OPTICAL MUSIC RECOGNITION WITH CONVOLUTIONAL SEQUENCE-TO-SEQUENCE MODELS

First Author
Affiliation1
author1@ismir.edu

Second Author
Retain these fake authors in submission to preserve the formatting

Third Author
Affiliation3
author3@ismir.edu

ABSTRACT

Optical Music Recognition (OMR) is an important technology within Music Information Retrieval. Deep learning models show promising results on OMR tasks, but symbol-level annotated data sets of sufficient size to train such models are not available and difficult to develop. We present a novel deep learning architecture called a Convolutional Sequence-to-Sequence model to both move towards an end-to-end trainable OMR pipeline, and improve the learning process by training on full sentences of sheet music instead of individually labeled symbols. The model is trained and evaluated on a human generated data set, with various image augmentations based on real-world scenarios. This data set is the first publicly available set in OMR research with sufficient size to train and evaluate deep learning models. With the introduced augmentations a pitch recognition accuracy of 81% and a duration accuracy of 94% is achieved, resulting in a note level accuracy of 80%.

1 Introduction

Optical Music Recognition (OMR) is an application of recognition algorithms to musical scores, to encode the musical content to some kind of digital format. In modern Music Information Retrieval (MIR), these applications are of great importance. The digitization of sheet music libraries is necessary first step in various data-driven methods of musical analysis, search engines, or other applications where digital formats are required.

OMR is an active area of research in MIR, and a classically hard problem. OMR systems need to deal with a large range of challenges such as low quality scans, ambiguous notation, long range dependencies, large variations in musical font, and handwritten notation. Multiple commercial applications are available, each with their own strengths and weaknesses [3], but the accuracy of these products is often too low to use without human supervision. An implementation that deals with these challenges in a satisfactory way has yet to be developed.

A traditional OMR system typically consists of multiple parts: score pre-processing, staff line identification and removal, musical object location, object classification and score reconstruction [13]. Each of these individual parts has its own difficulties, resulting in OMR systems with low confidence. More recently, there has been a trend towards less segmented systems involving machine learning methods, such as OMR without staffline removal [11] or without symbol segmentation [14]. However, a major difficulty of these algorithms is the need for large amounts of training data. Typically, scores need to be annotated on musical symbol level to train such machine learning pipelines, but large corpora of sufficiently diverse symbol-annotated scores are difficult and expensive to produce [13].

In this study, we propose a novel deep learning architecture to both move towards an end-to-end trainable OMR pipeline, and greatly reduce the data requirements for training. This is achieved by using two common deep learning architectures: *Convolutional Neural Networks* (CNN) and *Recurrent Neural Networks* (RNN). Convolutional architectures have been a popular choice of algorithm in various MIR related tasks, due to the ability to learn local structures in images, and combining them to useful features. In our method, we use a CNN to learn a feature representation of the input scores. Continuing, a *Sequence-to-Sequence* model [4, 16] is used, which is a stack of two RNN's used commonly in machine translation tasks. This model directly produces a digital representation of the score from the learned representation by the CNN. The combination of these two architectures is called a *Convolutional Sequence-to-Sequence* model. By using a Sequence-to-Sequence architecture, we cast the problem of OMR as a translation problem. Instead of training on individual segmented symbols without context, full lines of sheet music are translated simultaneously. This approach has two major advantages; Firstly, by training the algorithm on full lines of sheet music, there is no need for symbol level annotated training data. This means that in principle any corpus of sheet music with corresponding digital notation could be used for training, opening up many new possibilities for data-driven OMR systems. Secondly, because in the proposed model each of the classically segmented OMR steps is done by a single algorithm, the model can use a large amount of contextual information to solve ambiguity and long-range dependency problems.

To train the proposed model, a large corpus of monophonic sheet music is generated from a MusicXML dataset

as described in Section 3. Additionally, in Section 3.2 various types of image augmentations based on real-world scenarios are proposed to enhance the models flexibility to different kinds of fonts and varying score quality. Finally in Section 5, the results of the method are discussed on both clean and augmented data, and the weaknesses of the model are examined.

