Applying End-to-end Trainable Approach on Stroke Extraction in Handwritten Math Expressions Images

Elmokhtar Mohamed Moussa, Thibault Lelore, Harold Mouchère

Applying End-to-end Trainable Approach on Stroke Extraction in Handwritten Math Expressions Images

Elmokhtar Mohamed Moussa[1,2], Thibault Lelore[1][0000−0002−7083−2422], and Harold Mouchère[2][0000−0001−6220−7216]

[1] MyScript SAS, Nantes, France
{elmokhtar.mohamed.moussa,thibault.lelore}@myscript.com
[2] LS2N, University of Nantes, UMR 64004
harold.mouchere@univ-nantes.fr

Abstract. In this paper, we propose a novel end-to-end system to extract strokes from offline math expressions. Using a multi-task neural network we simultaneously predict the location of the pen and the pen state. Our approach is based on a recent state-of-the-art image-to-sequence method limited to small fixed-sizes images. We generalize it to large and multi-symbol images without preprocessing steps such as skeletonization or binarization. This architecture allows an end-to-end training. A curriculum learning strategy have been used to address the complexity of the images. We achieve comparable results to the state of the art on the UNIPEN English character dataset considering the next point prediction. We propose a stroke level metrics that allows us to measure the stroke reconstruction. Experiments show the advantages and limitations of the adopted Image-to-Sequence method when scaling up to large and complex images such as math equations.

Keywords: Stroke Extraction · End-to-end Trainable System · Handwritten Mathematical Expressions.

1 Introduction

Traditionally, handwriting recognition approaches are split in two categories: off-line systems which use as input an image of a scanned document; and on-line systems which use the pen trajectory recorded with e-pen or touch-sensitive surfaces. In most of real use cases, there is no choice in the input modality. Finger traces on a smartphone are on-line data, ancient documents are always digitized as images. However, each modality has its own advantages. The on-line signal keeps the dynamic of the pen trace which is useful for most of the recognition algorithms. In addition, on-line signals facilitate editing to the users. Image based systems have the advantage to better take into account local 2D context or the global layout of a full document. Multi-modal systems try to keep advantages from both worlds [18,19] but they need the two modalities.

Very few datasets provide both modalities natively [17]. Rasterization or on-line world to the off-line conversion is a straightforward problem. This is very useful to produce training pairs for multi-modal systems from on-line data [3] or to produce new training image samples (data augmentation) [10]. In this last case, rendering realistic raster images from on-line signals usually involves adding background noise, variation in pen tip width to simulate movement speed and different artifacts [7]. The reverse operation, off-line to on-line conversion, is mainly used in two different contexts.

The first one is the vectorization which attempts to model a line drawing image as a set of mathematical primitives (polygons, parametric curves, *etc.*) associated to vector elements found in vector image format such as SVG. Application can be for technical drawing vectorization [6] and 2D animation [7]. In this case, retrieving temporal information is less relevant. The ordering between the different primitives is not an interest here and parametric curves have no drawing direction.

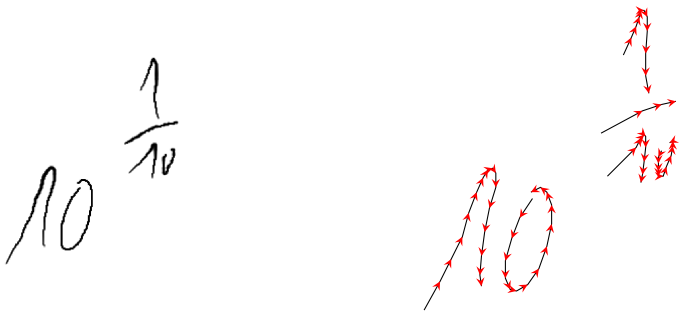
The second one is pen trajectory recovery from image of handwritten documents focus on retrieving the original temporal information. It is a crucial step for many applications, such as handwriting recognition and signature verification. The availability of temporal information in online systems often makes them better performing than their offline analogue [13]. In 2019, the Competition on Recognition of Online Handwritten Mathematical Expressions (CROHME) [12] included for the first time a offline recognition task. It has sparked since a great interest for offline to online conversion [4].

Recovering the stroke structure (or skeleton) is one of the initial steps in most document image analyses and understanding systems. It plays a key role in document processing since its performance affects quite critically the degree of success in a subsequent character segmentation and recognition. Degradation appear frequently and can be due to several reasons which range from the acquisition source type to environmental conditions [11]. Sketches and rough pencil drawings can add difficulties to the process when multiple overlapped lines should be merged into a single line [16]. Skeletonization is a ubiquitous step in classical approaches [13] of on-line to off-line conversion. Those approach suffer from the aforementioned limitations of skeletonization.

