Spatio-Temporal Handwriting Imitation

Martin Mayr[0000−0002−3706−285X], Martin Stumpf[0000−0002−2278−8500], Anguelos Nikolaou[0000−0003−3818−8718], Mathias Seuret[0000−0001−9153−1031], Andreas Maier[0000−0002−9550−5284], and Vincent Christlein[0000−0003−0455−3799]

Pattern Recognition Lab, Friedrich–Alexander-Universität Erlangen–Nürnberg
{firstname.lastname}@fau.de
https://lme.tf.fau.de/

Abstract. Most people think that their handwriting is unique and cannot be imitated by machines, especially not using completely new content. Current cursive handwriting synthesis is visually limited or needs user interaction. We show that subdividing the process into smaller subtasks makes it possible to imitate someone's handwriting with a high chance to be visually indistinguishable for humans. Therefore, a given handwritten sample will be used as the target style. This sample is transferred to an online sequence. Then, a method for online handwriting synthesis is used to produce a new realistic-looking text primed with the online input sequence. This new text is then rendered and style-adapted to the input pen. We show the effectiveness of the pipeline by generating in- and out-of-vocabulary handwritten samples that are validated in a comprehensive user study. Additionally, we show that also a typical writer identification system can partially be fooled by the created fake handwritings.

Keywords: Offline Handwriting Generation, Style Transfer, Forgery, Handwriting Synthesis

1 Introduction

Handwriting is still a substantial part of communication, note making, and authentication. Generating text in your handwriting without the need to actually take a pen in your hand can be beneficial, not only because we live in the age of digitization but also when the act of writing is physically impaired due to injuries or diseases. Handwriting synthesis could also enable to send handwritten messages in a much more personal way than using standard handwriting fonts, e. g., for gift messages when sending presents. Personal handwriting could also be useful in virtual reality games, where parts could be written in a famous handwriting or in the player's own handwriting in order to identify more strongly with the avatar. Similarly, it could be used for augmented reality. Why not adapting someone's handwriting style when translating a note from one language into another one?

Since the most deep learning methods need large datasets for training, another use case could be the improvement of automatic Handwritten Text Recognition

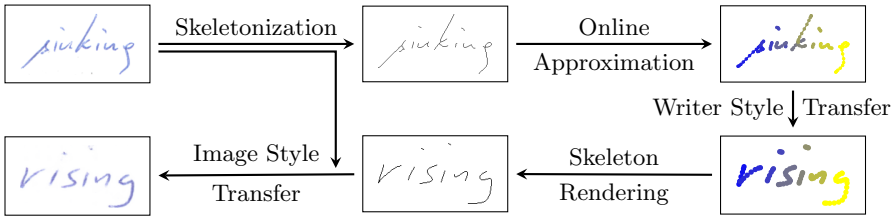


Fig. 1: Offline-to-Offline Handwriting Style Transfer Pipeline

(HTR). In fact, the simulation of single handwritten words is already used for data augmentation during the training of HTR methods [3].

With the help of our proposed method, it is possible to generate handwritten text in a personal handwriting style by using just a few paragraphs of handwritten samples of the person to be imitated. These handwritten samples are denoted as *offline* data in comparison to *online* handwriting, which stores for each data point not only its position but also the temporal information, representing the actual movement of a pen. This can for example be recorded with special pens or pads. We make use of an existing online handwriting synthesis method [10] and embed it into our work. Therefore, the offline data is transferred to the temporal domain, see Fig. 1 top branch. Finally, this online data is rendered again in the style of the writer to be imitated, cf. Fig. 1 bottom branch.

The main contributions of this work are as follows: (1) creation of a full pipeline for the synthesis of artificial handwriting, recreating both the visual and the writer-specific style; (2) development of a novel conversion method to approximate online handwriting from offline data to be able to utilize an existing online-based style transfer method; (3) the adaptation of conditional Generative Adversarial Networks (GANs) to compute a robust handwriting skeletonization and a visual style transfer to adapt to the used pen. For the former, we introduce an iterative knowledge transfer to make the offline skeletons more similar to the online training data. For the latter, we modified the well-known pix2pix framework [12] to incorporate the writing style information. (4) Finally, our method is thoroughly evaluated in two ways. First, by means of a writer identification method to quantitatively assess that the writing style is preserved and second, by a user study to evaluate to what extent humans can be fooled by the generated handwriting.

2 Related Work

Online Handwriting Synthesis. To produce convincing handwriting, it needs to reproduce the given content exactly, while keeping the style consistent, but not constant. Real human handwriting will repeat the same content almost identically, but still with some variance. This requires a solid long term memory, combined with some guidance from the required content. In the seminal work by Alex

Graves [10] this is achieved by the use of a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) cells, which enables the network to make predictions in context of the previous pen positions. Each hidden layer uses skip connections to give the network control over the learning flow. Additionally, an early form of attention mechanism was developed that decides which part of the content the network focuses on.

A problem of Graves' method [10] is that it tends to change the writing during the sequence generation. This can be overcome by the use of Variational RNNs (VRNNs) to generate consistent handwriting [6]. Another method [2] builds on the idea of predicting single pen positions. Instead of relying on the network's internal memory to store style, the goal was to explicitly extract style and content from the input. This is achieved by utilizing a Conditional VRNN (CVRNN) to split the input into two separate latent variables, representing style and content. A drawback of this method is the need to split the input strokes into words and characters in form of begin- and end-of-character (EOC) tokens, which typically cannot be automatically determined from offline handwriting. In contrast to Graves' attention mechanism, it uses the generated EOC tokens to switch letters. This could remove some of the predictive capability of the network, since it can only foresee which letter follows when the next letter is already about to be written. In a subsequent work [1], the CVRNN was replaced by Stochastic Temporal CNNs (STCNNs) showing more consistent handwriting generation.

Offline Handwriting Synthesis. There are approaches that use printed text style synthesis for augmentation [9] or text stylization for artistic typography creation [24]. Similarly, cursive handwriting synthesis [3] can be generated using a GAN combined with an auxiliary text recognition network. While the augmentation of HTR improves the recognition accuracy, the generated words cannot imitate a specific handwriting. Another approach [5] synthesis handwriting from public fonts by finding the best character matches in public handwritten fonts. The results are convincing but still far away from the actual user's style.

The closest work to ours is the work by Haines *et al.* [11]. To the best of our knowledge, this is the only other method using offline cursive handwritings as input. They produce convincing output by creating glyph models from segmented labeled ligatures and individual glyphs. The method has two main short-comings. First, the selection of the glyphs and ligatures involves human assistance. Second, only letters present in a handwriting sample can be reproduced.

Another work [16] employs a similar pipeline as our method, i. e., offline data is converted to online strokes and then rendered and style adapted. Therefore, glyphs need to be segmented and matched to characters. From these matched glyphs, strokes are extracted by registering them to a trajectory database, and sampled with regular points. In contrast, we propose a method based on maximum acceleration that uses more points for curved strokes. The user's style in [16] is learned by a feed forward neural network, which is added during the rendering process. While no human interaction is needed to generate handwritten Chinese fonts with a large amount of characters, the "characters should be written

separately without touching each other in a given order and consistently in size and style." [16]. This issue is targeted by Nakamuar *et al.* [18] who generate Chinese characters for samples with an incomplete character set by choosing the closest learned character distribution.

