Published in the *Proceedings of the 20th International Conference of the International Graphonomics Society, IGS 2021, Intertwining Graphnomics with Human Movements*, 7-9 June 2022, Las Palmas de Gran Canaria,Spain, which should be cited to refer to this work.

DOI: https://doi.org/10.1007/978-3-031-19745-1_5

Improving handwriting recognition for historical documents using synthetic text lines

Martin Spoto[1], Beat Wolf[1], Andreas Fischer[1,2], and Anna Scius-Bertrand[1,2,3]

[1] iCoSys, HES-SO, Fribourg, Switzerland
{martin.spoto, beat.wolf, andreas.fischer, anna.scius-bertrand}@hefr.ch
[2] DIVA, University of Fribourg, Switzerland
[3] EPHE-PSL, Paris, France

Abstract. Automatic handwriting recognition for historical documents is a key element for making our cultural heritage available to researchers and the general public. However, current approaches based on machine learning require a considerable amount of annotated learning samples to read ancient scripts and languages. Producing such ground truth is a laborious and time-consuming task that often requires human experts. In this paper, to cope with a limited amount of learning samples, we explore the impact of using synthetic text line images to support the training of handwriting recognition systems. For generating text lines, we consider lineGen, a recent GAN-based approach, and for handwriting recognition, we consider HTR-Flor, a state-of-the-art recognition system. Different meta-learning strategies are explored that schedule the addition of synthetic text line images to the existing real samples. In an experimental evaluation on the well-known Bentham dataset as well as the newly introduced Bullinger dataset, we demonstrate a significant improvement of the recognition performance when combining real and synthetic samples.

Keywords: Handwriting recognition · synthetic handwriting · meta-learning strategies · lineGen · HTR-Flor.

1 Introduction

The state of the art in handwritten text recognition (HTR) for historical documents has improved greatly in the past decade, leading to relatively robust systems for automated transcription and keyword spotting [3]. However, the main limitation of such systems is the need to access thousands of annotated training samples, which have to be produced by human experts for each script and handwriting style anew.

A promising approach to alleviate this limitation is to support the training of the recognition system with synthetic samples. Recent progress include the use of Generative Adversarial Network (GANs) for generating synthetic handwriting based on examples of existing handwriting styles.

In this paper, we aim to investigate whether or not synthetic handwriting samples can help to improve the recognition performance for historical documents when only few real labeled samples are available. To the best of our knowledge, this question has not been addressed comprehensively so far. We consider a recent GAN-based approach, lineGen [2], for style transfer and synthesis of text line images, and use the synthetic learning samples for training a state-of-the-art recognition system, HTR-Flor [14]. Several meta-learning strategies are investigated to schedule the addition of synthetic samples to the real ones.

Two datasets are considered for experimental evaluation. First, the well-known Bentham collection [5], which contains a single-writer collection of English manuscripts from the 18th and early 19th century. Secondly, the newly introduced Bullinger dataset, a work in progress that aims to make the letter correspondence of Heinrich Bullinger, a Swiss reformer, available in an electronic edition. The letters were written in Latin and German in the 16th century and encompass a considerable number of writers who are represented with only one or few letters in the collection. Handwriting synthesis is particularly interesting in this scenario, as it may allow to adapt a handwriting recognition system to the particular writing styles of these letters.

In the following, we discuss related work, describe the handwriting datasets, introduce the synthesis and recognition methods as well as the meta-learning strategies, and present the experimental results. The paper is concluded with an outlook to future work.

1.1 Related work

We found relatively few examples of using synthetic handwriting data to help with the training of HTR systems. One recent example is TrOCR [9]. It uses a transformer-based architecture as well as pre-trained image and text transformers to achieve state-of-the-art recognition performance on both printed and handwritten text datasets. One issue of transformer-based architectures is that they require huge amount of training data. The solution implemented by TrOCR is to use synthetic data to augment existing datasets. They did not however use a handwriting generator, but instead generated training data using publicly available fonts with both printed and handwritten style.

There have been several attempts at handwritten text generation in recent years since the introduction of Generative Adversarial Networks [6], but only in a few cases, the resulting synthetic data has been used to train an HTR system.

To the best of our knowledge, Alonso et al. [1] is the first attempt at generating handwritten text images by using a Generative Adversarial Networks (GAN) [6] trained on offline data. While it is able to generate legible French and Arabic words, it suffers from several limitations. It is only able to generate fixed width images with consequently a fixed character count. It is also unable to properly disentangle style from content, making it impossible to control the style of the generated images. Despite these limitations, the generated images were

used to augment the IAM dataset with 100k new entries and train a handwritten text recognition system, but to no noticeable improvements.