Based on the work of Zhao *et al.* [20], we propose a fully convolutional neural and multitask network, based on U-Net [15] to predict the pen location and the skeleton. A final fully connected layer is added for pen state classification. Pen state prediction enable the reconstruction of strokes and to halt the iteration framework (thanks to *END* state). Training data is generated using variable width strokes [7] making the system robust to stroke width variations.

1.1 Related Works

Pen trajectory recovery Over the years, researchers have proposed many methods to recover the temporal information. Usually they follow similar steps: topology extraction, local ambiguous regions detection (junctions, double traced



(a) (b)

Fig. 1: offline image (a) rendered from a math expression online signal (b).

strokes, *etc.*). The ambiguities are then resolved using handcrafted heuristics. Existing approach often fall into three categories: recognition-based, topology-based and tracking-based. Recognition-based methods [5] were initially addressed to drawing composed of regular shapes (diagrams, engineering drawing, *etc.*). They detect those shapes using geometric primitives. By nature, this approach is not well-suited to handwriting and limit the user to a limited graphical vocabulary. Topology-based approaches build a representation from topological information in original image (skeleton, contour, *etc.*) and express the pen trajectory recovery as a global or local optimization problem. Qiao *et al.* proposed a weighted graph approach to recover the pen trajectory by finding the best matching paths [14]. They achieve good performance on English characters. Tracking-based approach iteratively estimates the relative direction of the pen. Bhunia *et al.* applied sequence-to-sequence modeling with an end-to-end Convolutional-BLSTM network obtaining excellent results on Tamil, Telugu and Devanagari characters [2]. However, Their approach is limited to single isolated characters. Although the mentioned scripts are closely related, they train a separate model for each script. Zhao *et al.* [20] proposed an image-to-sequence iterative framework to generate pen trajectories with a CNN followed by fully connected layers to predict the pen position at each time step. They obtain good results on Chinese and English handwriting datasets. However, these approaches suffer from the same drawback: the complexity of the model is directly dependent on the offline image resolution. Small resolutions (such as 28×28 or 64×64) can be sufficient for characters level applications but will lead to illegible images in the case of larger content like math equations. Using higher resolutions implies a quadratic increase of the total number of parameters in the model, stemming from the fully connected layers. Their method also relies on the skeleton to guide the prediction and to end the iteration framework. Skeletonization of real-world offline image often results in noisy and incomplete skeletons.

Line drawing vectorization is an essential step for 2D animations and sketching. Vectorization focus on converting the drawing images to vector graphics.

Artist usually start by sketching up a first version of their work with a pen and paper and then manually vectorize and finalize their work digitally. Vectorization of rough and complex real-world sketches is a difficult task. Multiple overlapping lines should be merged into a single line, noisy background and non-essential lines (*e.g.* construction lines) need to be cleaned. Simo-Serra *et al.* [16] proposed a fully convolutional simplification network augmented with a discriminator network to clean high resolution sketches. Guo *et al.* [7] presented a two-phase method using two networks to vectorize clean line drawings. A first multi-task CNN extract the skeleton and junction images. The skeleton is subdivided to many lines by removing junctions. A second CNN reconstruct the line connectivity around junctions. They achieve state of the art on the public *Quick, Draw!* [9] dataset. Nonetheless, their approach is limited to junction of valence 3 to 6 of small size 32×32.

2 Proposed method

2.1 Overview

Our approach is based on Zhao *et al.* image to sequence method [20]: we design a pen trajectory prediction network to model pen position frames and infer the pen trajectory from handwriting offline images by iteratively predicting the next pen position with the said network. Our approach goes beyond limitations of the original method by being applicable to arbitrary image resolution, reconstructing the different strokes using pen states classification and eliminating the need for a preprocessing skeletonization step. The Fig. 2 shows the inputs and outputs of our model. The input of our network consists of fives images, the previous and current images $F_{i-1}, F_i \in \{-2, -1, 0\}^{h \times w}$, the grayscale offline image $I \in [0, 1]^{h \times w}$. We also provide the pixel coordinates as two images $I_X, I_Y \in [0, 1]^{h \times w}$. In fact, the network has a receptive field size of 32×32, the spatial clues we provide are global information that can help improve the network decisions in the local regions. The network outputs three images, the full skeleton I_S of the input image, all the stroke end points I_E, and the next pen position I_{POS}^{i+1}. Furthermore, we also predict a pen state in {Down, Up, End}. The redundancy in the different outputs allows to guide the training of the main task (the pen position and pen state). We train our network on synthetic off-line images with a variable stroke-width generated from the on-line signals *c.f.* 3.1. Contrary to [20], the network learns the skeleton image and can handle different image resolutions.

