The same approach was used to find the hyper-parameters for the models used to extract styles from the images (explained

in section V). However, we did this separately, after the hyper-parameters for the generator has been found, in order to reduce the final search space.

B. Training

We follow the similar approach to the work done in image captioning [11]. Each handwritten character is encoded as shown in figure 2a. An *End-of-Sequence* (EOS) symbol is added at the end of each modality (freeman code and speed) in each sequence (letter tracing). Padding is then performed to make the different sequences equal in their length.

The first time step represents the bias we use for the model, as seen in 2b. The bias is projected to the same dimension as the rest of the letter sequence. For example, if we use the letter identity only as bias (a one-hot encoded vector, with 26 dimensions), and dimensions of our sequence is 34 (16 + 1 for freeman codes + EOS, and another 16 + 1 for speed + EOS), then we use a MLP to convert the 26 dimensions into 34 dimensions, and the result is used as the first time step for the model. After that, the letter sequence (freeman codes and speed) is fed to the model. To formalize this:

$$hidden_0 = Zeros \tag{1a}$$
$$s_0 = MLP(I) \tag{1b}$$
$$hidden_{t+1} = GRU(s_t, hidden_t), t \in \{0...N\} \tag{1c}$$
$$p_{speed_{t+1}}, p_{freeman_{t+1}} = SoftMax(MLP(hidden_{t+1})) \tag{1d}$$
$$speed_{t+1} = argmax\ p_{speed_{t+1}} \tag{1e}$$
$$freeman_{t+1} = argmax\ p_{freeman_{t+1}} \tag{1f}$$

where $S = (s_1...s_t), t \in \{1...N\}$ is the input letter trace, where N is the trace length, and I is the model bias - the model bias, $hidden_{t+1}$ is the new hidden from the GRU, and $p_{modality_{t+1}}$ is the probability of different values for this modality. The final output is the $argmax$ of the probability of each modality.

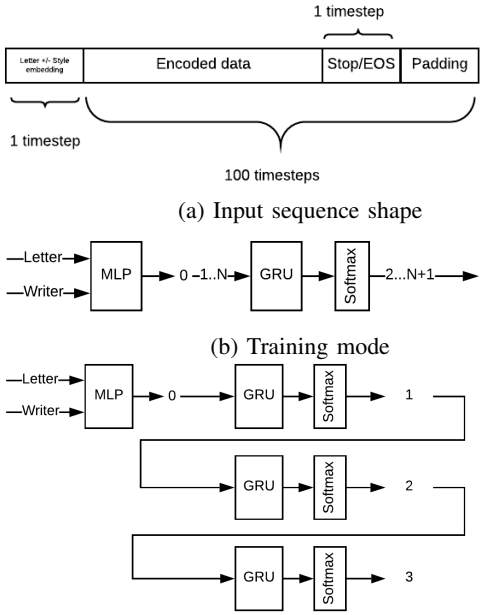
The loss used to optimized the network parameters is the negative log likelihood of the correct trace point at each time step, calculated as follows:

$$L(S, I) = -\sum_{t=1}^{N} \log p_{speed_t} + \log p_{freeman_t} \tag{2}$$

C. Inference

During inference, figure 2c, the first time step has the embedding information, used to bias the model. The network then generates the first frame. This frame is then feedback to the network's input for generating the second frame. This continues until an EOS symbol is generated.

In order to infer/generate the tracing of the letter, we use the *SoftMax Sampling* strategy: at each time step, we generate a two multinomial distributions: one for the directions, the other for the speed). At time step t, we sample both distributions according to a *temperature* level, and use these samples to feed the model's input for the next time step $t + 1$. This method is the one we use in this work. After testing multiple temperature values in the range between 0 (equivalent to an $argmax$ operation) and 1, we found that a temperature of 0.5 provides the best result.



