TISMIR

Schneider, D., et al. (2021). Automatic Transcription of Organ Tablature Music Notation with Deep Neural Networks. *Transactions of the International Society for Music Information Retrieval,* 4(1), pp. 14–28. DOI: https://doi.org/10.5334/tismir.77

RESEARCH

Automatic Transcription of Organ Tablature Music Notation with Deep Neural Networks

Daniel Schneider, Nikolaus Korfhage, Markus Mühling, Peter Lüttig and Bernd Freisleben

Organ tablature music notation differs considerably in structure and form from the music notation used today. The manual transcription of organ tablature compositions to modern music notation is time-consuming and often prone to errors. In this paper, we present a deep learning approach to automatically recognize organ tablature notation in scanned documents and transcribe it to modern music notation. Our approach is aimed at generating a uniform transcription that remains as close as possible to the original sheet music and therefore does not perform automatic error correction or musical interpretation. The artificial neural network model developed for the recognition of tablature characters is trained using a combination of real annotated tablature staves and tablatures produced by a synthetic data generator. The results of our experiments are evaluated on tablatures taken from two tablature books. We identify several types of error and validate that these are primarily caused by the poor legibility of relevant parts of some tablature scans. Overall, our approach achieves an accuracy of 97.2% and 99.3% correctly recognized bars, depending on whether note pitch and rest characters or note duration and special characters are considered, respectively.

Keywords: Organ Tablature; Automatic Transcription; Deep Learning; OCR; OMR

1. Introduction

The analysis of historical music notation is a major research topic in the field of musicology. Often, a manual transcription of the source material into modern music notation is required to make the material accessible to a wider audience and facilitate musicological analyses. Manual transcription, however, is a time-consuming and error-prone process.

The *New German Organ Tablature* is one such old music notation. It is studied by musicologists and is important to improve our knowledge about renaissance music. Several archives contain large numbers of organ tablatures, some of which have neither been digitized nor been transcribed to modern notation yet (Motnik, 2011; Wojnowska, 2016).

In this paper, we present a deep learning approach that automatically transcribes scanned organ tablature pages to modern music notation. First, our method segments each input image into the corresponding tablature staves and recognizes tablature characters in the resulting partial images using a deep neural network. Then, the results of this process are converted to the format of Lilypond,[1] an open-source music notation program that can be used to generate a graphical output in modern notation. An example of such an automatic transcription is shown in **Figure 1**.

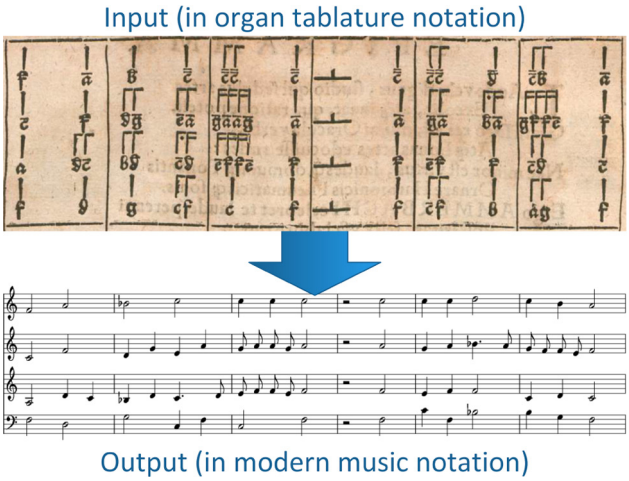

Figure 1: Transcription of a tablature row into modern music notation. The transcribed row consists of four tablature staves that are converted into a four-part score in modern notation.

We utilize two scanned organ tablature books as data sources for training our deep neural network. Using data augmentation and a synthetic data generator that we developed as part of our work, we generated a data set of sufficient size to perform the training. This data set and the tools to create it are available online.[2]

We present the results of an experimental evaluation of the performance of the proposed approach. The neural network achieves an accuracy of 97.2% and 99.3% correctly

Department of Mathematics and Computer Science, University of Marburg, Marburg, DE

Corresponding author: Daniel Schneider (schneider@informatik.uni-marburg.de)

recognized bars, depending on whether note pitch and rest characters or note duration and special characters are considered, respectively. On the average, an error occurs every 220th pitch/rest character and every 833rd duration/special character.

The contributions of this paper are as follows:

- We apply, for the first time, a deep learning method to automatically transcribe organ tablatures into modern music notation.
- We present a deep neural network architecture called Character Sequence Pair (CSP) network that is trained to recognize character sequences arranged on two lines without requiring bounding box annotations.
- We present synthetic data generation and augmentation tools for organ tablature music that could be adapted for other tasks in musicology.
- We provide a data set with ground-truth label sequences for training a neural network to recognize sequences of tablature characters.

The paper is organized as follows. Section 2 gives an introduction to organ tablature notation. Section 3 reviews related work. Section 4 presents our deep learning approach to recognize tablature notation. Section 5 explains our data generation and augmentation process. Section 6 presents experimental results. Section 7 concludes the paper and outlines areas for future work.

2. Organ Tablature Notation

Organ tablature music notation originates from the mid-14th century as a representation of multipart vocal music for keyboard or string instruments (Wolf, 1919). It differs from modern musical notation with 5 staves. There is a Spanish and a German form of organ tablature letter notation. The German form can be further divided into an older and a newer one. The *New German Organ Tablature* notation was not only used to spread free popular compositions, but also in the guild-based education of organists in the 17th and 18th centuries. The most prominent example is Johann Sebastian Bach, who used organ tablature notation for transcriptions of works of Dietrich Buxtehude and Johann Pachelbel during his lessons with Georg Böhm (Maul and Wollny, 2007). Organ tablature notation, which saved space and paper when writing down compositions, disappeared from the organists' horizon with the decline of church music in the 18th century.

2.1 Musicological Background

It was only with the rediscovery of early music, especially vocal music in mensural notation (which began in the second half of the 19th century (Bellermann, 1858)) that musicological interest in organ tablature notation was reestablished. Since then, transcriptions of organ tablatures into modern music notation have formed the basis for the music-making patterns required to perform this music (Apel, 1967, 2006). In the contemporary training of organists, however, organ tablature is found only in a few exceptional cases.

On December 6, 2017, UNESCO has included organ manufacturing and organ music in Germany in the list of the intangible cultural heritage of mankind (German UNESCO Commission, 2018). Therefore, the identification and philologically correct transcription of organ tablatures is becoming an important topic.

Due to the different nature of their notation and a dwindling knowledge of organ tablatures in general, some tablatures have not even been recognized as pieces of music. For example, Johann Sebastian Bach's oldest manuscript, the Weimar organ tablature (Maul and Wollny, 2007), was for a long time considered a cabalistic work and was therefore assigned to the field of theology. For the scientifically unambiguously identified tablature sources, transcriptions to modern notation do not always exist. For example, a significant part of the organ tablatures with intabulations of vocal music have not (or have only partially) been transcribed (Motnik, 2011). This includes works that explore previously unknown collections, but do so without a complete source-critical transfer (Wojnowska, 2016). Furthermore, the problems of transcription and reconstruction of music manuscripts preserved in the New German Organ Tablature notation have only been examined more closely in individual cases, as in Warsaw (Hulková, 2015) or Prague (Horyna, 2018).

But even when transcriptions into modern notation exist, they are not always uniform, transparent, and philologically accurate. **Figure 2** shows an example of two transcriptions of a few bars of the "Orgel oder Instrument Tabulaturbuch" ("Organ or instrument tablature book") by Elias Nikolaus Ammerbach (Ammerbach, 1583), one of the first printed books containing New German Organ Tablature. The differences in the transcriptions of Cecil Warren Becker (Becker, 1963) and Hans-Thomas Müller-Schmidt (Müller-Schmidt, 2017) are apparent. Müller-Schmidt octaves the alto voice in bars two and three, while in Becker's transcription it is in the position that the original dictates. In the bass voice, too, the second note in bar two is transcribed once as 'A' (incorrect), the other time as 'B' (correct). These differences demonstrate that the results depend on the particular transcription approach and the individual knowledge of the two authors. This example indicates that uniform methods of transcription, as currently used in the field of music of the 15th and 16th centuries (Huang et al., 2015; Calvo-Zaragoza et al., 2016, 2019), are indispensable.