2.2 Network architecture

Motivated by the successful application of fully convolutional neural networks (FCNNs) in the recent work of [16] for sketch simplification, We adapt our model from the U-Net architecture [15]. U-Net is a FCNN used for image segmentation in biomedical applications. It consists of a downsampling and an upsampling

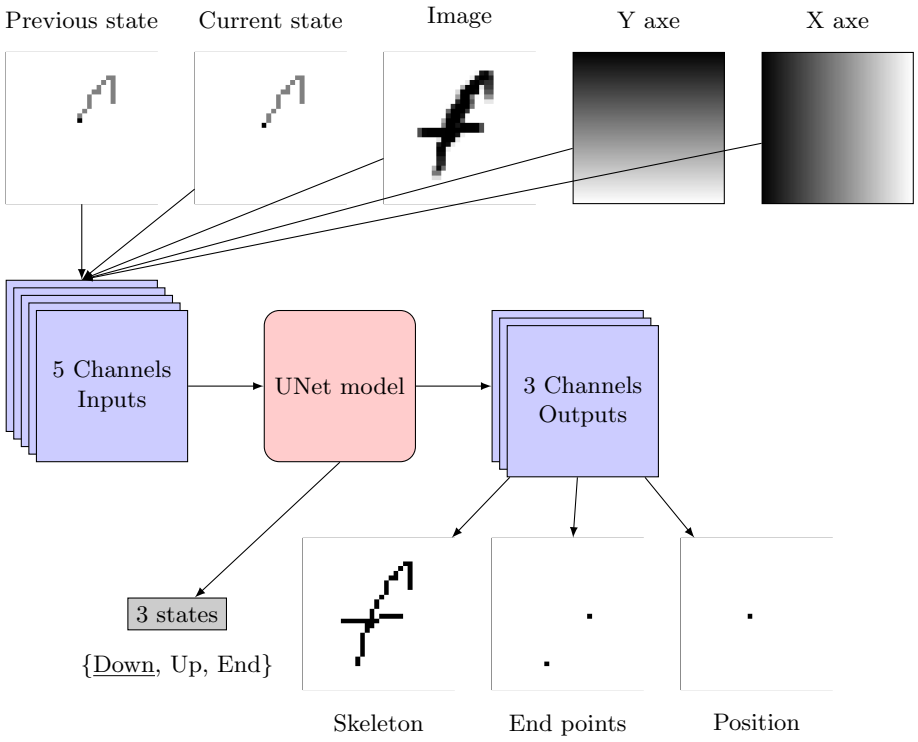


Fig. 2: Inputs and outputs of our model.

path. The downsampling path is a succession of 3×3 convolutions followed by $ReLU$ and a 2×2 max pooling layers. It can be seen as an encoder, that encodes the input in a small hidden feature map H. The upsampling path decodes the feature map H to the original resolution using up convolutions with a stride of 2×2. The high-resolution information is reused thanks to shortcut connections from the downsampling layers to the upsampling layers.

As shown in Fig. 3, the input frame images (F_{i-1}, F_i), the offline image I and the coordinate images I_X, I_Y are encoded to a small size hidden feature map $H \in \mathbb{R}^{448}$:

$$H = Encoder(F_{i-1}, F_i, I, I_X, I_Y).$$

The upsampling path decodes the feature map H to a map $O \in R^{h \times w \times 28}$ at the original resolution and outputs the target skeleton image $\hat{I}_S$, the stroke ends positions $\hat{I}_E$ and locate the next pen position $\hat{I}_{pos}^{i+1}$. The equations (1) to (3) explain how these images are computed.

