HAL
open science

Style Transfer and Extraction for the Handwritten Letters Using Deep Learning

Omar Mohammed, Gérard Bailly, Damien Pellier

To cite this version:

Omar Mohammed, Gérard Bailly, Damien Pellier. Style Transfer and Extraction for the Handwritten Letters Using Deep Learning. ICAART 2019 - 11th International Conference on Agents and Artificial Intelligence, Feb 2019, Prague, Czech Republic. hal-02049006

HAL Id: hal-02049006

https://hal.science/hal-02049006

Submitted on 26 Feb 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Style Transfer and Extraction for the Handwritten Letters Using Deep Learning

Omar MOHAMMED[1,2], Gérard BAILLY[1], Damien PELLIER[2]
[1]*Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, 38000 Grenoble, France*
[2]*Univ. Grenoble Alpes, CNRS, LIG, 38000 Grenoble, France*
omar-samir.mohammed@grenoble-inp.fr

Keywords: Generative models, Deep Learning, Online Handwriting, Style Extraction

Abstract: How can we learn, transfer and extract handwriting styles using deep neural networks? This paper explores these questions using a *deep conditioned autoencoder* on the IRON-OFF handwriting data-set. We perform three experiments that systematically explore the quality of our style extraction procedure. First, We compare our model to handwriting benchmarks using multidimensional performance metrics. Second, we explore the quality of style transfer, i.e. how the model performs on new, unseen writers. In both experiments, we improve the metrics of state of the art methods by a large margin. Lastly, we analyze the latent space of our model, and we see that it separates consistently writing styles.

1 Introduction

One aspect of a successful human-machine interface (e.g. human-robot interaction, chatbots, speech, handwriting . . .) is the ability to have a personalized interaction. This affects the overall human experience, and allow for a more fluent interaction. At the moment, there is a lot of work that uses machine learning in order to learn to model for such interactions. However, most of these models do not address the issue of personalized behavior: they try to average over the different examples from different people in the training set. Identifying the human styles during the training and inference time open the possibility of biasing the models output to take into account the human preference. In this paper, we focus the problem of styles in the context of handwriting.

However, defining and extracting handwriting styles is a challenging problem, since there is no formal definition for these styles (i.e. it is an ill-posed problem). A style is both social – depends on writer's training, especially at middle school – and idiosyncratic – depends on the writer's shaping (letter roundness, sharpness, size, slope . . .) and force distribution across time. To add to the problem, till recently, there were no metrics to assess the quality of handwriting generation.

There are two questions: what is the task itself? and what is the style used to achieve this task?. In handwriting, the task space is well defined (i.e. which letter we want to write), thus, allowing us to focus on the second part, of extracting styles for achieving this task.

In this paper, we address the problem of style extraction by using an conditioned-temporal deep autoencoder model. The conditioning is on the letter identity. The reason we use an autoencoder is that there is no explicit way that we know about to evaluate the quality of the handwriting styles other than using them to generate handwriting, and evaluate this generation. (Mohammed et al., 2018) introduced benchmarks and evaluation metrics in order to assess the quality of generating handwritten letters. In comparison to the those benchmarks and metrics, we achieve higher performance, while extracting a meaningful latent space.

We also hypothesize that the latent space of styles is generic, i.e. that it will generalize over unseen writers, thus achieving a "transfer of style". To test this hypothesis, we assess our model on 30 new writers. We compare the tracings generated by this model to a benchmark model already proposed for online handwriting generation.

In addition, we explore the latent space of our model for each letter separately. This revealed that there is a limited number of 'unique' styles per letter, categorical as well as continuous. We report our analysis for some of the letters, since a full analysis is out of the scope for this paper.

Thus, our contributions in this paper are the following:

- We test and compare our deep conditioned autoencoder with the state of the art benchmarks. We show that this model greatly improves the generation performance over a state of the art benchmark model.
- We experiment on performing style transfer on new writers using this model achieves, and we show that it achieves much better results than the benchmark model.
- Finally, and maybe most interestingly, we further analyze the extracted the latent space from our model to show that there is a limited number of styles for each letter and that the style manifold is not a continuous space.