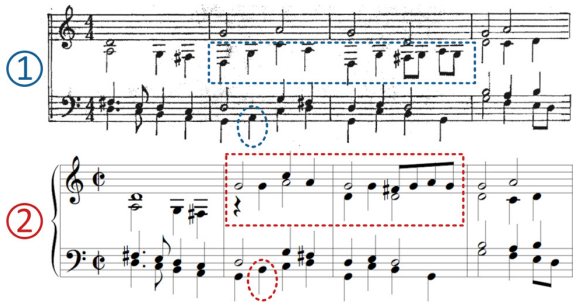

Figure 2: Deviations in transcriptions of Ammerbach's "Orgel oder Instrument Tabulaturbuch" by (1) Becker (1963) and (2) Müller-Schmidt (2017).

2.2 Automatic Organ Tablature Transcription

The use of automated methods for the transcription of organ tablatures promises both to simplify the time-consuming task of manual transcription and to guarantee a standardized transcription. Automated methods can also increase the number of available notation examples for the analysis of musical semantics, for instance with regard to harmonic features or contrapuntal processes. This would greatly support musicological research in this area.

However, a method applicable to all kinds of organ tablature seems quite difficult to develop, due to the large variation. Organ tablatures are mostly preserved as handwritten manuscripts and rarely exist as printed versions. In general, tablatures from different sources can vary considerably in layout and font style.

In the context of this work, we focus on two publications of printed tablature music by the German organist and arranger Elias Nikolaus Ammerbach: the "Orgel oder Instrument Tabulaturbuch" ("Organ or instrument tablature book") (Ammerbach, 1583) and "Ein new künstlich Tabulaturbuch" ("A new artificial tablature book") (Ammerbach, 1575).

Using Ammerbach as an example for developing an automatic organ tablature transcription is justified by the fact that his tablature books are among the first printed works on this subject. In addition, Ammerbach offers a complete basic course for organists: up to the fingering, all topics are covered in his books. Last but not least, Ammerbach's books have been a formative influence for many organists due to their wide distribution in the German-speaking world.

2.3 Character Set

Unlike modern musical notation, the New German Organ Tablature does not use staff lines, and the pitch is not indicated by the positioning of note marks. Instead, a letter notation is used, in which the pitches are specified as a sequence of note names. The octave position of the notes is indicated by upper and lower case letters and additional horizontal strokes above the characters. The note duration is indicated by rhythmic symbols placed above the pitch symbols, whose appearance is similar to the note stems and bars of modern music notation. For pause signs and special characters, additional symbols exist, which also resemble symbols of modern music notation.

Figure 3 shows examples of the different types of tablature characters taken from Ammerbach's tablature books (labeling from left to right):

1. Note duration symbols: whole note, half note, two quarter notes, four quavers
2. Note pitch symbols for the note *g* (from high to low): two-line octave position, one-line octave position, small octave position (without strokes), great octave position (in capital letters)
3. Pause signs: whole rest, half rest, quarter rest, eighth rest
4. Special characters: repetition mark, time change (three-four time)

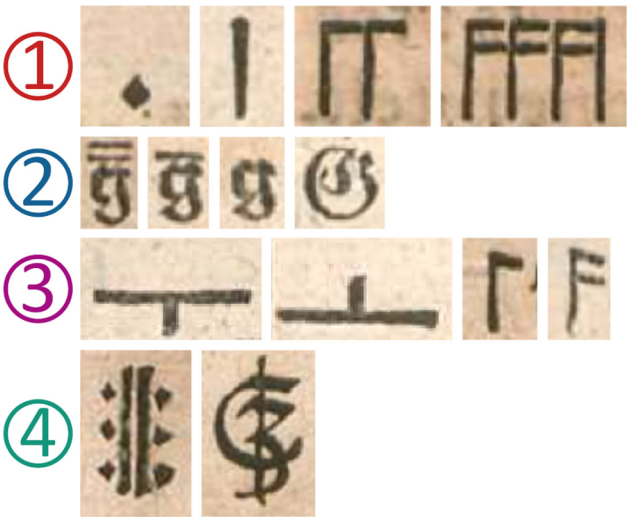

Figure 3: Examples of the different types of organ tablature symbols taken from Ammerbach (1583): (1) Note duration symbols; (2) Note pitch symbols; (3) Pause signs; (4) Special characters.

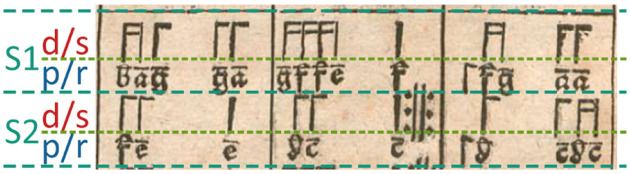

Figure 4: The layout of printed tablature characters using the example of Ammerbach (1583). Each staff (*S1*, *S2*) inside the row consists of two lines (*duration/special (d/s)* and *pitch/rest (p/r) line*) in which the tablature characters are arranged.

2.4 Page Layout

An organ tablature page consists of several *rows* separated by horizontal lines. Some editors additionally divide their tablatures into *bars* by vertical lines. Each row consists of *staves* arranged one below the other, separated only by a small distance, with one staff for each voice of the composition.

The tablature characters of a voice are arranged in two lines, as shown in **Figure 4**. The upper line contains note duration symbols, while the lower line contains note pitch symbols and pause signs. The position of special characters may vary from editor to editor, but they are usually located in the upper line. We therefore refer to the upper line as *duration/special line* and to the lower line as *pitch/rest line*.

3. Related Work

The analysis and transcription of organ tablature music notation has not been a research topic outside the field of musicology and thus no research on this particular topic exists in the field of computer science. However, the research areas Optical Character Recognition (OCR) and Optical Music Recognition (OMR) deal with the automatic recognition of handwritten or printed text and notes on images, respectively, and are thus related to the topic of organ tablature character recognition.

In all of these tasks, the goal of the analysis is to recognize characters of a given alphabet in an input image. In OMR, however, the positioning of the characters in relation to each other also plays an important role. The pitch, for example, cannot be derived from the symbols alone, but is determined by the position of the note head on the staff. The necessity of analyzing these semantic relationships makes note recognition more complex than text recognition, in which individual characters can be recognized and transcribed independently of each other (Calvo-Zaragoza et al., 2020; Bainbridge and Bell, 2001).

The automated analysis of organ tablatures poses similar challenges. Here, the alphabet consists of a comparatively large number of characters, many of which are quite similar. The relative position of the characters to each other plays a role as well, in particular for note duration and note pitch characters that together represent one note.

In OCR and OMR, the recognition process is usually carried out in several stages that commonly pass through the same basic steps (Pansare and Joshi, 2012; Awel and Abidi, 2019; Patel and Thakkar, 2015; Rebelo et al., 2012; Calvo-Zaragoza et al., 2020). First, the input images are pre-processed to better discriminate the foreground and background from each other. Commonly performed steps are noise removal (application of a smoothing filter), binarization (conversion to grayscale with thresholding), and deskewing (correction of image orientation and reduction of distortions) (Awel and Abidi, 2019; Patel and Thakkar, 2015; Rebelo et al., 2012; Calvo-Zaragoza et al., 2020).

After pre-processing, the actual character analysis is performed. In the past, several methods have been used to perform OCR, such as pattern matching (e.g., Chain Code Histogram (CCH)), statistical models (e.g., Hidden Markov Model (HMM)), k Nearest Neighbor (kNN) and kernel-based machine learning methods (e.g., Support Vector Machine (SVM)) (Purohit and Chauhan, 2016). In recent years, however, deep neural networks, in particular Convolutional Neural Networks (CNNs), have increasingly become the standard for image analysis tasks and often provide the best results in this area. Almost all current approaches for text or note recognition use artificial neural networks, but in some cases they are combined with other machine learning methods such as HMMs or SVMs (Patel and Thakkar, 2015; Purohit and Chauhan, 2016; Rebelo et al., 2012).

A common approach to automatically process printed documents is to first segment the input images into individual objects (e.g., single letters or musical symbols) to be analyzed further by a neural network. In this case, bounding boxes indicating the position of the individual characters are required for labeling the training data for the neural network. When an input is analyzed, the characters are recognized individually and are not combined into a single output as part of the recognition task. However, semantic information such as the coherence of characters (e.g., the position of note heads on staves) is lost when characters are recognized independently. This information has to be restored for merging the results, which requires additional effort. Examples of this approach can be found in the works of Feng et al. (2017) for handwritten text and Tuggener et al. (2018) for scanned sheet music.

