Obstacles with Synthesizing Training Data for OMR

Jiří Mayer*, Pavel Pecina†
Institute of Formal and Applied Linguistics
Charles University, Prague, Czech Republic
Email: *mayer@ufal.mff.cuni.cz, †pecina@ufal.mff.cuni.cz
ORCID: *0000-0001-6503-3442, †0000-0002-1855-5931

Abstract—**Training with synthetic data has been successfully used in many domains of deep learning where authentic training data is scarce. Optical Music Recognition (OMR), especially recognition of handwritten music, greatly benefits from training on synthetic data too. In this paper, we explore the challenges of synthesizing images of sheets of music for training deep learning OMR models and compare such synthesis to the process of digital music engraving. We also contrast that with the architecture of our synthesizer prototype, which was used to achieve state-of-the-art results by training on the synthetic images only.**

Index Terms—**Optical Music Recognition, Synthetic Training Data, Data Augmentation, Deep Learning**

I. Introduction

Most recent advances in Optical Music Recognition (OMR) have been possible thanks to the use of deep learning models [1], [4]–[6]. These models, however, require large amounts of annotated training data, which are difficult to obtain for this task. Manual annotations in complex schemes, such as *Music Notation Graph* (MuNG) [3], are very costly to be produced in amounts required by supervised deep learning models [2].

In other domains, the exploitation of synthetic training data has greatly helped with this problem, covering various areas of computer vision, such as handwritten text recognition, natural scene text recognition, or optical-flow estimation [10]–[16]. Synthetic data is generated by a computer simulation of a real-world process. It can be used in situations where the data-generating process can be accurately simulated.

This paper outlines the challenges of synthesizing training data for OMR, both printed and handwritten. The second part of this paper describes the inner workings of our handwritten music synthesizer prototype *Mashcima* (Figure 1, top). We evaluated suitability of our synthetic images for OMR model training in our previous paper [24], where a model trained on our synthetic data was compared to other previously published state-of-the-art models, and the results indicated superior performance of our approach.

II. Synthesis of Music Scores

The idea of using synthetic training data for OMR is not completely new. Calvo-Zaragoza and Rizo [6] published the

This work described in this paper has been supported by the Czech Science Foundation (grant no. 19-26934X), CELSA (project no. 19/018), and has been using data provided by the LINDAT/CLARIAH-CZ Research Infrastructure (https://lindat.cz), supported by the Ministry of Education, Youth and Sports of the Czech Republic (project no. LM2018101).

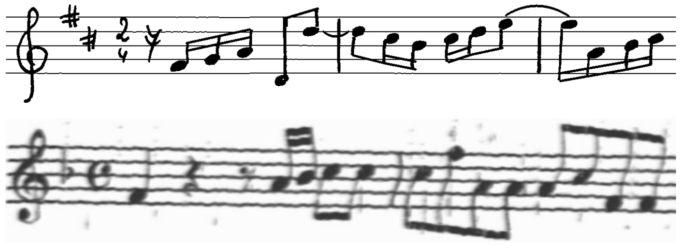
Fig. 1. A sample image created by the Mashcima synthesizer using symbols from single writer (41) from the MUSCIMA++ dataset (top) and a sample image from the Camera-PrIMuS synthetic dataset (bottom).

PrIMuS dataset containing images of printed monophonic music that were created digitally using the engraving tool Verovio[1]. An augmentation of this dataset was introduced later, called Camera-PrIMuS (Figure 1, bottom). It contains the same music as PrIMuS but the images are distorted and blurred to simulate the process of taking pictures of physical sheets of music under various conditions [7]. A similar approach was also used by Baró et al. [4] to produce images of historical music documents. The DeepScores dataset of printed music [8] was created in a similar way to the PrIMuS dataset but was not constrained to monophonic music. DoReMi [9] is a new synthetic dataset that also contains printed music and it provides multiple annotation schemes, each best suitable to a different stage of OMR.