ScrabbleGAN [4] improves on Alonso et al. [1] by using a fully convolutional generator and a filter bank to handle character style. These improvements allow for variable word length and control over the generated style, but the character width is still fixed, making generated cursive text look unrealistic. Generated data was evaluated by mixing 100k images to existing datasets of modern handwriting, specifically IAM and RIMES, and retraining the HTR system. A improvement of around 1% was measured on both datasets. More interestingly, they highlighted the possibility of using such a generation system in domain adaptation scenarios.

GANWriting [8] improves on ScrabbleGAN [4] by removing the character width limitation, and therefore showing huge improvements in generation quality for cursive and tight handwriting styles. Unfortunately, they did not evaluate their data for HTR system training.

SmartPatch [10] is the latest improvement of GANWriting [8]. Custom patch discriminators are used to improve generation quality by removing some common artifacts produced by GANWriting. In a human evaluation, SmartPatch was thought to look better than GANWriting 70.5% of the time, and it even seemed more real than the true real data 54.4% of the time. They did not however evaluate their results on a HTR model either.

LineGen [2] is based on Alonso et al. [1]. It works directly on entire lines and is capable of extracting style information from only a few samples. It uses an additional spacing network to allow much better variation in character output width. It makes use of an autoencoder-like architecture to introduce perceptual and pixel-wise reconstruction loss, enabling for high-fidelity results. However, we find once again no evaluation of the results on a HTR model.

In this paper, we consider lineGen for generating synthetic handwriting because of convincing visual results and its ability to generate complete text lines, which are the standard input for current HTR systems. The synthetic training samples are studied in the context of handwriting recognition with HTR-Flor [14], a relatively lightweight convolutional recognition system with state-of-the-art performance.

2 Data

2.1 Bentham

The Bentham collection [5] [13] is a set of manuscripts images written by the English philosopher Jeremy Bentham during the 18th and early 19th centuries. It is therefore a single writer dataset. It contains 433 pages of scanned letters, totaling 11,473 lines. The scans are of high quality, with a clearly legible black ink on grey background handwriting, as can be seen in Figure 1.

The dataset comes as either directly the pages or the lines, along with a ground truth indicating what is written on each image. There is no word-level

Fig. 1. Example of a line from the Bentham dataset.

isolation available. That means that, as can be seen on Figure 1, the entire lines are cut from the pages. The skew, i.e. the inclination of the text lines, is not corrected, so some of the lines are not perfectly horizontal and may have a slight upward or downward angle.

2.2 Bullinger

The Bullinger dataset is a novel, work-in-progress dataset originating from the Bullinger Digital project[1]. This project aims to scan and associate transcriptions to letters sent by and to the Swiss reformer Heinrich Bullinger (1504-1575), which leads to the presence of different handwriting styles. While the scans are of high quality, the dataset itself is a challenge for handwriting recognition. It features 16th century style handwriting with ink on paper that is often hard to read even for a human observer. It is also a multi-language dataset, featuring letters mostly in Latin but also in German. A line example is shown in Figure 2.

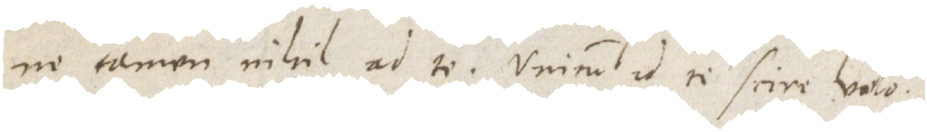

Fig. 2. Example of a line from the Bullinger dataset.

The dataset is composed of a set of scanned letters with the line location information.

As the dataset is still a work in progress, we did not have access to all the data at once. We therefore used two distinct releases. The first release, the small Bullinger dataset, is composed of 1,488 lines after preprocessing. The second release, the large Bullinger dataset, is composed of 18,925 lines.

There are also some caveats with the provided ground truth and segmentations. While some of the content and segmentations have been proofread and are human-verified, most of the data comes from a Transkribus [7] based transcription alignment system. Although transcription alignment has a very high precision, the resulting ground truth still contains a few errors. In particular, some abbreviations that are commonly used in handwritten Latin text may be written out in full. While this makes sense when providing a transcription of a

[1] https://www.bullinger-digital.ch/

letter to a human reader, it is not ideal for training an HTR system, because it leads to a mismatch between the abbreviation character visible in the image and the word written out in full in the transcription. Nevertheless, an experimental evaluation of this dataset is still feasible, since we only compare relative HTR performances measured on the same data.