In contrast, our method works fully automatic for cursive handwriting without any user interaction. The method is able to produce out-of-vocabulary (OOV) letters and words as long as the trained model has seen some instances during training, but not necessarily from the writer to be imitated.

3 Offline-to-Offline Handwriting Style Transfer Pipeline

We decided to split the offline to offline handwriting style transfer system into several subtasks. While background, pen and writer style are static problems that could be solved with Convolutional Neural Network (CNN), the text content has a structural component and therefore makes some temporal generation, e. g., in form of an RNN preferable. Each of the subtasks is trainable on its own, allowing human prior knowledge to guide the process, and to evaluate single steps separately. Also, not all of those tasks require a neural network, where an algorithmic solution is beneficial. For a writer imitation, we need to apply a style transfer on two levels: (1) the arrangement of writing strokes and (2) the pen style, i. e., thickness and color distribution.

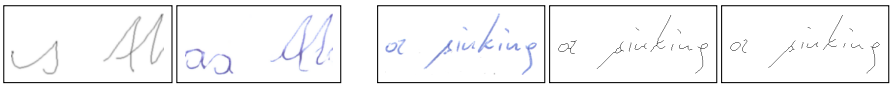
Fig. 1 gives an overview over our pipeline.[1] First, a skeleton is computed from the input sequence, which is used as the writing style to imitate (commonly several words long). The skeleton is converted to online data with a novel sampling process that puts emphasis on the curved structure of handwriting. Afterwards, we make use of the writing generation method of Graves [10], which creates new online text in a given writing style. This handwriting sequence is rendered as a skeleton and transferred to the visual appearance of the priming sequence using additional data created through the initial skeletonization process. Each step of the pipeline is described in the following subsections.

3.1 Skeletonization

In this stage, we convert images of real handwriting to their corresponding skeletons. These skeletons are then subsequently mapped to the temporal domain and fed into the generative network. There are sophisticated learning-free [4] and Deep Learning (DL) [23]-based methods, which learn a mapping between natural objects and skeleton. We face the challenge that individual datasets exist of both real handwriting images and skeletons (in form of online data). However, to the best of our knowledge, there is no dataset that annotates a mapping between those two.

Given an offline handwritten sample, the challenge is to produce a skeleton similar enough to the online data used in the generative network. Specifically, we

[1] Code and models available below https://github.com/M4rt1nM4yr/spatio-temporal_handwriting_imitation



(a) (b)

Fig. 2: (a) shows a skeleton rendered from the IAM-Online dataset (left) and the estimated skeleton by the CycleGAN (right). (b) shows an offline example excerpt (left), skeleton output of a basic method [25] (middle), and through our skeletonization network (right).

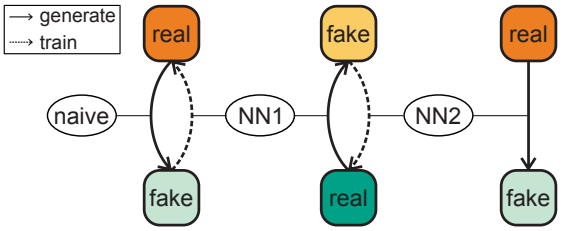


Fig. 3: Iterative knowledge transfer from naive algorithms. See text for details.

use the CVL dataset [14] as the source for real offline data and the IAM-Online dataset [17] for real skeleton data, where the latter is also used for training the generative network.

A natural choice would be the use of CycleGAN [26], which enforces cycle consistency, i. e., the output of a transfer from source to target and back is similar to the source again (and vice versa). A shortcoming of CycleGAN is that it has to guess the transfer function. This could lead to several problems because it is not guaranteed that the resulting mapping will still satisfy spatial consistency. It could freely add/remove strokes, as long as the result is GAN-consistent and contains enough information to perform an inverse mapping, see Fig. 2a for an example.

Conversely, there are basic skeletonization algorithms [25,15]. These could be used to guide the training to incorporate prior knowledge about the mapping, cf. Fig. 2b. Therefore, we propose *iterative knowledge transfer* to extract the knowledge of one of those algorithms and transfer it to a neural network. The proposed method is not limited to this specific use case. It is rather a general method to transfer knowledge from an existing mapping function to a neural network while improving and generalizing it along the way. It requires a naive mapping function and two non-paired datasets for which we would like to achieve a mapping and consists of the following steps, as illustrated in Fig. 3.

(1) Generation of a synthetic dataset from the real source dataset using the naive mapping. This dataset is expected to be erroneous.
(2) Training of an inverse mapping (NN1) based on the synthetic dataset, enforcing generalization. The network capacity needs to be sized correctly, so

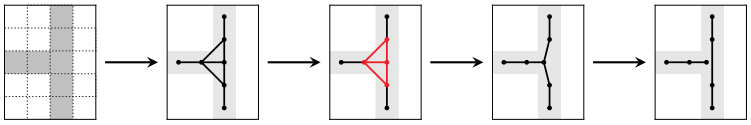


Fig. 4: Conversion of a skeleton image to strokes. (a) Original skeleton image. (b) Initial generated graph by connecting neighboring pixel. (c) Detected cluster. (d) Graph after the cluster got replaced by its mean node. (e) Final strokes after resolving intersections.

that generalization happens without picking up on the errors of the naive implementation.

(3) Generation of a synthetic inverse dataset from the real destination dataset using the trained mapping. The network was only ever trained to output real data, so the expectation at this step is that the switch from erroneous input data to real input data will not produce additional artifacts, but real-like output data that matches the input data.

(4) This will finally create a dataset that maps real destination data to real-like source data, and we can now take this dataset to train our final network.

3.2 Online Sequence Approximation

The purpose of this stage of the pipeline is to take skeleton images as input and convert them to an approximate online representation. As skeletons do not contain temporal annotations, this step will require a heuristic approach to synthesize realistic online data analytically. This stage consists of two steps: (1) conversion of the bitmap representation to strokes, and (2) temporal resampling and ordering.

Conversion to Strokes. The schematic diagram in Fig. 4 shows the conversion of image skeletons to strokes. First, the bitmap data is converted to a graph. This is done by connecting the neighboring pixels because we assume that strokes are connected lines in the skeleton. The skeletonization mainly influences the characteristics of the strokes.

Our target is to create strokes without cycles, but the generated graph contains pixel clusters, as can be seen in Fig. 5. These pixel clusters are defined as triangles having at least one common edge. Graph cycles which do not consist of triangles are assumed to be style characteristics of the writer, like drawing the dot on the "i" as a tiny circle. The cluster artifacts are solved by replacing all the cluster nodes that are not connected to outside nodes with a single node at the mean position of the group.