$$O = \text{Decoder}(H) \tag{1}$$

$$\hat{I}_S, \hat{I}_E = \sigma(\text{conv}_2(O)) \tag{2}$$

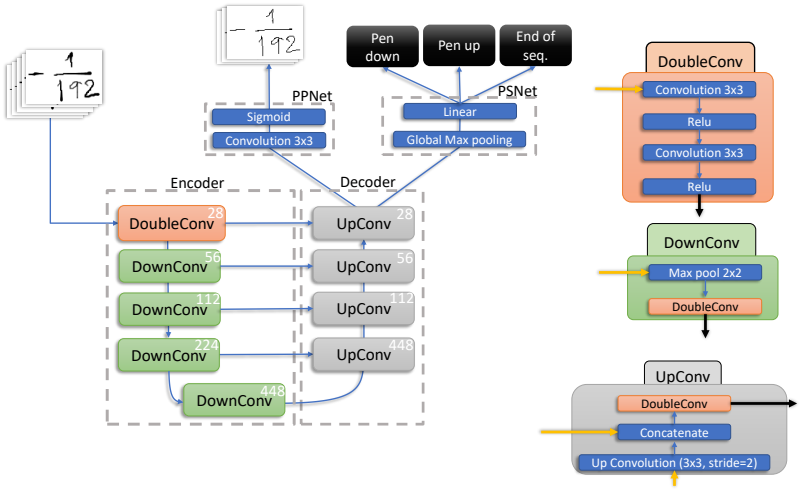


Fig. 3: Network architecture. The number of filters used in each convolution is shown on every block. The encoder and decoder are shared by PSNet and PPNet.

$$\hat{I}_{pos}^{i+1} = \sigma(\text{conv}_1(O) \odot I_S) \tag{3}$$

The conv_1 and conv_2 functions are classical convolution layers with respectively 1 and 2 kernels. The σ function is the sigmoid function. The product with the skeleton I_S before the output of the next position allows to constraint the next position to be on the predicted skeleton. This simplifies the task of the last conv_1 operation. A similar operation is done in [20] but as a post-processing step, using the skeleton extracted separately.

The encoder and decoder define the pen position prediction network `PPNet`. We modify U-Net by adding third path, a pen state classification network `PSNet` with one fully connected layer. We aggregate the decoder output O to a fixed size vector with global max pooling and input it to the classification layer:

$$\hat{P}_{i+1} = \text{PSNet}(O). \tag{4}$$

This allows the classification network to have a complete view of the input image whereas the decoder has a fixed size receptive field to make it prediction. Fortunately, the max pooling layers exponentially increase the receptive field. The pen can take three different state values. In addition to the standard *pen down* and *pen up*, we define an *end* state indicating that the scripter has finished writing. We consider that an *end* is a *pen up*, which implies multi-label classification. The *end* state is necessary, it's a stopping condition for the iterative framework used in the inference. Checking if every pixel from the skeleton has been visited (as done in [20]) is insufficient, as a scripter can draw certain pixels multiple times (pixels at junctions and double traced segments).

2.3 Loss functions

We train our network with a multi-task loss composed of a binary-cross entropy loss of the outputed pen state $\hat{P}$, a soft-F1 loss of the predicted skeleton $\hat{I}_S$ and for the predicted end of stroke positions $\hat{I}_E$ and L2 loss of the predicted pen position $\hat{I}_{pos}$.

$$\begin{aligned}
&\mathcal{L} = \mathcal{L}_P + \mathcal{L}_S + \mathcal{L}_E + \mathcal{L}_{POS} \\
&\mathcal{L}_P(\hat{P}, P) = \frac{1}{3} \sum_{y \in \{down, up, end\}} -(P_y \log(\hat{P}_y) + (1 - P_y) \log(1 - \hat{P}_y)) \\
&\mathcal{L}_S(\hat{I}_S, I_S) = \frac{1}{h \times w} \sum_{h,w} \text{softF1}(\hat{I}_S, I_S) \\
&\mathcal{L}_E(\hat{I}_E, I_E) = \frac{1}{h \times w} \sum_{h,w} \text{softF1}(\hat{I}_E, I_E) \\
&\mathcal{L}_{POS} = \frac{1}{h \times w} \sum_{h,w} \left\| I_{pos} - \hat{I}_{pos} \right\|_2^2
\end{aligned} \tag{5}$$

Where $I_S, I_E, I_{pos} \in \{0, 1\}^{h \times w}$ are the ground-truth skeleton, next pen position and strokes ends images.

3 Experimental results

This section describes the used data sets and their preparation, the training protocol with curriculum learning and finally the proposed evaluations using the stroke level metrics.

3.1 Dataset preparation

We use the online data from the CROHME 2019 [12] dataset to create a synthetic offline dataset of handwritten math expressions. We also use the isolated symbols of UNIPEN online handwriting [8] to allow a fair comparison with [20]. We apply the same prepossessing steps on both datasets.

The following preprocessing steps are applied to the online data:

- Symbol-wise normalization: the online signals are recorded with different resolutions and written in different handwriting sizes. To reduce handwriting size variation, we normalize the writing size to an average stroke diagonal size (so about a symbol size) of 32 pixels.
- Rasterization: We raster the online signals to gray-scale images with four stroke thickness selection strategies as in [7]. We vary the strokes-widths between 1 and 3 pixels.
- Frames generation: Each frame indicates the position of the pen. Similar to [20], we set background pixels to -2, already drawn pixels that are on the pen trajectory to -1. We encode the current pen position with 0 value.