Instead, so-called sequence-to-sequence approaches, in which larger units (e.g., entire rows) are recognized at once, can be used. For this purpose, Recurrent Neural Networks (RNNs) are usually employed in combination with CNNs. When recognizing larger sequences, semantic relationships between characters in the sequence are preserved, which simplifies the combination of results considerably. In addition, the creation of ground truth data required for training a neural network is less expensive, since bounding boxes for each character are usually not required. Examples of this approach can be found by Su and Lu (2017) or Dutta et al. (2018) for handwritten texts, and Calvo-Zaragoza et al. (2017); Calvo-Zaragoza and Rizo (2018) or Alfaro-Contreras et al. (2019) for sheet music.

Typically, a post-processing step follows after the recognition step is finished. If the recognition was performed on smaller units, the individual results are now combined. When recognizing notes, the semantic relationships of the detected characters must be determined (for example, the position of a note on the staff) and transferred to a data structure in which these relationships are modeled. After the results have been merged, an automatic error correction can be performed, in which the syntax and semantics of the analysis results are examined using dictionaries, and identified analysis errors are corrected. In OMR, the results are finally encoded in the desired output format (e.g., MusicXML or MIDI) (Patel and Thakkar, 2015; Rebelo et al., 2012; Calvo-Zaragoza et al., 2020).

4. Deep Tablature Transcription

Our deep learning approach for organ tablature transcription receives scanned documents in organ tablature notation as its input and outputs a corresponding transcription of the musical score in modern notation (i.e., Lilypond[1] format).

For multi-page documents, the individual scanned pages are first extracted from the input document and processed sequentially. The transcription process consists of three successive steps with several sub-steps:

1. Pre-processing: deskewing and segmentation of the input images into tablature rows and staves
2. Recognition: recognition of tablature characters in the individual staves
3. Post-processing: merging of the recognition results into a combined result and generation of output files

4.1 Pre-processing

In this first step, the input data is prepared so that OCR can be performed. This includes image deskewing and segmentation into rows and individual staves.

4.1.1 Deskewing

The quality of scans of old documents can vary significantly. Many documents are marked by age and signs of use, which limits readability. Often, the paper is yellowed and in some places the print has faded. Moreover, due to the woodcut printing technique of the 16th century, the print is often irregular and many pages are printed skewed or appear distorted due to the age of the paper.

This is a challenge not only for recognition, but also for segmentation. If the tablature rows run at an angle due to

distortions, it is possible that characters are cut off when the input images are divided into partial images for the individual tablature staves. At the same time, however, no unnecessary margins should be added.

To meet these challenges, the first pre-processing step is to correct distortion and rotation of the image. The algorithm used for deskewing first performs a horizontal line detection on the input image by applying the morphological opening operation with a gain in *x*-direction to the binarized input. The resulting connected areas represent the lines. On these lines, points are sampled at regular *x*-intervals. For each of these points, the amount of offset required to align the sample points horizontally is calculated. Based on these offsets, a transformation matrix is created to align the lines horizontally.

4.1.2 Segmentation

The pages of Ammerbach's tablature books contain multiple rows of tablature separated by horizontal dividing lines. Each row consists of a number of staves, one for each voice of the piece of music. Since tablature recognition can only be performed for individual staves, the input must be segmented accordingly.

While the number of rows on each page can be determined by detecting the horizontal dividing lines, the number of voices within a row is difficult for an algorithm to determine, since there are rarely clear boundaries between the staves. Since an incorrect assumption of the number of voices would lead to major errors in character recognition, the number of voices in each row is not automatically determined by the program, but must be specified by the user.

The dewarped input image is first split into partial images for rows. For this purpose, line detection is performed to find all horizontal dividing lines. The detected middle *y* positions of each line serve as separation edges. The row images are then divided into images of individual staves. Since no further dividing lines exist between the different voices and since it is not always possible to draw clear cutting edges due to overlaps, the voices' positions are estimated by dividing the image pitch by the given number of voices. The pitch and width of the partial images to be generated are set according to the size of the input layer of the neural network. The row image is then cropped so that the estimated middle position of the staff is located at the center of the resulting partial image. The segmentation process is illustrated by an example in **Figure 5**.

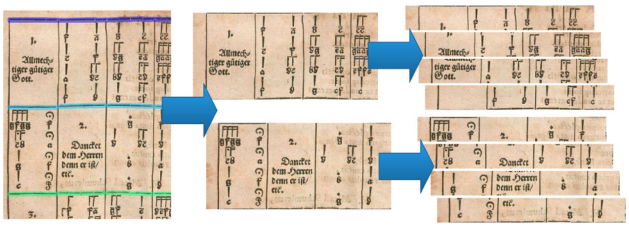

Figure 5: Segmentation of the input image. The input image is split into separate images for each row on the displayed horizontal dividing lines. Afterwards, the results are split into images for each staff using the estimated voice positions.

4.2 Recognition

We use an RNN architecture that contains CNN components for recognizing tablature staves. The network is trained on a large and diverse data set of annotated images of tablature staves and learns to recognize tablature character sequences. After training, the network can recognize tablature characters displayed on previously unseen images. The proposed Character Sequence Pair (CSP) network is implemented in Keras.[3]

4.2.1 Character Sequence Pair Network

The biggest difference between the recognition of arbitrary text lines and organ tablature staves in images is that tablature staves each consist of two lines on which the tablature characters are arranged. The upper line contains note duration symbols and special characters, while the lower line contains note pitch symbols and pause signs (see **Figure 4**). To consider this two-line arrangement, we present the Character Sequence Pair (CSP) recognition network. It has two output heads (called *duration/special head* and *pitch/rest head*) that are trained to predict the upper and lower line characters, respectively. This follows the basic idea of multitask learning as originally introduced by Caruana (1997), where the tasks in this case are to recognize the two different character sets.

Splitting the output is necessary because the recognition network is not built to recognize individual characters at specific positions, since this would require tedious labeling with bounding boxes. Instead, for each *x* coordinate of the input image, a probability distribution over all possible labels at this position is produced. From these probability distributions, the label sequence maximizing the probabilities for each coordinate is determined.

With only one output head, all combinations of duration/special and pitch/rest line characters that could occur at the same *x* position would have to be encoded by different labels. This would increase the number of labels immensely, which in turn would lead to a significantly higher training duration and a larger number of required training samples. By using two output heads, the characters of the two lines can be recognized independently of each other, which reduces the training effort significantly.

The CSP network consists of a CNN part, followed by an RNN part. **Figure 6** shows the architecture of the network. The input layer has a size of 2000 × 128 neurons corresponding to the pixels of the input image. This size provides space for one tablature staff and leaves sufficient room for rows that are not aligned perfectly horizontally.

We utilize two blocks of convolutional layers that first detect simple and then more complex shapes in the input image. In the first convolutional block, we use two sets of 64 filters, and in the second block we use two sets of 128 filters (due to the increased complexity level). After each block, we use max-pooling that cuts the size of the input down to half of its size. This results in a reduction of the weights to be trained and thus in a faster convergence of the training, but additionally increases the invariance of the network with respect to small displacements (Goodfellow et al., 2016).

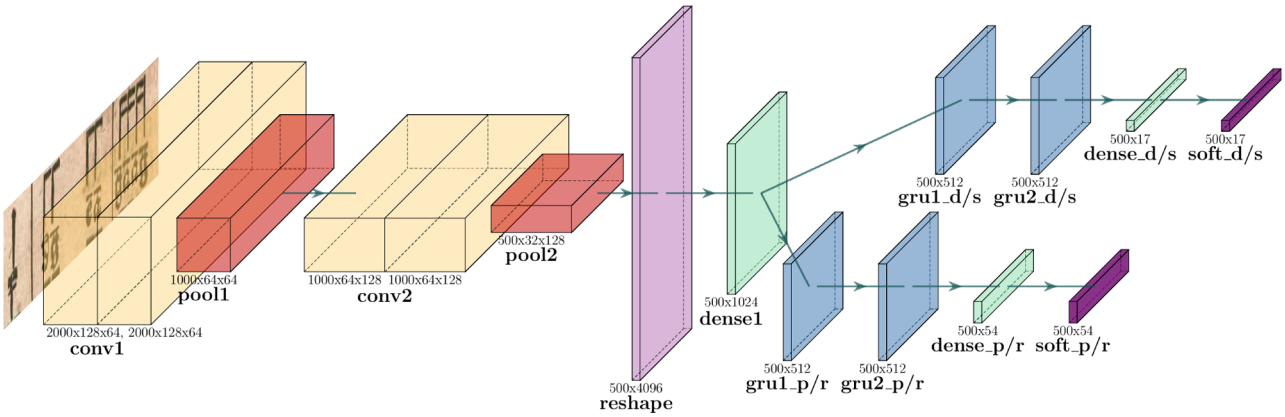

Figure 6: Architecture of the CSP tablature recognition network. The different layer types are color-coded and the size is indicated below each layer. This image was generated with PlotNeuralNet (Iqbal, 2018).