In all of the above-mentioned works, the synthetic data was produced using some music engraving tool – a software for producing *printed* sheet music. There are no well-used datasets of synthetic handwritten music, but efforts in such direction exist. Baró et al. [5] introduced a data augmentation method based on measure shuffling where a staff of handwritten music from the MUSCIMA++ dataset [2] was sliced into individual measures and those were then shuffled and joined again. Our synthesizer prototype is based on the same idea taken further – shuffling individual symbols [24].

The production of printed music is called *engraving* and there is a variety of software tools available for this task (MuseScore, Verovio, and Lilypond, to name a few). Using them to produce synthetic training data is possible but requires certain modifications. Training data should be diverse in style

[1]https://www.verovio.org/

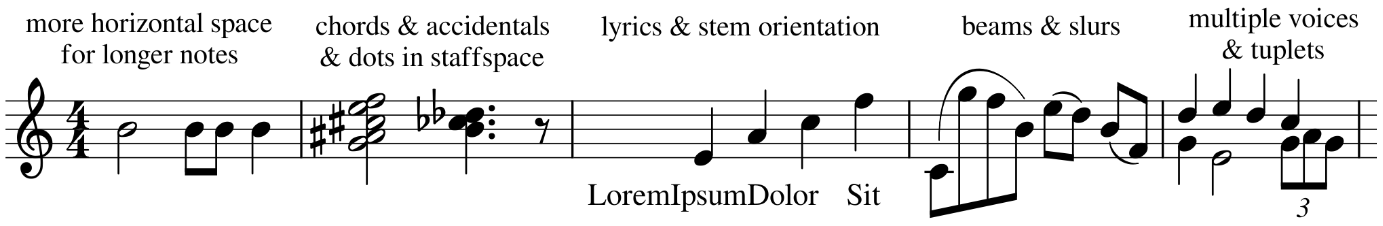
more horizontal space for longer notes
chords & accidentals & dots in staffspace
lyrics & stem orientation
beams & slurs
multiple voices & tuplets
LoremIpsumDolor Sit
3

Fig. 2. Demonstration of various challenges of music engraving process.

so that the trained model learns to generalize to unseen music. Such diversity can be increased e.g. by using multiple music fonts [6], [8] or by distorting and blurring the produced images to simulate imperfections introduced by scanning and paper degradation [4], [7], [23].

While these modifications let us better synthesize printed music, synthesis of handwritten music requires much more control over the process. There are music fonts that mimic a handwritten style, such as the Petaluma font family[2], but the resulting image still contains the same symbol shape in all of its occurrences. Similarly, the layout system will not introduce any variability to symbol placement. This is what motivated us to build a dedicated handwritten music synthesizer, instead of modifying an existing engraving tool [24].

Straightforward candidates for data synthesis are generative deep learning models, such as Generative Adversarial Networks (GANs) [17] that were successfully used to generate synthetic images of faces, text, or even cuneiforms [11], [18], [19]. While GANs are typically suitable for generating individual symbols, synthesizing entire sheets of music is very difficult. They also have limited control over their output and require large amounts of training data. For all these reasons, GANs seem as not a solution for synthesizing OMR data.

A. Complexities of Digital Music Engraving

This section provides a brief overview of music engraving, because engravers are currently used for printed music synthesis and we believe most of their architecture will likely be reused in a handwritten music synthesizer.

The defining problem of music engraving is the positioning of musical symbols on a blank page. An empty music document contains only stafflines and these also define the spatial unit – the staff space (Figure 3).

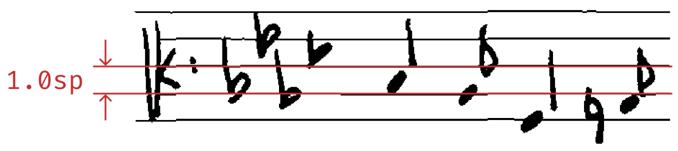

Fig. 3. Definition of the staff space (sp) unit (image from CVC-MUSCIMA).

There are many ways to draw a graph-like structure, for example, a force-directed layout could in-theory be used [20]. In practise, though, a more bottom-up approach is typical. MuseScore places objects called *segments* from left to right onto the staff. The horizontal size of each segment is computed based on its type and content, e.g. a chord, rest, barline, clef. While this segment approach solves the high-level horizontal placement, there are lots of specialized systems and rules that handle various edge-cases within and in-between segments. The complexity of music engraving stems from the unexpected interactions of these systems, resulting in awkward spacing or symbol collisions. Below, we provide a list of some of the challenges and illustrate them in Figure 2.