(a) Input sequence shape

(b) Training mode

(c) Generation framework

Fig. 2: Illustration for biasing the generative model using letter + writer. The MLP, receives letter/writer embedding is responsible for down-/up-sizing the input dimension to the frame dimension of the tracings (34 i.e 16+1 hot encoding for direction and speed together with EOS feature). In this example, the model is biased using the letter and writer code.

V. Biasing the network with a style input

We assess multiple methods to bias our handwritten letter generator, and evaluate their ability to capture of writing styles. We know their cardinal order of the power of these methods (depends on the kind of information available to each method). Knowing this information beforehand, we can use it to ground our performance metrics. The methods are:

Letter bias the letter id only is used as bias. No style information is thus included. The model will try to average over the different example for the same letter. We consider this as a lower baseline.

Letter + Writer bias the letter id and writer id are used as a bias. Thus, the model has an explicit access information about the writer. This method is expected to perform the best. This model will also serve as a upper baseline.

Image classifier embedding We train a convolution neural network (CNN) to classify the letters images[1], as shown in figure 3. We use an intermediate layer as to extract embeddings, that will encode information about the letter images. This model should perform the same or a more

[1]This letter classification task achieves 95.1% classification accuracy, which we consider very good.

performance that the letter bias only, since it learns to clusters the letters, and there are classification errors. But we expect it to perform less than the letter + writer bias.

Image auto-encoder latent space we train a letter image autoencoder, using reconstruction error, and use the latent space as a representation of the letter + style bias. The architecture we use can be seen in figure 3. The latent space encodes the similarity between the letters. This model should perform worse than *Letter bias*, since, while it capture the similarity between the letter images, it does not capture discriminative features about each letter itself.

From this discussion, we can say that cardinal power of the different biases is:

$$autoencoder < letter \leq classifer < letter + writer \quad (3)$$

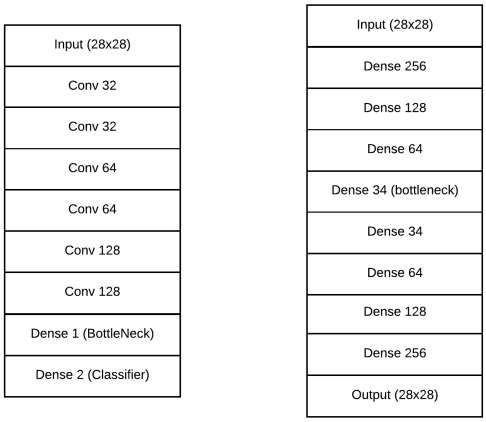


Fig. 3: Left: architecture of the CNN letter classifier. Batch normalization is used after each convolution layer. The *Dense 1* layer is the embedding that is used to bias our generator. Right: the autoencoder architecture we used. The first *Dense 34* layer provides the latent space used to bias the generator.

VI. Evaluation metrics

Evaluation, in generative models, is by far the most challenging part. Ideally, we want metrics to capture the distance between the generated and the reference distributions. In order to objectively compare the proposed style embeddings, we propose the following metrics:

BLEU score [20] It is a widely used metric to evaluate the quality of text generation areas, like in machine translation [9] and image captioning [11]. In this work, we test the hypothesis that the BLEU score is also relevant to the generation of handwriting[2]. The idea is that such metric can capture the similarity between the curves of the generated letter and the reference letter, in terms of segments of the curvature. In this study, we report the BLEU scores for 1, 2 and 3 grams, for the freeman codes and the speed separately. The higher number of grams, the larger the segments we compare. The final score is calculated as follows:

$$BLEU_N = \frac{\sum_{C \in G} \sum_{N \in C} Count_{Clipped}(N)}{\sum_{C \in G} \sum_{N \in C} Count(N)} \quad (4)$$

$$Score_N = \min\left(0, 1 - \frac{L_R}{L_G}\right) \prod_{n=1}^{N} BLEU_n \quad (5)$$

where: G is all the generated sequences, N is the total number of N-grams we want to consider. $Count_{Clipped}$ is clipped N-grams count (if the number of N-grams in the generate sequence is larger than the reference sequence, the count is limited to the number in the reference sequence only), L_R is the length of the reference sequence, L_G is the length of the generated sequence. The term $\min(0, 1 - \frac{L_R}{L_G})$ is added in order to penalize short generated sequences (shorter than the reference sequence), which will deceptively achieve high scores.