The processing of the input by the CNN part yields 128 activation maps with a size of 500 × 32 neurons each, thus a three-dimensional output. To further process this information in the RNN part, it has to be condensed into a two-dimensional form. For this purpose, we use a reshape layer that arranges the activations of the individual filters vertically one below the other, thereby creating an activation map of 500 × 4096 neurons. To reduce the number of parameters of the subsequent recurrent layers and thus speed up training, we insert a fully-connected layer with a smaller number of neurons (500 × 1024) after the reshape layer.

After the fully-connected layer, the path through the network, which has been shared so far, splits into two paths for the recognition of the duration/special characters and the pitch/rest characters of a staff. The recurrent networks we use are bidirectional Gated Recurrent Units (GRUs) with a size of 500 × 512 neurons. GRUs (Cho et al., 2014) are designed for processing sequential data (such as sequences of organ tablature characters).

On each of the two paths of the CSP network, we use two bidirectional GRUs, followed by a fully connected layer with softmax activation function. The softmax layers serve as output heads of the network for the characters of the upper (duration/special head) and lower (pitch/rest head) lines. These layers differ in size due to the different character sets to be recognized. There are 15 different characters to be recognized by the duration/special head, and there are 52 characters for the pitch/rest head (each plus two reserved labels: `blank` and `no output`).

4.2.2 Training and Recognition

The neural network is trained with Stochastic Gradient Descent (SGD). The cost function used is the Connectionist Temporal Classification (CTC) loss (Graves et al., 2006; Hannun, 2017). This makes it possible to train the network using training data without bounding boxes. The loss values are calculated independently for the duration/special head and the pitch/rest head. The total loss optimized during the training is calculated as the sum of the duration/special and the pitch/rest loss. Details of the training and validation process are explained in Section 6.

When performing a prediction on a tablature staff, the neural network outputs probability distributions for the duration/special and the pitch/rest line for each x coordinate of the input image, representing the occurrence of all possible labels at that position. From these, we use an adapted version of beam search (Hannun, 2017) to determine the label sequences with the highest overall probabilities independently of each other for the two lines of the staff.

4.3 Post-processing

In this step, the results of the recognition network for each staff are combined into an overall result, from which a Lilypond file is generated. This file is subsequently used to generate a graphical output in modern music notation.

4.3.1 Result Combination

The label sequences for duration/special head and pitch/rest head determined independently for each given staff are now merged into a single sequence. This is achieved by repeatedly combining one note duration symbol with one note pitch symbol, since these together represent a single note in modern notation. Pause signs and special characters can be transferred directly to the result sequence during this process. If a complete matching is not possible due to analysis errors, the remaining characters are added individually to the result, but formatted with an x as note head (if no pitch sign was found) or without note stem (if no duration sign was found) to indicate that a matching error occurred for this character.

When analyzing a sequence of tablature staves, the result sequences for each staff are combined into a single overall result. Thereby, the character sequences of all staves that are assigned to the same voice are concatenated to a single long sequence. Thus, the analysis of a four-voice organ tablature piece, for example, will result in four character sequences, regardless of the number of pages analyzed.

4.3.2 Output Generation

The results are saved in a Lilypond file to be able to generate a graphical output in modern music notation. The Lilypond file format is a LaTex-like structured format

in which each voice of a polyphonic composition is recorded as a separate string. In these, letters of the note names indicate note pitch (with commas and quotation marks for the octave position) and numbers indicate note duration. Owing to its simple but clear structure, this file format is well suited for musicological analyses, especially statistical studies.

The labeling of the training data for the network has been largely adapted to Lilypond's notation scheme, making it very easy to transfer network outputs to Lilypond. For example, the note duration symbols are labeled with the corresponding numbers (e.g., `4` for a quarter note) and the note pitch labels each consist of a note name letter followed by quotation marks or commas to indicate the octave position (e.g., `d''` for the note d in the two-line octave, the second octave above middle c).

We only use abbreviations for special characters and a few other exceptions that require a more complex command in Lilypond. These abbreviations are replaced by their corresponding Lilypond commands during the combination step. The result for each voice is therefore a string with the recognized character sequence in Lilypond format.

To generate a Lilypond file from the result sequences, a template is used containing all necessary commands for the desired layout. The strings for the individual voices are inserted into this template at the appropriate places. Further down the file, the voices are then assigned to corresponding staves, for example, to create a four-part composition. Finally, the Lilypond file is used to generate graphical sheet music output in the desired format (pdf, png) or digital music output (MIDI).

5. Tablature Data Set

Since no data sets for training a neural network for organ tablature recognition existed until now, we created such a data set in the context of our work.

Two organ tablature books by Elias Nikolaus Ammerbach are the basis of our data set. First, the "Orgel oder Instrument Tabulaturbuch" ("Organ or Instrument Tablature Book") (Ammerbach, 1583) is used; it consists of 213 pages in tablature notation. Second, "Ein new künstlich Tabulaturbuch" ("A new artificial tablature book") (Ammerbach, 1575) is used; it has 170 printed tablature pages. Both works contain printed tablatures and share the same font style, but differ slightly in layout.

To create our data set, we manually annotated 1,200 staves from each book with label sequences. However, this number of tablature staves is not sufficient to train a neural network to deliver highly accurate results, as we will show in Section 6.4.3. Thus, we increased the amount of available data by employing data augmentation and artificially generating tablature rows using a data generator that we developed for this purpose.

The generator produces images of organ tablature rows similar to those in Ammerbach's tablature books by randomly arranging images of single tablature characters. When placing the characters, the generator ensures that duration and pitch characters are combined appropriately, but no semantic relationships are considered, neither in the same voice nor between voices. Instead, the selection of a character from a category is purely random, which is why no logical melody progressions are created, no rhythmic structure is observed, and no harmonic rules between the voices are followed. Thus, the neural network is trained to recognize sequences of independent characters.

We use the generated images to focus on cases where characters are poorly printed or appear distorted to make the network more robust to these challenges. Furthermore, in the original images, some characters appear very frequently, while others are severely underrepresented. This imbalance is mitigated by the generator as well.

Figure 7 shows a comparison between a real tablature row, an augmented tablature, and two generated tablatures. While font style and basic layout are identical, the original image is more clearly structured and has higher contrast than the generated ones. Augmentation introduces variation by making rather small changes to the original image. The artificially generated images provide an even higher variability and thus allow a better generalization of the model to more diverse data. The data set with annotated staves and the open-source code of our data generation and augmentation tools are available for download.[2]

5.1 Data Generator

The generator receives a set of images of all tablature characters from Ammerbach's tablature books as its input. Several variations of each character are needed to ensure that the generated results are as diverse as possible. We use 20 to 30 sample images of each character. The individual characters had to be selected manually from the original documents and then had to be cropped using an image processing program. This tedious process was necessary to ensure that the generated images could be blended well and the results appear as similar as possible to the originals. In addition to the tablature characters, some examples of bar lines, page borders, and text segments were extracted to represent these elements in the generated data.

To help the network to separate background and content during training, a number of areas without characters were taken from the tablature books as backgrounds for the generated images. The probabilities for the random selection of each type of tablature character and minimum distances to neighboring characters can be specified by additional metadata.

The generation process starts with an empty image of the desired size. First, a random background image is selected and randomly positioned in the image section. Then, the image is filled with tablature characters in bars. Each bar is framed by randomly selected bar lines or special characters on both sides. Within a bar, random sequences of tablature characters are generated one after the other for each voice. Note duration and note pitch characters are always generated together to ensure that these characters only occur in combination. The symbols are placed on the duration/special line or the pitch/rest line of the corresponding voice according to their type.

After all staves within a bar have been filled in this way, the bar is completed and the next bar is started. This procedure is repeated until the entire width of the image has been filled in.