2 Related work

2.1 Generative models

Recent advances in deep learning (Goodfellow et al., 2016) architectures and optimization methods led to remarkable results in the area of generative models. For static data, like images, the mainstream research builds on the advances in *Variational Autoencoders* (Kingma and Welling, 2013) and *Generative Adversarial Networks* (Goodfellow et al., 2014).

For generating sequences, the problem is more difficult: the model generates one frame at a time, and the final result must be coherent over long sequences. Recent recurrent neural networks architectures, like *Long-Short Term Memory* (LSTM) (Hochreiter and Schmidhuber, 1997) and *Gated Recurrent Units* (GRU) (Chung et al., 2014), achieve unprecedented performance in handling long sequences.

Theses architectures has been used in many applications, like learning language models (Sutskever et al., 2014), image captioning (Vinyals et al., 2015), music generation (Briot and Pachet, 2017) and speech synthesis (Oord et al., 2016).

Focus was dedicated to use these powerful tool in order to extract meaningful latent space. One such work that inspired the investigation in this paper is (Ha and Eck, 2017). In their work, they investigated the problem of sketch drawing (Google, 2017) using a Variational Autoencoder. The latent space emerged encoded meaningful semantic information about these drawings. In our work, we simple a similar architecture, without the variational part, showing that similar behaviour.

2.2 Data Representation

For handwriting, a continuous coordinate representation (e.g. continuous X, Y) seems the natural option. However, generating continuous data is not straightforward. Traditionally, in neural networks, when we want to output a continuous value, a simple linear or *Tanh* activation function is used in the output layer of the neural network.

However, Bishop (Bishop, 1994) studied the limitations of these functions and showed that they can not model rich distributions. In particular, when the input can have multiple outputs (one-to-many), these functions will average over all the outputs. He proposed the use of *Gaussian Mixture Model* (GMM) as the final activation function of a neural network. The alliance of neural networks and GMMs is called *Mixture Density Network* (MDN). The training consists in optimizing the GMM parameters (means, covariances). The inference is done by sampling from the GMM distribution.

To simplify the process, and focus our study on investigating of styles, we extract two features for the tracings: directions and speed (explained in section 3), and we quantize these features. Thus, we can model each point in the letter tracings as a categorical distribution, and use a simple *SoftMax* function as the output of the network, which is much simpler than MDN. This was inspired by the studies done in (Oord et al., 2016), where they report impressive results on originally continuous data, using suitable quantization policy. A categorical distribution is more flexible and generic than continuous ones.

2.3 Evaluation metrics

The objective evaluation of a generative model is a challenging task, since there is no consensus for objective evaluation metrics. In many cases, a subjective evaluation is performed to overcome this problem. For handwriting of Chinese letters, (Chang et al., 2018) proposed two metrics: *Content accuracy* and *Style discrepancy*. In the first metric, a classifier is trained to determine the type of the letter on the reference letters, then it is used to evaluate the generated letters. However, it is not clear how to reliably use the classifier trained on one distribution (reference letters) to evaluate new distribution (the generated letters). The second metric is not applicable to our case, since it assumes the use of *Convolution Neural Network* (CNN) on the image of the letter, while we us the pen sequence of drawing the letter (i.e., temporal data) with RNNs.

(Mohammed et al., 2018) also addressed the problem of evaluation of handwriting generation. They

used the *BLEU score* (Papineni et al., 2002) (a metric widely used in text translation and image captioning) and the *End of Sequence* (EoS) analysis (both metrics are explained in section 5). They showed that these metrics correlate with the quality of the generated letter. We use these metric in our experiments.

3 Dataset

In this study, we use the *IRON-OFF* Cursive Handwriting Dataset (Viard-Gaudin et al., 1999), which contains isolated handwritten letters. To summarize this dataset:

- Around 700 writers in total. We use the 412 writers who have written isolated letters.
- 10,685 isolated lower case letters, 10,679 isolated upper case letters, 4,086 isolated digits and 410 euro signs.
- The gender, handiness (left or right handed), age and nationality of the writers.
- For each example (letter, digit, euro sign), we have that example's 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.

We focused on the uppercase letters only, and we did not use the pen state or the pen pressure. The idea was to limit number the possible style factors, so that we can better study them. 90% of the data is used for training, and 10% for validation.

One challenging issue with this dataset however is that we have only one example for each writer-letter combination. This makes the task more difficult, because it is hard to extract a writer style using very few items (the 26 letters/writer in this case).