Generated Sequence Length Another aspect that we measure, is the relationship between the length of the generated sequence and the reference sequence. We use *Wilcoxon signed-rank test* [25] to compute the statistical significance between the distribution of the length of generated letters and the reference letters. We also calculate the *Pearson correlation coefficient*, in order to better quantify the relation between the generated and the reference sequences.

VII. Results and Discussion

We train our different model and generate the traces from them as explained earlier. In this section, we compare the different models using the evaluation metrics discussed before. We observe the consistency of the reported metrics with the prior information about the cardinal power of the different methods, equation 3. This is how we ground our metrics

A. BLEU score

The final results using the BLEU score can be seen in table I. The results vary when measuring BLEU-1. But, as we increase the number of grams, BLEU-2 and BLEU-3, to measure the similarity between larger segments of the traces, we can observe:

- The letter + writer bias performed better than all other biases, thus showing that having access to information about the writer, like the writer id, improve the quality of the handwriting synthesis.
- The image classifier bias performs better than the letter only bias, but less than the letter + writer bias. Since the classifier is trained on a single objective only (to classify the letters), and the classifier performs well, we expect the embedding to cluster the letters well, as seen in figure 4b. We can expect the model to capture some of the writer style, possibly in the inter-cluster variance. This is an interesting result, suggesting that some fine tuning for the image classifier while in the generation task could be beneficial to capture more details about the styles.

[2]In text evaluation, while the BLEU score is usually used when there is multiple reference sentences, there is no constraint on using it with one reference sentence only.

- The image autoencoder bias performed the worst. To understand why, we plot a 2-D projection of its latent space using t-SNE [26], figure 4a. Since the autoencoder is trained to minimize the reconstruction error, the distance in the latent space encode the proximity between the images. It can be observed also that this latent space does not encode discriminative features for the letters. Using this latent space for our generator, we find the model gets confused between nearby letters, resulting sometimes in generating different letters than requested.

Aspect/Feature	Speed			Freeman		
Model / B-score	B-1	B-2	B-3	B-1	B-2	B-3
Letter bias	49.7	37.3	24.2	47.4	36.6	26.8
Image classifier	50.9	38.2	24.6	48.5	37.9	28.1
Image autoencoder	51.9	37.9	23.1	46.4	35.0	24.5
Letter + Writer bias	51.5	41.4	25.1	56.7	39.4	28.3

TABLE I: Comparing different approaches for style extraction using clipped n-grams

B. Sequence length

As mentioned earlier, we performed a statistical test between the paired distributions of lengths of the generated and the reference tracings. The results are shown in table II. We can see the following:

- The results from the statistical test shows that the letter + writer bias outperform the rest of the biases, achieving p-value < 0.05. This is quite reassuring, since it is also in line with the results from the BLEU score.
- The results from the Pearson correlation coefficients are also consistent with the rest of the results. High coefficients are given to the letter + writer bias, compared to the other methods. The image classifier and autoencoder gives the lowest results. This could be due to insufficient information about the letter length that can be inferred from the image. For the image classifier, as noted earlier, a fine-tuning during the generation task is worth exploring.

Models	Pearson coefficient	p value
Letter bias	0.38	0.84
Image classifier	0.32	0.62
Image autoencoder	0.25	0.29
Letter + Writer bias	0.55	0.04

TABLE II: Pearson correlation coefficients and associated p-values for the EOS distributions of the different style biases.

VIII. Conclusions and future work

In this paper, we proposed and compared the following biases for the task of handwriting synthesis: letter, letter + writer, letter image classifier, and letter image bottleneck. In order to evaluate their quality, the we proposed two performance metrics: BLEU score (adapted from machine translation) and sequence length analysis. In order to ground our metrics, we leveraged our prior knowledge over the cardinal power of different biases: we observe that the results satisfy the prior information about the cardinal power of the different methods, equation 3, thus providing good evidence that the metrics are valid for this task.