The CROHME 2019 dataset provides 9,993 math equations for the training set, 986 and 1199 for the validation and test sets respectively (see table 1). To reduce computational time, the validation set is reduced to a smaller subset of 40 randomly selected equations. The Unipen dataset only contains small images and thus allows comparison of both systems. The CROHME dataset contains larger and more complex images with much more strokes per samples.

Dataset	Number	Training	Validation	Test
CROHME	# of equations	9,993	40	1,119
	# of strokes	137K	539	17K
	# of frames	6M	25K	770K
Unipen	# of characters	8,000	500	2,000
	# of strokes	12K	722	2,852
	# of frame	407K	26K	102K

Table 1: Dataset split between training, validation and test sets. And the corresponding total frame images.

3.2 Training

During training we follow a curriculum learning [1] strategy. We start by selecting frame images of one symbol (image resolution of 4096 pixels, *e.g.* 64×64) and equally sub-sample each pen state class. We compute the total f1-score of the substasks with eq. (6) on the validation set regularly during training. When the validation f1-score has not improved over 10 evaluation steps, the image resolution selection threshold is doubled. We stop at an image resolution of $202,752$ pixels (*e.g.* 256×792). As all images in a mini-batch should have the same size, each image is padded with white background inside each mini-batch. To optimize the mini-batch content, each one is built with images of about the same size. Its means that mini-batches with small images contain more samples than one with large images.

$$\begin{aligned}\mathcal{L}_C(\hat{I}_S, I_S) = &\frac{1}{h \times w} \sum_{h,w} \Big[\text{softF1}(\hat{I}_E, I_E) + \text{softF1}(\hat{I}_S, I_S) \\ &+\text{softF1}(\hat{I}_{pos}, I_{pos})\Big] + \sum_{y \in \{down,up,end\}} \text{F1}(\hat{P}_y, P_y)\end{aligned} \tag{6}$$

The Fig. 4 illustrates the training progression showing the F1 score of the validation dataset which always contains the same images, and the F1 score of the training set which is updated with new larger images every time the validation F1-score converges. As the validation set contains simple (small images with few strokes) and complex (large images with numerous strokes), the f1 score is

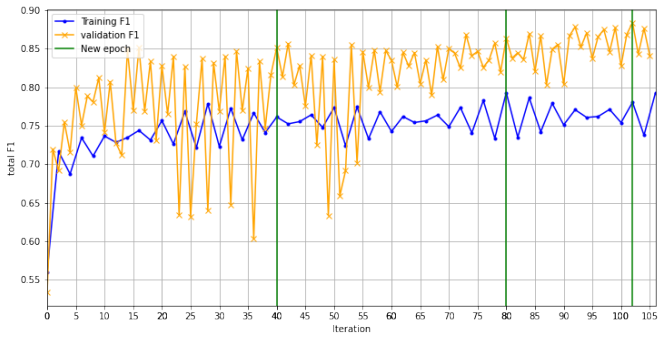


Fig. 4: Network training and validation learning curve. The different phases of curriculum learning are illustrasted with the green vertical lines.

progressing after each update of the training set. However the training score remains stable as complex images are smoothly introduced.

We use *Adam* optimizer with a learning rate of $1e^{-3}$. The maximum batch size is set to 10. The network is trained on a single NVIDIA GeForce RTX 2080 Ti 11GB GPU, taking 18 hours to be completed.

3.3 Evaluation & Metrics

In this section, we provide a comprehensive performance analysis of our network architecture. First we evaluate each subtasks such as skeletonization, next frame prediction and pen state classification. We then assess the efficiency of their combination in the iteration framework using an end-to-end image-to-sequence scenario. We also compare our results with [20] on the Unipen dataset.

Skeletonization, pen location and state prediction The output of `PPNet` jointly predicts the position of the next point, the structure of the skeleton as well as the end point of the strokes, in three separate output layers. We define the position of the next point as the pixel with the highest activation in the dedicated layer and compute the TOP-1 error as an evaluation metric for the pen position prediction. We choose the F1 score metric for the skeleton and end points extraction. To evaluate `PSNet`, we compute the per-class f1-score for pen state classification. The evaluation results on UNIPEN and CROHME datasets are listed in Table 2.

In comparison, [20] achieves a pen position top-1 error rate of 1.2% on UNIPEN. No public implementation of their work is available. We implemented and trained their neural network on UNIPEN. Some meta parameters were not available in the paper, thus we optimized them on the validation dataset. We obtained a 4.6% error rate. This result should be compared with the 15.5% of error obtained by our network on the same dataset in Table 2.