We represent each letter tracing by two features: directions and speed of the pen between each two consecutive points. Each feature is quantized into 16 levels and represented as a one-hot encoded vector. Freeman codes (Freeman, 1961) is used in order to encode the direction feature. It belongs to a family of compression algorithms called *Chain Codes*. We can use N freeman codes (where N are the number of directions), depending on the needed resolution.

4 Model architecture

The model architecture is illustrated in figure 1. The trace of the letter is first fed to encoder module. The final hidden state of that module summarizes the letter. In order to allow this module to focus on learning the style embedding, we complement this last hidden state with the one-hot encoding of the letter identity, and use a projection of them as the bias input to the generator. Thus, we decouple the *task space* – the letter – from the *style space*: the encoder is free from the need to learn the letter identity, and can focus learning additional information that enables the generator to better approximate the ground truth tracings.

In the decoder, we follow the framework proposed by (Vinyals et al., 2015) in order to bias the model: we create an extra time step at the beginning, which has the information we want to bias the model with. In this case, this time step is the projection of the encoder last hidden state and the letter encoder. This has a much lower dimension than encoder hidden state (the hyperparameters are discussed in section 4.1). This further encourage the model to learn only necessary style information, as suggested in (Skerry-Ryan et al., 2018).

4.1 Hyper-parameter tuning

We ran random hyper-parameter search for a wide range of parameters (learning rate, size and the number of layers for the encoder and the decoder, dropout percentage, etc). GRU layers (Chung et al., 2014) is being used in this model. We use *Adam* (Kingma and Ba, 2014) optimizer. In order to allow for faster exploration of different hyper-parameters, we use an early stopping of 20 epochs (no improvement happens during these epochs).

4.2 Training

The encoder and the decoder parts have the target of modeling the next time step in the sequence, x_{t+1}, given the previous time steps, or in other words, $P(x_{t+1}|x_1, x_2, ..., x_T)$, where x_t is the tracing point at time t, and T is the length of the input sequence. To achieve this, the model is given the ground truth input of points $x1, x2, ..., x_{T-1}$ and is asked to output the sequence $x2, x3, ..., x_T$.

The model is trained to minimize the negative log likelihood loss of the correct point at each time step. For each feature (speed and freeman codes), it is calculated as in equation 1. The final loss is the average loss of the two feature, as in equation 2.

$$\begin{aligned} Loss &= -\log \prod_{t=1}^{T} p(x_t|x_1, x_2, ..., x_{t-1}) \\ &= -\sum_{t=1}^{T} \log p(x_t|x_1, x_2, ..., x_{t-1}) \end{aligned} \tag{1}$$

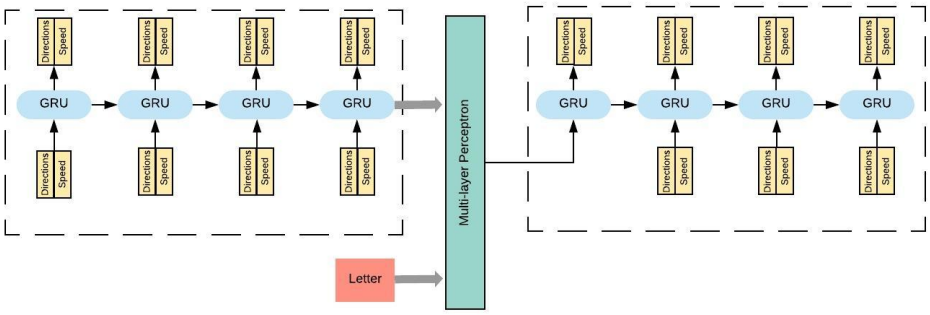

Figure 1: Schematic diagram of the model we used. Both the encoder and the decoder have 2 layers, with size of 128. A dropout of 0.2 is used for the decoder. Learning rate selected is 0.001. During the training time, the input to the model is always the ground truth. During the inference time however, the input to the decoder (generator) part at each time step is its own predication in the previous time step.

$$TotalLoss = (Loss_{speed} + Loss_{freeman})/2.0 \quad (2)$$

During the training, the output of the model at each time step is the:

$$x^g_{t+1} = argmax_x p(x|x_t, h_t) \quad (3)$$

where x^g_{t+1} is the generated/predicted next time step by the model, x_t is the ground truth input at the current time step t, and h_t is the hidden state of the GRU at the current time step. To sample from the model, we used the *Temperature Sampling* strategy from he *Softmax* output.