3 Methods

In this section, we describe our choice of methods for evaluating the impact of synthetic training data. First, the GAN-based approach to generate synthetic text lines using lineGen, secondly, the HTR system, and finally the meta-learning strategies for mixing real and synthetic data.

3.1 Text-line image generation

To generate new text-line images we use lineGen [2]. It works directly on entire lines and is capable of extracting style information from only a few samples of the target style. It uses an additional spacing network to allow much better variation in character output width. It makes use of an autoencoder-like architecture to introduce perceptual and pixel-wise reconstruction loss, enabling for high-fidelity results.

The network is composed of six components: a style extractor, a space predictor, a pre-trained HTR system, a generator, a discriminator and a pre-trained encoder. The style extractor takes a single image as input and outputs a style vector. The space predictor takes a line of text and the style vector as input and outputs spaced text. Both the style vector and the spaced text are then fed to the generator which outputs a generated image that should have the content of the given line of text with the style of the given image example.

Three loss functions are considered. First, the generated image is used by the pre-trained HTR system to compute a Connectionist Temporal Classification (CTC) loss. Second, the discriminator computes an adversarial loss and, third, the pre-trained encoder computes a perceptual and pixel-wise reconstruction loss.

We use this network on two datasets described below, Bentham and Bullinger. The data is split randomly into independent sets for training, validation, and testing, respectively. The exact split is indicated in Table 2 in the experiments Section 4.1.

Bentham To generate synthetic text lines for the English Bentham dataset, we write lines from *The Lord of the Rings* in the style of the Bentham dataset. We successively trained the encoder (25'000 iterations), the internal HTR system (15'000 iterations), and the generator (50'000 iterations). Figure 3 shows the results obtained after this training process. Visually the lines look like the original database lines. They are legible with relatively few artifacts. Some do appear, especially with characters that go below the baseline like "y" or "g". We can

also see some artifacts appear around punctuation marks like "!" or ",". Some of those artifacts are due to the data, in particular the non-frequent characters. For example, we count only 19 occurrences of "!" in the whole Bentham dataset.

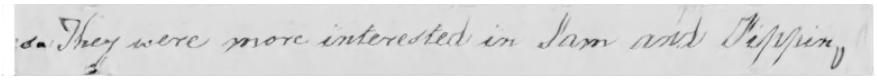

Fig. 3. Example of generated lines using lineGen on the Bentham dataset after 50'000 iterations

The artifacts of letters like "y" and "g" are harder to understand. One possible explanation is that they are somewhat often cut or overlapped in the dataset, due to the segmentation of the lines. One other interesting point regarding those letters is that they seem to share the same defects across sentences. Figure 4 shows a close up of those artifacts from multiple different lines. We can clearly see that the defaults as well as the general shape are similar for each instance. It seems that the generator does not produce enough variations. We tried to add some variations by introducing a normal noise $X \sim \mathcal{N}(0, 0.5)$ to the style vector centered around the mean. We do not observe any significant difference with or without additional noise, hinting that adding noise to the style vector is not enough to fix the variation issues of the generator.

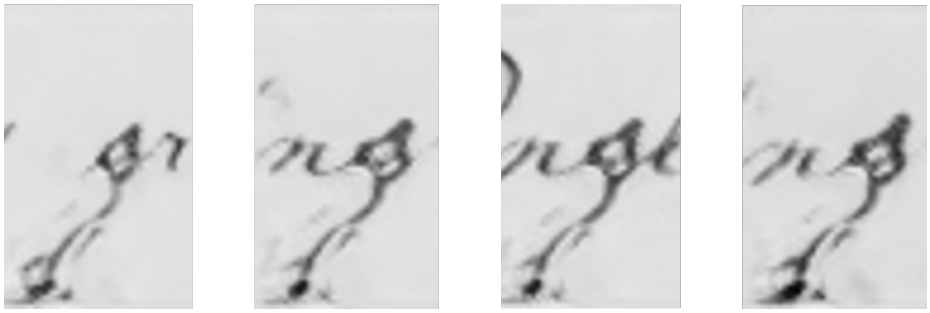

Fig. 4. Example of artefacts in a line generate by lineGen on the Bentham dataset

Bullinger On the Bullinger dataset, the same lines from *The Lord of the Rings* were synthesized. The training of lineGen was done in two parts. It was first attempted on the small original dataset containing only 1,190 transcribed lines. As the results were not satisfactory, training was then continued on the bigger 17,033 lines dataset.

