Article

Staff, Symbol, and Melody Detection of Medieval Manuscripts written in Square Notation using Deep Fully Convolutional Networks

Christoph Wick [1], Alexander Hartelt [1] and Frank Puppe [1]

[1] Chair for Artificial Intelligence and Applied Computer Science, Univeristy of Würzburg, 97074 Würzburg, Germany; {prename.surname}@informatik.uni-wuerzburg.de

* Correspondence: christoph.wick@informatik.uni-wuerzburg.de

Abstract: Even today, the automatic digitisation of scanned documents in general but especially the automatic optical music recognition (OMR) of historical manuscripts still remain an enormous challenge, since both handwritten musical symbols and text have to be identified. This paper focuses on the Medieval so-called square notation developed in the 11th-12th century, which is already composed of staff lines, staves, clefs, accidentals, and neumes, that are roughly spoken connected single notes. The aim is to develop an algorithm that captures both the neume and pitch, that is melody information that can be used to reconstruct the original writing. Our pipeline is similar to the standard OMR approach and comprises a novel staff line and symbol detection algorithm, based on deep Fully Convolutional Networks (FCN), which perform pixel-based predictions for either staff lines or symbols and their respective types. Then, the staff line detection combines the extracted lines to staves and yields an F_1-score of over 99% for both detecting lines and complete staves. For the music symbol detection we choose a novel approach that skips the step to identify neumes and instead directly predicts note components (NCs) and their respective affiliation to a neume. Furthermore, the algorithm detects clefs and accidentals. Our algorithm recognises these symbols with an F_1-score of over 96% if the type is ignored and predicts the true symbol sequence of a staff with a diplomatic symbol accuracy rate (dSAR) of about 87%. If only the NCs without their respective connection to a neume, all clefs, and accidentals are of interest the algorithm reaches an harmonic symbol accuracy rate (hSAR) of approximately 90%.

Keywords: Optical Music Recognition; historical document analysis; Medieval manuscripts; neume notation; fully convolutional neural networks

1. Introduction

The digital acquisition of historical documents is an ongoing area of research for many scientists. Especially, since the vast amount of cultural heritage that suffers particularly from degradation and non-standardised fonts, glyphs or layouts, currently relies heavily on human effort, novel techniques in the area of artificial intelligence are required to capture the encoded data in a computer readable form. This paper focuses on the digitisation of medieval monophonic music written in square notation which was developed and used from the 11th-12th century onwards [1]. Compared to even earlier forms, this writing of music is already similar to modern music notation in the sense that it uses four staff lines arranged in a third (musical interval), clefs, accidentals and more or less discrete notes that are drawn as squares (compare Figure 1). However, unlike modern notation, notes are mostly connected to groups, the so-called neumes witch depict a small segment of melodic motion. Historically, neumes arise from single stroke drawings that visually followed the pitch levels. Figure 1 also shows an actual transcription of the upper line taken from our data set in a modernised form where each NC is a single note. Graphical connectivity of two NCs which is in most cases straightforward to decide is marked by a slur (a). Logical connected NCs are visualised by a smaller space between two notes (c). Example (b) shows two notes that are two single tone neumes and therefore are notated with a larger space. This,

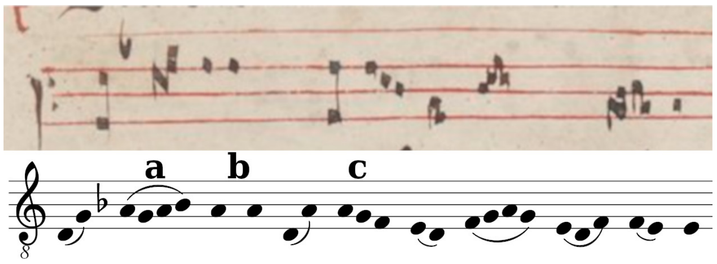

Figure 1. An example transcription equivalent in modern notation (bottom image) based on a neume notation (upper image). A neume consisting of graphical connected NCs (a) is visualised by a slur, each new neume starts with an increased space (b), logical connected NCs are notated with small note space (c). The modern notation can be used to fully reconstruct all neumes.

however, is in some cases difficult to decide because there is no difference between a neume consisting of multiple single NCs or multiple neumes comprising one NC. In this example the syllables each belonging to different neumes dissolve this ambiguity (not shown).

The aim of this paper is on the one hand to detect the staff lines that form staves, and on the other hand to locate all musical symbols including the discrete note position of the neumes, so that the musical relevant information is fully recognised. Both tasks are conducted separately by usage of two FCNs which predict pixel based locations of staff lines or music symbols, respectively. Further postprocessing steps extract the actual positions and symbol types.

In contrast to the works of other researchers [2,3] that first detect and classify neumes and then resolve them into single NCs to gain the distinct pitches, our algorithm combines this step by directly yielding the described transcription equivalent. However, the actual neumes in the original manuscript can be reconstructed, and thus no information is lost. The advantage of our approach is that we bypass the ambiguities of neumes that are very similar to each other but yield the same pitches. Furthermore, no subsequent faults can arise. Eventually, we conduct an evaluation that both ignores and includes the neume connectivity to measure either only the quality of the detected melody or the capability to digitise the complete square notation.

To summarise, the main contributions of this work regarding an OMR of manuscripts of the 12th century are the following:

- Development of a very robust staff line detection algorithm based on FCNs, which identifies almost all staves and their related staff lines.
- Proposal of a symbol detection algorithm which applies an FCN to the detected staves. The algorithm locates clefs, accidentals, and notes with their respective affiliation to neumes. Thereby, it is possible to fully capture and reconstruct the melodic sequence.
- Evaluations of both new algorithms on 49 pages comprising 510 staves and 16,731 annotated symbols. We consider, among other things, the effect of only a few training examples and the generalisation to new manuscripts.

The remainder of this paper is structured as follows: First, related work is listed and discussed. Afterwards, the algorithms for staff and symbol detection are described, and we evaluate the algorithms independently, including the experiments that measure the amount of required training data. We conclude this paper by giving an outlook on our future work.

2. Related Work

Especially, the detection on manuscripts is an ongoing area of research on music but also on textual documents. The special focus of this paper lies on monophonic music, where only one musical symbol must be detected at a point in time. In the following, we list work related to OMR on monophonic music detection and to music detection on historical documents in general. With the recent overwhelming success of deep learning, CNNs acting on raw input data instead of preprocessed images (e.g. staff line removal) became very popular and yielded new state-of-the-art results. Since OMR is in principal a sequence to sequence task, also recurrent networks such as LSTMs are promising to use.

2.1. OMR on contemporary notation

Baró-Mas et al. [4] use (bidirectional) LSTM networks to predict the pitch and rhythm of notes and other musical symbols e.g. rests and clefs in line images containing monophonic music. The outputs of the network are two sequences. The first one aims to predict the pitch of notes or the type of other symbols like clefs (54 classes in total), while the second one indicates the rhythm of notes or a *no note* label (26 classes). For training the network, they implement two different loss functions that consider both target sequences. One computes the weighted euclidean loss, the other implements a multi-label soft margin loss. To evaluate their algorithm two different data sets are used which both consist of monophonic music in modern notation, however one is printed and one handwritten. Experiments on the first dataset which comprises incipts from the RISM data set [5] yield a symbol/pitch error rate of 1.5% and a rhythm error rate of 2.0%. The total error for both properties to be correct is 2.8%. As second data set, they use one page of the CVC-MUSICMA dataset [6]. After manually labelling six staves, they extend the number of different staves by varying the order of bars within a single staff. During training, data augmentation is used, and all lines from the printed data set were added. The error of the symbol/pitch and rhythm detection is 47% and 43%, respectively, yielding a total error rate of 65%.

A similar attempt to learn music recognition on modern monophonic printed music was proposed by van der Wel and Ullrich [7]. However, compared to [4], they use preliminary CNN layers and an encoder/decoder structure for their neural network. The CNN/LSTM based encoder maps a line image into one fixed size state representation by processing the line sequentially. The decoder consecutively decodes this state and predicts a sequence of pitches, duration, and finally a stop mark. This overall procedure is based on and very similar to the sequence to sequence task used for machine translation [8]. In total there are 108 pitch and 48 duration categories in their data set, which is compiled from MusicXML [9] scores from the MuseScore[1] sheet music archive. In contrast to the normalised edit-distance, a rather strict metric is used that can not handle insertions or deletions. They align the prediction and ground truth (GT) sequence label by label and count the number of correct predictions. Thus, if a label is deleted or inserted all subsequent notes are usually false. Their final model which is trained with data augmentation, yields a pitch and duration accuracy of 81%, and 94%, respectively. The total accuracy of notes is 80%.

A third very promising attempt to predict monophonic music was made by Calvo-Zaragoza and Rizo [10]. They use CNN/LSTM hybrids combined with a CTC-loss-function [11] as model architecture, a technique that already succeed in handwritten and printed optical character recognition (OCR) [12,13], or speech recognition [14,15], which are both sequence to sequence tasks. The advantage of this loss function is that it does not require position accurate labelling in the GT. Solely the target sequence and input data is obligatory out of which the network automatically learns the alignment. The drawback of this method is that each combination of pitch and rhythm semantically requires a distinct label. Moreover, a key-signature can for instance either be a single symbol or be dissolved in individual accidentals. The first so-called semantic representation requires 1,781 classes in total, while the second agnostic representation only uses 758 classes. To perform experiments they created the so-called PrIMuS[2] (Printed Images of Music Staves) data set containing 87,678 real-music incipts which are rendered by Verivio [16]. The evaluation yields a sequence error rate of 17.9% and 12.5% in the agnostic and semantic representation, respectively, which is explainable by the higher number of classes. However, the individual symbol error rate is approximately 1% in both representations.

[1] https://musescore.org/
[2] available at https://grfia.dlsi.ua.es/primus/

Compared to the upper work, handwritten music recognition can also be regarded as an object detection task. This approach was proposed by Pacha et al. [17]. Their pipeline uses existing state-of-the-art deep neural networks for object detection such as Faster R-CNN [18] or R-FCN [19] with custom preprocessing and training. A cropped image of a single staff without staff line removal serves as input. They evaluate on the MUSCIMA++ dataset [20] which contains over 90,000 symbol annotations of handwritten music distributed in 71 classes. The object detection achieves a mean average precision of over 80%.