5 Evaluation metrics

Evaluation is a challenging problem when using generative models. We want metrics to capture the distance between the generated and the ground truth distributions. Similar to the work done in (Mohammed et al., 2018), we use the same two evaluation metrics in our model:

- **BLEU score** (Papineni et al., 2002) It is a well known metric to evaluate text generation applications, like image captioning (Vinyals et al., 2015) and machine translation (Sutskever et al., 2014). Since we discretized the letter drawings, this fits nicely within our work. The general intuition is the following: if we take a segment from the generated letter, did this segment happen in the ground truth letter? We keep doing this for segments of increasing length (the length of the segment here is the number of grams used in the BLEU score). For our work, we report the results on segments from 1 to 3 time steps. Each part of the letter has two parallel segments: freeman codes and speed, thus, we report the BLEU score for both of them. The equation to compute the BLEU score is the following:

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

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

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.

- **End of Sequence (EoS)** The length letter is another aspect of the style. The distribution of length in the generated examples should follow the ground truth examples. In order to perform this analysis, we compute *Pearson correlation coefficient* between the generated examples and the ground truth data.

6 Experiments and results

6.1 Letter generation with style preservation

The objective here to compare the quality of the generated letters to the state-of-the-art benchmarks. As

mentioned earlier, we compare using the BLEU score metric and the EoS analysis. The BLEU score results can be seen in table 1, and the results for EoS analysis results are in table 3. We can see that the BLEU-3 score results of our model achieves 32.3% accuracy in Speed feature and 38.7% accuracy in Freeman feature, compared to 25.1% and 28.3% accuracy using the benchmark model on both features respectively.

The same goes for the EoS analysis. In comparing the Person Coefficient, our model achieves 0.99 score compared to 0.55 for the benchmark model (the highest score is 1.0). This is a support that our model capture the style of handwriting better than the benchmark.

Examples for the generated letters can be found in figure 11.

6.2 Style transfer across writers

One of the hypotheses we want to test is whether there is a limited number of styles needed, to generalize over new writers. To achieve this, the learned representation for styles should extract generic information about the styles.

In order to test this hypothesis, we expose our model to 30 writers that have not been seen before. We compare our model performance on these writers with a model is biased by the writer and letter identities (the benchmark model). The latter model was not constrained from seeing those writers (thus, the reported results of the comparison overestimates the actual performance of that model).

The BLEU scores can be seen in table 2. Our model achieves on BLEU-3 score 32.2% and 42.1% accuracy on the Speed and Freeman code features, compared to 25.3% and 27.7% on the benchmark model for the same features respectively.

The EoS analysis can be seen in table 4. Our model achieves a coefficient value of 0.99, compared to 0.5 for the benchmark. Thus, the new model clearly outperform the current benchmarks on the transfer task, on both BLEU score and EoS analysis.

6.3 Styles per letters

One of the nice consequences of using our model is that we can have a better look at the styles. We explore the latent space for multiple letters, and see that we can uncover interesting writing styles. A full scale analysis is beyond the scope of this paper. We project the latent space using *Principal Components Analysis* (PCA) (Jolliffe, 2011) and t-SNE (Maaten and Hinton, 2008).

As a start, we take a look at letter X. Beforehand, we identified a style feature in letter X: some writer draw X clockwise, and some draw it anti-clockwise. We manually annotated the whole dataset for this feature; the result can be seen in figure 2. Almost half of the writers draw the letter X clockwise, and the other half draw it anti-clockwise. If our assumption is correct, our model should be able to capture this feature. We project the latent of the model using PCA on all the letter X, which can be seen in figure 3. The model latent space clusters almost perfectly based on rotation. Examples for letters from both clusters are in figure 4.

Encouraged by the results on letter X, we explored more letters. For letter C, we can see the latent space project in figure 5. It can be seen that there are at least two main clusters. Examples from this cluster in the red ellipse are in figure 7. The indicated cluster represents the Edwardian handwriting style. The rest of the writers (in the big cluster) have a very similar style (this is expected, since the drawing of the letter C is quite simple).