After training our text-line generator on the small dataset we reach a Character Error Rate lower than 1% on the training data and 30% on the validation

data. These results show clearly an overfitting, as should be expected with such a small dataset. Nevertheless, we went ahead with the training of the generator to see how it would perform in such bad conditions. It was trained for 50,000 iterations (about 54 hours). As visible in Figure 5a), the generator struggled to output something coherent, and it took over 40'000 iterations just to start seeing something that was remotely similar to what we would expect. After another 10,000 iterations, the output stabilized to very blurry but still coherent text, as can be seen in Figure 5b). We can however see that the general look of the generated image matches the expected style of the Bullinger dataset.

After this initial training, we continued on the larger one. The generator was trained for another 10,000 steps. This resulted in a significant improvement in generation quality, as can be seen in Figure 5c). The text is sharper and mostly legible. We can however observe the same artifacts as on the Bentham dataset, particularly visible on the "y" letter. This effect is amplified here because we generated an English sentence, which has vastly more occurrences of "y" than the original Latin or German languages of the dataset.

As an attempt to further increase the generation quality, both the encoder and internal HTR were also trained on the larger dataset. The encoder was trained for an additional 54,000 iterations, bringing the total to 80,000. The internal HTR was trained for 15,000 more iterations, for a total of 30,000. This renewed training ended up with a Character Error Rate of 6% on the training data and 15% on the validation data, i.e., we see much less overfitting than after the initial training, as it shows on the results obtained on the validation data. Using those new pre-trained parts, the generator was then trained for another 20,000 iterations, reaching a total of 80,000. An example is shown Figure 5d).

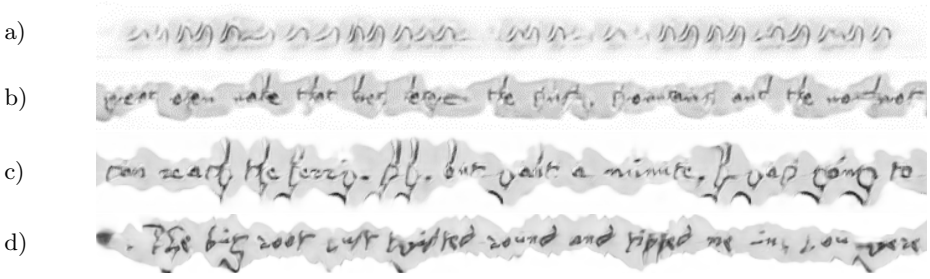

Fig. 5. Example of generated line images using lineGen on the small Bullinger dataset after a) 40'000 iterations, b) 50 000 iterations and on the large one with c) 60'000 and d) 80'000 iterations

While the results may look worse than before at first glance, they are actually closer to the original dataset. This is explained by the fact that the internal HTR got significantly better. Before the additional data and training, it was biased towards "easy to read" characters, and this bias got carried over to the generator. By increasing the ability of the internal HTR, the generator is able

to create a wider variety of character styles that are still correctly recognized, and is therefore not incorrectly penalized by the character loss.

Note, however that the final results are overall worse than on the Bentham dataset, with more artifacts appearing. This is easily explainable by the nature of the two datasets. The Bentham dataset being a cleaner, single writer dataset, it is obviously easier than the Bullinger one and its multiple writers with hard to read handwriting, even for an human observer. The synthetic Bullinger style is expected to contain predominant character styles across the whole database but not to mimic one writer in particular, although it will be biased towards the style of Bullinger himself who wrote the largest number of letters.

3.2 Recognition system

To choose the recognition system, three networks were compared on the Bentham and IAM dataset: HTR-Flor [14], TrOCR [9] and PyLaia [12], used in commercial tools like Transkribus [7]. The results of the papers mentioned above have been reported in Table 1. TrOCR has the lowest CER rate but is the network with the largest number of parameters and the longest decoding time. Also, this network needs more training data than the two other networks.

For evaluating the impact of synthetic training samples on the recognition performance, we chose HTR-Flor as our baseline recognition system, because it offers an excellent trade-off between recognition performance and computational effort. It is lightweight, relatively fast, and still manages to outperform other models like PyLaia.

Model	# of params	Decoding Time	CER (Bentham)	CER (IAM)
HTR-Flor	0.8 M	55 ms/line	3.98%	3.72%
PyLaia	9.4 M	81 ms/line	4.65%	4.94%
TrOCR	558 M	600 ms/line	-	2.89%

Table 1. Comparison between different recognition models, results from [14] and [9]

Additionally, having less parameters also means that the network can be trained with less data, and with less risks of overfitting when data is scarce. This is also an advantage in our case, since our final use case aims to be able to train the HTR system on as little as a single page of real text for a particular writer.