2.2. OMR on historical notations

In the area of historical OMR, Calvo-Zaragoza et al. [21,22] apply Hidden Markov Models (HMM) and a n-gram language model on line images written in mensural notation of the 17th century. This notation is comparable to modern notation, since it is already ruled by very similar symbols. Their handwritten corpus comprises 576 staves with 13,863 individual symbols assigned to 200 different classes. As metrics they measured the glyph error rate (GER, i.e. note duration or clef) and height error rate (HER, position relative to the staff lines) separately, but also computed the combined symbol error rate (SER). Their best model reached a GER of 35.2%, a HER of 28.2%, and a SER of 40.4%.

Ramirez and Ohya [3] did notable work on the automatic recognition of handwritten square notation which is also the focus of our paper. They built a dataset based on 136 pages from the Digital Scriptorium repository[3] comprising 847 staves and over 5,000 neumes. A huge challenge are the greyscale images of the 14th century which suffer from physical degradation, variability in notation styles, or non-standardised scan conditions. Their first task detects and extracts staves. Thereto, they use a brute-force algorithm which matches the original image with a staff template built up from four straight staff lines by varying line and staff distance, and orientation. The found optima indicate locations for staves, which are then extracted. The advantage of this method is that individual staff lines need not to be detected, however the handwritten staff lines require to be equally distant and almost straight. In total, 802 of all 847 staves are correctly detected (95% recall). In a second task, they perform a symbol detection and classification algorithm. A pattern matching algorithm fed with several different templates for each neume first marks possible symbol locations and finds approximately 88% of all symbols. Then, a SVM classifies the symbols into different neume types with an accuracy no lower than 92% across all classes. Compared to our approach, they do not resolve the individual neumes into single NCs nor detect clefs. But likewise, accidentals are ignored or do not occur.

Vigliensoni et al. [2] focus on the pitch detection in their work, however on documents of the Liber Usualis[4] printed in 1961. Using the staff line detection of Miyao [23], the staff line removal of Roach and Tatem (compare e.g. [24]), and an automatic neume classification algorithm trained on 40 pages, they evaluate the pitch detection on 20 pages consisting of 2,219 neumes and 3,114 pitches. The pitch of the first component of a neume is correctly detected for 97% of all neumes, while only 95% of all note components including single-tone neumes are found.

In [25], Calvo-Zaragoza et al. propose an approach known from historical document analysis [26] in the area of OMR. A deep CNN is trained to segment scans of two manuscripts of the 14th and 16th century pixel-wise, i.e. each pixel is assigned a class selected from background, text, staff line, or symbol. Approximately 80% of the pixels are background, 10% are text, 6% are staff lines and 4% are symbols. The results show that especially at the margin of changing label types, the classification is incorrect, which however has only a minor impact on proceeding steps. Furthermore, only a small number of GT instances must be manually created to obtain good results. The evaluation measures the correct label of pixels located particularly at the edge of different symbols and yields an average

[3] http://www.digital-scriptorium.org/
[4] a digital version is available at https://archive.org/details/TheLiberUsualis1961

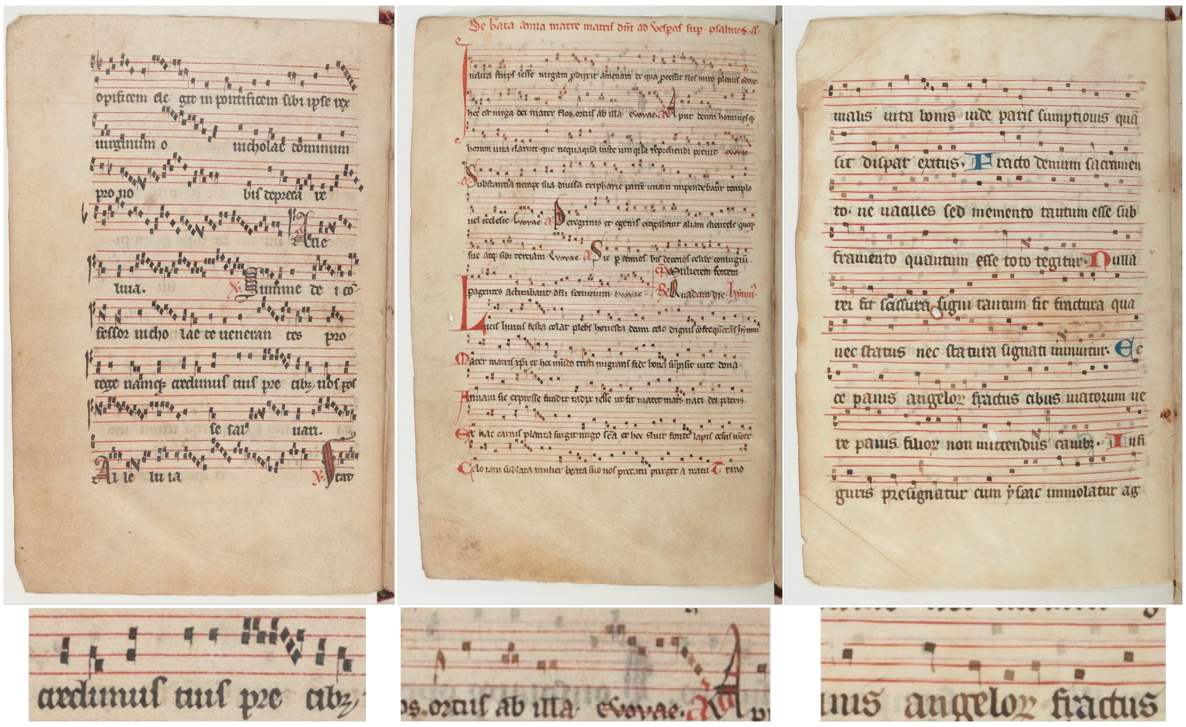

Figure 2. Each page represents part 1, 2, and 3, respectively. The bottom images provide a zoomed view on the upper pages. While the first page is very clean, both other pages suffer from bleeding and fainter writing. The staff lines in the first and second part are very straight most certainly due to usage of a ruler, whereas the staff lines of the third part are freehand drawn.

F_1-score of around 90%. In general, this algorithm only solves one step in an OMR pipeline, which is why additional steps are required to extract actual staff lines, symbols, or text.

3. Dataset

As dataset we use 49 pages of the manuscript "Graduel de Nevers" (accessible at the Bibliothèque nationale de France[5]) published in the 12th Century. The handwritten music comprises different neume notation styles, only a part is written in square notation, which are folios 2-9 and 246-263. These pages were extracted and further split into three parts that share a similar layout or difficulty based on our human intuition (compare Figure 2). The first part contains pages with the best available notation quality with fading bleeding. Its staff lines are mostly straight possibly due to usage of a ruler, and all neumes are very clear and distinct. The second part highly suffers from bleeding staff lines and is written very narrowly yielding the most difficult notation. The third part comprises to some extend very unclear neumes and very wavy staff lines. In our experiments, we will use these parts to estimate how well our trained algorithms generalise onto unseen layouts by not using all parts for training.

The GT was manually annotated under the supervision of music scientists. For each stave, we store four staff lines each as polyline whose coordinates are relative to the image. The exact start and end of a line is ambiguous due to occasional severe degradation. We further defined the symbols clef, accidental, and neume as part of a staff. A clef consists of a type (C or F), its centre as absolute position on the image, and its location relative to the staff lines, i.e. which staff line is denoted. In the example line in Figure 1 the F-clef is located on the second line. Since all occurring accidentals are a flat B, it is sufficient to store their absolute centre position and their staff line location. Each neume is comprised of single NCs. For each NC, we take its absolute position, its location on the staff lines, and whether it

5 https://gallica.bnf.fr/ark:/12148/btv1b8432301z

Table 1. Overview of the data set statistics. The data set is split manually into three groups that share similarities in layout or handwriting.

Part	Pages	Staves	S.-Lines	Symbols	Notes	Clefs	Accid.
1	14	125	500	3,911	3,733	153	25
2	27	313	1,252	10,794	10,394	361	39
3	8	72	288	1,666	1,581	84	1
Total	49	510	2,040	16,371	15,708	598	65

is graphically connected to the previous NC. For example the first neume in Figure 1 consists of two NCs. The first one is located on the first staff line while the second one is graphically connected and is positioned in the second space. The neume (c) in the example line comprises three NCs of which none is graphically connected. Note, that in general the symbol location in a staff line can be computed using the positions of the staff lines and the global position of the symbol, however there exist many ambiguities whether a NC is actually located on a line or in a space. For instance the first NC of the second last neume in Figure 1 might also be an G instead of an F. In this work, we do not distinguish additional note types such as liquecents or oriscus. Furthermore, unlike the approach in [25], we do not label the images on a pixel-basis, since detecting the actual shape and extent of neumes, clefs, or accidentals is out of the focus of this paper. We are solely interested in transcribing historical neume notations in a digitised form that preserves all melodic information which is the desired output in almost all cases.

Using the described storage scheme, all required information for training and evaluation is preset or can be computed. Table 1 gives an overview of each part's properties by listing its number of pages, the number of staves on these pages, each consisting of four staff lines, and the symbol counts. In total, we annotated more than 16,000 symbols located in more than 500 staves. Compared to the number of symbols, the 65 flat accidentals occur very rarely, which makes the learning of their detection very challenging. Approximately 4% of the symbols are clefs, yet at least up to 600 clefs can be used as training examples.

4. Methodology

This section describes the general workflow of the staff line and symbol detection. Furthermore, we provide an introduction to FCNs which are heavily used by the algorithms.

4.1. Workflow

The typical OMR pipeline usually breaks down in the following four stages as shown in [27]:

1. image preprocessing,
2. staff line detection and recognition of musical symbols,
3. reconstruction of the musical information, and
4. construction of a musical notation model.

Since the focus of this paper is to output a sequence of music symbols that fully captures the written neume notation of the original manuscripts only the first three steps are included in the proposed workflow. The omitted reconstruction of the output into a modern notation is, however, straightforward (compare Figure 1). Therefore, our pipeline starts with a scanned colour image and ends with the reconstructed symbols that are recognised and positioned in a detected staff as seen in Figure 3.