	Pen position error	Skeleton	End points	Pen state		
				Down	Up	End
CROHME	0.100	0.993	0.754	0.990	0.770	0.870
UNIPEN	0.155	0.939	0.6691	0.990	0.682	0.789

Table 2: Evaluation results of `PPNet` and `PSNet` on the Unipen and CROHME test datasets. The table shows the pen position TOP-1 error, the skeleton and end points f1-score and the per-class f1-score for pen state classification.

Stroke extraction We adapt the iteration framework [20] to extract strokes with our network. At every iteration, in addition to the next point prediction, the pen state output is used to recover the strokes. The next point is constrained to be on the skeleton by multiplying the heat map I_{POS}^{i+1} by the predicted skeleton I_S. We evaluate the proposed stroke extraction algorithm on UNIPEN and CROHME. We use the stroke intersection over union `SIoU` from [7] defined as

$$\texttt{SIoU} = \frac{1}{n} \sum_{i=1,\dots,n} \max_{j=1,\dots,m} \frac{P_i \cap \hat{P}_j}{P_i \cup \hat{P}_j}, \tag{7}$$

with n being the number of strokes in the ground-truth online signal and m the stroke number in the predicted one. A groundtruth stroke P_i is matched with the predicted stroke $\hat{P}_j$ with the highest `IoU`. We add a new metric, `SIoU 75%` which is the rate of strokes for which the `SIoU` is greater than 75%.

Table 3 compare these metrics for the two systems on UNIPEN dataset. The method from [20] does not provide pen up information. Thus, we consider that a pen up state has been reached if the next pen positions is not 8-connected to the current position.

Method	SIoU	SIoU 75%	Number of parameters
Zhao et al. [20]	0.3587	0.4673	17.8 Millions
Ours	0.568	0.283	5.9 Millions

Table 3: Stroke extraction evaluation and comparison on UNIPEN dataset.

We can observe that our approach has a better `SIoU` than the other system. However both have quite low results, in the best case, there is only 56.8% of matching between the predicted strokes and the ground-truth strokes. The `SIoU75%` shows that the reasons are different for both systems. It seems that our approach over segment the strokes (only 28.3% of the strokes match at more than 75%) and the system from Zhao et al. seems to merge strokes. We can also notice that our system has 3 times less parameters than the other one.

To better understand the behavior of the proposed system on large images, we study in Table 4 on different subsets of the CROHME data set depending of the number of strokes in the original ink. We can observe a small decrease

of SIoU for ink between 10 to 20 strokes, but the results are globally stable arround 53.2%. The low value of the SIoU75% shows that we over segment most of the strokes as only 22% of the stroke match more than 75% of the original stroke. However, we note surprisingly that the lowest result is for small expression (17.7% for less than 5 strokes). We think that this is because the system well succeeded in long straight strokes (as fraction bar, integral, equals, ...) which are rare in small expressions.

	[1,5]	]5,10]	]10,15]	]15,20]	]20,25]	All
SIoU	0.510	0.511	0.498	0.498	0.535	0.532
SIoU75%	0.177	0.178	0.192	0.217	0.229	0.220

Table 4: Matching rate of predicted stroke on the CROHME data set, considering different sizes of ink.

Figure 5 presents results of complete inference on two samples with different sizes. We can see in the original images the used variability in the pen styles (width and gray level). The second line in the figure shows the ground-truth strokes. We can see in the last line that the predicted strokes correctly follow the true skeleton. We can notice that the strokes end at the extremity of the skeleton and no symbol or part of symbol is missing. However, we globally observe an over segmentation of the ink. The system well predicts long strait lines (fraction bars, equal parts) and smooth curves (parts of the α, θ or t). The difficulties rise in the complex parts at crossing strokes (in *alpha*, θ symbols), at inflection points (in 7, 1, or tan symbols).

4 Discussions

One of the main difficulties we faced in this work was the absence of absolute ground-truth for the strokes order. In the case of equations, the great diversity which exists in the way of writing a single number, but also the various corrections made *a posteriori* on the beginning of an equation (*e.g.* the prolongation of a fraction bar) means that the order of the strokes captured by the user does not always correspond to a logical and semantically valid visual order. One solution to overcome this issue would be to use an online recognition system to evaluate if the produced ink is recognizable. We think that it would not completely overcome this problem because these systems generally put a rather strong prior knowledge on the dynamics and the temporality of the entries, and for the moment none produce a perfect recognition. Moreover, the non-differentiable nature of this approach prevents the backpropagation from using the stroke order tolerance of the online math recognition engine to improve the stroke extraction. That is why we proposed the SIoU and SIoU75% metrics which evaluate the