For letter A, our model latent space create two main clusters, figure 6. We give examples from those two in figure 8, where we can see clear difference in the style. Some people start drawing the letter from down-left, other writers start from the top of letter A, move down, then continue drawing of the letter.

Another example is for letter S bottleneck, figure 9. There are three resulting clusters which we investigated. The indicated cluster (in red) is clearly different from the other two clusters (not indicated). Examples can be seen in figure 10. The indicated cluster is again for people with Edwardian handwriting style. We did not find a clear difference between the other two clusters though, but this is an expected outcome of using t-SNE (since it does not have the clear objective of clustering styles).

These examples show is that we can use our model to extract verbose style information.

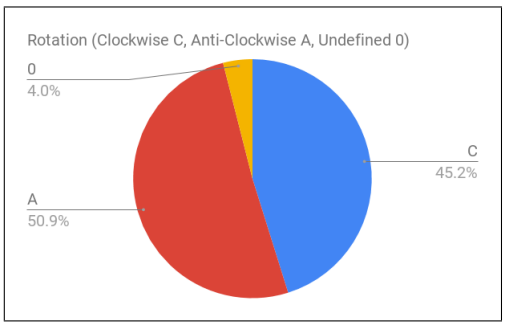

Figure 2: Results of the manual annotation for the rotation of letter X drawings over the whole dataset. Almost half the writers drew X clockwise, the other half anti-clockwise. The undefined styles were unclear to determine.

Aspect/Feature	Speed			Freeman		
Model / B-score	B-1	B-2	B-3	B-1	B-2	B-3
Letter + Writer bias	51.5	41.4	25.1	56.7	39.4	28.3
Style Extractor	71	51.7	32.3	65.6	51.5	38.7

Table 1: BLEU scores for different models for known writers.

Aspect/Feature	Speed			Freeman		
Model / B-score	B-1	B-2	B-3	B-1	B-2	B-3
Letter + Writer bias	55.4	39.6	25.3	50.2	38.6	27.7
Style Extractor	72.4	52.4	32.2	70.4	55.6	42.1

Table 2: BLEU scores for different models for style extraction for 30 new writers (style transfer).

Models	Pearson coefficient
Letter + Writer bias	0.55
Style Extractor	0.99

Table 3: Pearson correlation coefficients for the End-Of-Sequence (EoS) distributions for the different models on the normal generation scenario

Models	Pearson coefficient
Letter + Writer bias	0.50
Style Extractor	0.93

Table 4: Pearson correlation coefficients for the End-Of-Sequence (EoS) distributions for the different models on 30 new writers (style transfer).

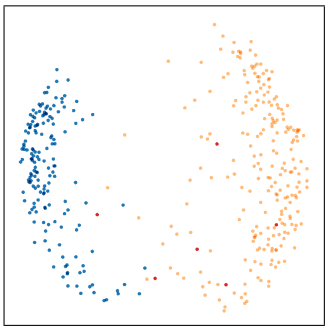

Figure 3: Projection for latent space for letter X using PCA. The colors show the ground truth of the X rotation: blue is counter clockwise, orange is clockwise, and the few red points are undefined.

7 Conclusions and future work

In this paper, we explored the concepts of styles of handwriting, using a deep neural network paradigm. We have approached the problem systematically. First, we compared our generation results to the benchmark reported in the state-of-the-art on this problem, and we show that our model outperforms the benchmark. Second, we explore the ability to perform style transfer, by testing the model's performance on 30 new writers. We hypothesize that there is a limited number of style components that describe handwriting, and a

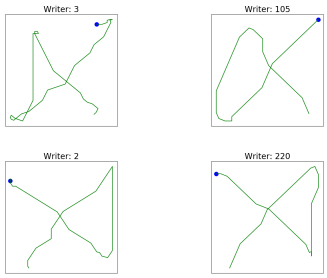

Figure 4: Examples for writing of letter X. Starting point is marked with the blue mark. Each raw is randomly sampled from each cluster in the bottleneck. The clusters shows that almost half the writers draw the letter clockwise (first row, first cluster), and the other half draw it anti-clockwise (second row, second cluster).