- **Horizontal spacing:** more space for longer notes and distributing evenly the remaining staff space[3]
- **Note clusters:** placing notes in a chord to not overlap
- **Stem orientation:** note orientation in general (pointing up or down)
- **Accidentals and ornaments:** allocating additional space for accidentals, preventing multiple accidentals from overlapping, placing dots in spaces and not on stafflines
- **Beam placement:** stretching stems to meet the beam, tilting the beam
- **Slurs and ties:** shaping a slur to not intersect notes, yet stay close to them
- **Multiple voices:** simultaneous notes from all voices should be horizontally aligned, two nearby voices must have stems pointing away from each other
- **Lyrics:** width of lyrics may influence note positioning
- **Tuplets:** tuplet markings and numbers interact with beams and slurs
- **Grace notes:** share features of notes and ornaments
- **Dynamics:** hairpins, dynamics, piano pedal marking
- **Text:** other text around the music, chord names, chord fingering diagrams, etc.

The precise documentation of how these subsystems are implemented by the existing tools is usually not available. However, most of the engraving systems we mentioned are open-source and the technical details can be inferred from their code. Some of them feature active forums and communities willing to answer questions (e.g., MuseScore, see *Engraving improvements in MuseScore 4.0*[4]).

The vector shapes of the symbols come from a *Music Font* where the very basic musical symbols (noteheads, stems, flags) are mapped onto Unicode characters and stored in a font file,

[2]https://www.smufl.org/fonts/

[3]https://musescore.org/en/node/299741 at 13th of October 2022

[4]https://musescore.org/en/node/330793 at 13th of October 2022

such as OTF[5]. The mapping from musical symbols to Unicode characters is standardized as SMuFL (Standard Music Font Layout)[6].

B. Technical Differences in Handwritten Synthesis

Most existing handwritten music datasets contain images in raster form because they come from scanned physical documents. It is therefore convenient to have the synthesizer use and produce images in raster form as well. This is in opposition to engraving systems which work with vector images only. The consequence of this is that we need to harmonize the data resolution throughout the synthesis process.

We should mention that some datasets of music symbols contain richer information. For instance, HOMUS [22] contains symbols that are stored as a path, traced by a stylus or a finger, as it was drawn. This opens doors for sophisticated pen models that could learn specific handwriting styles. However, to the best of our knowledge, no such attempts have been published so far and we will focus on raster-based generation pipelines.

C. Understanding the Style of Handwriting

The most important difference between printed and handwritten music synthesis is the need to model the diverse visual style of handwritten music. We identify three areas that together compose the handwritten style:

- Layout style
- Symbol style
- Environment style

Layout style is the way in which the writer positions musical symbols relative to each other. In the previous section we described how engraving systems focus heavily on achieving the perfect layout. Such level of precision is not necessary for handwritten music synthesis because the hand is not as precise as a printer. While the layout style can, and should, be modeled, Figure 4 shows that our synthesizer with fixed layout rules still preserves most of the overall style of the source writer. To learn the layout style for future synthesizers, we can leverage the dataset MUSCIMA++ [2].

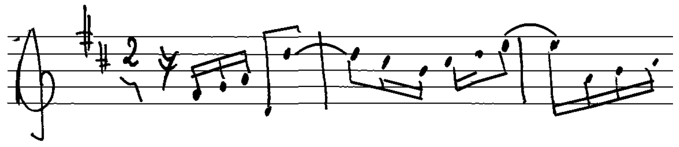
Fig. 4. A synthetic image with the sloppy handwriting style of writer 49 of the MUSCIMA++ dataset.

Engraving systems usually justify the staff to fit the width of the paper. Some human writers do this as well, first laying out measures and then filling them in, while others just write eagerly from left to right and leave empty space at the end of the staff when no more measures fit.