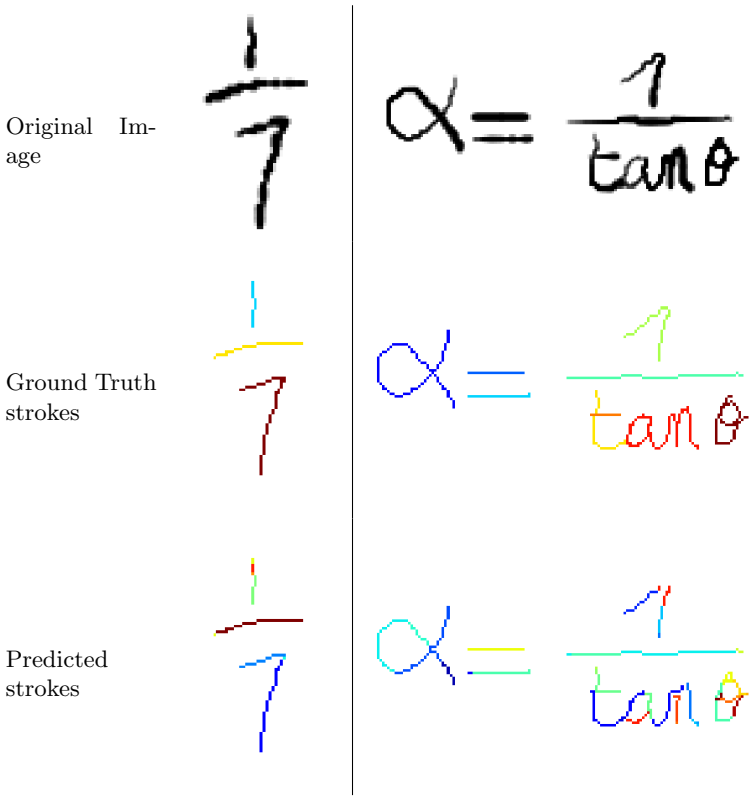


Fig. 5: Stroke extraction visualisation for a simple and a more complex image. Each stroke is drawn with a different color (better view in color).

strokes independently of their global order. However, the stroke direction (order of points in a stroke) should be evaluated, but the over/sub-segmentation of strokes makes it difficult. A metric allowing this evaluation still needs to be proposed (maybe based on DTW distance).

The proposed approach very well succeed to produce the skeleton and constraining the next pen position to be on the skeleton was an interesting proposal. However, the pen state strategy proposed too much pen-up. This decision needs local and global context. Indeed, the same local configuration (*e.g.* crossing points) can be solved differently depending on the symbol level context. On the one hand, the global pooling layer allows a concurrency between all candidate points of the image for this type of decision. On the other hand, it brings confusion on difficult points. The network favors solving obvious regions (*e.g.* straight lines) of the image before ambiguous regions (*e.g.* junctions), resulting in an over segmentation of the strokes. Furthermore, the network learns a short temporal transition ($t \rightarrow t+1$) from a temporal context limited to the previous state $t-1$ therefore, by design it doesn't necessarily learn to model longer term temporal order.

5 Conclusion

In this paper, we presented image-to-sequence approach based on FCNN network to extract strokes from off-line images. The network simultaneously predicts the next pen position, skeleton image, stroke end points and pen state at a given time step. The iterative framework from [20] is complemented with the pen state information enabling stroke extraction. To the best of our knowledge, we are the first to tackle stroke extraction from arbitrary image resolution with a neural network approach. The skeletonization and stroke end points extraction shows good results that encourage a convolutional approach. However, we show the limitations of the Image-to-sequence with a FCNN approach on this type of problem, the network over segments the strokes because of a lack of mid-term target. A solution can be to model longer temporal transition and to provide a longer temporal context combining an attention based approach with a recurrent framework, keeping the CNN backbone.