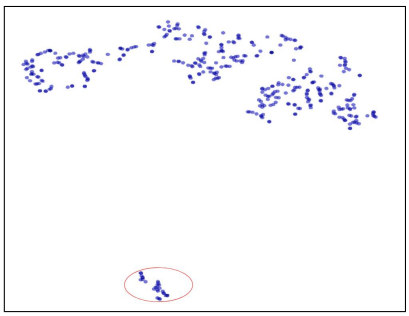

Figure 5: Projection for latent space for letter C using t-SNE. The cluster surrounded by the red circle has a clear interpretation, where writers have a cursive style.

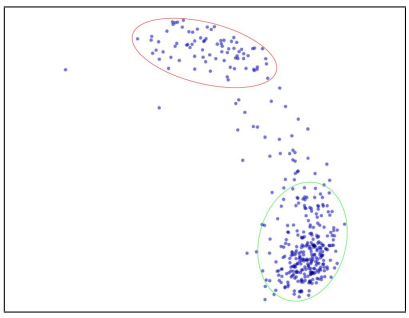

Figure 6: Projection for latent space for letter A using PCA.

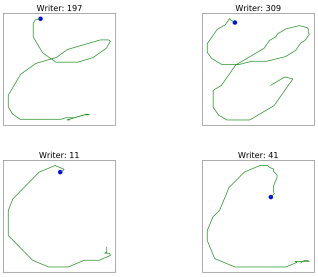

Figure 7: Examples for writing of letter C from the selected cluster (first row) versus the rest of the letter drawings (second row). Starting point is marked with the blue mark. The drawings from the selected cluster show people with Edwardian style of handwriting.

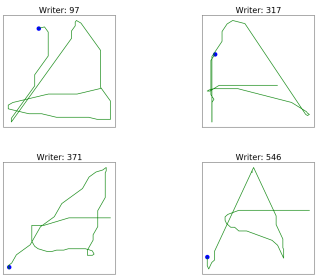

Figure 8: Examples for writing of letter A from the selected clusters. Starting point is marked with the blue mark. Each row is from one cluster. The first row show people who start drawing the letter from the top, going down, and then continue the drawing of the letter. The second row show people who start drawing from down directly.

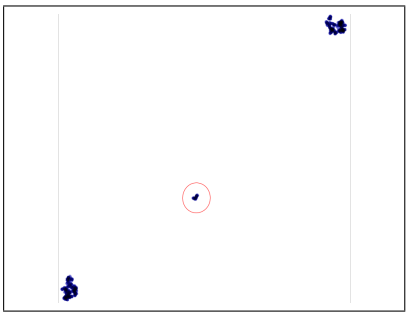

Figure 9: Projection for latent space for letter S using t-SNE. We manage to interpret the indicated cluster as the Edwardian style in drawing. The other two clusters (not indicated) did not show clear difference in the style, but this is an expected behavior from using the t-SNE algorithm, since it does not try to cluster styles as an objective.

good style extraction model should generalize well to new writers. Last, we analyze the latent space of our model for multiple letters, and show that the model separate the different styles in different clusters. We are interested in further investigating the concept of style transfer. In this work, we fixed the task (the uppercase letters), and performed transfer of style across

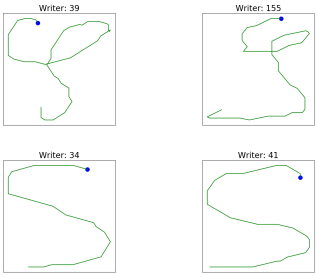

Figure 10: Examples for writing of letter S from the selected cluster (first row) versus the other two clusters (second row). Starting point is marked with the blue mark. The drawings from the selected cluster is always Edwardian style.

writers. Our plan is to investigate style transfer while changing the task (e.g., learn style on uppercase letters, and transfer them to the lowercase writers).

Based on the results of the latent space analysis, our next objective is to build an latent space structure and objective function that disentangle the style manifold. So far, we used multiple projection techniques in order to explore the style information in the latent space. We would like this to emerge on its own in the latent space. This step is usually known as *Knowledge Restructuring*, which enable the addressing of several interesting questions, like: What are all the different styles available for different letters? Can we use the styles from those different letters to build a footprint for each writer (i.e. style embedding for the writer)? If so, how good is this embedding in learning to generate letters using it as a prior knowledge only?