This work presents an essential first step towards further study and analysis for styles in handwriting, enabling further techniques to be developed and compared.

Multiple points can be done in order to enhance our results, and to extend our study:

Extract styles from examples : The letter + writer bias has explicit access to the writer id, which we argue is the simplicity possible style information about the writer. The advantage is that it is quite simple, yet it does not have much information about the writer. For examples, for the *X* letter, some people draw it clockwise and some counter-clockwise. Some people start from the left side, and some started from the right side.

Style transfer : From our observation of the data, although there are 400 writers, there are some components for writing styles, like the ones mentioned in the letter *X* in the previous point (although it is not possible to enumerate them). One way to test the quality of a style extraction method is by performing a style transfer: leveraging the information from different writers to make a quick adaption to a new unseen writer. One interesting method we are investigating at the moment to extract the writer style is to adapt the method used in *FaceNet* [27], where they want to create an embedding for human faces. They introduced a loss function, *Triplet Loss*, which is generic enough to be used in other applications, like identifying the speaker turn [28]. Also, recent work has been performed in style transfer in the domain of speech synthesis [29], [30] for separating textual input from voice and expressivity shows promising results.

Task specific metrics : The proposed metrics in this paper are quite generic, allowing us to evaluate the system as a whole. Yet, a better understanding and analysis for the different systems requires more task-specific metrics. This is also in-line with the previous points, since it will give better insight on developing better methods for writer style extraction.

Acknowledgment

This work is supported by PERSYVAL (ANR-11-LABX-0025) via the project-action RHUM.

References

[1] C. Viard-Gaudin, P. M. Lallican, S. Knerr, and P. Binter, "The ireste on/off (ironoff) dual handwriting database," in *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, Sep 1999, pp. 455–458.

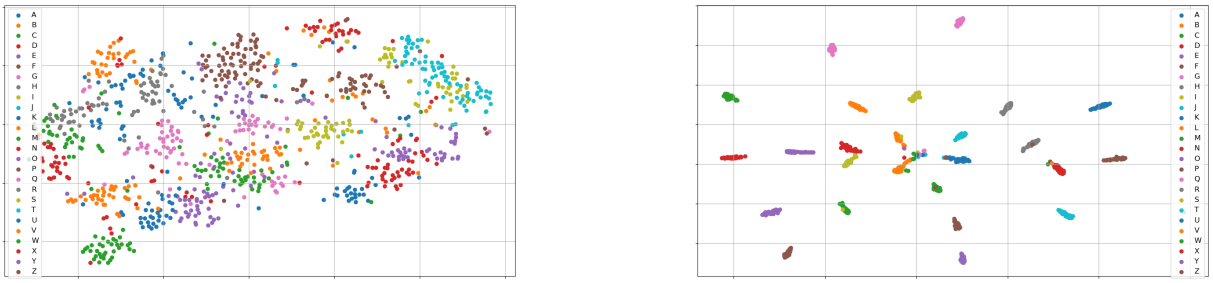
[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[3] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.



(a) (b)

Fig. 4: a) In the the autoencoder latent space, there is no clear separation between letters; the encoding is based on the similarity of the images only. b) In the classifier embedding, there is a clear separation between the letters - with few exceptions -.

[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[8] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 1017–1024.

[9] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, pp. 3104–3112. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969033.2969173

[10] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3128–3137.

[11] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015, pp. 3156–3164.

[12] J.-P. Briot and F. Pachet, "Music generation by deep learning-challenges and directions," *arXiv preprint arXiv:1712.04371*, 2017.

[13] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[14] C. M. Bishop, "Mixture density networks," 1994.