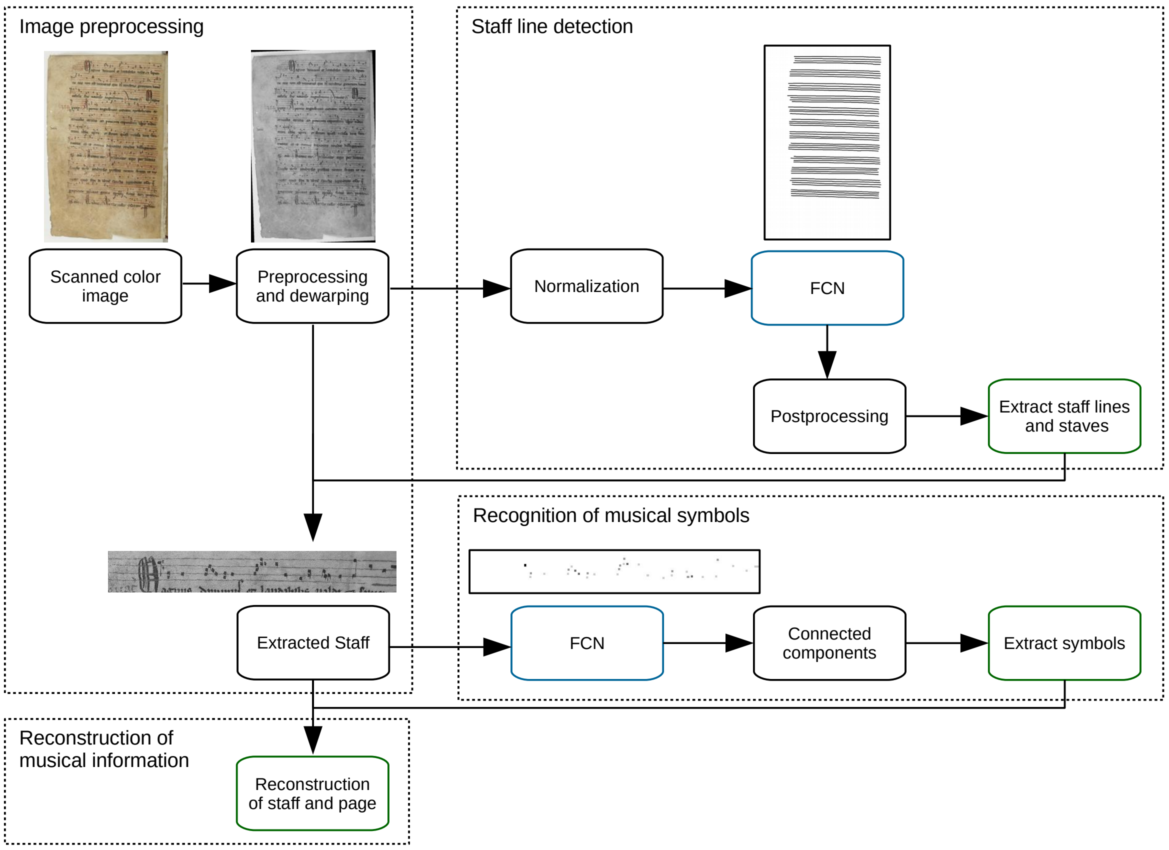
Image preprocessing
Staff line detection
Scanned color image
Preprocessing and dewarping
Normalization
FCN
Postprocessing
Extract staff lines and staves
Recognition of musical symbols
Extracted Staff
FCN
Connected components
Extract symbols
Reconstruction of musical information
Reconstruction of staff and page

Figure 3. Schematic view of the workflow of the staff line and symbol detection. The small images show the original image and the input and output of the respective FCN.

During the preprocessing, first, the raw image is deskewed and converted into grayscale by using OCRopus[6]. The deskewing algorithm was initially designed for processing textual documents in an OCR-pipeline, however it generalises flawlessly on our musical data. The algorithm first applies an array of small rotations on the image. Afterwards, for each rotated image all rows are averaged pixel-wise and the variance across the resulting vector is computed, which yields a score for each initial rotation angle. Finally, finding the maximum yields the best rotation angle.

Next, our staff line detection algorithm is applied to the preprocessed image. The resulting staff lines are represented as polylines and grouped into single staves. Each staff is then cut out of the preprocessed image and the symbol detection is applied, which yields a list of symbols including their absolute pixel position. Finally, this pixel position is converted to a location relative to the recognised staff lines to produce the final output. The last step for an actual transcription is a straightforward mapping which decodes these positions based on the detected clefs to actual note names, i.e. the pitch. This step can induce subsequent faults if the clef was misclassified. The melody of a line however, which is defined by the intervals that are relative to the NCs, is independent of the clef which only defines the global pitch.

Both the staff line and symbol detection make use of FCNs, which we will introduce in the following. Afterwards, the individual algorithms are described is greater detail.

[6] https://github.com/tmbdev/ocropy/blob/master/ocropus-nlbin

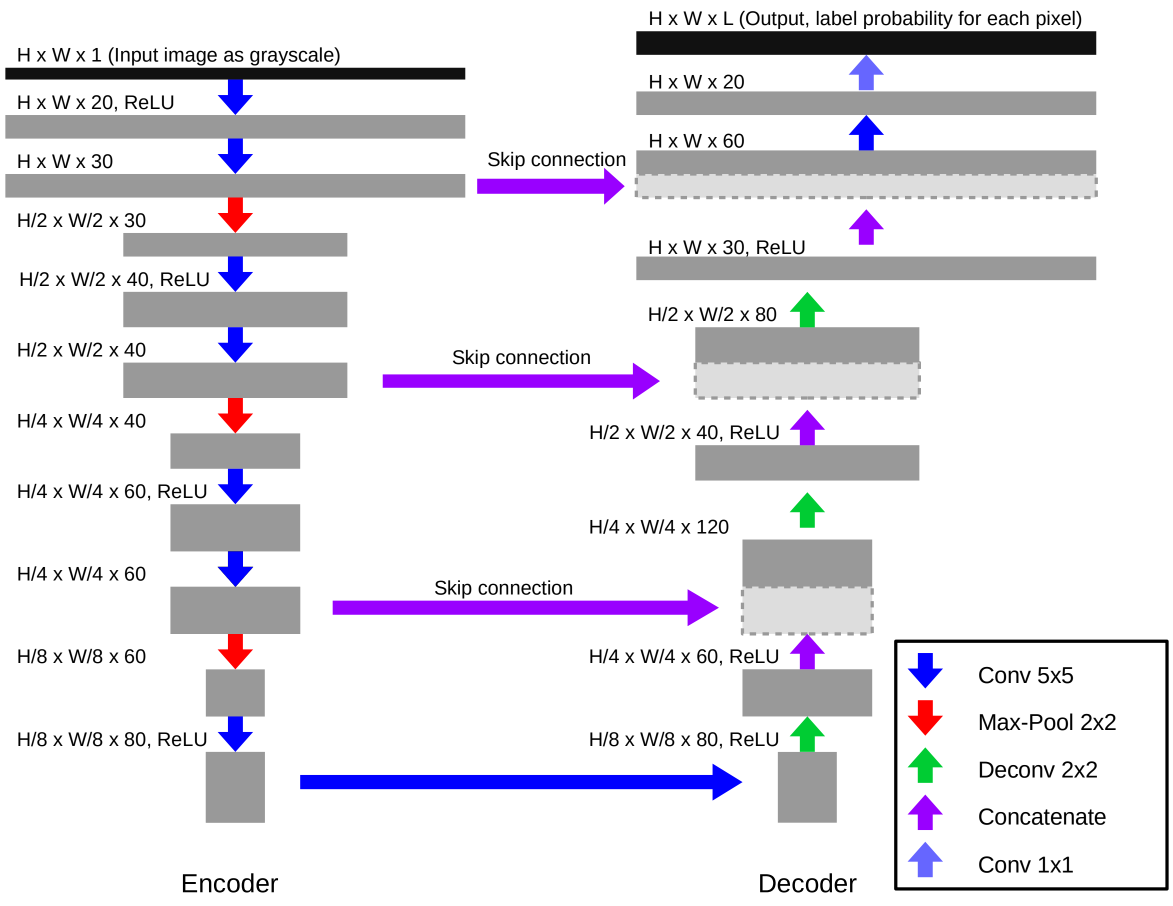

Figure 4. FCN network architecture used for the staff line detection and symbol detection. Both the type and the dimension of each layer is shown.

4.2. FCN Architecture

FCNs are a novel method to classify an input image with variable size ($H \times W$) pixel-wise, i.e. a target class is assigned to each pixel in the input image. Especially the U-Net structure of [28] poses a new state-of-the-art in several areas. Our network architecture which is shown in Figure 4 is a bit simpler than the original U-Net since it has a reduced number of filters in the Convolution Layers and only down scales to a factor of 8.

In general, the U-Net comprises an encoder/decoder structure that down- and up-scales the input image and thereby applies convolution and pooling operations. Skip-connections directly link the outputs of each scale level of the encoder to the outputs of the decoder by concatenation. Thus, the next higher layer of the decoder knows both the widespread information of the deeper levels and the more fine-granular features of the higher levels of the encoder. The output of the network is a probability distribution over all allowed labels for each input pixel.

The network is trained by using a pair of input (e.g. greyscale) image and a corresponding label matrix as GT. The neural network tries to predict the desired label with a probability of one, while all other classes shall be zero. The corresponding loss function is the negative-loss-likelihood which treats each individual pixel prediction independently.

4.3. Staff Line Detection

Our staff line detection algorithm aims to extract staves and their respective staff lines as group of polylines from an input image. Each polyline is a list of points located in the original image.

Our algorithm (compare Figure 3) for the staff line detection acts on the deskewed greyscale image of a full page as input, and expects as parameters the number of staff lines per staves (four in the case of this paper) and the average staff line distance d_{SL} which is however approximated by the algorithm of [29] and [30]. d_{SL} is used to normalise the original input to an image where d_{SL} is a fixed value of 10px. The automatic staff line distance approximation algorithm of [30] acts on the binarised version of the input image which is computed by the OCRopus binarisation tool. The idea of the algorithm is based on run-length encoding (RLE) for each column of the binary image. RLE encodes an array of binaray black and white pixels in sequence length, e.g. the sequence $\{1110011111000111010\}$ is encoded as 3, 2, 5, 3, 3, 1, 1, 1. The most common black runs represent the height of a single staff line, while the most common white runs denote the staff space, 1 and 3 in the upper example, respectively. We adapt this algorithm, by constraining the staff space to be at least twice the staff line height to gain more robust results with noisy data. The actual d_{SL} is finally approximated by the sum of the staff line height and space. An preliminary experiment shows that on all available pages this algorithm correctly approximates d_{SL}, hence all images can be normalised automatically. This normalisation step is only required for the staff line detection but not for the later symbol detection.