The final step for creating strokes is to remove intersections and cycles. This implies that the nodes of every line segment have either one or two neighbors. So intersections are defined at nodes with more than two neighbors. Intersections with even number of neighbors are solved by connecting neighbors of opposite

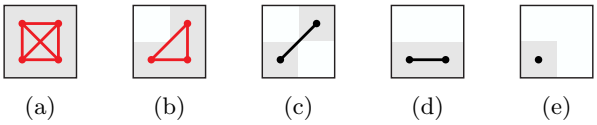


Fig. 5: All possible triangle group configurations in a 2×2 square. There are rotated versions of all patterns except (a). Each square can contain either (a) four, (b) one, or (c)–(e) no triangles. A square cannot contain triangles from two different groups because all triangles in (a) must be in the same group as they are touching each other.

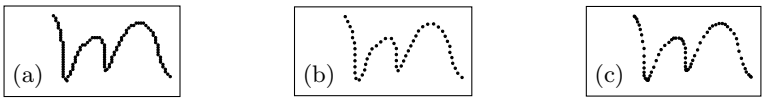


Fig. 6: Visualisation of different resampling methods. (a) no resampling, (b) constant velocity resampling, (c) proposed maximum acceleration resampling.

sides and removing the crossover node. The other intersections require a little more math to become useful. Every neighbor has two crossover candidates that are on the opposite side of the crossover. We compute all the angles between all candidates, and iteratively connect the neighbors with the largest angle until only one neighbor is left. We keep that one neighbor connected to the original crossover, which is now the end of a curve. We skip to join two nodes if they would create a new cycle.

Removing cycles is done as follows. We step through each subgraph until we either reach a node with only one neighbor or the starting node. Once we found a cycle, we always split it on the upmost node to achieve a consistent behavior. Note, humans also tend to split circles at the upmost position.

Resampling. To obtain online handwriting, we still need to incorporate temporal information. The simplest way would be to sample at constant time steps. However, the training of the subsequent online network becomes very difficult because very small time steps are needed. Therefore, we propose *maximum acceleration resampling* on the computed lines to imitate the dynamics of human writing. We constrain the velocity to be zero at both extremities of a line, increase it on straight parts, and decrease it on curved parts. This has the advantage that the network focuses on important parts of the handwriting. The difference between constant resampling and the proposed maximum acceleration resampling can be seen in Fig. 6.

In detail, the algorithm consists of the following steps: (1) Resampling of the curve to sufficiently small intervals. (2) Creating a reachability graph between nodes to prevent cutting corners. (3) Analyzing acceleratability, i. e., the acceleration required between two nodes. This will push the problem into a 4D space: (x,y,v_x,v_y). (4) Searching shortest path using directed Dijkstra.

Constant Pre-sampling. The Dijkstra search [7] does not include the possibility to cut lines into pieces. It can only select a set of optimal nodes from existing nodes. Consequently, we need to make sure that the existing nodes are spaced appropriately by resampling them into small constant intervals. We empirically set the distance to 1/3 of the maximum acceleration value.

Reachability Graph. We have to avoid cutting corners, such that during curved sections the sampling rate is higher. Therefore, we create a graph, encoded in a boolean matrix of size $N \times N$ (where N is the number of nodes in the pre-sampled graph), which stores the pairwise reachability between all nodes. We define points $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+n}$ to be reachable, when $\max_{j=i..i+n} d(\boldsymbol{p}_j) < t$ with $d(\boldsymbol{p}_j)$ being the distance of the point $\boldsymbol{p}_j$ to the line between $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+n}$ and t being a given threshold. We found the threshold parameter to be optimal at 3 times the node distance of the pre-sampling step.

Acceleratability. So far, our nodes are two dimensional: $\boldsymbol{p} = (x, y)$. We now enhance them with two more dimensions: the incoming velocities $\boldsymbol{v} = (v_x, v_y)$. The incoming velocity $\boldsymbol{v}_i$ of node $\boldsymbol{p}_i$ from node $\boldsymbol{p}_{i-n}$ is computed by $\boldsymbol{v}_i = \boldsymbol{p}_i - \boldsymbol{p}_{i-n}$. Thus, only one 4D node exists for each preceding 2D node, with multiple possible edges from different velocities of that preceding node. We now create a 4D acceleratability graph based on the 2D reachability graph that connects all 4D nodes $(\boldsymbol{p}_i, \boldsymbol{v}_i)$ and $(\boldsymbol{p}_j, \boldsymbol{v}_j)$ that fulfill the following 'acceleratability' criterion:

$$\boldsymbol{v}_j = \boldsymbol{p}_j - \boldsymbol{p}_i \quad \wedge \quad ||\boldsymbol{v}_j - \boldsymbol{v}_i|| < a \tag{1}$$

with a being the maximum acceleration hyperparameter. As we never go back, we have a directed graph, and thus it contains all possible pen trajectories that create the given curve.

Shortest Path Search. The set of possible paths is quite large, therefore we use a Dijkstra shortest path search with some optimizations specific to the 4D case. Since we have a directed graph, we will always start at one end of the stroke and move towards the other one. We can therefore step through the curve, 2D node by 2D node, and compute all optimal paths *to* that node for all possible velocities *at* that node. The number of possible velocities is quite limited and is equivalent to the number of incoming edges to that node.

Computing an optimal path to a given position $\boldsymbol{p}$ and velocity $\boldsymbol{v}$ can be done as follows. First, we get the position of the previous node: $\boldsymbol{p}_{prev} = \boldsymbol{p} - \boldsymbol{v}$. We now take all the possible velocities $\boldsymbol{v}_{prev}$ at position $\boldsymbol{p}_{prev}$ that can reach $\boldsymbol{p}$, based on the acceleratability criterion, cf. Eq. (1). The shortest path l to $(\boldsymbol{p}, \boldsymbol{v})$ is then:

$$l(\boldsymbol{p}, \boldsymbol{v}) = \min_{\boldsymbol{v}_{prev}} l(\boldsymbol{p}_{prev}, \boldsymbol{v}_{prev}) + 1 \tag{2}$$

We start the entire algorithm at one end of the curve, which we define as the starting point. Further, we set $\boldsymbol{p} = \boldsymbol{p}_{start}$, for which the only valid velocity is $\boldsymbol{v}_{start} = 0$ and $l(\boldsymbol{p}_{start}, \boldsymbol{v}_{start}) = 0$. We then iterate through the entire curve until we reach $\boldsymbol{p}_{end}$. As we defined both start and end velocities to be zero, we then look at $l(\boldsymbol{p}_{end}, \boldsymbol{v}_{end})$ with $\boldsymbol{v}_{end} = 0$ and backtrack to get the optimal path.

Ordering. Finally, the points are ordered by computing the mean of every stroke and ordering them from left to right to sort the list of strokes. In real human writing there are cases where this is not true, but in this way we produce a consistent behaviour, which is necessary for the further stages in the pipeline.

3.3 Writer Style Transfer

The produced online sequence is used to prime an online writing synthesis algorithm. We employ Graves' algorithm [10] who showed that LSTM cells are capable of generating complex structures with long-term contextual dependencies. To be able to predict both the content and the style at the same time, the content of the text is added to the network as a side input to one of the intermediate layers. The network does not see the entire content sequence at once, instead, a form of attention is used. To achieve this, the mixture density output from intermediate layers decide which part of the content gets delivered to the network. Note that we are not forced to use Gaves' method but can use any online handwriting generation approach.