[15] A. Graves, "Generating sequences with recurrent neural networks," *CoRR*, vol. abs/1308.0850, 2013. [Online]. Available: http://arxiv.org/abs/1308.0850

[16] U.-V. Marti and H. Bunke, "A full english sentence database for off-line handwriting recognition," in *Document Analysis and Recognition, 1999. ICDAR'99. Proceedings of the Fifth International Conference on*. IEEE, 1999, pp. 705–708.

[17] L. Theis and M. Bethge, "Generative image modeling using spatial lstms," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 1927–1935. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969442.2969455

[18] A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, pp. 1747–1756. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045390.3045575

[19] P. Koehn, *Statistical Machine Translation*, 1st ed. New York, NY, USA: Cambridge University Press, 2010.

[20] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.

[21] S. Banerjee and A. Lavie, "Meteor: An automatic metric for mt evaluation with improved correlation with human judgments," in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 2005, pp. 65–72.

[22] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4566–4575.

[23] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Transactions on Electronic Computers*, vol. 2, pp. 260–268, 1961.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945. [Online]. Available: http://www.jstor.org/stable/3001968

[26] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[27] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *CoRR*, vol. abs/1503.03832, 2015. [Online]. Available: http://arxiv.org/abs/1503.03832

[28] H. Bredin, "Tristounet: Triplet loss for speaker turn embedding," *CoRR*, vol. abs/1609.04301, 2016. [Online]. Available: http://arxiv.org/abs/1609.04301

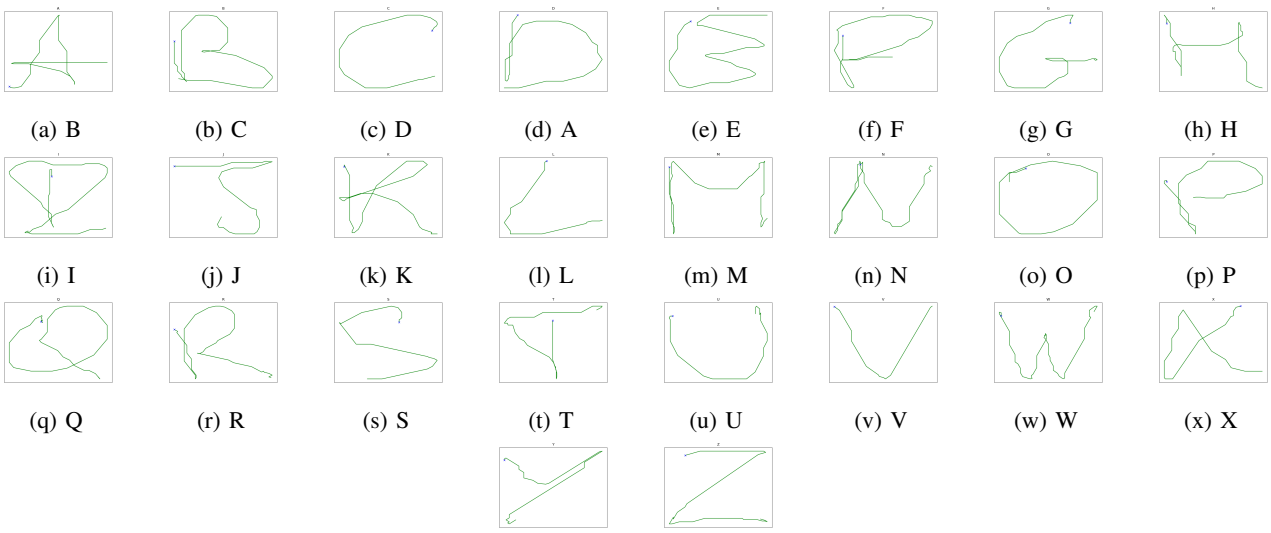
[29] R. J. Skerry-Ryan, E. Battenberg, Y. Xiao, Y. Wang, D. Stanton, J. Shor, R. J. Weiss, R. Clark, and R. A. Saurous, "Towards end-to-end prosody transfer for expressive speech synthesis with tacotron," *CoRR*, vol. abs/1803.09047, 2018. [Online]. Available: http://arxiv.org/abs/1803.09047