The preprocessed and normalised image is then fed into the FCN which is trained to discriminate between staff line and background pixels. To prune artefacts, we then apply a horizontal RLE to drop row-wise pixel groups with a length smaller than 6px. The remaining connected components are then used to determine preliminary polylines by computing the vertical centre at each horizontal position for each component. Since the staff lines are sometimes interrupted, we apply a postprocessing step that combines polylines which are on a similar height level. Finally, we cluster all staff lines into staves by first computing the most common distance of staff lines $\bar{d}_{\mathrm{SL}}$ and then group staff lines with a distance of smaller than $1.5 \cdot \bar{d}_{\mathrm{SL}}$. We only keep staves with four lines, by what staves that might be wrongly classified noise (e.g. bleeding), text or the page margin, are dropped. The remaining grouped polylines represent the final result of the staff line detection algorithm.

The FCN is trained on GT that solely contains the human annotated staff lines drawn with a thickness of 3px. During training, we optionally augment the data by randomly rotating the image up to 2°, flipping the image vertically or horizontally, and varying the contrast and brightness up to a factor of $2^{[-0.8,0.8]}$ and value of 8%, respectively. Furthermore, the data is scaled with a factor of up to $2^{[-0.1,0.1]}$ on each axis independently.

4.4. Symbol and Note Component Detection

The symbol detection algorithm acts on the image of a single staff (see Figure 3). To extract these images, we determine the bounding boxes of each known staff which are extended to the top and bottom by adding d_{SL} which is computed using the given staff lines. This line image is finally scaled to a fixed height h_{staff}, which we choose as 80px (see Section 5.3.2), so that the resolution of the input data is normalised. An example line is shown in the first row of Figure 5.

Similar to the staff line detection, the symbol detector employs the proposed FCN-architecture to classify the input image pixel-wise into seven different classes: background, clef-c, clef-f, accidental-flat, initial NC (start of a neume), logical connected NC, and graphical connected NC. The GT mask for each line is generated by drawing a small box of the respective label at the centre of each symbol (compare the second image in Figure 5). When training the FCN it aims to predict the correct labels pixel wise, which is why actual symbols and their position and types must be extracted in a proceeding step. For this purpose, we first assign the most probable class to each pixel to receive a label map. Afterwards, connected components are detected on a binary image that either denotes background or any symbol. Each component yields a single symbol whereby the centre of the component equates the symbol position and the type defined by majority voting on the label map.

Finally, the relative note position is computed based on the detected staff lines (see left image of Figure 6), whereby the relative distance of the y-coordinate of each symbol in the staff determines the closest space or staff line. It shows, that notes are often "on the line" even though it is visually closer to

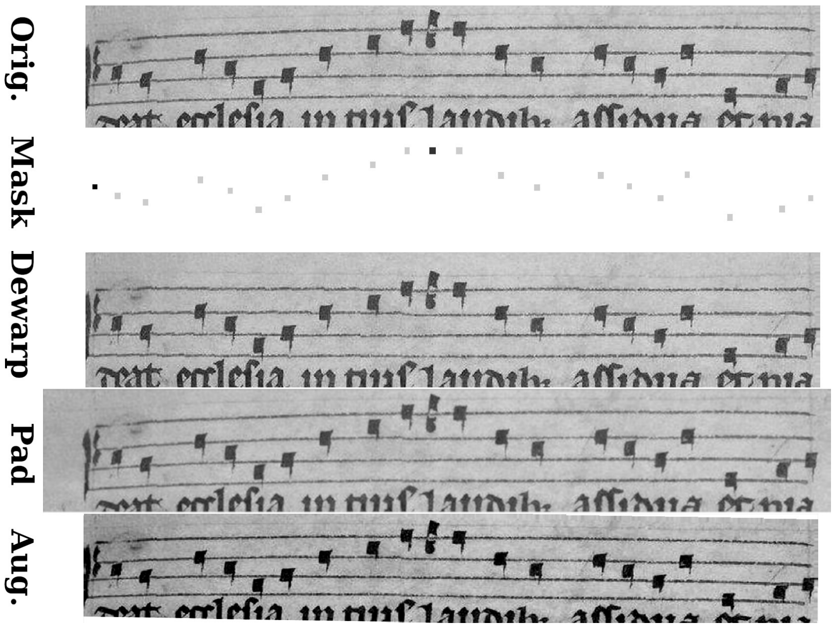

Figure 5. The first line shows the cut off original input image. The next one is the target mask for the FCN in which notes and clefs are marked. The next three lines show the dewarped, padded and an augmented version of the original image.

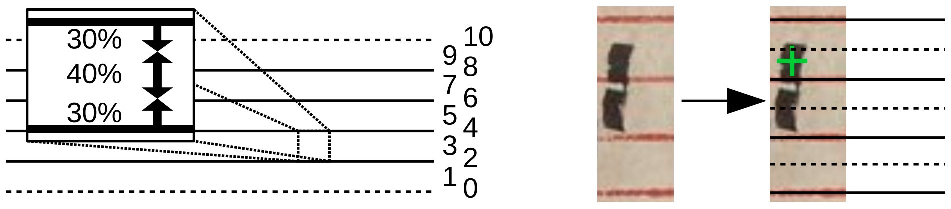

Figure 6. The left image shows the relative staff line positions shown on the right. The zoomed window marks the area sizes that are assigned as line or space, while the dashed lines indicate ledger lines. The right image is an example for a NC (green mark) that is visually closer to the space (dashed) however actually a NC on a line (solid).

a space which is why spaces and lines are not distributed equidistant, instead the ratio is 2/3 (see the left zoomed sub figure and for an example the right image of Figure 6).

To improve the quality of a line, we implemented and tested several preprocessing steps applied to the line image before being processed by the FCN (compare third to fifth image of Figure 5). First, we dewarp the line by transforming the image so that the staff lines are straight. Furthermore, we pad extra space to the left and right of a staff to ensure that all symbols especially clefs are fully contained. We optionally use data augmentation that varies the contrast and brightness up to a factor of $2^{[-0.1,0.1]}$ and a value of 4%, respectively. Also, the data is scaled with a factor of up to $2^{[-0.1,0.1]}$ independently in x- and y-direction.

5. Experiments

In this section, we first introduce the data setup i.e. how we chose parts for training and testing the FCNs. Then, we evaluate and discuss the proposed staff line and symbol detection algorithm. Finally, we measure the computation times required for training the deep models and prediction.

5.1. Data Setup

In general, we report the average value of a five cross-fold as generated by the scheme shown in Figure 7. The different experimental setups are:

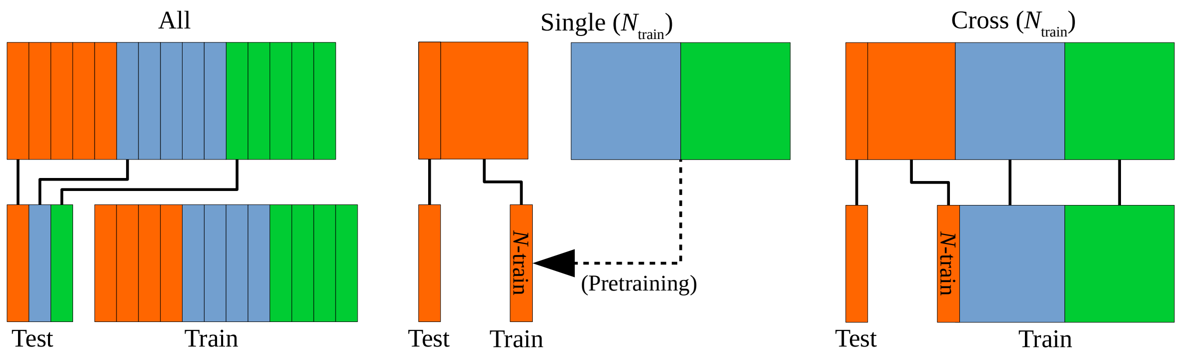

Figure 7. Scheme of the used cross fold training. Each colour represents a part that is split into a five fold (smaller boxes). Depending on the experiment different sections are chosen for training and testing. Optionally, a pretrained model on remaining data is used. For the cross-part scheme N_{train} can be set to zero to only train on other parts.

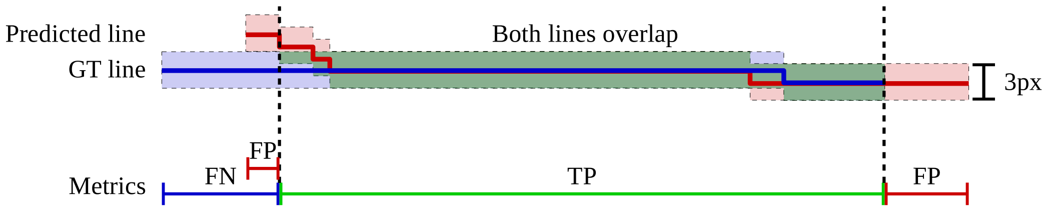

Figure 8. Visualisation of the metric used to evaluate the staff line detection. The blue and red solid lines in the upper image denote the GT and prediction of a single staff line, respectively. The coloured area is the allowed space of 3px where lines must overlap to be counted as a match. If at least one vertical pixel is green it is counted as correct. The bottom line shows the intervals which are counted as FN, TP, and FP. The deviation in the first part counts both as FN and FP.

1. Use *all* parts for training. Each part is split into a five fold whereof one fold of each part is chosen for testing, the remainder for training.
2. Only use a five fold of one *single* part. One fold is used for testing while N_{train} data of the remainder serves for training. Optionally, a model pretrained on the two remaining parts can be utilised as starting point for the training on the actual part.
3. *Cross*-Fold: The same as experiment 2, however all pages of the remaining two parts are added to the training data. Setting N_{train} to zero yields experiments that only incorporates data of the other parts and thus generate the models that are used for pretraining in the previous setup.