3.4 Image Style Transfer

The produced new sequence of online data is transferred back to offline handwriting by means of drawing lines between the points of the online data. The last stage of the pipeline is to produce realistic offline handwriting by imitating the ink and style of the input image given the new skeleton of the produced online handwriting. Therefore, we modify the pix2pix [12] architecture to output the correct style.

Pix2pix Architecture Modification The network consists of an encoder and a decoder network. The first step in creating a style transfer network is to extract the style information from our input image Y. We use the encoder part of the pix2pix network with the style image as input. It is important that the style only contains global information, otherwise the discriminator could start to discriminate by content instead. Hence, we take the max-pooled outputs of all activation maps as style information, Then, we feed the style information into the pix2pix generator network by concatenating it with the innermost layer of the network, as seen in Fig. 7. To keep the size-agnostic property of the network, we repeat the style information along the two spatial axes to match the size of the innermost layer.

Loss Function To include the style extraction network in the training process, we add the output of the style extraction network to the input of the discriminator. Incorporating this into the objective of pix2pix [12] results in

$$\begin{aligned}\mathcal{L}_{\mathrm{cGAN}(G,D)} &= \mathbb{E}_{X,Y}[\log D(X, Y, \mathrm{style}(Y)] \\ &+ \mathbb{E}_{X,Y}[\log(1 - D(X, G(X, \mathrm{style}(Y)), \mathrm{style}(Y)))] \\ &+ \lambda \mathbb{E}_{X,Y}[||Y - G(X, \mathrm{style}(Y))||_1] \,.\end{aligned} \tag{3}$$

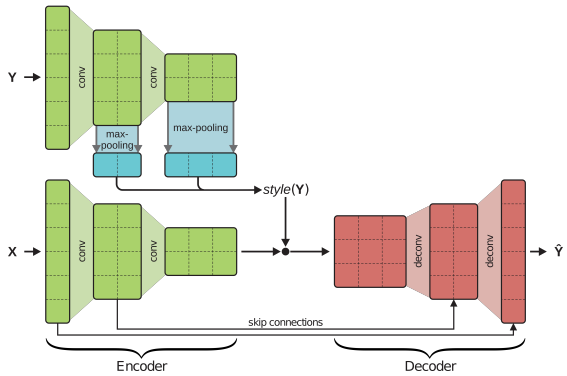


Fig. 7: Modified pix2pix generator network for conditional style transfer. The green network depicts the encoder of the pix2pix generator network. The max-pooling extracts the style information of all activation maps of style image Y. The style information gets concatenated with the output of the encoder for newly rendered image X used in the decoder to generate the output $\hat{Y}$.

Feeding the style information to the discriminator has the effect that the discriminator can now differentiate between real and generated images by comparing their style output. This forces the generator to generate images that include the style, as well as the style extraction network to produce meaningful styles so that the generator can use them.

3.5 Postprocessing

During pre-evaluations, we encountered that the online generation network of Graves [10] needs more contextual knowledge to adapt to the writing style. Thus, for our user study, where single words are evaluated, we duplicate the target words two times and take the last word as representative. In future, we would like to evaluate other generative models, e. g., STCNNs [1] for the online synthesis. Furthermore, we encounter that the conditional pix2pix model sometimes produces wrong pen colors while the stroke width and all other characteristics are transferred correctly. Therefore, we find the most important principal component of the style image and transfer it to the output image.

4 Evaluation

We conduct a large-scale user study to assess how well a human could be fooled by our method. Furthermore, we utilize writer identification methods to have a non-biased measurement of how well the results are imitating the original writer.

4.1 Datasets

The following datasets are used to train the models, perform the experiments and user studies.

IAM On-Line Handwriting Dataset. The IAM On-Line Handwriting Database [17] consists of handwritten samples recorded on a whiteboard. The dataset consists of 221 writers with 13 049 isolated and labeled text lines in total containing 86 272 word instances of a 11 059 large word dictionary.

CVL Dataset. The CVL Database [14] consists of offline handwriting data providing annotated text line and word images. We used the official training set consisting of 27 writers contributing 7 texts each.

Out-of-vocabulary Dataset. For the OOV words, we used the 2016 English and German Wikipedia corpora [8], containing both 100 000 words. Words containing less than four characters and words already part of the CVL training set are removed from these dictionaries. For the user study, the number of German words is reduced such that the ratio of English and German words is equal to the ones of the CVL dataset.

4.2 Implementation Details

Pix2pix Models. For the skeletonization, we make use of the proposed iterative knowledge transfer algorithm due to the nonexistent dataset for our problem description, see Sec. 3.1. We use the pix2pix framework [12] as the mapping network, trained with augmentations, such as added noise, resizing, cropping and color jitter. For the pen style transfer, we use another pix2pix model for which we employ an asymmetric version of the U-Net [21]. It is shortened to 4 layers (original 8) and reduced number of filters: [16, 32, 64, 64] for the encoder, [32, 64, 128, 128] for the style extractor, and [192, 256, 128, 64] for the decoder. The goal of this network is to synthesize a real-looking handwriting image given a skeleton, which was produced earlier through the pix2pix skeletonization, and a random sample of the CVL dataset as style input, cf. Sec. 3.4. Both models are trained with standard pix2pix [12] training parameters. We use ADAM with a learning rate of 0.0002, momentum parameters $\beta_1 = 0.5$, $\beta_2 = 0.999$, and $\lambda_{L_1} = 100$ for weighting the L_1 distance between Y and $\hat{Y}$.

Online Handwriting Style Transfer. The RNN model is trained using the IAM On-Line Handwriting Database. The model consists of 400 LSTM cells. It is trained with RMSProp with $\beta_1 = 0.9$, a batch size of 32 and a learning rate of 0.0001 and gradient clipping in $[-10, 10]$.

4.3 Qualitative results

Fig. 8 shows four distinctive outputs. Row 1 and 2 show three synthesized words, depicting two good results and one containing typical pipeline failures with the anchor (=style) line at top. Row 3 and 4 show two common outputs of the network with the anchor line on the left. For example, the addition of superfluous

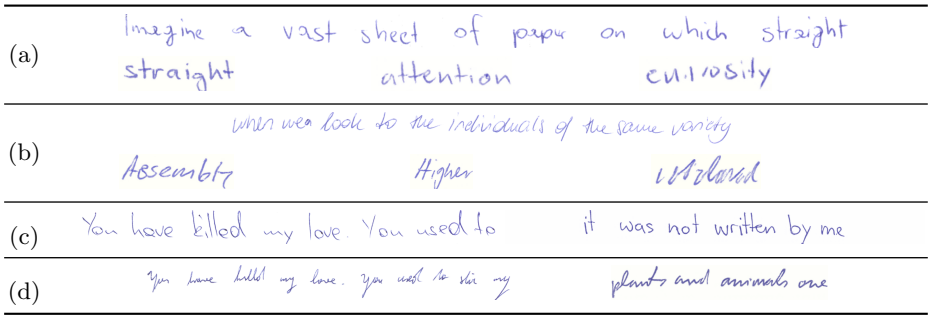


Fig. 8: (a) and (b) show three generated words each - two good results and one bad one with the given style line on top. (c) and (d) show two generated lines (right) with two style lines (left).

lines, incorrect splitting and positioning of lines, detail removal, such as missing t-, f- lines or i-dots, or skipped letters at the beginning of the message. We notice that the less legible a writer's handwriting is, the harder it is to imitate the writing style, see for example writer Figure 8b and 8d.