[5]http://www.microsoft.com/en-us/Typography/OpenTypeSpecification.aspx
[6]https://www.smufl.org/

TABLE I
SYNTHESIS STAGES IN MASHCIMA

Phase	Tasks
1. Input parsing	Building internal representation, adding cross-references (beams, slurs)
2. Symbol synthesis	Selecting symbol images
3. Layout synthesis	Placing notes (with ornaments), orienting stems, placing beams, placing slurs
4. Render	Building the final image

Symbol style describes shapes of individual symbols, such as noteheads, stems, accidentals. A music font cannot be used, because each instance of the same symbol has slightly different appearance. This is probably the most interesting area of handwritten music synthesis and a lot of research can be performed here. Smaller symbols like noteheads and accidentals can be synthesized by convolutional models, elongated symbols like stems, slurs, and beams could benefit from sequential models. These sequential models could be trained on online symbol datasets, such as HOMUS [22]. We would argue that a pen model also belongs to this category, as certain pens affect the way a music symbol is drawn.

We define the *environment style* as all the influences that turn an ideal binarized image to the actual scanned or photographed image. This includes paper degradation, camera distortions, blur, noise, stains, etc. Erosion and dilation can also be used to alter the pen style. These methods have been used before in printed music recognition and more general document recognition [4], [7], [23]. We can call this environment simulation process *postprocessing*.

III. Prototyping a Handwritten Music Synthesizer

In this section, we present a prototype implementation of a handwritten music synthesizer that can be used to generate training data for OMR. It is called *Mashcima* and is available on GitHub[7]. Initially, we tried to modify existing tools to produce handwritten-like music scores, however the particularities of handwritten music described in the previous section lead us to build a new tool. Our experiments show that training with synthetic data produced by Mashcima lead to the state-of-the-art model for handwritten OMR [24].

A. Synthesizer Pipeline

The synthesizer accepts as input a sequential encoding containing some monophonic music. This encoding is very similar to the *PrIMuS Agnostic Encoding* [6]. Agnostic means the encoding contains note positions, rather than note pitches, making the recognition task slightly easier. Then, four stages shown in Table I process this input and build the resulting image. This image contains only one staff, as seen in all examples in this paper. Mashcima was designed for monophonic music because it simplified the synthesis and allowed us to use the PrIMuS dataset as a source of input sequences [6].

[7]https://github.com/Jirka-Mayer/Mashcima

The first stage converts the input encoding into the internal representation. The core object is the *Canvas*, which acts as a container for *Canvas Items* and implements the layout synthesis logic. *Canvas Items* are larger groups of symbols that take up horizontal space on the staff, for example a quarter note with accidental and a duration dot; a clef; or a key signature. A staff is represented as a sequence of *Canvas Items* that are placed next to each other, similar to how text characters are arranged by a word processor (Figure 5). In this phase we also construct representations of beams and slurs, which break the sequential nature of the representation.

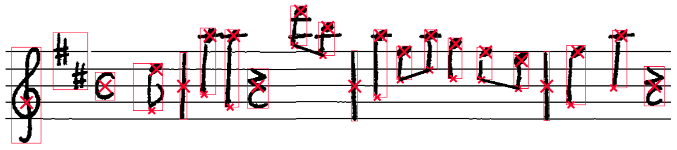
Fig. 5. *Canvas Items* (the red boxes) form the basis of horizontal spacing. The red crosses denote key positioning points, such as notehead centers and stem tops.

The second stage performs symbol synthesis. In the previous section of this paper we described various symbol synthesis approaches, but our synthesizer uses the simplest approach possible – printing symbol bitmaps from symbol datasets with no modifications. Specifically, we use symbol masks from the MUSCIMA++ dataset [2], including the empty staff image onto which other symbols are placed. This makes the synthesized images have similar style as the dataset. In the future we plan to utilize additional symbol datasets [22], [25], [26].

The bitmap sampling can be restricted to a subset of writers, giving us control over the handwriting style. The only modification we do to the sampled bitmaps is the vertical stretching of stems so that they meet the beam. Also, beams and slurs are not sampled due to their varied appearance and complex shape (they should align well with other symbols). Instead, they are rendered as straight lines and parabolic arcs, respectively. They are the least believable parts of the synthetic score, but a dedicated synthesizer would be too costly for a prototype.