For both the staff line and symbol detection, setup 1 serves for a general evaluation of the proposed algorithm. Then, we use setup 3 with $N_{\text{train}} = 0$ to test how a model generalised on unseen data with a somewhat different layout. Finally, setup 2 with and without pretraining and setup 3 with $N_{\text{train}} \geq 1$ conduce to test how the inclusion of actual data of the new part influences the accuracies.

5.2. Staff Line Detection

The staff line detection algorithm takes a preprocessed page image and outputs a list of staves each containing four staff lines stored as polyline. To evaluate this algorithm, we first introduce the used metrics and then present the results when training the FCN on different data sets. First, all parts are joined, then the impact of training on two and testing on the remaining one is investigated (compare Figure 7). Finally, we evaluate the effect of increasing the number of training examples.

Table 2. Staff line detection and length metrics with and without data augmentation. The last column combines the two individual F_1 measures by multiplication.

	Line Detection			Length Fit			
Model	**Prec.**	**Rec.**	F_1^{D}	**Prec.**	**Rec.**	F_1^{LF}	**Total** F_1
Default	0.996	0.998	0.997	0.977	0.995	0.985	0.982
Data aug.	0.987	0.991	0.989	0.972	0.995	0.983	0.972

Table 3. Staff detection and the number of recognised lines in a detected staff with and without data augmentation. The last column combines the two individual F_1 measures by multiplication.

	Staff Detection			Hit lines			
Model	**Prec.**	**Rec.**	F_1^{S}	**Prec.**	**Rec.**	F_1^{HL}	**Total** F_1
Default	0.996	0.998	0.997	1.000	1.000	1.000	0.997
Data aug.	0.988	0.992	0.990	0.999	0.999	0.999	0.989

5.2.1. Metrics

To evaluate the staff line detection algorithm we use two different pixel-based metrics. A true positive (TP) pixel pair of GT and prediction is allowed to have a distance of maximal 3px, any other pixel of a line is treated as false positive (FP) or false negative (FN) (compare Figure 8).

Similar to an object detection task, we first measure the percentage of staff lines that are detected (recall) and if too many lines were found (precision), which then yields the line detection F_1^{D}-score. A staff line is marked as TP if the overlap of GT and prediction is greater than 0.5, i.e. more than 50% of the line must be hit, but the detected line may not be twice as long. This threshold is fairly harsh, since the detection of only a short section of a line causes both an FP and FN error.

The second metrics computes how many pixels of detected staff line are hit in length, i.e. whether the prediction was too long or too short (compare Figure 8). Thereto, we evaluate the precision, recall, and F_1^{LF}-score of all TP-lines in the upper measure. Finally, multiplication of both individual metrics yields the total F_1-score.

Furthermore, we evaluate the detection rate of a complete staff consisting of four staff lines (F_1^{S}). A staff is marked as found if at least two staff lines ($=$ 50%) are correctly detected. This arbitrary threshold is common for general object recognition tasks. For all detected staves we also count the relative amount of how many of the four lines are correctly hit (F_1^{HL}).

5.2.2. Results on all data

The results of the staff line detection using a five cross-fold on all available data are shown in Table 2. Clearly over 99% of all staff lines are found (recall) and over 99% of all predicted staff lines are correct (precision). The resulting F_1^{D}-score is about 99.7%. The length of the correctly detected staff lines has a F_1^{LF}-score of 98.5%, however the recall that is higher than the precision shows that lines mostly are too long.

Typical errors are exemplarily shown in Figure 9. The green, red and blue colors indicate TP, FP and FN pixels, respectively, though FN are rare. Most errors are either staff lines that are too long, e.g. caused by falsely extending a staff, or staff lines that are predicted on background. Both error types are caused e.g. by background noise such as bleeding staves or foreground text or drop capitals. Naturally, other errors occur on damaged pages or very thin lines that must almost be guessed. Furthermore, there are always small inaccuracies at the exact beginning and at the end of a staff line which however are also a problem of the GT itself. The severity of the errors on the symbol detection will be discussed in Section 5.3.5.

Table 3 lists the recognition rate of complete staves. It is astonishing that the scores of the staff detection are (almost) identical to the staff line detection, however, this shows that the staff line

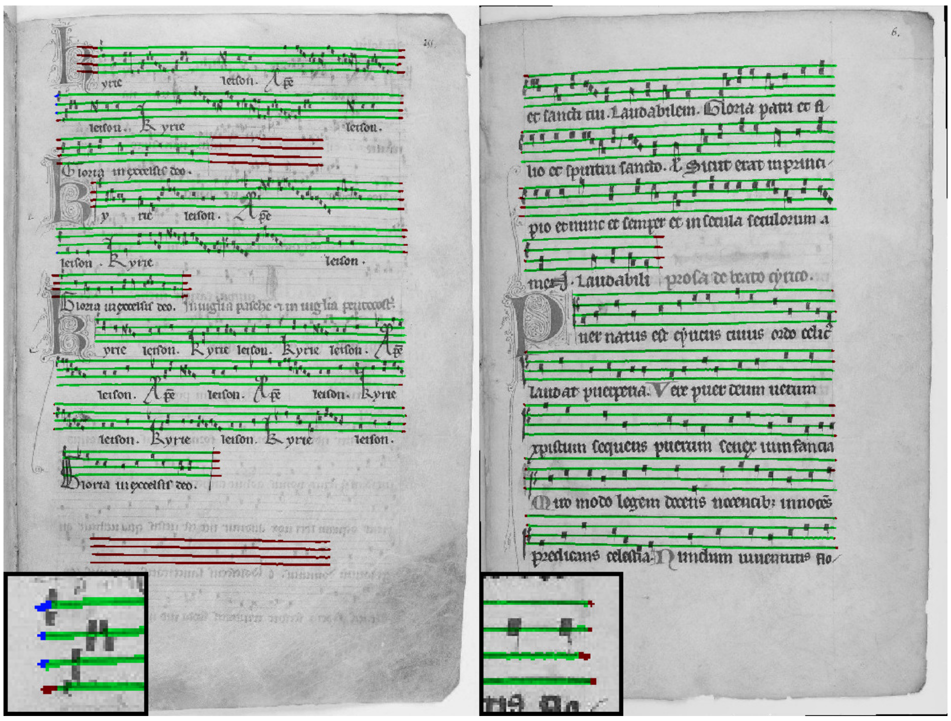

Figure 9. The left image shows typical errors during the staff line detection. Green pixels indicate TPs, red FPs, blue FNs. Either all lines in a staff are too long (third staff) or bled background is recognised (last staff). The right image shows a page with weak bleeding. In this image all staff lines are detected correctly. The zoomed sub images focus on the start and end of a staff.

detection only predicts wrong lines across a whole staff, e.g. all four lines are too long or a whole staff is predicted on a blank page, as seen in Figure 9. This is also shown by a precision and recall of 100% in the measure of hit lines. This states, that if a staff is detected all lines are correct. In the following, we will only report the staff line detection accuracy because it is almost equivalent to the staff accuracy, but also includes the metric concerning the exact length of the lines.

Applying data augmentation yields no improvements in both metrics, instead the results clearly worsen. However, especially the precision drops compared to the recall. This is justified by errors that are mostly confusions if a staff is bleeding or real, which is possibly caused by the strong augmentations in contrast and brightness levels.

5.2.3. Results of cross-part training

The results of cross-part training with and without data augmentation are shown in Table 4. In general, it can be observed, that on any split our algorithm obtains an F_1^{D}-score of 98.8% or higher in the staff line detection. On part 1 even 100% are reached, i.e. all staff lines are found with an overlap of at least 50%. This shows, that the staff line detection algorithm is very robust to new layouts. It is significant that part 2 suffers from data augmentation more clearly than part 3, while on part 1 still a score of 100% is reached. The prediction of the line length is untouched. This can be explained by the different kind of quality of the parts. While part 3 suffers very weakly from bleeding staves and its lines are drawn very clearly, part 2 consists of pages where bleeding lines are present and some staff lines are barely visible and thus are more similar to background noise. Data augmentation creates, among other things, pages where the contrast of data is adjusted, which can explain why it is harder to distinguish actual staff lines, background and bleeding in part 2. In general, part 2 also yields the lowest F_1^{D}-score of 96.8% which also shows that this part is the most complicated one.

The staff line lengths are detected very similar across the parts with a mean F_1^{LF}-value of 98.6% and 98.7% (with and without data augmentation), which is almost identical to F_1^{LF} when training and

Table 4. Comparing staff line detection using cross data set training: Two parts are chosen for training while the remainder is chosen as test data set. The training data is split into five cross folds (four folds training, one for validation) of which the average scores are shown. The total score is computed as the product of the individual F_1-scores. The lower half of the table shows the results when using data augmentation.

			Detection			Length Fit			
	Train	**Test**	**Prec.**	**Rec.**	F_1^{D}	**Prec.**	**Rec.**	F_1^{LF}	**Total F_1**
Def.	2, 3	1	1.000	1.000	1.000	0.977	0.990	0.983	0.983
	1, 3	2	0.986	0.993	0.990	0.982	0.995	0.988	0.978
	1, 2	3	1.000	0.978	0.988	0.990	0.990	0.990	0.978
	Mean				0.993			0.987	0.980
Aug.	2, 3	1	1.000	1.000	1.000	0.976	0.993	0.984	0.984
	1, 3	2	0.959	0.978	0.968	0.968	0.997	0.982	0.951
	1, 2	3	0.989	0.967	0.977	0.988	0.995	0.991	0.969
	Mean				0.982			0.986	0.968

evaluation on all data of 98.5% (compare Table 2). Hence, we conclude that the crucial component of the algorithm is to detect where staff lines are located and not to find the correct length of staff lines.

Note, that the accuracies of these experiments are averaged over the parts while the results in the previous Table 3 are the mean of all pages, which is why the values should not be directly compared since part 2 consists of more instances than the other parts.

5.2.4. Data amount

In this section the effect of increasing the amount of training examples is investigated. The results are summarised in Table 5. Each value is computed by first averaging all cross-folds of a single part and then taking the mean of all three respective experiments for all parts. As expected, if no additional data is available, an increasing number of training examples yields a higher F_1 score. The model trained on only one page already detects staves with $F_1^{\mathrm{D}} = 98.3\%$ and the length of a detected line matches with $F_1^{\mathrm{LF}} = 96.9\%$. Applying data augmentation on this data improves the results significantly, resulting in a F_1^{D} of over 99% and a F_1^{LF} of 98.0% if one page is used. Using more pages slightly increases the total F_1-score up to 97.9%.