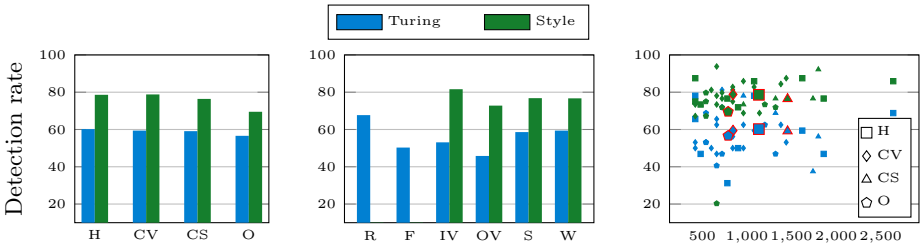
4.4 User Study

We conducted a user study[2] to estimate the degree to which humans can distinguish real from synthesized data. Random samples from the CVL dataset serve as real data, and the synthesized data is generated with words from the CVL and OOV corpora while line images from CVL are randomly chosen as style inputs. Two sub-studies were performed. In the first one (▬), humans are shown a sample and asked if it was "written by a human" or "created by machine". This is a kind of Turing test asking to differentiate between human and synthesized handwriting. In the second one (▬), we wanted to evaluate the ability to generate a person-specific handwriting. The users had to select between the assertion *"The LEFT word is written by the SAME writer as the line."* and *"The RIGHT word is written by the SAME writer as the line."*.

Every subject had to answer 96 questions without time limit: 32 for the *Turing* task and 64 for the *Style* task. For each task, a background representing notebook paper was artificially introduced [20,22] in exactly 50 % of the queries in order to investigate the impact of such a distraction. In total, 59 people participated in our user study with different knowledge background: *Humanities* (H), i. e., paleographers, book scientists, etc.; people working in *Computer Vision* (CV) or image processing; general *Computer Science* (CS); and *Other* (O).

Fig. 9a shows that for the Style task, the average results per user category range from 56.6 % to 60.2 % correctly recognized samples. Experts in the humanities are slightly more accurate than others. There is very little difference between working in the field of computer vision or image analysis and other computer scientists

[2] https://forms.gle/MGCPk5UkxnR23FqT9



(a) Demographics (b) Ablation study (c) Time [s]

Fig. 9: (a) shows the results of different people categories (H: humanities scholars, CV: computer vision/image analysis background, CS: computer scientist (general), O: other). (b) shows the results of different ablation studies (see text) while (c) shows the results obtained on either task individually, with regard to the total experimental time of each subject. Larger red encircled results are the means of each demographic group.

(59.4 % vs. 59.1 %). For the Style task, the average results per user category range from 69.5 % to 78.6 %. This time, however, people with computer vision background reach an accuracy similar to the one of scholars in the humanities (78.8 % and 78.6 %, respectively). The gap to computer scientists and others is larger with a difference of 2.4 %, and 9.3 %, respectively.

Fig. 9b shows several ablation studies. R marks the performance of humans on identifying real samples as such and F their performance on synthetic samples; it can be seen that humans can barely detect fake samples. It seems that samples generated from the training set (IV) are easier to distinguish than words coming from pages of an unknown corpus (OV). This might come from the usage of a different, more complicated lexical field. There is no accuracy difference when using samples with synthetic background (S) in comparison to real, white background (W). Finally, Fig. 9c depicts the total time users spent on the test. Most users needed between 8 and 24 minutes to answer the 96 questions. We can guess a trend: people taking more time tend to be, in average, more accurate, which was to be expected. Interestingly, people with computer vision background (CV) achieved about the same accuracy in fewer time than the humanities group (H).

4.5 Writer Identification with Synthesized Offline Handwriting

We tested samples of our proposed method against an automatic learning-free writer identification approach [19]. We made a joint retrieval database having our synthetic paragraphs and a paragraph of the CVL training set from all 27 writers of the dataset, i. e., a total of 54 samples. For every query sample, we removed the real sample of the writer in question and retrieved the most similar ones from the 53 database samples. This procedure effectively benchmarks our forged samples vs. the most similar writer to the real one, which could also be considered a forgery system. We used the specific experimental pipeline to obtain a strong baseline for forgeries and in order to asses the importance of the pix2pix

Table 1: Evaluation of generated vs. nearest human writer.

Query	DB	Skeleton	mAP %	Acc. %
Real	Real+Fake (OV)	Naive	31.94	18.52
Real	Real+Fake (IV)	Naive	26.18	7.41
Real	Real+Fake (OV)	pix2pix	29.66	14.82
Real	Real+Fake (IV)	pix2pix	37.13	25.92

skeletonization trained by our proposed iterative knowledge transfer vs. a naive skeletonization.

Tab. 1 (last row) shows that our method outperforms the nearest forgery on 25.9 % of the queries when our forgery is created with text from the same corpus. When the samples are generated with out-of-corpus words, performance drops significantly (row three). It can also be seen that substituting our proposed skeletonization for the naive skeletonization (row 1 and 2) has also a large impact on system performance. As a sanity check, we tested the retrieval of real writers from real writers, naive synthetic from naive synthetic, and pix2pix synthetic from pix2pix synthetic. In all cases, we achieved 100 % performance. It should be pointed out that automatic methods operate on the page/paragraph level while human users were challenged at the word level. Any comparison between humans and automatic writer identification methods must not be implied from this experiment. An extensive report on results from all automatic writer identification experiments is available in the supplementary material.

5 Discussion and Conclusion

In this work, we proposed a fully automatic method to imitate offline handwriting using spatial-temporal style transfer. The pipeline is capable of producing letters and words which are not in the vocabulary of the writer's samples and therefore could be applied to many use cases. We show that generated results just from a small number of writings show auspicious results, often indistinguishable from real handwriting. Every step of the pipeline is dependant of the previous actions. If the skeletonization doesn't robustly produce complete skeletons, the output in the end will be easily detectable as machine produced. Additionally, the pipeline is highly reliant on the online synthesis method. Graves method [10] has several shortcomings. The RNN needs a minimum amount of input composed of images and transcriptions to generate strokes with a new text sequence robustly. The current model has also problems with punctuation marks and synthesizing samples for writers with a bad handwriting does not work reliably. In future work, we will evaluate different online synthesis methods to improve this part of the pipeline.

We note that a concurrent work produces realistic words [13] using a GAN-based system. This could partially be used in our method for improved stylization, with the advantage that our method produces full line images.