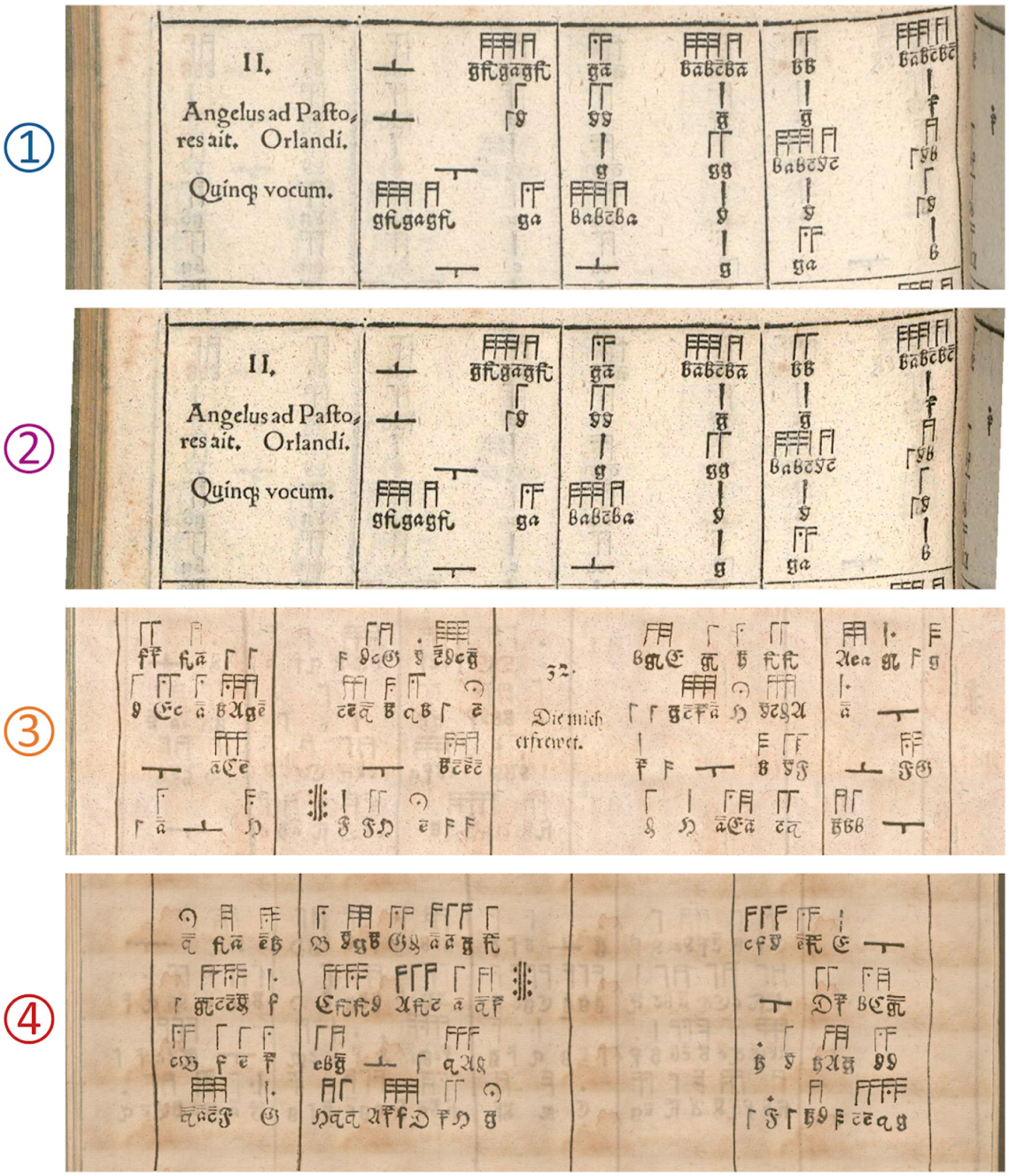

Figure 7: Comparison of (1) a real tablature row from Ammerbach's tablature books; (2) an augmented version of the same tablature row; (3,4) two tablature rows artificially created by the data generator.

Finally, the generated image, which contains several staves, is split into partial images for the individual staves. For each of these images, an additional text file containing the ground truth label sequences (one with duration and special characters and the other with pitch and rest characters) is created.

5.2 Data Augmentation

To increase the variety of the images in the data set, we utilize data augmentation to create several slightly modified variations of each image (both real and generated images).

During the artificial data generation, data augmentation is also performed at several points, e.g., to slightly modify the individual images of the randomly selected characters.

The augmentation operations include random changes of image parameters such as color, brightness, and contrast. Furthermore, the images are randomly scaled, rotated, and distorted. By setting some pixels to random color values, some noise is added as well. In addition, the entire image content can be shifted slightly in a random direction.

The probability and magnitude of each augmentation operation was determined experimentally and can be adjusted by the program parameters.

6. Experiments

We have conducted a series of experiments to evaluate the quality of the transcriptions created by the presented approach. The correctness of the results primarily depends on the accuracy of the outputs of the recognition network, which is why the experiments focus on the evaluation of metrics for the network outputs.

In our experiments, we compare the performance of different variations of the CSP network and different training setups and analyze the errors made on the test data set.

6.1 Data Sets

The data sets we created consist of labeled organ tablature staves, grouped into the three subsets: training, validation, and test data. We combine real tablature staves extracted from the scans of two tablature books with artificially composed tablatures generated with the synthetic data generator described above. To increase the number of images in the data sets, we use data augmentation to create several slightly modified variations of each image. We investigate the influence of augmentation and the use of randomly generated data on network training by comparing training runs with different partial data sets in Section 6.4.3. We started our experiments using a large training set with a high number of generated images (*trainSetL*). During our experiments, we found that the results are slightly better using a smaller number of generated training images (*trainSetS*). Therefore, we use this data set for the final evaluation. **Table 1** shows the composition of the data subsets.

6.2 Metrics

For a quantitative evaluation of the trained neural networks we calculate the following metrics on the test set of 1,000 tablature staves:

- *Top-k accuracy*:

$$acc_{\text{Top-}k} = \frac{\#\text{ images with correct Top-}k\text{ recognition}}{\#\text{ analyzed images}}$$

- *Bar accuracy*:

$$acc_{\text{bar}} = \frac{\#\text{ correctly recognized bars}}{\#\text{ analyzed bars}}$$

- *Levenshtein distance*:

$$edist_{\text{staff}} = \frac{\#\text{ editing steps per image}}{\#\text{ analyzed images}}$$

- *Normalized Levenshtein distance*:

$$edist_{\text{char}} = \frac{\#\text{ editing steps per image}}{\#\text{ characters on analyzed images}}$$

A challenge in the automated quantitative evaluation of organ tablature transcriptions is that comparatively long character sequences are considered. Here, single errors can heavily bias metrics such as Top-1 accuracy. To address this problem, we additionally consider the Top-10 and Top-5 accuracy, but especially the accuracy per bar, since single errors have a smaller impact here.

An even better evaluation criterion is the Levenshtein edit distance. It expresses the degree of difference between two strings as the average number of editing steps required to transform one string into the other string. This indicates the degree to which two sequences match. To be able to compare sequences of different lengths on different images, we additionally normalize the edit distance to an average number of editing operations per character.

6.3 Network Architectures

As described in Section 4.2.1, the presented CSP recognition network is divided into two paths that are trained to recognize characters using the duration/special head and the pitch/rest head, respectively. During training, both paths are trained together, and outputs for an input image are also produced simultaneously by both heads.

An alternative to this approach is to use two independent networks, each containing only one of these paths. Each of these networks has significantly less parameters, which simplifies training. The two networks can be trained independently of each other and only learn to recognize the characters of one line of characters. To compare the performance of these two approaches, we trained both a CSP network and two partial networks, one for each path, with identical parameters on the same training data set. The results of this evaluation can be found in Section 6.4.1.

To verify that the chosen network configuration is appropriate for this problem, we compare the results of various simplifications of the CSP network in Section 6.4.2.

We trained each network over a period of 40 epochs. The learning rate was initially set to 0.015 and reduced to 0.00001 over the training period using a cosine function (i.e., cosine learning rate decay).

6.4 Results

In our experiments, the networks were trained on an Nvidia GeForce GTX TITAN X installed in an Intel Core i7-5930K having 12 cores at 3.50 GHz and 64 GB of

Table 1: The data set, consisting of training, evaluation, and test sets. The values indicate the number of images taken from each source. The numbers in parentheses indicate the factor by which this number has been enlarged by data augmentation.