Of course using a pretrained model on the remaining data (PT) or including the remaining data into the data set (Inc.) clearly outperforms the standard training without data augmentation. However, it shows that including the data is in general better than just using a pretrained model if only one or two pages are available, which makes this approach more robust for few pages. If more data is used the error differences are caused by the training process itself e.g. choosing the random seed for initialisation of the model. But it is to be expected that if many GT pages are available PT is superior since the model can directly fit on the respective data. Data augmentation improves the results when PT is used, however not if the data is added to the dataset.

In general, it is remarkable that using $N_{\mathrm{train}} = 0$, i.e. only using data of the other parts for training, yields very competitive results compared to any other experiment. Similar results are achieved if only training on one page of a single part but using data augmentation. Therefore, in practice it can be expected that the staff detection algorithm generalised well on unseen layouts.

5.3. Symbol and Note Component Detection

The symbol detection algorithm acts on extracted images that contain a single staff. To extract these images as described in Section 4.4, we generally used the GT staff lines to gain scores that are independent of the staff line accuracy, however in Section 5.3.5, we will also evaluate on predicted staff lines to investigate the impact of staff lines that are too short or too long. Similar to the staff line

Table 5. Evaluation how the number of training instances influence the accuracy of pages with an unknown layout. All results are the averages of a five fold of each data set part. Only N_{train} pages are chosen for actual training. The major row grouping shows if pages of the remaining parts are included during training: Either no data is used (Default), a pretrained model on the data is used (PT), a large data set comprising all other data which is extended by N_{train} is used (Inc.) The first row shows the cross-training (CT.) results of Table 4 without any data of the target layout. The two major column sections show the results with and without data augmentation on the training data. The best values for each N_{train} are marked bold.

		No data augmentation			**Data augmentation**		
	N_{train}	F_1^{D}	F_1^{LF}	**Tot.** F_1	F_1^{D}	F_1^{LF}	**Tot.** F_1
CT.	0	0.993	0.987	**0.980**	0.982	0.986	0.968
Default	1	0.983	0.969	0.953	0.991	0.980	0.971
	2	0.986	0.976	0.962	0.990	0.985	0.975
	4	0.994	0.979	0.973	0.994	0.984	0.978
	All	0.989	0.984	0.974	0.994	0.986	0.979
PT	1	0.992	0.980	0.972	0.988	0.985	0.973
	2	0.992	0.981	0.973	0.993	0.984	0.977
	4	0.999	0.983	**0.982**	0.993	0.987	0.980
	All	0.995	0.984	0.980	0.998	0.986	**0.984**
Inc.	1	0.992	0.988	**0.980**	0.985	0.988	0.973
	2	0.992	0.987	**0.979**	0.985	0.984	0.969
	4	0.994	0.986	0.980	0.986	0.986	0.972
	All	0.994	0.988	0.982	0.988	0.986	0.974

detection, we first present the used metrics. Afterwards, we determine different hyperparameters and settings for the algorithm. Finally, the outcomes of training on different parts and varying the number of training examples are shown.

5.3.1. Metrics

We use several metrics for the symbol detection evaluation. First, we measure both the detection rates of symbols in general (F_1^{all}) and also the subgroups of notes (F_1^{note}) and clefs (F_1^{clef}). Thereto, we compare all GT and prediction symbols and match a pair as TP if two symbols are closer than 5px, all other symbols in the prediction and GT are treated as FP and FN, respectively. In the subgroups, we only account for symbols that match a note or clef in the GT or prediction. In the following, accidentals are not evaluated separately because no accidental is predicated at all.

Moreover, we measure how well types and staff locations of the note or clef subgroups are predicted. For each subgroup, our measure only takes TPs into account and counts the number of correctly predicted types ($Acc_{\text{type}}^{\text{note}}$ and $Acc_{\text{type}}^{\text{clef}}$), i.e. neume start and graphical connection for notes and clef type (C or F) for clefs, or staff positions ($Acc_{\text{pos}}^{\text{note}}$ and $Acc_{\text{pos}}^{\text{clef}}$).

Our last measure evaluates either the diplomatic or the harmonic transcription. Thereto, we compare the produced sequence with the GT and count the number of insertions, deletions or substitutes in the sequence, which is similar to OCR tasks. As diplomatic transcription, we compare staff position and types of notes and clefs, and their order, however, the actual horizontal position is ignored. Thus, only the resulting transcription is evaluated. The harmonic transcription is similar to the diplomatic one, but ignores the type of notes, which is why only the harmonic information i.e. the melody is captured. We represent these two sequence accuracy rates by dSAR and hSAR.

5.3.2. Preliminary Experiments

The first experiments listed in Table 6 test variations of the hyperparameters. Thereto, we use the setup of Figure 7 where a cross-fold on all available data is taken. We list all scores, however the

Table 6. Metrics of various hyperparameter settings for the symbol detection. All experiments were conducted using a cross-fold of size five using all available data. The best values for each metric are highlighted.

	Detection			Type		Position in Staff		Sequence	
	F_1^{all}	F_1^{note}	F_1^{clef}	$Acc_{\text{type}}^{\text{note}}$	$Acc_{\text{type}}^{\text{clef}}$	$Acc_{\text{pos}}^{\text{note}}$	$Acc_{\text{pos}}^{\text{clef}}$	hSAR	dSAR
$h_{\text{staff}} = 40$	0.926	0.936	0.523	0.929	**0.925**	0.963	0.992	0.834	0.792
$h_{\text{staff}} = 60$	0.953	0.961	0.655	0.944	0.911	0.968	0.994	0.877	0.844
$h_{\text{staff}} = 80$	**0.956**	0.964	**0.693**	0.949	0.897	**0.970**	0.990	**0.885**	**0.856**
$h_{\text{staff}} = 100$	**0.956**	**0.965**	0.673	**0.950**	0.911	**0.970**	**0.995**	0.884	**0.856**
Pad	0.962	0.966	**0.829**	0.949	**0.982**	0.967	0.990	0.893	0.866
Dewarp	0.952	0.959	0.708	0.946	0.873	0.969	0.987	0.877	0.845
Aug.	0.959	0.968	0.671	**0.953**	0.912	0.970	**0.992**	0.889	0.862
Pad, Aug.	**0.964**	**0.969**	0.818	0.952	**0.982**	**0.971**	**0.992**	**0.898**	**0.871**

Table 7. Symbol detection scores when testing on one and training on the remaining parts. The upper and lower rows show the results without and with data augmentation, respectively. Padding and a line height of 80px are used for all experiments.

			Detection			Type		Position in Staff		Sequence	
	Test	Train	F_1^{all}	F_1^{note}	F_1^{clef}	$Acc_{\text{type}}^{\text{note}}$	$Acc_{\text{type}}^{\text{clef}}$	$Acc_{\text{pos}}^{\text{note}}$	$Acc_{\text{pos}}^{\text{clef}}$	hSAR	dSAR
Def.	1	2, 3	0.956	0.965	0.782	0.914	1.000	0.981	0.993	0.906	0.860
	2	1, 3	0.912	0.916	0.638	0.928	0.975	0.955	0.990	0.797	0.757
	3	1, 2	0.950	0.952	0.907	0.958	0.968	0.981	1.000	0.895	0.867
	Mean		0.939	0.944	0.776	0.933	0.981	0.972	0.994	0.866	0.828
Aug.	1	2, 3	0.958	0.964	0.822	0.914	1.000	0.980	0.990	0.907	0.862
	2	1, 3	0.939	0.945	0.699	0.947	0.975	0.965	0.990	0.848	0.817
	3	1, 2	0.943	0.945	0.901	0.959	0.956	0.981	1.000	0.886	0.859
	Mean		0.946	0.951	0.807	0.940	0.977	0.976	0.993	0.881	0.846

main focus lies on the general symbol detection, hSAR, and dSAR, since these metrics do not act on subgroups but on the whole prediction.

The upper block shows the results when the resolution of the input line is varied. The bold numbers highlighting the best value for each score indicate that the accuracies seem to stagnate which is why in the following we use a line height of 80px.

Using this hyperparameter, we evaluated the outcome of using padding, dewarping, and data augmentation. Since padding is designed to add space to the left of a staff particularly where clefs are located, the highest score of clef type detection are achieved. As a result, the overall accuracies improve as well. Dewarping yields slightly worse results in any metric, which is why in the following, it was omitted in each experiment. We explain this unexpected behaviour with the induced distortions which complicates the detection of notes and especially their connection. Data augmentation slightly improved the results compared to the default model. Since 80% of all data is used for training it is to be expected, that data augmentation improves the results more clearly if less data is used for training. We show this behaviour in Section 5.3.4. The last row shows the best overall result reached when combining data augmentation, padding, and choosing a line height of 80px.

5.3.3. Results of cross-part training

Table 7 show the result of cross-part training, i.e. testing on one of the three parts and training on the two remaining ones. In general, as expected, data augmentation improves the results. The highest accuracies are achieved on part 1 and 3 in almost all metrics, which is explained by the greater similarities of those two parts and in general a clearer notation.

Table 8. Symbol detection scores when varying the number of training examples. The first line is the average of the cross-part training (CT.) experiment of Table 7 where no pages of the target part are used. The next rows (Default) only use N_{train} of the target data set for training. Afterwards, a pretrained model is used (PT.) or the additional part data is included (Inc.).