2 Related Work

A starting point for any OMR research is the overview paper by Rebelo et al. [13], which contains a complete introduction to OMR systems and a description of the current state of the field. The paper describes four main stages that are necessary for any OMR pipeline: Image pre-processing, musical symbol recognition, musical information reconstruction and construction of musical notation. The second component, as the name suggests, is where the main recognition work is done. Detecting and removing staff lines, segmenting individual symbols, and classifying symbols. Systems where steps are conducted by different methods we call segmented systems.

Not all methods follow this model, recent data-driven approaches suggest merging or omitting some of these segmented steps. An example of this is an approach suggested by Pugin et al. [10, 11], which applies *Hidden Markov Models* (HMM) to the recognition stage, without performing staff line removal. Shi et al. [14] incorporate a deep learning approach with an HMM and conduct a preliminary experiment on sheet music data without staffline removal with excellent results. Symbol classification involving neural networks has been researched by several authors [12, 18]. However, Shi et al. [14] are the first to incorporate a convolutional architecture.

In a different paper, Rebelo et al. [12] research the use of Deformable Templates [7] with various classifiers to make symbol recognition invariant to changes in musical font. This method is similar to the Elastic Transformations [15] used in this research.

3 Dataset

The dataset used in this research is compiled from monophonic MusicXML scores from the MuseScore sheet music archive [1]. The archive is made up of user-generated scores, and is very diverse in both content and purpose. As a result, the dataset contains a large variation in type of music, key signature, time signature, clef, and notation style. To generate the dataset, each score is checked for monophonicity, and dynamics, expressions, chord symbols, and textual elements are removed. This process produces a dataset of about 17 thousand MusicXML scores. For training and evaluation, these scores are split into three different subsets. 60% is used for training, 15% for validation and 25% for the evaluation of the models. A specification to reproduce the data set is publicly available online. [1]

[1] `https://github.com/*anonymized*`

3.1 Preprocessing

From the corpus of monophonic MusicXML files, a dataset of images of score fragments and corresponding note annotations is created. Each MusicXML score is split into fragments of up to four bars, with a two bar overlap between adjacent fragments. The fragments are converted to sheet music using MuseScore [1], each image containing a single staff line. The corresponding labels are represented with a pitch and duration vector, containing all information about the notes and rests within the same four bars. Pitch values are indicated by their scientific names, and duration values are notated in quarter length. Each musical symbol is represented with two classes: a pitch, and a duration. In case of a rest, the pitch is a special rest indicator, which we indicate with r. Complex durations, durations that cannot be expressed with a single notehead, are split to their elementary constituents. The first note of the complex duration is indicated with the pitch, pitches of subsequent complex notes are replaced with a tie indicator, which we indicate with t.
As an example, a quarter rest followed by a note with pitch C5 and a complex duration of a tied quarter note and a sixteenth note is notated as $((r, 1), (C5, 1), (t, 0.25))$. Applying this method to the full score fragments produces the pitch and duration vector, and is a suitable representation for the model. A maximum of 48 events per fragment is used to put a limit on the sequence length the model has to decode. Finally, at the end of each pitch and duration vector an extra event is added to indicate the sequence has ended. This indicator is implemented as a rest with duration of zero quarter notes.

Each generated image is padded to the same width and height, and images containing notes with more than five ledger lines are discarded. These notes are extreme outliers and do not occur in normal notation. The resulting fragments have a dimension of 2261×400 pixels.

3.2 Image Augmentation

The computer generated score fragments contain no noise or variation in musical symbols. To make the proposed model robust to lower quality inputs and different kinds of musical fonts we propose four different augmentations, each simulating a real world source of input noise. Additionally, for each augmentation, we choose two separate settings. For the augmented training data, the parameters are chosen such that the input sheet music is greatly deformed but still readable. For the augmented evaluation set, parameters are chosen such that they resemble real-world sheet music, with less deformation than the training data. The larger amount of deformation in training will force our model to learn to recognize musical symbol in any situation, and should improve the accuracy of our model on both non-augmented and augmented evaluation data.