The third synthesis stage is responsible for positioning symbols relative to each other. It is performed eagerly from left to right, where each *Canvas Item* requires some space and variable padding is added in between. Before a *Canvas Item* is placed, we first calculate its internal layout depending on its content, for example, accidentals and dots around notes make the *Canvas Item* wider.

Most of the layout rules determining a symbol position have the form of a fixed offset plus a random offset drawn from a uniform distribution. This introduces variability into the final score, however, this variability has the same hardcoded distribution for all writers. We do not perform any layout style learning.

In the final rendering phase, bitmaps are copied onto the output image. These bitmaps are binarized, but their position is non-integer, causing interpolation, so the resulting image is grayscale.

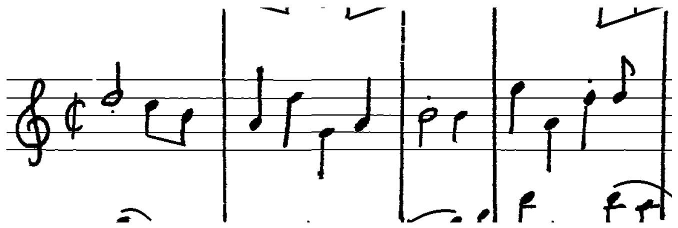
Fig. 6. A synthetic image with staves above and below the main staff mimicking the looks of a cropped image.

In the previous section of this paper we defined the *environment style* and called *postprocessing* the stage that simulates it. Our synthesizer can perform only minor affine transformations of the image. We plan to add a more capable postprocessing stage in the future.

B. Miscelaneous

In our experiments [24], we noticed that our model did not learn to output *empty sequence* when it was given an empty staff. For this reason, we started inserting random gaps in the synthetic music, which our model has to learn to ignore. This spacing algorithm modification was however disabled for images shown in this paper.

The synthesizer can also render three staves into one image (possibly using tall barlines), thus further improving the feel of a cropped image (Figure 6). It also helps the trained model to learn to ignore the surroundings.

A major assumption we made is the separation of symbol synthesis from layout synthesis into two stages. When a human creates a score, they interleave these two stages constantly – they can, for example, draw notes smaller and tighter when not enough space is available because of symbols already drawn. This would require feeding the spatial constraints into the symbol synthesizer, which makes it only more complicated.

IV. Conclusion

A sheet music image synthesizer consists of many smaller, semi-independent subsystems (notehead synthesis, slur synthesis, beam placement, horizontal spacing). For this reason, developing a synthesizer is very costly. However, contrasting that with the cost of annotating enough music sheets in the MuNG scheme, it becomes a viable alternative, especially given the promising results shown by our relatively simple synthesizer. While professional music engraving is, in certain situations, needlessly complex, we can focus our efforts on the average case. In such settings, synthetic data becomes a viable way of building OMR systems. We believe that training data synthesis will play a crucial role in solving the task of optical music recognition.

References

[1] Jorge Calvo-Zaragoza, Jan Hajič Jr., and Alexander Pacha, "Understanding optical music recognition" ACM Computing Surveys, vol. 53, no. 4, 77, 2020

[2] Jan Hajič jr., and Pavel Pecina, “The MUSCIMA++ Dataset for Handwritten Optical Music Recognition” 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 2017, pp. 39-46

[3] Alexander Pacha, and Jan Hajič jr. “The music notation graph (mung) repository” https://github.com/OMR-Research/mung, 2022

[4] Arnau Baró, Carles Badal, and Alicia Fornês, “Handwritten Historical Music Recognition by Sequence-to-Sequence with Attention Mechanism” 17th International Conference on Frontiers in Handwriting Recognition (ICFHR), Dortmund, Germany, 2020, pp. 205-210

[5] Arnau Baró, Pau Riba, Jorge Calvo-Zaragoza, and Alicia Fornés, “From Optical Music Recognition to Handwritten Music Recognition: A baseline” Pattern Recognition Letters, vol. 123, pp. 1-8, 2019