		Detection			**Type**		**Position in Staff**		**Sequence**	
	N_{train}	F_1^{all}	F_1^{note}	F_1^{clef}	$Acc_{\text{type}}^{\text{note}}$	$Acc_{\text{type}}^{\text{clef}}$	$Acc_{\text{pos}}^{\text{note}}$	$Acc_{\text{pos}}^{\text{clef}}$	**hSAR**	**dSAR**
CT.	0	0.946	0.951	0.807	0.940	0.977	0.976	0.993	0.881	0.846
Default	1	0.846	0.854	0.533	0.862	0.963	0.954	0.986	0.730	0.661
	2	0.899	0.905	0.706	0.901	0.989	0.962	0.998	0.806	0.749
	4	0.926	0.930	0.775	0.927	0.982	0.970	0.990	0.846	0.804
	All	0.954	0.958	0.861	0.945	0.987	0.977	0.998	0.897	0.867
PT.	1	0.925	0.929	0.760	0.907	0.978	0.956	0.992	0.834	0.777
	2	0.939	0.942	0.796	0.928	0.981	0.966	0.999	0.862	0.821
	4	0.953	0.956	0.871	0.944	0.987	0.972	0.996	0.890	0.858
	All	0.969	0.971	0.910	0.950	0.989	0.975	0.997	0.916	0.888
Inc.	1	0.948	0.953	0.808	0.936	0.982	0.977	0.997	0.882	0.846
	2	0.952	0.958	0.793	0.939	0.978	0.976	0.997	0.887	0.854
	4	0.959	0.964	0.817	0.948	0.980	0.976	0.996	0.900	0.870
	All	0.961	0.966	0.846	0.950	0.979	0.977	0.996	0.905	0.878

The dSAR of any part is considerably lower than the value of 86.6% and 87.1% yielded when training on all data (compare Table 6) with and without data augmentation, respectively. Also the ratio of detected symbols F_1^{all} is significantly lower, due to the variations in style across the parts. $Acc_{\text{pos}}^{\text{note}}$ and $Acc_{\text{pos}}^{\text{clef}}$ are similar to training on all data which shows that if a symbol is correctly detected, its position on the staff lines is usually correct. This behaviour is expected because these errors are almost always intrinsic ambiguities which are independent of the layout. Furthermore, the average clef type accuracy $Acc_{\text{type}}^{\text{clef}}$ of 97.7% in average is also very high. Therefore the network is mostly certain if a C- or F-clef is written even though they share similarities. The decision whether a NC is graphically or logically connected is correct in only 94.0% of all detected notes and slightly lower than the 95.2% reported in Table 6. This shows, that the FCN can transfer information about the arrangement of NCs that build typical neumes.

5.3.4. Varying the number of training examples

In this section, we also train on two parts and test on the remaining part, however actual data of the target part is included. Table 8 lists the results when using 0, 1, 2, 4, or finally all pages.

As expected, an increasing number of training examples leads in general to improved recognition rates and thus higher scores in any experiment. However, it is remarkable that the model which only comprises one page without any other techniques already yields an dSAR of 66.1% and a hSAR of 73.0%. Nevertheless, the scores significantly improve if other data is incorporated even if no page of the target part is included at all ($N_{\text{train}} = 0$), yielding a dSAR of 84.6% and a hSAR of 88.1%. Only if all available pages are used for training (dSAR $= 86.7\%$), the default model is slightly better than training on the other parts only (dSAR $= 84.6\%$).

However, there are huge differences when to use pretraining or data inclusion. If only few data of the target set is available (1, 2, 4) including the data yields better results, but if many examples are used during training PT is superior. The reason is that the PT model yields a good starting point which is then optimised for one specific part if enough target data is used. Whereas, if all data is included, the model is forced to generalise on any part and not on a specific one.

Furthermore, the experiments show that the accuracy of the position in staff increase with a higher number of training instances in the default model or when using PT, but are almost constant if zero or

Table 9. Error analysis: The relative amount of errors of the output sequence is categorised.

Group		Error
False Negatives		*51.7%*
1.	Missing Note	21.2%
2.	Wrong Connection	10.8%
3.	Wrong Location	14.3%
4.	Missing Clef	3.5%
5.	Missing Accidental	1.9%
False Positives		*48.3%*
6.	Additional Note	20.6%
2.	Wrong Connection	10.8%
3.	Wrong Location	14.3%
7.	Additional Clef	2.6%
8.	Additional Accidental	0.0%
	Total	100%

all instances are added to the training set. Obviously, having more data available it is easier to find and remember good rules that state how to resolve incertitudes.

On the whole, the model training if no data of the target part already yields a very robust F_1^{all}- and dSAR-score. Even if PT and all data is used the dSAR increases only from 88.1% to 91.5%, which is a relative improvement of about 4%. However, we highly expect that if the differences between training and testing data are even more clear, the effect will increase.

5.3.5. Symbol detection on predicted staves

Finally, we test the combination of both algorithms. Thus, in this experiment, the impact of shortened or too long staff lines of the predicted staves can be investigated. Thereto, we only evaluated on staves that are marked as TPs which is why for an combined result the amount of detected staves must be included, but which is almost 1. We evaluate on the model which is trained on a cross-fold of all pages combined.

The experiment yields a hSAR of 88.9% and a dSAR of 86.2% which is slightly worse than the results obtained if predicted on the corrected lines (89.8% and 87.1% respectively). The sole new errors that can occur are FPs if the staves are expanded into other content such as drop capitals or text, or FNs if the staves are too short.

In conclusion the dSAR decreases by only 1% on detected staves due to wrong staff lengths. The score is further reduced by approximately 1% when accounting for FPs and FNs during the staff prediction.

5.3.6. Error analysis

Regarding the error analysis, we compare the predicted and GT sequences and count how many deletions or insertions of a specific type are required to gain the GT. We use the model, when training and evaluating on a cross fold comprising all data, which yields a dSAR of 87.1% (compare Table 6). Note that in this section replacements are treated as two errors: one insertion and one deletion. Table 9 lists eight groups of errors and their relative impact on the total dSAR whereas Figure 10 shows examples for the errors. The first result is that the combined amount of FPs and FNs are balanced (48% vs 52%). Notes are missed (1) most often if the contrast between notes and background is too low or if neumes are written very dense, which also highly affects the prediction accuracy of connection types (2). Wrong note positions (3) occur only at locations where even for a human it is difficult to decide whether the NC is located on a line or in a space. Mainly neumes with stacked NCs e.g. a so-called pes (first neume in Figure 1) or scandicus suffer from these errors. These neumes also induce errors in reading order which is only captured by SAR metrics. Since the NCs are ordered left to right based on

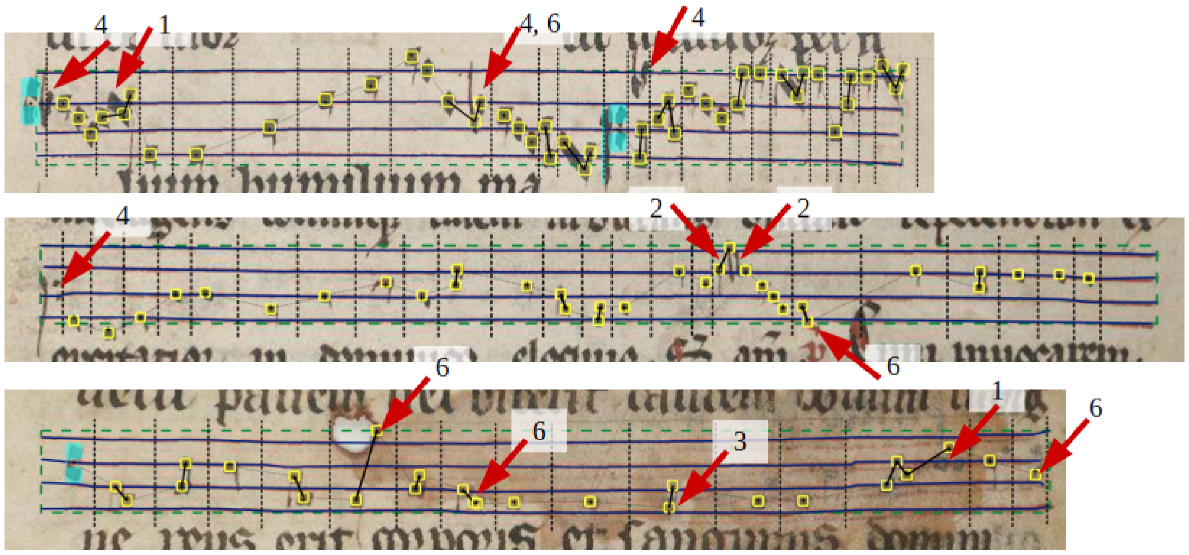

Figure 10. The three lines exemplarily show typical errors of the symbol detection algorithm. Refer to Table 9 for the definition of the numbers indicating different groups. The yellow boxes mark single NCs, solid lines between two NCs indicate that the second note is graphically connected to the previous. The dashed vertical lines represents logical connections between neumes. The cyan symbols denote clefs. The connecting thin dotted line represents the reading order of the symbols.

Table 10. Prediction and training times in seconds. For the staff line detection the numbers denote the time to process a whole page, while for the symbol detection the reference is a single staff. Prediction includes pre- and postprocessing, however training only refers to a single iteration.

	Staff Line		**Symbols**	
	GPU	**CPU**	**GPU**	**CPU**
Predict	3.5	3.7	0.15	0.12
Train	0.12	3.6	0.04	0.42

their prediction position, it happens, that the upper note which should be sung later, is predicted a little to far to the left compared to the lower note. Observations show, that the first clef in a staff was almost always detected, whereas a clef in a line was very often missed (4) and sometimes misrecognised as two notes. Accidentals are not detected at all (5) possibly due to their underrepresentation in the available data set.

Additional notes (6) mostly occur if clefs or accidentals are labeled as two notes, or if noise or bleeding is falsely detected as symbols. Wrong connections and locations count also as FP which is why their ratio is the same. The amount of FPs that are clefs (7) is very low, and since no accidentals are predicted at all no FPs (8) can occur.

Concluding, it can be observed that the majority of errors are as expected note related, however missing (21.2%), additional (20.6%), wrong connection (21.6%), and wrong location (28.6%) are roughly balanced.

5.4. Timing

Finally, we measure the times required for predicting staff lines or the symbols of a staff line including data loading and pre- and postprocessing steps (see Table 10). The time for loading the model which is only required once is included. All times are measured on a Nvidia Titan X GPU and an Intel Core i7-5820K processor, while the FCN is implemented in Tensorflow. Moreover, we obtain the results when using only the CPU or if a GPU is available.

Table 11. Overview of the main results of this paper.