A popular choice of augmentation is *Additive White Gaussian Noise* (AWGN). This augmentation introduces a normally distributed random deviation in pixel intensities, to mimic noise introduced by low quality scans or photos.

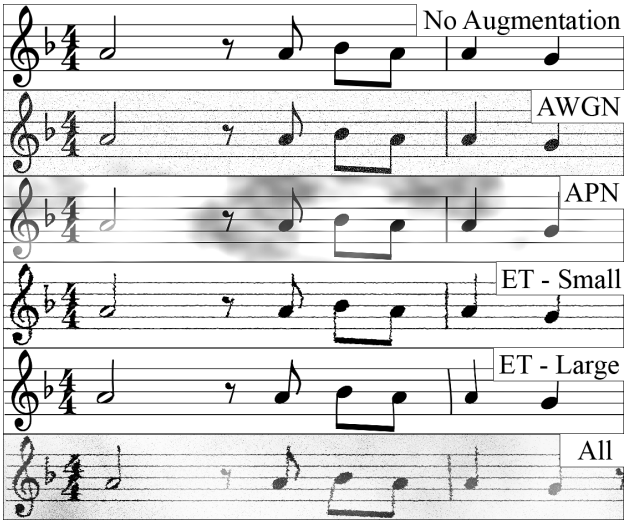

Figure 1. An example of each of the used image augmentations from the evaluation dataset. From top to bottom: No Augmentations, Additive White Gaussian Noise (AWGN), Additive Perlin Noise (APN), Small scale Elastic Transformations (ET small), Large scale Elastic Transformations (ET large), and all combined augmentations.

This noise has a mean μ, which is chosen to be the same as the mean pixel intensity of the full dataset. The standard deviation σ is different between our training and evaluation set. In the training set, the σ of pixel intensities in our non-augmented data set is used. The evaluation set has a σ of half that value.

The second type of noise augmentation used is *Additive Perlin Noise* [9]. Perlin noise is a procedurally generated gradient noise, that generates lighter and darker areas in the image at larger scales than AWGN. This effect mimics quality differences in parts of the score. Some symbols might be faded and parts of staff lines less visible, and dark areas in the image are created. The mean size of generated clouds is controlled by a frequency parameter. For each augmented score, this frequency is chosen to be a random value between the size of one note head and the mean width of a full bar, to generate noise structures at different scales. The maximum intensity of the noise in our training set is chosen to be a strength of 0.8. The evaluation set uses a maximum intensity of half this value.

The final two augmentations are achieved with *Elastic Transformations* (ET) [15], which apply a smoothed field of local random affine transformations, resulting in wave-like displacements in the augmented image. Two parameters are used to control an elastic transformation: a strength factor σ, which reduces the strength of the distortion if a larger value is used, and a smoothing factor α, which controls the scale of deformations. A very large α will apply a nearly linear translation to the image, while an α of zero applies fully random displacements on individual pixels.

The first type of Elastic Transformation is applied on very small scales, to change the characteristics of lines and smaller symbols. Lines might appear to be drawn by pencil or pen, and the edges of symbols become less defined. α is chosen to be a random value between 2 and 8, with a σ of 0.5 for the training data, and a σ of 2 for the evaluation data.

The second type of Elastic Transformation is applied on a large scale to change the shape and orientation of musical symbols. Barlines and note stems get skewed or bent, note heads can be compressed or elongated, and many new shapes are introduced in the score. This transformation mimics the use of different musical fonts, or even handwritten notation. An α between 2000 and 3000 is used, with a σ of 40 for the training data, and 80 for the evaluation data. To maintain straight and continuous stafflines, the original algorithm is slightly adapted to reduce vertical translations of pixels by reducing the vertical component of transformations by 95%.

In Figure 1, an example of each of these four augmentation is shown, with the setting used for generating the evaluation data. The last example shows a combination of all four augmentations.