Subset	Staves taken from "Orgel oder Instrument Tabulaturbuch"	Staves taken from "Ein new künstlich Tabulaturbuch"	Generated Staves	Sum
trainSetL	500 (*100)	500 (*100)	140,000 (*5)	**800,000**
trainSetS	500 (*100)	500 (*100)	20,000 (*5)	**200,000**
valSet	200 (*25)	200 (*25)	8,000 (*5)	**50,000**
testSet	500	500	0	**1,000**

RAM. After completing the training of the networks, we evaluated the metrics described above on the test data.

6.4.1 Experiment 1: Partial Networks

In this experiment, we compare the CSP network, in which the paths for the duration/special head and the pitch/rest head are trained together, with two partial networks, each consisting of only one of these paths. All networks were trained on the *trainSetL* data set. **Table 2** shows the results.

It is evident that the difference in the quality of the results between the CSP network and the two partial networks is quite small. For pitch/rest, the partial network delivers slightly better results (difference in edit distance relative to CSP: –0.070), but for duration/special, the CSP network is slightly better (edit distance difference +0.016).

Since the CNN part of the CSP network is traversed for both output paths, this network has a smaller number of parameters to be trained than two independent partial networks, which results in lower computational requirements. The number of floating point operations (FLOPs) required for the combined network is 27.93 M as opposed to a sum of 36.83 M of both partial networks. This is reflected in a shorter computing time for CSP network training. On the Nvidia/Intel platform mentioned above, the 40 training epochs for the CSP network lasted 47 hours and 34 minutes, whereas the training of the two partial networks, one after the other, lasted 60 hours and 45 minutes in total.

The CSP network thus offers approximately the same quality of the results with reduced computational requirements and thus shorter training times.

6.4.2 Experiment 2: Simplified Networks

In this experiment, we compare the CSP network with several simplified network architectures to investigate whether a simpler architecture exists that provides equivalent or better results. The networks were again trained on the *trainSetL* data set. The results are presented in **Table 3**.

All three simplified architectures perform worse than the full CSP network for each of the calculated metrics. The use of unidirectional GRUs provides the least degradation (edit distance difference +0.009 for duration/special or +0.092 for pitch/rest), while the reduction to one CNN block leads to a much larger deterioration (+0.025 resp. +0.181). The use of only one GRU (+0.028 resp. +0.196) is even worse. We conclude that all components of the CSP network architecture are necessary for this problem, and the use of a smaller architecture is likely to decrease the performance.

6.4.3 Experiment 3: Training Data

In this experiment, we compare the training of the CSP network on different data sets to examine the influence of data augmentation and synthetic data generation on the quality of the results. The results are shown in **Table 4**.

The use of data augmentation *(Aug)* greatly improves the metrics compared to training with unaugmented

Table 2: Comparison of the CSP network with two partial networks, one for each path. The table shows the metrics evaluated on the test data set, the number of floating point operations (FLOPs), and the training time required for 40 epochs.

Network	Characters	Accuracy				Edit Distance		Computation Time	
		Top-10	Top-5	Top-1	Bar	Staff	Char	FLOPs	Training
CSP	*Duration/Special*	0.994	0.992	0.970	0.989	0.070	0.00162	27,925,688	47h34m
	Pitch/Rest	0.944	0.943	0.876	0.971	0.320	0.00515		
Partial 1	*Duration/Special*	0.991	0.990	0.963	0.988	0.086	0.00192	18,377,884	28h12m
Partial 2	*Pitch/Rest*	0.963	0.960	0.896	0.972	0.250	0.00423	18,453,660	32h33m

Table 3: Comparison of the CSP network with three simpler variations of the network. The table shows the differences in the network configurations, the metrics evaluated on the test set, as well as the number of FLOPs for each network.

Network	Configuration		Characters	Accuracy		Edit Distance		FLOPs
	CNNs	GRUs		Top-1	Bar	Staff	Char	
Full CSP	2 blocks	2 bidir.	*Duration/Special*	0.970	0.989	0.070	0.00162	27,925,688
		2 bidir.	*Pitch/Rest*	0.876	0.971	0.320	0.00515	
1 CNN	1 block	2 bidir.	*Duration/Special*	0.954	0.989	0.095	0.00218	27,483,320
		2 bidir.	*Pitch/Rest*	0.840	0.958	0.501	0.00835	
unidir. GRUs	2 blocks	2 unidir.	*Duration/Special*	0.962	0.988	0.079	0.00190	18,415,780
		2 unidir.	*Pitch/Rest*	0.835	0.960	0.412	0.00690	
1 GRU	2 blocks	1 bidir.	*Duration/Special*	0.950	0.984	0.098	0.00207	21,634,212
		1 bidir.	*Pitch/Rest*	0.818	0.949	0.516	0.00859	

data *(noAug)* (edit distance difference –0.285 for duration/special or –0.876 for pitch/rest). If we extend the 100,000 augmented real images by the same number of synthetically arranged tablatures *(Aug1Gen1)*, we achieve the best results in our tests (difference further –0.008 resp. –0.108). For tests with larger data sets, we did not achieve any further improvements, neither by adding more generated data *(Aug1Gen2)* (difference +0.013 resp. +0.021) nor by increasing the number of augmentations *(Aug2Gen2)* (difference +0.004 resp. +0.006). We conclude that with larger amounts of generated data, the effect of overfitting the images of individual characters that serve as the basis for the generator is observable. The same is true for frequent augmentation of a small number of real tablature staves. Thus, further improvements of the results require either additional annotated real tablatures or more samples of each character as inputs for the data generator.

Furthermore, we compared two test runs in which we only used a small number of real training images, (a) without generated data *(PtAug)*, (b) with augmented and generated data *(PtAugGen)*. The addition of generated data leads to a significant improvement of the results (edit distance difference –0.516 resp. –1.141). We conclude that the use of our data generator is very helpful for smaller data sets where many characters are underrepresented. With an increasing amount of real data available, the positive influence of generated data on accuracy decreases. A greater improvement in the results is then achieved by using data augmentation on the real data.

Finally, we investigated the benefits of the different augmentation operations. We trained our network with real training images augmented in different ways, with only one type of augmentation performed at a time, and compared the results with a training run on the same images without augmentation. We found that changes in image parameters and the addition of noise leads to small improvements of the results. However, shifting the image and applying scaling and distortion significantly improves the results.

6.4.4 Evaluation of the CSP Network

After having compared various network configurations and training with different data sets, we now present the results of the final configuration in detail. For this purpose, the CSP network was trained on the *trainSetS* data set. The results are summarized in **Table 5**.

Table 4: Comparison of training the CSP network with different data sets. The table shows the differences in the number of real and generated tablature images (as well as the factor by which this amount was increased by augmentation) for each training set and the metrics evaluated on the test data.

Train Data	Number of Tablature Staves			Characters	Accuracy		Edit Distance	
	Real	Generated	Sum		Top-1	Bar	Staff	Char
noAug	1000	0	1000	*Duration/Special*	0.844	0.952	0.348	0.00746
				Pitch/Rest	0.641	0.895	1.270	0.02258
Aug	1000 (*100)	0	100,000	*Duration/Special*	0.966	0.995	0.063	0.00134
				Pitch/Rest	0.851	0.967	0.394	0.00645
Aug1Gen1	1000 (*100)	20,000 (*5)	200,000	*Duration/Special*	0.970	0.993	0.055	0.00120
				Pitch/Rest	0.875	0.972	0.286	0.00455
Aug1Gen2	1000 (*100)	40,000 (*5)	300,000	*Duration/Special*	0.967	0.990	0.068	0.00143
				Pitch/Rest	0.873	0.970	0.307	0.00510
Aug2Gen2	1000 (*200)	40,000 (*5)	400,000	*Duration/Special*	0.971	0.990	0.059	0.00128
				Pitch/Rest	0.870	0.972	0.292	0.00475
PtAug	100 (*100)	0	10,000	*Duration/Special*	0.725	0.919	0.782	0.01608
				Pitch/Rest	0.527	0.849	2.005	0.03460
PtAugGen	100 (*100)	50,000 (*5)	260,000	*Duration/Special*	0.871	0.952	0.266	0.00583
				Pitch/Rest	0.709	0.906	0.864	0.01539

Table 5: Evaluation of the CSP network trained on the *trainSetS* data set. The table shows the metrics calculated on the test data set.