Task	Score
F_1-scores of staff line & staff detection	$> 99\%$
F_1-score of detected Symbols	$> 96\%$
Diplomatic sequence accuracy (dSAR)	$\approx 87\%$

To process a full page the staff line detection requires 3.7 (CPU) and 3.5 (GPU) seconds, i.e. about 3 minutes for the full dataset. The preprocessing step which normalises the input image takes about half of the time. The application of the FCN follows with about 40%, however the difference between CPU and GPU is minor. This can be explained by the shorter model loading time when using the CPU and the faster computation time but a required data transfer when using the GPU. The remaining time is consumed by the postprocessing which generates the final output.

The symbol detection requires about 0.12 (CPU) and 0.15 (GPU) seconds for the prediction of a single staff, i.e. the complete dataset takes one minute to predict. Because no preprocessing besides cutting the image is required and the postprocessing step only consists of a connected component analysis it is clearly faster than the pre- and postprocessing of the staff line detection, yet it is the step consuming the most time. Compared to the staff detection the CPU is even a bit faster due to the smaller input line image. In total about 4 minutes are required to predict both the staves and symbols on all 49 pages.

Especially for training, GPU usage is almost indispensable to enable a fast back-propagation which computes the weight differences required for training the network. Compared to the CPU the staff line training on the GPU that acts on the full page yields a speed-up of 30 which is justified with the convolution and pooling operations of the FCN that are highly optimised and perfectly suited for the GPU. Note that training a single instance is faster than prediction because pre- and postprocessing steps are not required.

6. Conclusion and Future Work

This paper introduces new methods for automatic staff line and symbol detection on historical medieval notations based on deep neural networks. The main results on our data set are (compare Table 11):

- Staff lines and staves are recognised with an F_1-score greater than 99%. The major error sources are staves that are detected on bled background or which are extended into other foreground (e.g. text).
- A trained staff line model generalises flawlessly to different layout types and thus needs no retraining.
- Our best model finds symbols with an F_1-score of above 96% and yields a dSAR of about 87%.
- Pretraining is helpful but not mandatory for the symbol detection, since the dSAR only increases about 4%.

To further improve the staff detection which is close to perfect though, more experiments must be conducted to evaluate the effect of data augmentation which is unclear in the proposed manner. Especially augmentations in brightness and contrast must be reconsidered to allow for more robust models. Furthermore, staves that are detected on background can be dropped by introducing a threshold of minimum symbols per staff, because the symbol detection only occasionally detects symbols on background. Another problem are staves that extend into text or drop capitals. However a symbol detection which is explicitly trained to ignore text could help to determine the boundaries between staff and text because no symbols should be detected there.

However, the primary goal of our future work is to further reduce the errors of the symbol detection. First, the error analysis shows that false notes are the main source of symbol errors. Especially the connection and location of notes induces many errors, which however also for humans is a difficult tasks. We propose to evaluate a two stage approach similar to [3] that first predict neumes as combined components and then separates them into individual NCs, however with incorporation of deep neural networks for object detection. Furthermore, to improve the quality of clef or accidental prediction more data must be gathered or over-sampled in the training data either by data augmentation or by presenting these lines more often.

A completely different approach which is based on the state-of-the-art networks in OCR is to directly use sequence-to-sequence networks as already proposed by [25] on contemporary OMR. A CNN/LSTM hybrid network with combination of a CTC-loss function directly predicts a NC sequence. The main advantage is that GT production is easier since only the sequence has to be transcribed but not the actual position of each single NC. However, it is to be expected that more GT is required since it has to learn autonomously both the shape of all music symbols, and the staff lines to infer the location of the symbols. It is still an open issue whether this CTC-approach yields in fact improved results. Furthermore, because this approach directly predicts a symbol sequence and no actual note positions relative to the image, it cannot be used if this information is mandatory.

Author Contributions: C. W. conceived and performed the experiments and created the GT data. C. W. and A. H. contributed the staff line detection algorithm. C. W. designed the symbol detection algorithm. C. W. and F. P. analysed the results. C. W. wrote the paper with substantial contributions of F. P.

Funding: This research received no external funding

Acknowledgments: We would like to thank Tim Eipert for helping to resolve ambiguities in the data set.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Network
CPU	Central Processing Unit
CTC	Connectionist Temporal Classification
dSAR	Diplomatic Sequence Accuracy Rate
FCN	Fully Convolutional Network
FN	False Negative
FP	False Positive
hSAR	harmonic Sequence Accuracy Rate
LSTM	Long Short-Term Memory
GPU	Graphical Processing Unit
GT	Ground Truth
NC	Note Component
OCR	Optical Character Recognition
OMR	Optical Music Recognition
PT	Pretraining
RLE	Run Length Encoding
TP	True Positive

References

1. Corbin, S.; Institut, B.M. *Palaeographie der Musik.[3]. Die Neumen*; Volk, 1977.
2. Vigliensoni, G.; Burgoyne, J.A.; Hankinson, A.; Fujinaga, I. Automatic pitch recognition in printed square-note notation. Proc. ISMIR, 2011.
3. Ramirez, C.; Ohya, J. Automatic recognition of square notation symbols in western plainchant manuscripts. *Journal of New Music Research* **2014**, *43*, 390–399.
4. Baró, A.; Riba, P.; Calvo-Zaragoza, J.; Fornés, A. Optical Music Recognition by Long Short-Term Memory Networks. International Workshop on Graphics Recognition. Springer, 2017, pp. 81–95.
5. Keil, K.; Ward, J.A. Applications of RISM data in digital libraries and digital musicology. *International Journal on Digital Libraries* **2019**, *20*, 3–12.

6. Fornés, A.; Dutta, A.; Gordo, A.; Lladós, J. CVC-MUSCIMA: a ground truth of handwritten music score images for writer identification and staff removal. *International Journal on Document Analysis and Recognition (IJDAR)* **2012**, *15*, 243–251.
7. van der Wel, E.; Ullrich, K. Optical Music Recognition with Convolutional Sequence-to-Sequence Models. Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017; Cunningham, S.J.; Duan, Z.; Hu, X.; Turnbull, D., Eds., 2017, pp. 731–737.
8. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. Advances in neural information processing systems, 2014, pp. 3104–3112.
9. Good, M. MusicXML for notation and analysis. *The virtual score: representation, retrieval, restoration* **2001**, *12*, 113–124.
10. Calvo-Zaragoza, J.; Rizo, D. End-to-end neural optical music recognition of monophonic scores. *Applied Sciences* **2018**, *8*, 606.
11. Graves, A.; Fernández, S.; Gomez, F.; Schmidhuber, J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. Proceedings of the 23rd international conference on Machine learning. ACM, 2006, pp. 369–376.
12. Breuel, T.M. High Performance Text Recognition Using a Hybrid Convolutional-LSTM Implementation. 14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017. IEEE, 2017, pp. 11–16. doi:10.1109/ICDAR.2017.12.
13. Wick, C.; Reul, C.; Puppe, F. Comparison of OCR Accuracy on Early Printed Books using the Open Source Engines Calamari and OCRopus. *JLCL* **2018**, *33*, 79–96.
14. Song, W.; Cai, J. End-to-end deep neural network for automatic speech recognition. *Standford CS224D Reports* **2015**.
15. Zhang, Y.; Chan, W.; Jaitly, N. Very deep convolutional networks for end-to-end speech recognition. 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2017, pp. 4845–4849.
16. Pugin, L.; Zitellini, R.; Roland, P. Verovio: A library for Engraving MEI Music Notation into SVG. Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014; Wang, H.; Yang, Y.; Lee, J.H., Eds., 2014, pp. 107–112.
17. Pacha, A.; Choi, K.Y.; Coüasnon, B.; Ricquebourg, Y.; Zanibbi, R.; Eidenberger, H. Handwritten music object detection: Open issues and baseline results. 2018 13th IAPR International Workshop on Document Analysis Systems (DAS). IEEE, 2018, pp. 163–168.
18. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 2015, pp. 91–99.
19. Dai, J.; Li, Y.; He, K.; Sun, J. R-fcn: Object detection via region-based fully convolutional networks. Advances in neural information processing systems, 2016, pp. 379–387.
20. Hajič, J.; Pecina, P. The MUSCIMA++ dataset for handwritten optical music recognition. 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2017, Vol. 1, pp. 39–46.
21. Calvo-Zaragoza, J.; Toselli, A.H.; Vidal, E. Early handwritten music recognition with hidden markov models. 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR). IEEE, 2016, pp. 319–324.
22. Calvo-Zaragoza, J.; Toselli, A.H.; Vidal, E. Handwritten music recognition for mensural notation: formulation, data and baseline results. 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2017, Vol. 1, pp. 1081–1086.
23. Miyao, H. Stave extraction for printed music scores using DP matching. *Journal of Advanced Computational Intelligence and Intelligent Informatics* **2004**, *8*, 208–215.
24. Dalitz, C.; Droettboom, M.; Pranzas, B.; Fujinaga, I. A comparative study of staff removal algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2008**, *30*, 753–766.
25. Calvo-Zaragoza, J.; Castellanos, F.; Vigliensoni, G.; Fujinaga, I. Deep neural networks for document processing of music score images. *Applied Sciences* **2018**, *8*, 654.
26. Wick, C.; Puppe, F. Fully Convolutional Neural Networks for Page Segmentation of Historical Document Images. 13th IAPR International Workshop on Document Analysis Systems, DAS 2018, Vienna, Austria, April 24-27, 2018. IEEE Computer Society, 2018, pp. 287–292. doi:10.1109/DAS.2018.39.

27. Rebelo, A.; Fujinaga, I.; Paszkiewicz, F.; Marcal, A.R.S.; Guedes, C.; Cardoso, J.S. Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval* **2012**, *1*, 173–190. doi:10.1007/s13735-012-0004-6.
28. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III; Navab, N.; Hornegger, J.; III, W.M.W.; Frangi, A.F., Eds. Springer, 2015, Vol. 9351, *Lecture Notes in Computer Science*, pp. 234–241. doi:10.1007/978-3-319-24574-4_28.
29. Fujinaga, I. Staff detection and removal. In *Visual Perception of Music Notation: On-Line and Off Line Recognition*; IGI Global, 2004; pp. 1–39.
30. Cardoso, J.S.; Rebelo, A. Robust Staffline Thickness and Distance Estimation in Binary and Gray-Level Music Scores. 20th International Conference on Pattern Recognition, ICPR 2010, Istanbul, Turkey, 23-26 August 2010. IEEE Computer Society, 2010, pp. 1856–1859. doi:10.1109/ICPR.2010.458.