4 Method

We introduce the Convolutional Sequence-to-Sequence model as applied to OMR tasks, translating lines of sheet-music to a sequence of (pitch, duration) pairs. Continuing, the training and evaluation methods are defined.

4.1 Model

We define a Convolutional Sequence-to-Sequence network as a stack of three components. First, a CNN encodes the input image windows to a sequence of vector representations. Then, an encoder RNN encodes the vector sequence to a fixed size representation, containing all information from the input score. Finally, a decoder RNN decodes the fixed size representation to a sequence of output labels. The following section describes each component in detail. Note that, while each component is described separately, the model will be trained as a single algorithm.

Sliding window input. The image input of the algorithm is defined as a sequence of image patches, generated by applying a *sliding window* over the original input score. The implementation has two separate parameters: the window width w and window stride s. By varying w, the amount of information per window can be increased or decreased. s defines how much redundancy exists between adjacent windows. Increasing the value of w or decreasing the value of s provides the model with more information about the score, but will raise the computational complexity of the algorithm. Thus when determining the optimal parameters, a balance has to be struck between complexity and input coverage. As a rule of thumb, we use a w that is approximately twice the width of a notehead, and an s of half the value of w. This will ensure that each musical object is shown in full at least once in an input window.

Convolutional neural network. To extract relevant features from the image patches, each patch is fed into a CNN. In this research, we keep the architecture of the CNN the same between different experiments, to ensure a

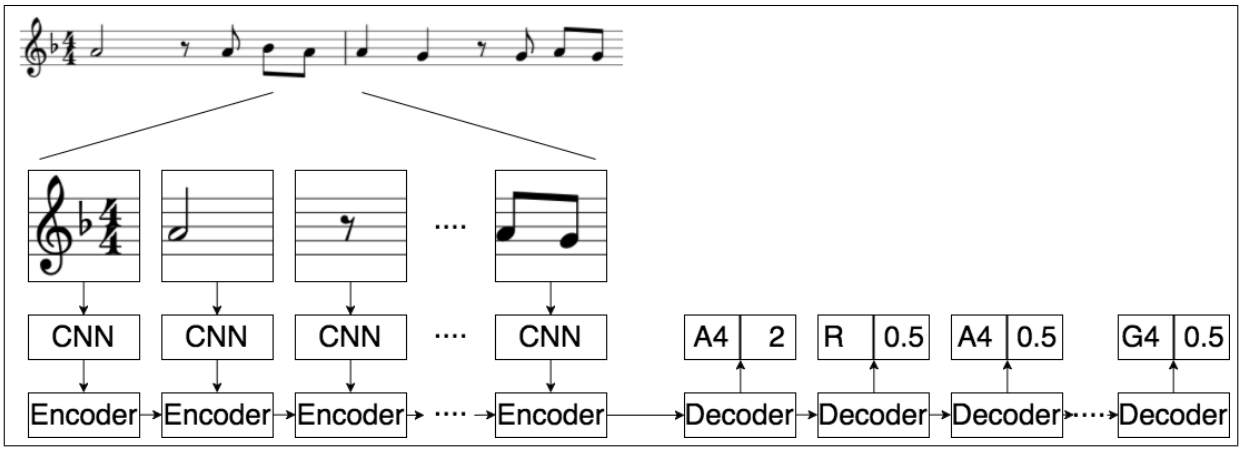

Figure 2. A diagram of the proposed Convolutional Sequence-to-Sequence model. On the left, a score fragment is processed by a CNN and Encoder RNN to a fixed size representation. This representation is used by the decoder RNN to create a sequence of (pitch, duration) pairs.

fair comparison. First a *max-pooling* operation of 3×3 is applied on the input window for dimensionality reduction. Then, a convolutional layer of 32 5×5 kernels is applied, followed by a *relu* activation and 2×2 max-pooling operation. These three layers are repeated, and a fully-connected layer of 256 units with relu activation is applied, so each input for the encoder will be a vector of size 256.