References

1. Aksan, E., Hilliges, O.: STCN: Stochastic Temporal Convolutional Networks. arXiv e-prints arXiv:1902.06568 (Feb 2019) 3, 10
2. Aksan, E., Pece, F., Hilliges, O.: Deepwriting: Making digital ink editable via deep generative modeling. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. CHI '18, Association for Computing Machinery, New York, NY, USA (2018) 3
3. Alonso, E., Moysset, B., Messina, R.: Adversarial generation of handwritten text images conditioned on sequences. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). pp. 481–486 (Sep 2019) 2, 3
4. Bai, X., Ye, L., Zhu, J., Zhu, L., Komura, T.: Skeleton filter: A self-symmetric filter for skeletonization in noisy text images. IEEE Transactions on Image Processing **29**, 1815–1826 (2020) 4
5. Balreira, D.G., Walter, M.: Handwriting synthesis from public fonts. In: 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI). pp. 246–253 (Oct 2017) 3
6. Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A.C., Bengio, Y.: A recurrent latent variable model for sequential data. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 2980–2988. Curran Associates, Inc. (2015) 3
7. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische mathematik **1**(1), 269–271 (1959) 7
8. Goldhahn, D., Eckart, T., Quasthoff, U.: Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. pp. 759–765 (2012) 11
9. Gomez, R., Furkan Biten, A., Gomez, L., Gibert, J., Karatzas, D., Rusiñol, M.: Selective style transfer for text. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). pp. 805–812 (Sep 2019) 3
10. Graves, A.: Generating Sequences With Recurrent Neural Networks. arXiv e-prints arXiv:1308.0850 (Aug 2013) 2, 3, 4, 9, 10, 14
11. Haines, T.S.F., Mac Aodha, O., Brostow, G.J.: My text in your handwriting. ACM Trans. Graph. **35**(3) (May 2016) 3
12. Isola, P., Zhu, J., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5967–5976 (July 2017) 2, 9, 11
13. Kang, L., Riba, P., Wang, Y., Rusiñol, M., Fornés, A., Villegas, M.: GANwriting: Content-Conditioned Generation of Styled Handwritten Word Images. arXiv e-prints arXiv:2003.02567 (Mar 2020) 15
14. Kleber, F., Fiel, S., Diem, M., Sablatnig, R.: CVL-DataBase: An Off-Line Database for Writer Retrieval, Writer Identification and Word Spotting. In: Proceedings of the 2013 12th International Conference on Document Analysis and Recognition. pp. 560–564. ICDAR '13, IEEE Computer Society, Washington, DC, USA (2013) 5, 11
15. Lee, T., Kashyap, R., Chu, C.: Building skeleton models via 3-d medial surface axis thinning algorithms. CVGIP: Graphical Models and Image Processing **56**(6), 462 – 478 (1994) 5
16. Lian, Z., Zhao, B., Chen, X., Xiao, J.: Easyfont: A style learning-based system to easily build your large-scale handwriting fonts. ACM Trans. Graph. **38**(1) (Dec 2018) 3, 4
17. Liwicki, M., Bunke, H.: IAM-OnDB - an On-Line English Sentence Database Acquired from Handwritten Text on a Whiteboard. In: 8th Intl. Conf. on Document Analysis and Recognition. vol. 2, pp. 956–961 (2005) 5, 11

18. Nakamura, K., Miyazaki, E., Nitta, N., Babaguchi, N.: Generating handwritten character clones from an incomplete seed character set using collaborative filtering. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR). pp. 68–73 (Aug 2018) 4
19. Nicolaou, A., Bagdanov, A.D., Liwicki, M., Karatzas, D.: Sparse radial sampling lbp for writer identification. In: 2015 13th International Conference on Document Analysis and Recognition (ICDAR). pp. 716–720. IEEE (2015) 13
20. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. In: ACM SIGGRAPH 2003 Papers, pp. 313–318 (2003) 12
21. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. pp. 234–241. Springer International Publishing, Cham (2015) 11
22. Seuret, M., Chen, K., Eichenbergery, N., Liwicki, M., Ingold, R.: Gradient-domain degradations for improving historical documents images layout analysis. In: 2015 13th International Conference on Document Analysis and Recognition (ICDAR). pp. 1006–1010. IEEE (2015) 12
23. Shen, W., Zhao, K., Jiang, Y., Wang, Y., Bai, X., Yuille, A.: Deepskeleton: Learning multi-task scale-associated deep side outputs for object skeleton extraction in natural images. IEEE Transactions on Image Processing **26**(11), 5298–5311 (Nov 2017) 4
24. Yang, S., Liu, J., Yang, W., Guo, Z.: Context-aware text-based binary image stylization and synthesis. IEEE Transactions on Image Processing **28**(2), 952–964 (Feb 2019) 3
25. Zhang, T.Y., Suen, C.Y.: A Fast Parallel Algorithm for Thinning Digital Patterns. Commun. ACM **27**(3), 236–239 (Mar 1984) 5
26. Zhu, J., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 2242–2251 (Oct 2017) 5

Supplementary Material to "Spatio-Temporal Handwriting Imitation"

Martin Mayr[0000−0002−3706−285X], Martin Stumpf[0000−0002−2278−8500], Anguelos Nikolaou[0000−0003−3818−8718], Mathias Seuret[0000−0001−9153−1031], Andreas Maier[0000−0002−9550−5284], and Vincent Christlein[0000−0003−0455−3799]

Pattern Recognition Lab, Friedrich–Alexander-Universität Erlangen–Nürnberg
{firstname.lastname}@fau.de
https://lme.tf.fau.de/

Table 1: Evaluation of WI in various modalities of the data, metrics given in percent.

Query	DB	Skeleton	mAP	Acc.	Soft2	Soft3	Soft4	Soft5
Fake (OV)	Real(IV)	pix2pix	48.6	33.3	44.4	51.8	62.9	74.0
Fake (OV)	Fake(IV)	pix2pix	100.0	100.0	100.0	100.0	100.0	100.0
Fake (OV)	Real(OV)	pix2pix	49.2	33.3	44.4	62.9	66.6	70.3
Fake (IV)	Real (IV)	pix2pix	51.0	37.0	51.8	51.8	55.5	74.1
Fake (IV)	Fake(OV)	pix2pix	100.0	100.0	100.0	100.0	100.0	100.0
Fake (IV)	Real(OV)	pix2pix	52.3	40.7	44.4	59.2	62.9	66.6
Fake (OV)	Real (IV)	naive	54.5	40.7	51.8	62.9	70.3	74.0
Fake (OV)	Fake(IV)	naive	100.0	100.0	100.0	100.0	100.0	100.0
Fake (OV)	Real(OV)	naive	53.9	40.7	48.1	66.6	66.6	66.6
Fake (IV)	Real (IV)	naive	49.0	29.6	51.8	62.9	74.0	74.0
Fake (IV)	Fake (OV)	naive	100.0	100.0	100.0	100.0	100.0	100.0
Fake (IV)	Real (OV)	naive	52.1	37.0	51.8	62.9	66.6	66.6
Fake (IV)	Fake(OV)	naive/pix2pix	100.0	100.0	100.0	100.0	100.0	100.0
Fake (IV)	Fake(OV)	naive	100.0	100.0	100.0	100.0	100.0	100.0
Real (IV)	Real(OV)	naive	100.0	100.0	100.0	100.0	100.0	100.0

1 Additional Experiments

1.1 Learning-free WI Algorithm

Tab. 1 shows all combinations of query and retrieval database between different modalities. All experiments operate on text paragraphs from 27 writer identities as in the train-set of the CVL dataset. Specifically we produced three modalities for every sample, one with the proposed pipeline, one with the proposed pipeline substituting the pix2pix skeleton with naive skeletons, and a real sample from the CVL train-set. Every modality is represented by an in-vocabulary (IV) and an out-of-vocabulary (OV) sample. Fake and real IV samples share the same

Table 2: Evaluation of WI using a different WI algorithm and different Query/DB settings, metrics given in percent. (a) Human page 4 as database. (b) All human pages as database.