[30] Y. Wang, D. Stanton, Y. Zhang, R. J. Skerry-Ryan, E. Battenberg, J. Shor, Y. Xiao, F. Ren, Y. Jia, and R. A. Saurous, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," *CoRR*, vol. abs/1803.09017, 2018. [Online]. Available: http://arxiv.org/abs/1803.09017

APPENDIX

EXAMPLE OF THE LETTERS

The design choices of our experiments (discretization, and ignoring the pen state) affects the final shape of the letters, yet, the letters and their style are quite recognizable. See examples for the original letters in figure 5. Examples for the generation with our methods are in figure 6.



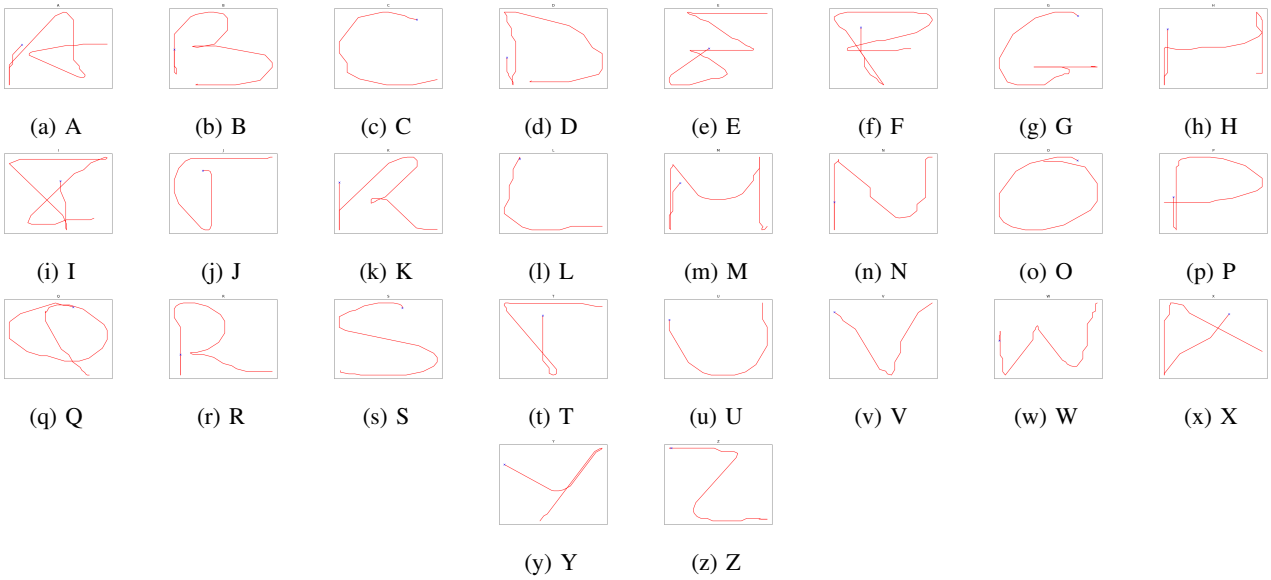
(a) B (b) C (c) D (d) A (e) E (f) F (g) G (h) H

(i) I (j) J (k) K (l) L (m) M (n) N (o) O (p) P

(q) Q (r) R (s) S (t) T (u) U (v) V (w) W (x) X

(y) Y (z) Z

Fig. 5: Examples of original letters. The blue x mark is the starting point. These ones are generated using the letter + Writer bias. E and F are visually harder to recognize, since we do not model the pen pressure, otherwise, the rest of the letters are well recognizable.



(a) A (b) B (c) C (d) D (e) E (f) F (g) G (h) H

(i) I (j) J (k) K (l) L (m) M (n) N (o) O (p) P

(q) Q (r) R (s) S (t) T (u) U (v) V (w) W (x) X

(y) Y (z) Z

Fig. 6: Examples of generated letters. The blue x mark is the starting point. These ones are generated using the letter + Writer bias. The general quality of this quite acceptable.