Sequence-to-Sequence network. After extracting a vector description of each image patch, the sequence of vectors is fed into a Sequence-to-Sequence network [4,16]. This architecture consists of two RNN's. The first RNN, the *encoder*, encodes the full input sequence to a *fixed size representation*. The second RNN, the *decoder*, produces a sequence of outputs from the encoded representation. In the case of the OMR task, this sequence of outputs is the sequence of pitches and durations generated from the MusicXML files. For both encoder and decoder, a single *Long Short-Term Memory* (LSTM) [6] layer is used with 256 units. To predict both the pitch and duration, the output of the decoder is split into two separate output layers with a *softmax* activation and *categorical cross-entropy* loss.

A diagram of the full model is shown in Figure 2, where four input patches and output predictions are shown. On the left side, A sliding window is applied to a 2 bar score fragment. Each image patch is sequentially fed into the same CNN. This CNN is connected to the encoder network, creating a fixed size representation of the two input bars. The decoder uses this representation to produce the output sequence of (pitch, duration) pairs. Note that the second predicted pitch is an r pitch, representing a rest symbol.

Using the described configuration, the model has an input sequence length of 70 windows and an output sequence length of 48 units. Shorter output sequences are padded to this maximum length and the loss function is masked after last element of the sequence. The number of pitch categories is 108, and the number of duration categories is 48. In total, the model contains approximately 1.67 million parameters.

4.2 Training

Six separate models are trained, one on each of the proposed augmented data sets: No augmentations, AWGN, APN, Small ET, large ET and all augmentations. All models are trained with a batch-size of 64 using the ADAM optimizer [8], with an initial learning rate of $8 * 10^{-4}$ and a constant learning rate decay tuned so the rate is halved every ten epochs. Each model is trained to convergence, taking about 25 epochs on the non-augmented dataset. A single Nvidia Titan X Maxwell is used for training, which trains a model in approximately 30 hours.

4.3 Evaluation Metrics

On the evaluation data, three different metrics are calculated, similar to [3]:

- Pitch accuracy, the proportion of correctly predicted pitches.
- Duration accuracy, the proportion of correctly predicted note durations.
- Note accuracy, the proportion of predicted events where both pitch and duration are correctly predicted.

The accuracy is measured over all notes before the stop indicator, and the stop indicator is not included in the calculation of accuracy. The model is not given any a priori knowledge about how many notes are in the input fragment, so a wrong number of notes could be predicted. In case of a shorter predicted sequence, the missing notes are automatically marked as incorrect. If the predicted sequence is longer than the ground truth, the additional predicted notes are cut and only the notes within the length of the ground truth are used. This method of measuring accuracy is quite strict, as an insertion or omission of a note in the middle of a sequence could mean subsequent notes are all marked as incorrect. This should be kept in mind when evaluating the results of the model, and perhaps more descriptive metrics could be applied in future work.

Augmentation	Pitch Accuracy	Duration Accuracy	Note Accuracy
None	0.79	0.92	0.76
AWGN	0.79	0.92	0.77
APN	**0.82**	0.91	0.79
ET - Small	0.78	0.91	0.76
ET - Large	0.79	**0.94**	0.78
All augmentations	0.81	**0.94**	**0.80**

Table 1. Measured accuracy on non-augmented scores. The accuracy scores for augmentations with the highest positive impact are in bold.

Augmentation	Pitch Accuracy	Duration Accuracy	Note Accuracy
AWGN	0.79	0.90	0.75
APN	0.81	0.89	0.76
ET - Small	0.78	0.89	0.74
ET - Large	0.78	0.94	0.75
All augmentations	0.79	0.92	0.77

Table 2. Measured accuracies on scores with augmentations. Each model trained on different augmented data is evaluated on an evaluation set with corresponding augmentations.

5 Results

5.1 Model Evaluation

The six trained models are evaluated on both a clean evaluation set, shown in Table 1, and augmented sets, shown in Table 2. The augmented evaluations are performed on data with the same augmentations the model is trained on, with the parameters described in Section 3.2.

5.2 Evaluation of Model Difficulties

To examine the difficulties the model has on different kinds of scores, three additional evaluations are performed on different subsets of the evaluation data.