3.3 Meta-learning strategy

The standard meta-learning strategy for using synthetic data is to add a fixed number s of synthetic samples to the real samples and then train the system until convergence. Because there is no limit in the number of synthetic text lines that can be produced by the generator, different values for s can be tested. The

expectation is that adding some synthetic data will be helpful because of the increased data quantity but adding too much synthetic data leads to a reduced performance because of the decreased data quality when compared with real samples.

We explore also a more detailed meta-learning strategy by gradually adding more synthetic data to the system as the number of training epochs increase. Our intuition is that real data is especially important at the beginning of the training process, in order to find good initial parameters for the HTR system with high-quality data, and that adding synthetic data is especially beneficial at the end of the training process to fine-tune the parameters with high-quantity data.

The suggested meta-learning strategy applies a sequence $L_1, \ldots, L_n$ of learning steps, where each step $L_i = (r_i, s_i, e_i)$ utilizes r_i real samples and s_i synthetic samples for training the HTR system during e_i epochs, with $r_i \geq r_{i+1}$ and $s_i \leq s_{i+1}$ to gradually increase the number of synthetic learning samples. The following strategies are considered in this paper:

- *Real-only*. Use only real samples $L = (r, 0, e)$.
- *Synthetic-only*. Use only synthetic samples $L = (0, s, e)$.
- *Fixed*. Use a fixed amount of real and synthetic samples $L = (r, s, e)$.
- *Increase*. Increase synthetic samples $L_1 = (r, s_1, e_1), \ldots, L_n = (r, s_n, e_n)$.
- *Replace*. Also decrease real samples $L_1 = (r_1, s_1, e_1), \ldots, L_n = (r_n, s_n, e_n)$.

4 Experiments

4.1 Setup

The Bentham and Bullinger datasets consist of independent sets of text lines for training, validation, and testing, as indicated in Table 2. To evaluate the impact of synthetic data for HTR training, we used HTR-Flor with its standard configuration. Because there was no need to optimize meta-parameters on an independent set, we used the training set to train the system and the validation set to evaluate the results, i.e., the test set was not used.

Dataset	Training	Validation	Test	Total
Bentham	9,198	1,415	960	11,573
Bullinger (large)	17,033	946	946	18,925

Table 2. Distribution of text lines partitions

After training the lineGen text line generator for the two datasets (see Section 3.1), we consider two scenarios for evaluating the HTR system:

- **Medium.** A medium amount of 1000 real text lines from the training set is used to train the HTR-Flor recognition system. Such a situation is encountered when ground truth is available for several pages of a historical manuscript.
- **Low.** A low amount of 200 real text lines is used. Such a situation is encountered when ground truth is prepared only for one or few pages of a historical manuscript.

The HTR system is trained for 75 epochs in total, which is sufficient for convergence. Depending on the meta-learning strategies employed, the 75 epochs are subdivided into several learning steps $L_i = (r_i, s_i, e_i)$ with $\sum_{i=1}^{n} e_i = 75$.

For the **Medium** scenario, the meta-learning strategies are evaluated:

- *Real-only:* 1000 real samples.
- *Synthetic-only:* 1000 synthetic samples.
- *Fixed:* 1000 real and 1000 synthetic samples.
- *Increase:* $(1000, 0, 20), (1000, 500, 10), (1000, 1000, 20), (1000, 2000, 25)$
- *Replace:* $(1000, 0, 25), (750, 250, 25), (500, 500, 15), (0, 1000, 10)$

For the **Low** scenario, the following meta-learning strategies are evaluated:

- *Real-only:* 200 real samples.
- *Synthetic-only:* 200 synthetic samples.
- *Fixed:* 200 real and 200 synthetic samples.
- *Fixed-4k:* 200 real and 4000 synthetic samples.
- *Fixed-8k:* 200 real and 8000 synthetic samples.
- *Increase:* $(200, 0, 20), (200, 200, 10), (200, 500, 20), (200, 1000, 25)$
- *Replace:* $(200, 0, 25), (150, 50, 25), (100, 100, 15), (0, 200, 10)$

In both scenarios, to put the results into context, a baseline is provided, which corresponds to a situation where a large amount of ground truth is created for a historical document collection.

- *Baseline:* Use all the available real training samples (see Table 2).