References

1. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09. pp. 1–8. ACM Press, Montreal, Quebec, Canada (2009). https://doi.org/10.1145/1553374.1553380
2. Bhunia, A.K., Bhowmick, A., Bhunia, A.K., Konwer, A., Banerjee, P., Roy, P.P., Pal, U.: Handwriting Trajectory Recovery using End-to-End Deep Encoder-Decoder Network. In: 2018 24th International Conference on Pattern Recognition (ICPR). pp. 3639–3644 (Aug 2018). https://doi.org/10.1109/ICPR.2018.8546093

3. Bluche, T., Louradour, J., Messina, R.: Scan, Attend and Read: End-to-End Handwritten Paragraph Recognition with MDLSTM Attention. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). vol. 01, pp. 1050–1055 (Nov 2017). https://doi.org/10.1109/ICDAR.2017.174
4. Chan, C.: Stroke Extraction for Offline Handwritten Mathematical Expression Recognition. IEEE Access **8**, 61565–61575 (2020). https://doi.org/10.1109/ACCESS.2020.2984627
5. Doermann, D., Intrator, N., Rivin, E., Steinherz, T.: Hidden loop recovery for handwriting recognition. In: Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition. pp. 375–380 (Aug 2002). https://doi.org/10.1109/IWFHR.2002.1030939
6. Egiazarian, V., Voynov, O., Artemov, A., Volkhonskiy, D., Safin, A., Taktasheva, M., Zorin, D., Burnaev, E.: Deep Vectorization of Technical Drawings. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020, vol. 12358, pp. 582–598. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-58601-0_35
7. Guo, Y., Zhang, Z., Han, C., Hu, W., Li, C., Wong, T.T.: Deep Line Drawing Vectorization via Line Subdivision and Topology Reconstruction. Computer Graphics Forum **38**(7), 81–90 (2019). https://doi.org/10.1111/cgf.13818
8. Guyon, I., Schomaker, L., Plamondon, R., Liberman, M., Janet, S.: Unipen project of on-line data exchange and recognizer benchmarks. In: Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 - Conference C: Signal Processing (Cat. No.94CH3440-5). vol. 2, pp. 29–33 vol.2 (1994). https://doi.org/10.1109/ICPR.1994.576870
9. Ha, D., Eck, D.: A NEURAL REPRESENTATION OF SKETCH DRAWINGS p. 16 (2018)
10. Le, A.D., Indurkhya, B., Nakagawa, M.: Pattern generation strategies for improving recognition of Handwritten Mathematical Expressions. Pattern Recognition Letters **128**, 255–262 (Dec 2019). https://doi.org/10.1016/j.patrec.2019.09.002
11. Lelore, T., Bouchara, F.: FAIR: A Fast Algorithm for Document Image Restoration. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(8), 2039–2048 (Aug 2013). https://doi.org/10.1109/TPAMI.2013.63
12. Mahdavi, M., Zanibbi, R., Mouchere, H., Viard-Gaudin, C., Garain, U.: ICDAR 2019 CROHME + TFD: Competition on Recognition of Handwritten Mathematical Expressions and Typeset Formula Detection. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). pp. 1533–1538 (Sep 2019). https://doi.org/10.1109/ICDAR.2019.00247
13. Nguyen, V., Blumenstein, M.: Techniques for static handwriting trajectory recovery: A survey. In: Proceedings of the 9th IAPR International Workshop on Document Analysis Systems. pp. 463–470. DAS '10, Association for Computing Machinery, New York, NY, USA (Jun 2010). https://doi.org/10.1145/1815330.1815390
14. Qiao, Y., Nishiara, M., Yasuhara, M.: A framework toward restoration of writing order from single-stroked handwriting image. IEEE transactions on pattern analysis and machine intelligence **28**(11), 1724–1737 (Nov 2006). https://doi.org/10.1109/TPAMI.2006.216
15. 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. Lecture Notes in Computer Science, Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28

16. Simo-Serra, E., Iizuka, S., Ishikawa, H.: Mastering Sketching: Adversarial Augmentation for Structured Prediction. ACM Transactions on Graphics **37**(1), 11:1–11:13 (Jan 2018). https://doi.org/10.1145/3132703
17. Viard-Gaudin, C., Lallican, P.M., Knerr, S., Binter, P.: The IRESTE On/Off (IRONOFF) dual handwriting database. In: Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318). pp. 455–458 (Sep 1999). https://doi.org/10.1109/ICDAR.1999.791823
18. Vinciarelli, A., Perone, M.: Combining online and offline handwriting recognition. In: Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings. pp. 844–848 (Aug 2003). https://doi.org/10.1109/ICDAR.2003.1227781
19. Zhang, J., Du, J., Dai, L.: Track, Attend, and Parse (TAP): An End-to-End Framework for Online Handwritten Mathematical Expression Recognition. IEEE Transactions on Multimedia **21**(1), 221–233 (Jan 2019). https://doi.org/10.1109/TMM.2018.2844689
20. Zhao, B., Yang, M., Tao, J.: Pen Tip Motion Prediction for Handwriting Drawing Order Recovery using Deep Neural Network. In: 2018 24th International Conference on Pattern Recognition (ICPR). pp. 704–709 (Aug 2018). https://doi.org/10.1109/ICPR.2018.8546086