[6] Jorge Calvo-Zaragoza, and David Rizo, “End-to-End Neural Optical Music Recognition of Monophonic Scores” Applied Sciences, vol. 8, no. 4, pp. 606, 2018

[7] Jorge Calvo-Zaragoza, and David Rizo, “Camera-PrIMuS: Neural End-to-End Optical Music Recognition on Realistic Monophonic Scores” 19th International Society for Music Information Retrieval Conference (ISMIR), Paris, France, 2018, pp. 248-255

[8] Lukas Tuggener, Yvan Putra Satyawan, Alexander Pacha, Jürgen Schmidhuber, and Thilo Stadelmann, “The DeepScoresV2 Dataset and Benchmark for Music Object Detection” 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 2021, pp. 9188-9195

[9] Elona Shatri, and György Fazekas “DoReMi: First glance at a universal OMR dataset” 3rd International Workshop on Reading Music Systems (WoRMS), Alicante, Spain, 2021, pp. 43-49

[10] Adrià Rico Blanes, “Synthetic handwritten text generation” 2018

[11] Eloi Alonso, Bastien Moysset, and Ronaldo Messina, “Adversarial Generation of Handwritten Text Images Conditioned on Sequences” 15th IAPR International Conference on Document Analysis and Recognition (ICDAR), Syndey, Australia, 2019, pp. 481-486

[12] C. V. Jawahar, and Shankar Balasubramanian, “Synthesis of Online Handwriting in Indian Languages” 10th International Workshop on Frontiers in Handwriting Recognition, La Baule, France, 2006

[13] Alex Graves (2014) “Generating Sequences With Recurrent Neural Networks” arXiv preprint arXiv:1308.0850

[14] Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox, “What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation?” International Journal of Computer Vision, vol. 126, pp. 942-960, 2018

[15] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman (2014) “Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition” arXiv preprint arXiv:1406.2227

[16] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox, “FlowNet: Learning Optical Flow with Convolutional Networks” IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 2758-2766

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, “Generative Adversarial Nets” 27th International Conference on Neural Information Processing Systems (NIPS), Montreal, Canada, 2014, pp. 2642-2680

[18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen, “Progressive Growing of GANs for Improved Quality, Stability, and Variation” 6th International Conference on Learning Representations (ICLR), Vancouver, Canada, 2018

[19] Kai Brandenbusch, Eugen Rusakov, and Gernot A. Fink, “Context Aware Generation of Cuneiform Signs” 16th IAPR International Conference on Document Analysis and Recognition (ICDAR), Lausanne, Switzerland, 2021, pp. 65-79

[20] Roberto Tamassia, “Handbook of Graph Drawing and Visualization” 1st edition, Chapman & Hall/CRC, 2016

[21] Alicia Fornés, Anjan Dutta, Albert Gordo, and Josep Lladós, “CVC-MUSCIMA: A ground truth of handwritten music score images for writer identification and staff removal” International Journal on Document Analysis and Recognition, vol. 15, pp. 243-251, 2011

[22] Jorge Calvo-Zaragoza, and José Oncina, “Recognition of Pen-Based Music Notation: The HOMUS Dataset” 22nd International Conference on Pattern Recognition (ICPR) Stockholm, Sweden, 2014, pp. 3038-3043

[23] Nicholas Journet, Muriel Visani, Boris Mansencal, Kieu Van-Cuong, and Antoine Billy, “DocCreator: A New Software for Creating Synthetic Ground-Truthed Document Images” Journal of Imaging, vol. 3, no. 4, 62, 2017

[24] Jiří Mayer, and Pavel Pecina, “Synthesizing Training Data for Handwritten Music Recognition” 16th IAPR International Conference on Document Analysis and Recognition (ICDAR), Lausanne, Switzerland, 2021, pp. 626-641

[25] Alexander Pacha, and Horst Eidenberger, “Towards a Universal Music Symbol Classifier” 12th IAPR International Workshop on Graphics Recognition, Kyoto, Japan, 2017

[26] A. Rebelo, G. Capela, and J. S. Cardoso, “Optical recognition of music symbols: A comparative study” International Journal on Document Analysis and Recognition, vol. 13, no. 1, pp. 19-31, 2010