4.2 Medium Scenario

The recognition results of the **Medium** scenario are summarized in Table 3 in terms of character error rate (CER). The best results on Bentham are achieved with the *Fixed* meta-learning strategy, with a CER of 11.38%, and the best results on Bullinger are achieved with the *Increase* strategy, with a CER of 23.55%. In both cases, the results of the *Fixed* and *Increase* strategies are very similar. When compared with the *Real-only* scenario, significant improvements of 3.39% (Bentham) and 3.24% (Bullinger) are obtained when using synthetic text lines during training. When compared with the *Baseline*, the achieved CER indicates that HTR remains feasible even when ground truth exists only for 1000 text lines.

Medium	Bentham	Bullinger
Baseline	05.01	10.69
Real-only	14.77	26.79
Synthetic-only	67.70	83.37
Fixed	**11.38**	24.09
Increase	11.99	**23.35**
Replace	30.63	55.96

Table 3. Character error rates for the **Medium** scenario. The best results among the different meta-learning strategies are highlighted in bold font.

When comparing the different meta-learning strategies, we can see that both datasets follow the same general trends. While the *Real-only* scenario reaches a solid performance of 14.77% and 26.79% on Bentham and Bullinger, respectively, the *Synthetic-only* one seems to quickly get stuck at 67.70% and 83.37%. This strongly hints that the HTR model is overfitting on the synthetic training data. The most likely explanation is that the generated images do not have enough variations. As previously shown in Figure 4 during the generator training, the individual characters do not seem to vary in a meaningful way, suggesting that the HTR model learns the shape of a few particular characters but cannot generalize to real variations present in the handwriting.

Figures 6 and 7 show the evolution of the CER in more detail during training with the different meta-learning strategies. Again, we observe consistent results among the two datasets.

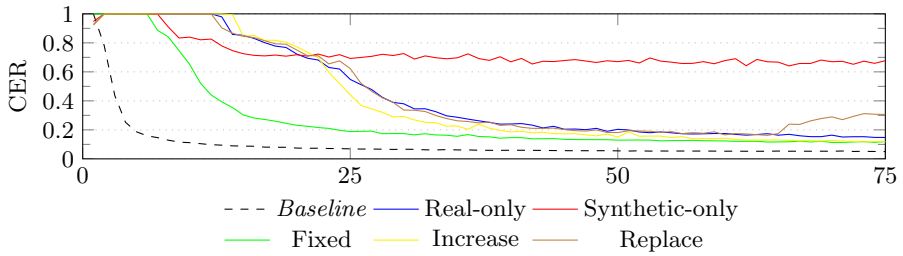

Fig. 6. Training behavior for the **Medium** scenario on the Bentham dataset.

Real-only and Synthetic-only In the **Medium** scenario, training with only real data achieves a reasonable performance but leaves room for improvements when

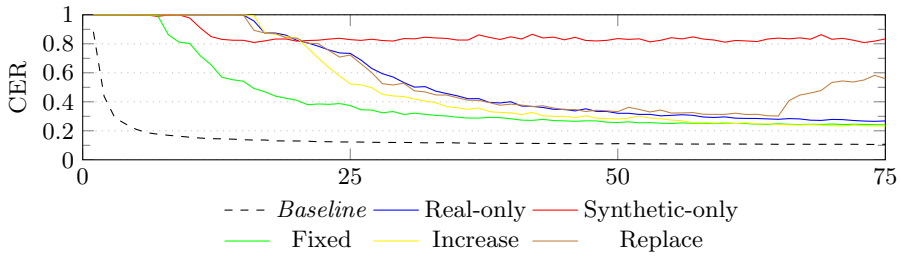

Fig. 7. Training behavior for the **Medium** scenario on the Bullinger dataset.

compared with the *Baseline*. Using only synthetic data fails and leads quickly to overfitting.

Fixed The *Fixed* scenario uses the same amount of real and synthetic images over 75 training epochs. It clearly outperforms the *Real-only* scenario, both learning faster and reaching a lower CER. This is a very encouraging result, as it shows that the additional generated data did indeed help training the HTR model.

Increase The *Increase* scenario gradually adds more synthetic data. As expected, this scenario stays close to the performances of the real-only one until the 20th epoch. It then starts to outperform it as additional generated data is included, reaching a final performance similar to the *Fixed* scenario. These results seem to indicate that it is not necessary to add the synthetic data progressively to "guide" the training.

Replace The *Replace* scenario both decreases the real data and increases the synthetic data gradually. This scenario is interesting because the training seems to follow the *Real-only* scenario, up to the 65th epoch, where we switch to generated data only. We can then see the CER climbing back up, presumably as the model overfits on the generated data, as theorized previously. It shows that the model overfits very quickly when presented with only synthetic data, even when previously "warmed up" with real data.