		(a) Single		(b) Full	
Query	Skeleton	mAP	Acc.	mAP	Acc.
Fake (OV)	Naive	44.9	29.6	32.4	29.6
Fake (IV)	Naive	39.3	18.5	32.7	29.6
Fake (OV)	pix2pix	46.0	29.6	38.5	40.7
Fake (IV)	pix2pix	47.5	29.6	36.0	37.0

transcription they are page 4 from the CVL dataset. While OV samples fake samples share the same transcription among them and OV real samples are page 1 from the CVL dataset. *SoftX* refers to the usual TOPX rates, i. e., average percentage that the correct writers are among the first X ranks.

1.2 Alternative WI Algorithm + Experiment using Full CVL Dataset

Tab. 2 shows additional writer identification results, where we evaluated a different writer identification pipeline. Therefore, we used a method by Christlein et al. [1], but replaced the CNN-based features with dirichlet-normalized SIFT [3] descriptors, which are whitened and dimensionality-reduced to 64 components by means of PCA. The rest of the pipeline stays unchanged, i. e., VLAD encodings [2] are computed using 100 clusters for k-means, which are aggregated using generalized max pooling [4] ($\lambda = 1000$). This process is repeated three times. All resulting representations are then jointly whitened by another PCA. For computing the k-means clusters and PCA matrices, we used the CVL test dataset. Similar to the learning-free method, which was employed throughout the main paper and in Tab. 1, Tab. 2a shows that the method works similarly well whether it uses non-vocabulary words or not. Furthermore, the skeletonization we proposed using the adapted pix2pix mesh has improved the skeletonization. The same behavior can be observed in Tab. 2b.

2 Additional Figures

2.1 Modification of the pix2pix Network for Pen Style Transfer

To include the style extraction network in the training process, we add the output of the style extraction network to the input of the discriminator, as shown in Fig. 1.

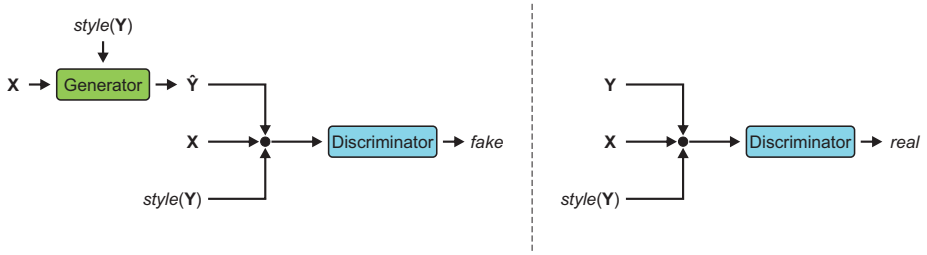


Fig. 1: Training process of pix2pix with added style information. GAN discriminators are always trained on both real and synthetic images, to learn how to distinguish between them. $\mathbf{X}$: input skeleton, $\mathbf{Y}$: real output image, $\hat{\mathbf{Y}}$: synthetic output image. *Left:* The training step for the generator and for the discriminator on synthetic images. *Right:* The training step for the discriminator on real images.

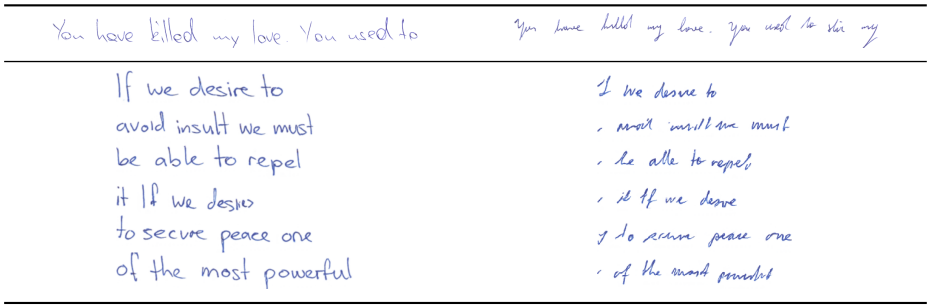


Fig. 2: Style input lines of two writers with their corresponding generated paragraphs.

2.2 Qualitative Results – Paragraphs

Fig. 2 illustrates different outputs of new paragraphs, i. e., containing out-of-vocabulary words. The left generated text looks convincing and is legible. Conversely, the right sample shows the output of a writer with a rather bad handwriting. This leads to artificats and an illegible output. Due to the fact that there is no groundtruth information of the punctuation, the pipeline sometimes produces artifacts at the beginning of the line when the style contains a punctuation.

2.3 User Study

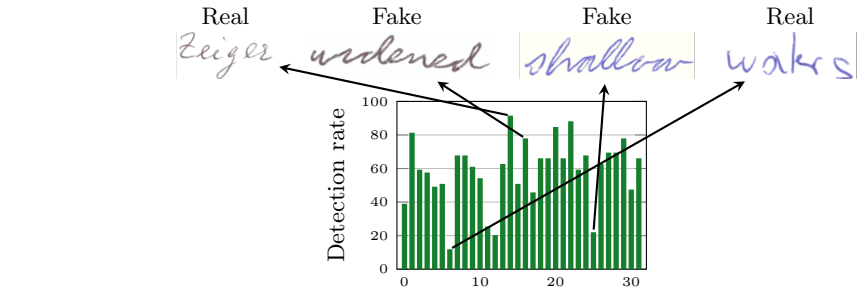
Tab. 3 shows the number of samples of in-vocabulary words (IV) and out-of-vocabulary words (OV) for both tasks. Note that also 50 % of all samples contain an additionally spliced background.

Fig. 3 shows the mean accuracy for each 96 samples. Additionally, the samples that were detected the best and the worst are depicted.

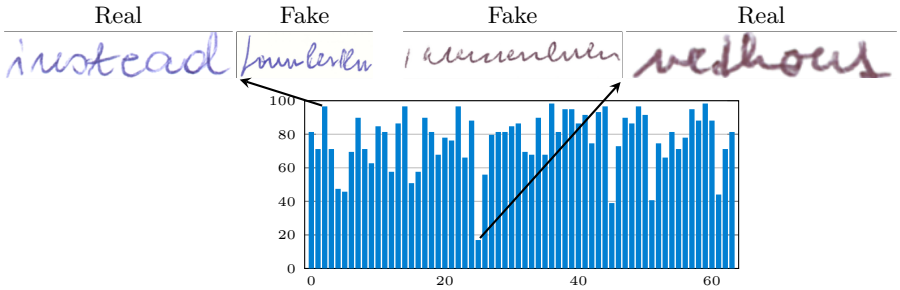
Fig. 4 and Fig. 5 show the individual results of all participants for the Turing and the Style tests. The user study comprised 59 individuals, 12 humanities

Table 3: Number of samples per category

	IV	OV	Total
Turing	10	6	16
Style	29	35	64



(a) Sample (Turing test)



(b) Sample (Style test)

Fig. 3: Per-sample results. Additionally, showing best and worst cases for real and fake samples, respectively.

scholars (H), 27 people of computer vision (CV), 10 computer scientists (CS), 10 others (O).