Characters	Accuracy				Edit Distance	
	Top-10	Top-5	Top-1	Bar	Staff	Char
Duration/Special	0.996	0.996	0.970	0.993	0.055	0.00120
Pitch/Rest	0.951	0.947	0.875	0.972	0.286	0.00455

The network achieves high accuracy values on the test data. The correct result is found in the Top-5 outputs of the network in 94.7% of the analyzed tablature staves for pitch/rest and in 99.6% for duration/special. When only the output with the highest probability is considered, the Top-1 accuracy for pitch/rest drops to 87.5%, while that of duration/special decreases to 97.0%. This imbalance can be explained by the larger alphabet of the pitch/rest characters and in particular by a larger number of highly similar symbols than in the duration/special symbols.

Looking at the percentage of correctly analyzed bars, the network reaches 97.2% for pitch/rest and 99.3% for duration/special. Here, the accuracy values are much closer, which shows that the errors mainly affect single characters and not larger areas. In most cases, only one bar of a staff is affected by an incorrect recognition, while the remaining bars are correctly recognized. This causes the staff accuracy to drop more strongly than the bar accuracy.

This observation is also evident when evaluating the Levenshtein edit distance metric. The average number of editing steps per staff is 0.286 for pitch/rest. Normalizing these results to the number of edits per character yields a value of 0.00455. This means that, on the average, an error occurs every 220th character. For duration/special, the edit distance is 0.055 per staff and 0.00120 per character. This means that an error occurs, on the average, at every 833rd character.

6.5 Error Analysis

In this section, we take a closer look at the detected errors and examine their causes.

The 1000 images of the test data set contain 105,117 characters – 51,367 for duration/special and 53,750 for pitch/rest. Our analysis revealed 258 wrongly recognized characters, 35 for duration/special and 223 for pitch/rest. Wrongly recognized spaces were not counted, because the characters of a label sequence are always separated by spaces. As a result, the network output always inserts or omits a space if a character is added or missed. The errors can be broken down into the four categories shown in **Table 6**.

In many cases, the errors can be traced back to the quality of the tablature sources. The print is uneven or incomplete in some places, or the color has faded due to the age of the document. This makes some characters unclear and difficult to distinguish. The neural network already finds the correct result in many of such unclear places, but in other places it fails. **Figure 8** shows some examples of such areas where the recognition of the test data is a challenge.

Table 6: The errors that occurred during the analysis, categorized into groups with the corresponding number of cases.

Duration/Special Errors		Pitch/Rest Errors	
Category	**Count**	**Category**	**Count**
Missed Symbol	21	***Missed Symbol***	47
Added Symbol	4	***Added Symbol***	17
Wrong Symbol	10	***Wrong Symbol***	74
		Wrong Octave	85

The error category *'Missed Symbol'* lists 68 cases in which characters could not be recognized by the CSP network. The most frequently affected characters are bar lines (29 times), the note pitch sign `e'` (5 times) and semiquaver notes (3 times). This type of error occurs especially in places where many characters appear in a small space (see **Figure 8** images 1 (incorrect) and 7 (correct)) or characters are poorly readable (see **Figure 8** images 2 (incorrect) and 8 (correct)). For example, the missed bar lines are nearly always caused by the fact that these bar lines are barely visible on the scans (see **Figure 8** image 3 (incorrect)).

The *'Added Symbol'* category contains 21 cases in which background elements or text blocks are incorrectly detected as tablature characters.

The most common analysis error is the confusion of similarly appearing characters. Here, a distinction is again made between two types of false analyses.

In the category *'Wrong Symbol'*, 84 cases are marked in which a completely different character was predicted than the one given in the ground truth data. Here, the note pitch symbols for `c` and `e` (7 times), `g` and `a` (6 times) as well as `b` and `bb` (also 6 times) were most often mistaken for each other. A confusion of these characters can be explained in many cases by an unclear print, because the appearance of the characters is quite similar (see **Figure 8** images 4, 5 (incorrect) and 8, 9 (correct)).

The errors where a correct note pitch sign was detected, but in the wrong octave position, are listed in the category *'Wrong Octave'*. The octave strokes above the note names are printed quite faintly in many places, so that the exact

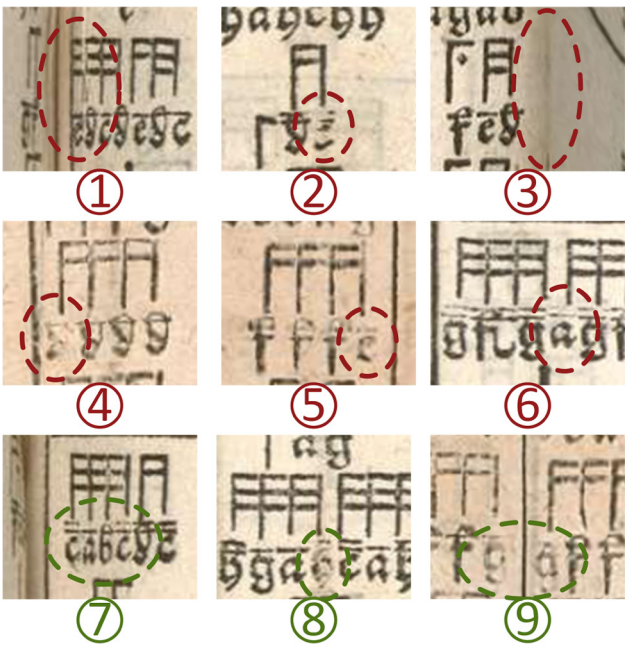

Figure 8: Examples of areas in the test data that are difficult to recognize. In images 1–6, an analysis error occurred in the marked areas, while in images 7–9, even in the circled areas, the poorly readable characters were correctly recognized.

identification of the octave position is often difficult (see **Figure 8** images 6 (incorrect) and 9 (correct)). Most often, the note pitch symbols for `d` (24 times), `c` (16 times) and `g` (12 times) were assigned to a wrong octave position.

6.6 Discussion

The experiments carried out show that the proposed CSP network is capable of recognizing scanned organ tablatures with low error rates. We have examined different network configurations and found that the presented CSP architecture is a suitable architecture and that the use of two output heads is reasonable. Training on different data sets showed that both data augmentation and synthetic data generation have a positive effect on network performance, especially with small amounts of available real data.

In our work, the network has only been used to recognize printed organ tablatures from two tablature books sharing the same font style, although differing in page layout. The experiments show that the CSP network generalizes well on these tablatures, and only minor recognition errors occur. Many of these errors can be blamed on the print quality and the age of the analyzed organ tablature books, which in some places make character recognition and differentiation difficult. To obtain a network that is even more robust to such challenges, a larger training data set is needed that covers these ambiguities to a greater extent.

7. Conclusion

We presented a deep learning approach to automatically transcribe scores from organ tablature music notation to modern music notation. The proposed approach processes scanned tablature pages by segmenting them into images of individual staves and performing character recognition on these sub-images using the presented Character Sequence Pair (CSP) neural network. The network recognizes one tablature staff at a time and provides two outputs for the two lines on which the tablature characters are located. In a post-processing step, these two outputs are combined into a single output and transferred to the desired music notation format. The transcription is performed without automatic error correction and interpretation and therefore delivers a result as close as possible to the original document.

Since no data sets for training neural networks to recognize organ tablatures existed up to now, we created a new data set based on two organ tablature books by Elias Nikolaus Ammerbach. We developed a synthetic data generator that randomly assembles artificial tablature rows from images of single characters. The network was trained using a combination of real staves from the tablature books and artificially composed organ tablature staves, using data augmentation to further increase the amount of available data.

The quality of the transcriptions generated by our approach was examined in several experiments. On a test data set obtained from the tablature books, the CSP network achieved an accuracy of 97.2% and 99.3% correctly recognized bars, depending on whether note pitch and rest characters or note duration and special characters are considered, respectively. On the average, an error occurs every 220th pitch/rest character and every 833rd duration/special character.

There are several areas for future work. First, we plan to train the CSP neural network to recognize further printed organ tablatures with different layouts and font styles, and possibly handwritten tablatures. Second, we intend to investigate whether extensions or alternative neural network architectures will be required to improve the results. Finally, we will investigate possibilities to further improve the data generator. Possible approaches are increasing the source image set or extending the generator with semantic rules. We also plan to develop a generator that uses digitized organ pieces as source material and transcribes them to tablature notation.

Notes

[1] https://lilypond.org.

[2] https://github.com/umr-ds/organ-tablature-ocr-dataset.

[3] https://keras.io/.