4.3 Low Scenario

The recognition results of the **Low** scenario are summarized in Table 4. The best results on Bentham are achieved with *Fixed-4k*, with a CER of 19.78%, and the best results on Bullinger are also achieved with *Fixed-4k*, with a CER of 40.24%. When compared with the *Real-only* scenario, drastic improvements of 61.31% (Bentham) and 47.79% (Bullinger) are observed, highlighting that synthetic data is especially helpful when only very few labeled samples are available in the ground truth, i.e. only a few pages or a single letter in a historical document collection.

Figures 8 and 9 illustrate the evolution of the CER during training. Again, consistent trends are observed among the two datasets.

Low	**Bentham**	**Bullinger**
Baseline	05.01	10.69
Real	81.10	88.03
Generated	68.41	80.71
Fixed	27.03	47.68
Fixed-4k	**19.78**	**40.24**
Fixed-8k	21.29	43.70
Increase	29.02	47.04
Replace	63.30	77.40

Table 4. Character error rates for the **Low** scenario. The best results among the different meta-learning strategies are highlighted in bold font.

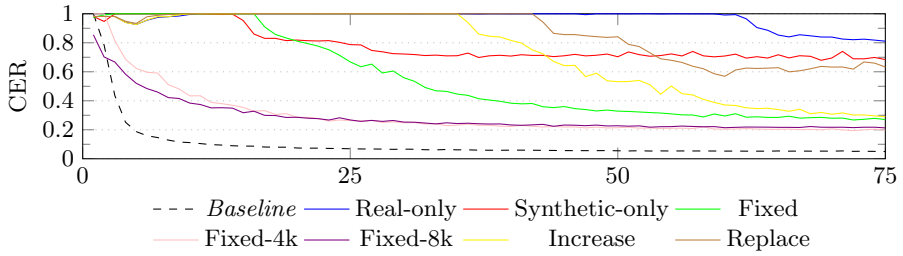

Fig. 8. Training behavior for the **Low** scenario on the Bentham dataset.

Real-only and Synthetic-only We see that the real data is not sufficient to train the model in the **Low** scenario. For *Synthetic-only*, the performance remains similar to the **Medium** scenario.

Fixed The different *Fixed* meta-learning strategies greatly outperform *Real-only* and *Synthetic-only*, demonstrating that a combination of real and synthetic data is of crucial importance when working in the **Low** scenario. We observe that adding more synthetic data using *Fixed-4k* improves the performance reaching a peak performance. Adding even more synthetic data with *Fixed-8k* does not further improve the result.

Increase As expected, we see the same "slow start" for *Increase* as with *Real-only*. We then see a rapid improvement as we add more data, the CER continuously decreases until the end of the training. On the Bullinger dataset, the strategy to gradually increase the synthetic data slightly outperforms the fixed combination of the *Fixed* strategies.

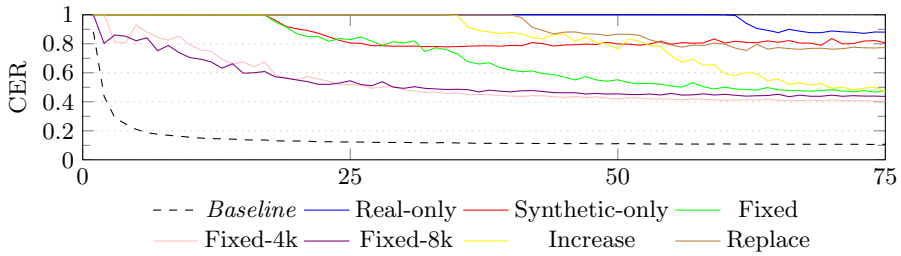

Fig. 9. Training behavior for the **Low** scenario on the Bullinger dataset.

Replace As for the **Medium** scenario, it is again not beneficial to remove the real samples. It leads to an overfitting to the synthetic data after the 65th epoch when all real data is removed. However, unlike the **Medium** scenario, the error rate does not increase significantly after this epoch, it just stops improving.

5 Conclusion

In this paper, we have studied the impact of GAN-based handwriting synthesis on the recognition performance, when training HTR systems with only few labeled data in the context of historical documents. With the right mix of real and synthetic data, supported by different meta-learning strategies, we were able to demonstrate a significant decrease in character error rate, ranging from 3% when using 1000 real text line images for training up to 60% when using only 200 real text lines. These results are especially promising for multi-writer document collections, such as the newly introduced Bullinger dataset, which contain a large number of unique writing styles.