In Fig. 6 a contextualized and two non-contextualized samples can be seen as they were presented to the subjects. All samples can be seen below https://forms.gle/MGCPk5UkxnR23FqT9.

3 Failure Cases

During our evaluation, we observed errors which occur more frequently in different stages of the pipeline. Note that these artifacts are often not severe and through the last pen style transfer, most of these artifacts vanish again. However, they show potential improvements for future work.

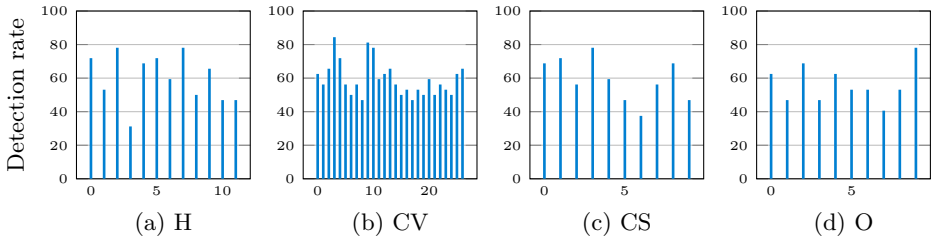


Fig. 4: All individual results (Turing test)

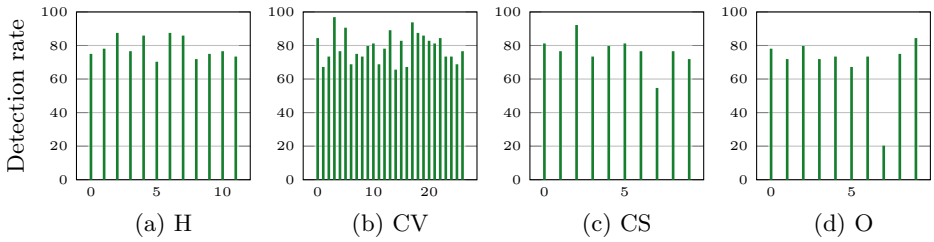


Fig. 5: All individual results (Style test)

Skeletonization The results of the *primitive skeletonization* often show merging of adjacent lines, breaking apart of continuous lines at intersections, removal of details or cutting corners, as seen in Fig. 7.

Also, *learned skeletonization* produces some common failure cases, described in Fig. 8. From top to bottom: input image, skeletonization network output, final skeleton. Fig. 8a and Fig. 8b show that crossing lines tend to be broken apart. Fig. 8c shows the loss of detail by over-smoothing. Fig. 8d shows another example where crossing lines get broken apart at the e and merged lines at the h, caused by the deblurring step. Fig. 8e shows a combination of them. The h gets distorted by both the network and the deblurring step, and the e ends up touching the h. The comparison of both methods is outlined in the main text of the paper.

Approximation of Online Representation Fig. 9 shows the most common errors occurring during the approximation of the online representation. The M of Fig. 9a demonstrates the inability of the algorithm to understand lines that are drawn on top of each other, causing the continuous line to be split into three segments. Further, the k and the e show cases where the algorithm fails to connect crossing lines properly. Fig. 9b describes the problem that the algorithm has with sharp edges. The entire word is mostly one continuous line, but the algorithm split it into lots of segments. Fig. 9c is an example for intersecting lines of the f which got connected incorrectly.

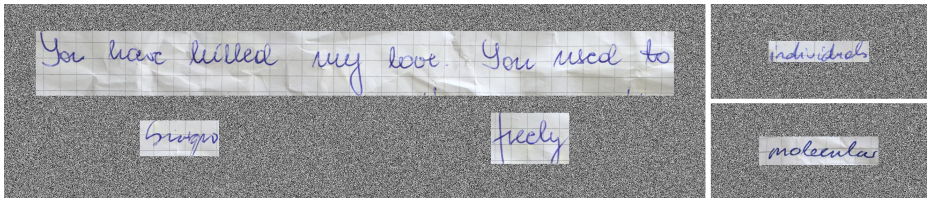
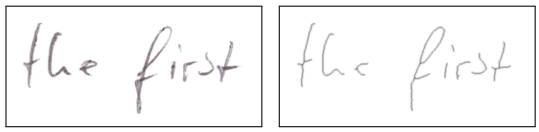


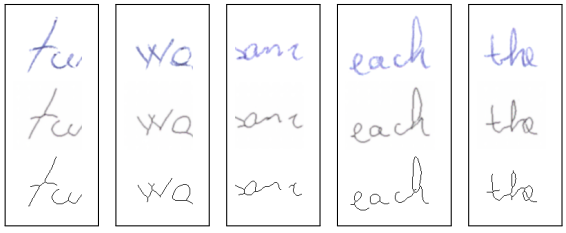
Fig. 6: Contextualized (left) and non-contextualized samples (right) as seen in the user study.

Writer Style Transfer The stage of the writer style transfer produces some common failures, like addition of superfluous lines (Fig. 10a), incorrect splitting of lines (Fig. 10b), incorrect positioning of lines, detail removal, like t-, f-lines or i-dots (Fig. 10c), skipping letters at the start of the message, which is an artifact that happens when the network was unable to correctly parse the input style.



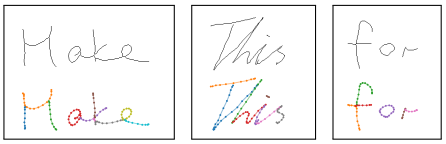
(a) real input image (b) primitive skeletonization

Fig. 7: Typical failure modes of the primitive skeletonization.



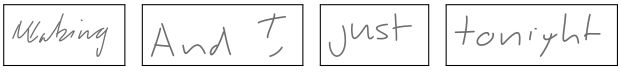
(a) (b) (c) (d) (e)

Fig. 8: Typical failure modes of the final skeletonization network.



(a) (b) (c)

Fig. 9: Typical failure modes of the online approximation.



(a) (b) (c) (d)

Fig. 10: Typical failure modes of the writer style transfer.

References

1. Christlein, V., Maier, A.: Encoding CNN Activations for Writer Recognition. In: 13th IAPR International Workshop on Document Analysis Systems. pp. 169—-174 (2018) 2
2. Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P., Schmid, C.: Aggregating Local Image Descriptors into Compact Codes. Pattern Analysis and Machine Intelligence, IEEE Transactions on **34**(9), 1704–1716 (2012) 2
3. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision **60**(2), 91–110 (2004) 2
4. Murray, N., Jegou, H., Perronnin, F., Zisserman, A.: Interferences in Match Kernels. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(9), 1797–1810 (2016) 2