Acknowledgements

This work is financially supported by the Deutsche Forschungsgemeinschaft (DFG, FR 791/15-1).

Competing Interests

The authors have no competing interests to declare.

References

Alfaro-Contreras, M., Calvo-Zaragoza, J., and **Iñesta, J. M.** (2019). Approaching End-to-End Optical Music Recognition for Homophonic Scores. *9th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, 11868: 147–158. DOI: https://doi.org/10.1007/978-3-030-31321-0_13

Ammerbach, E. N. (1575). *Ein new künstlich Tabulaturbuch*. Nürnberg.

Ammerbach, E. N. (1583). *Orgel oder Instrument Tabulaturbuch*. Nürnberg.

Apel, W. (1967). *Geschichte der Orgel- und Klaviermusik bis 1700*. Bärenreiter, Kassel.

Apel, W. (2006). *Die Notation der polyphonen Musik 900–1600*. Breitkopf & Härtel, Wiesbaden, 5th edition.

Awel, M. A., and **Abidi, A. I.** (2019). Review on Optical Character Recognition. *International Research Journal of Engineering and Technology (IRJET)*, 06(6): 3666–3669.

Bainbridge, D., and **Bell, T.** (2001). The Challenge of Optical Music Recognition. *Computers and the Humanities*, 35(2): 95–121. DOI: https://doi.org/10.1023/A:1002485918032

Becker, C. W. (1963). *A Transcription of Elias Nikolaus Ammerbach's Orgel oder Instrument Tabulaturbuch*. PhD thesis, University of Rochester.

Bellermann, H. (1858). *Die Mensuralnoten und Taktzeichen des XV. und XVI. Jahrhunderts*. Georg Reimer, Berlin.

Calvo-Zaragoza, J., Hajič Jr., J., and **Pacha, A.** (2020). Understanding Optical Music Recognition. *ACM Computing Surveys*, 53(4). DOI: https://doi.org/10.1145/3397499

Calvo-Zaragoza, J., and **Rizo, D.** (2018). End-to-end Neural Optical Music Recognition of Monophonic Scores. *Applied Sciences*, 8(4). DOI: https://doi.org/10.3390/app8040606

Calvo-Zaragoza, J., Rizo, D., and **Quereda, J. M. I.** (2016). Two (Note) Heads are Better Than One: Pen-Based Multimodal Interaction with Music Scores. In *Proceedings of the 17th International Society for Music Information Retrieval Conference*, pages 509–514.

Calvo-Zaragoza, J., Toselli, A. H., and **Vidal, E.** (2019). Handwritten Music Recognition for Mensural Notation with Convolutional Recurrent Neural Networks. *Pattern Recognition Letters*, 128: 115–121. DOI: https://doi.org/10.1016/j.patrec.2019.08.021

Calvo-Zaragoza, J., Valero-Mas, J. J., and **Pertusa, A.** (2017). End-to-end Optical Music Recognition Using Neural Networks. *Proceedings of the 18th International Society for Music Information Retrieval Conference*, pages 472–477.

Caruana, R. (1997). Multitask Learning. *Machine Learning*, 28(1): 41–75. DOI: https://doi.org/10.1023/A:1007379606734

Cho, K., van Merriënboer, B., Bahdanau, D., and **Bengio, Y.** (2014). On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111. Association for Computational Linguistics. DOI: https://doi.org/10.3115/v1/W14-4012

Dutta, K., Krishnan, P., Mathew, M., and **Jawahar, C. V.** (2018). Towards Accurate Handwritten Word Recognition for Hindi and Bangla. *Communications in Computer and Information Science (CCIS)*, 841: 470–480. DOI: https://doi.org/10.1007/978-981-13-0020-2_41

Feng, Z., Yang, Z., Jin, L., Huang, S., and **Sun, J.** (2017). Robust Shared Feature Learning for Script and Handwritten/Machine-Printed Identification. *Pattern Recognition Letters*, 100: 6–13. DOI: https://doi.org/10.1016/j.patrec.2017.09.016

German UNESCO Commission. (2018). *Jahrbuch der Deutschen UNESCO-Kommission 2017–2018*.

Goodfellow, I., Bengio, Y., and **Courville, A.** (2016). *Deep Learning*. MIT Press.

Graves, A., Fernández, S., Gomez, F., and **Schmidhuber, J.** (2006). Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proceedings of the 23rd International Conference on Machine Learning*, page 369–376, New York, NY, USA. Association for Computing Machinery. DOI: https://doi.org/10.1145/1143844.1143891

Hannun, A. (2017). Sequence Modeling with CTC. DOI: https://doi.org/10.23915/distill.00008

Horyna, M. (2018). Medieval Organ Tablature on a Manuscript Fragment from the National Museum Library. *Musicalia*, 10(1–2): 6–42. DOI: https://doi.org/10.1515/muscz-2018-0001

Huang, Y.-H., Chen, X., Beck, S., Burn, D., and **Van Gool, L.** (2015). Automatic Handwritten Mensural Notation Interpreter: From Manuscript to MIDI Performance. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, pages 79–85.

Hulková, M. (2015). Central European Connections of Six Manuscript Organ Tablature Books of the Reformation Era from the Region of Zips (Szepes, Spiš). *Studia Musicologica*, 56(1): 3–37. DOI: https://doi.org/10.1556/6.2015.56.1.1

Iqbal, H. (2018). PlotNeuralNet. DOI: https://doi.org/10.5281/zenodo.2526396

Maul, M., and **Wollny, P.,** editors (2007). *Weimarer Orgeltabulatur: die frühesten Notenhandschriften Johann Sebastian Bachs sowie Abschriften seines Schülers Johann Martin Schubart*. Bärenreiter, Kassel, New York.

Motnik, M. (2011). Deutsche Tabulatur: Gebreuchlich oder verdrießlich? *Musicological Annual*, 47(2): 125–137. DOI: https://doi.org/10.4312/mz.47.2.125-137

Müller-Schmidt, H.-T. (2017). Orgel oder Instrumenttabulaturbuch 1583 von Elias Nikolaus Ammerbach. https://imslp.org/wiki/Special:ReverseLookup/505757.

Pansare, S., and **Joshi, D.** (2012). A Survey on Optical Character Recognition Techniques. *International Journal of Science and Research (IJSR)*, 3(12): 1247–1249.

Patel, M., and **Thakkar, S. P.** (2015). Handwritten Character Recognition in English: A Survey. *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, 4(2): 345–350. DOI: https://doi.org/10.17148/IJARCCE.2015.4278

Purohit, A., and **Chauhan, S. S.** (2016). A Literature Survey on Handwritten Character Recognition. *International Journal of Computer Science and Information Technologies (IJCSIT)*, 7(1): 1–5.

Rebelo, A., Fujinaga, I., Paszkiewicz, F., Marcal, A. R., Guedes, C., and **Cardoso, J. S.** (2012). Optical Music Recognition: State-of-the-art and Open Issues. *International Journal of Multimedia Information Retrieval (IJMIR)*, 1(3): 173–190. DOI: https://doi.org/10.1007/s13735-012-0004-6

Su, B., and **Lu, S.** (2017). Accurate Recognition of Words in Scenes without Character Segmentation Using Recurrent Neural Network. *Pattern Recognition*, 63: 397–405. DOI: https://doi.org/10.1016/j.patcog.2016.10.016

Tuggener, L., Elezi, I., Schmidhuber, J., and **Stadelmann, T.** (2018). Deep Watershed Detector for Music Object Recognition. *Proceedings of the 19th International Society for Music Information Retrieval Conference*, pages 271–278.

Wojnowska, E. (2016). *Thematic Catalogue of 17th-Century Organ Tablatures from the Liegnitz Bibliotheca Rudolphina*. Warschau: Biblioteka Narodowa.

Wolf, J. (1919). *Handbuch der Notationskunde Teil 2*. Breitkopf und Härtel.

How to cite this article: Schneider, D., Korfhage, N., Mühling, M., Lüttig, P., & Freisleben, B. (2021). Automatic Transcription of Organ Tablature Music Notation with Deep Neural Networks. *Transactions of the International Society for Music Information Retrieval,* 4(1), pp. 14–28. DOI: https://doi.org/10.5334/tismir.77

Submitted: 23 September 2020 **Accepted:** 28 December 2020 **Published:** 24 February 2021

]u[*Transactions of the International Society for Music Information Retrieval* is a peer-reviewed open access journal published by Ubiquity Press.

OPEN ACCESS