There are several lines of future research. First, the handwriting generation system itself also depends on training data to perform a successful style transfer. This might not always be feasible for smaller datasets, as the initial training with the small Bullinger dataset showed. It would be interesting to further investigate the behavior of generators when trained using low quantity of data, or to explore new ways of training the generators, such as transfer learning with fine-tuning.

Secondly, all experiments in this paper have been realized using lineGen. It would be beneficial to test other generators as well, in particular to verify if the same drop in performance is observed when using only synthetic data.

Finally, the synthetically generated samples, although matching the target style well, were lacking some natural variability. Future work should investigate methods to increase the variability and make it as natural as possible. Using the kinematic theory of rapid human movements [11] for this purpose seems especially promising.

Acknowledgements

This work has been supported by the Hasler Foundation, Switzerland. We would like to thank the anonymous reviewers for their detailed and helpful comments.

References

1. Alonso, E., Moysset, B., Messina, R.: Adversarial generation of handwritten text images conditioned on sequences. In: Proc. 15th Int. Conf. on Document Analysis and Recognition (ICDAR). pp. 481–486 (2019)
2. Davis, B., Tensmeyer, C., Price, B., Wigington, C., Morse, B., Jain, R.: Text and style conditioned GAN for generation of offline handwriting lines. In: Proc. 31st British Machine Vision Conference (BMVC). pp. 1–13 (2020)
3. Fischer, A., Liwicki, M., Ingold, R. (eds.): Handwritten Historical Document Analysis, Recognition, and Retrieval – State of the Art and Future Trends. World Scientific (2020)
4. Fogel, S., Averbuch-Elor, H., Cohen, S., Mazor, S., Litman, R.: ScrabbleGAN: semi-supervised varying length handwritten text generation. In: Proc. Int. Conf on Computer Vision and Pattern Recognition (CVPR). pp. 4324–4333 (2020)
5. Gatos, B., Louloudis, G., Causer, T., Grint, K., Romero, V., Sánchez, J.A., Toselli, A.H., Vidal, E.: Ground-truth production in the Transcriptorium project. In: Proc. 11th Int. Workshop on Document Analysis Systems (DAS). pp. 237–241 (2014)
6. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Proc. 27th Int. Conf. on Neural Information Processing Systems (NIPS). pp. 2672–2680 (2014)
7. Kahle, P., Colutto, S., Hackl, G., Mühlberger, G.: Transkribus - a service platform for transcription, recognition and retrieval of historical documents. In: Proc. 14th Int. Conf. on Document Analysis and Recognition (ICDAR). pp. 19–24 (2017)
8. Kang, L., Riba, P., Wang, Y., Rusiñol, M., Fornés, A., Villegas, M.: GANwriting: content-conditioned generation of styled handwritten word images. In: Proc. 16th European Conf. on Computer Vision (ECCV). pp. 273–289 (2020)
9. Li, M., Lv, T., Cui, L., Lu, Y., Florencio, D.A.F., Zhang, C., Li, Z., Wei, F.: TrOCR: transformer-based optical character recognition with pre-trained models. CoRR (abs/2109.10282) (2021)
10. Mattick, A., Mayr, M., Seuret, M., Maier, A., Christlein, V.: SmartPatch: improving handwritten word imitation with patch discriminators. In: Proc. 16th Int. Conf. on Document Analysis and Recognition (ICDAR). pp. 268–283 (2021)
11. Plamondon, R., Marcelli, A., Ferrer, M. (eds.): The Lognormality Principle and its Applications in e-Security, e-Learning and e-Health. World Scientific (2020)
12. Puigcerver, J.: Are multidimensional recurrent layers really necessary for handwritten text recognition? In: Proc. 14th Int. Conf. on Document Analysis and Recognition (ICDAR). pp. 67–72 (2017)
13. Sánchez, J.A., Romero, V., Toselli, A.H., Vidal, E.: ICFHR2014 competition on handwritten text recognition on transcriptorium datasets (HTRtS). In: Proc. 14th Int. Conf. on Frontiers in Handwriting Recognition (ICFHR). pp. 785–790 (2014)
14. de Sousa Neto, A.F., Bezerra, B.L.D., Toselli, A.H., Lima, E.B.: HTR-Flor: a deep learning system for offline handwritten text recognition. In: Proc. 33rd Conf. on Graphics, Patterns and Images (SIBGRAPI). pp. 54–61 (2020)