Acknowledgements

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

REFERENCES

Bishop, C. M. (1994). *Mixture density networks*. Aston University.

Briot, J.-P. and Pachet, F. (2017). Music generation by deep learning-challenges and directions. *arXiv preprint arXiv:1712.04371*.

Chang, B., Zhang, Q., Pan, S., and Meng, L. (2018). Generating handwritten chinese characters using cyclegan. *CoRR*, abs/1801.08624.

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

Freeman, H. (1961). On the encoding of arbitrary geomet-

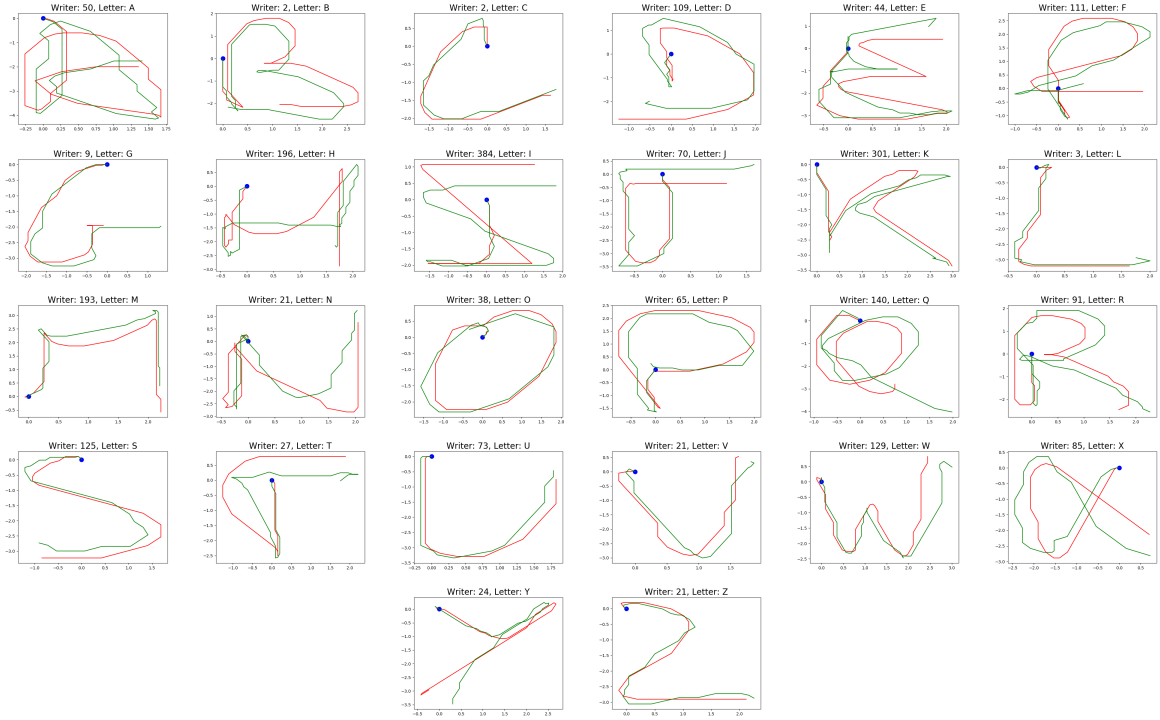

Figure 11: Examples of generated letters. The blue mark is the starting point. The traces in green is the ground truth, and the red is the generated ones by our model.

ric configurations. *IRE Transactions on Electronic Computers*, 2:260–268.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

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

Google (2017). The quick, draw! dataset.

Ha, D. and Eck, D. (2017). A neural representation of sketch drawings. *CoRR*, abs/1704.03477.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jolliffe, I. (2011). Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Mohammed, O., Bailly, G., and Pellier, D. (2018). Handwriting styles: benchmarks and evaluation metrics. In *First International Workshop on Deep and Transfer Learning - Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, Valencia, Spain. IEEE.

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

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

Skerry-Ryan, R. J., Battenberg, E., Xiao, Y., Wang, Y., Stanton, D., Shor, J., Weiss, R. J., Clark, R., and Saurous, R. A. (2018). Towards end-to-end prosody transfer for expressive speech synthesis with tacotron. *CoRR*, abs/1803.09047.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

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

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In

Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on, pages 3156–3164. IEEE.