First, an investigation into the impact of key signature on the note level accuracy on the non-augmented evaluation data is conducted. Just like human performance, the added complexity of many sharps or flats in the key signature of a fragment impacts the accuracy of the model. The results of this experiment are displayed in Figure 3 (top). At zero sharps or flats, the reported accuracy is 0.86, achieving 0.06 higher than the mean accuracy of 0.80. With more than 4 sharps or flats in the key signature the note accuracy starts diminishing, down to a minimum of 0.66 for key signatures with seven sharps or flats.

Continuing, the nine most common time signatures and their accuracies are examined. While the output notation does not encode any direct information about time signature, the model could use structural information imposed by the time signature on the score to aid in note recognition. This evaluation will both look at if that is the case, and investigate which time signatures are potentially more difficult to transcribe. The results in Figure 3 (bottom) do not show a significant difference between the measured accuracy of different time signatures. The complex time signatures of $7/8$ and $5/4$ both are slightly less accurate, but this observation could be caused by a random deviation, or by features correlating with complex time signatures such as number of notes in a fragment.

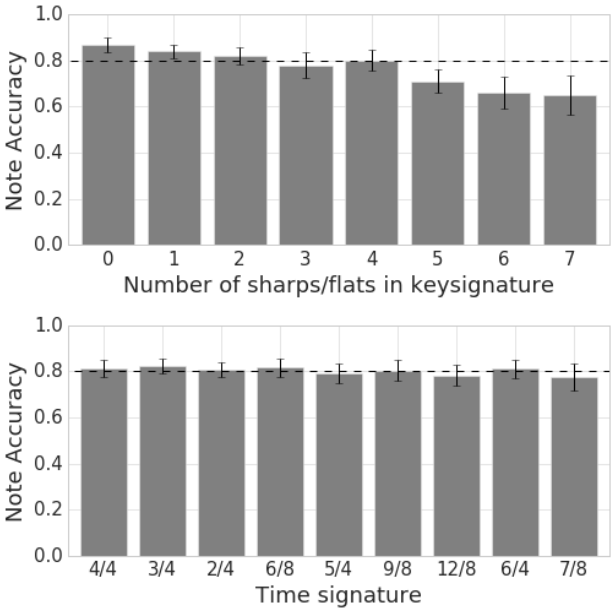

Figure 3. Top: The note-level accuracy for each number sharps/flats in the key signature. Bottom: The note-level accuracy for the most common time signatures. The dotted lines indicate the mean accuracy of 0.80.

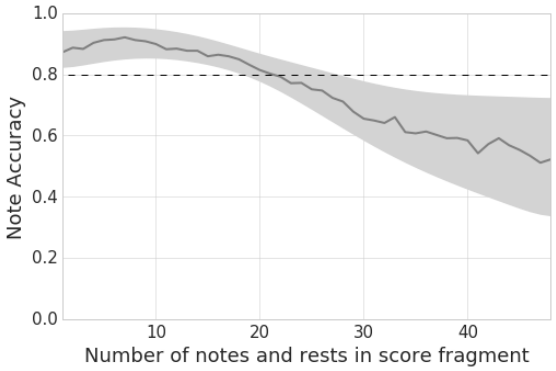

Figure 4. The mean note-level accuracy for each number of notes per fragment, with a confidence interval at a 95% level fit with a Gaussian Process.

As a final evaluation, we look at the correlation between number of notes in a fragment and accuracy. The model capacity and the representation between encoder and decoder are of a fixed size, which forces the model to represent more notes in the same space for fragments with a higher note density. This higher density could cause a loss in accuracy. Figure 4 shows clear evidence that this is the case; fragments containing more than 25 notes have a significantly lower accuracy than the measured mean.

6 Discussion

We propose the Convolutional Sequence-to-Sequence model to deal with the difficulties OMR presents for learning systems. By using an end-to-end trainable sequential model, we completely move away from segmented symbol recognition, and perform the full OMR pipeline with a single algorithm. By incorporating Sequence-to-Sequence models into OMR, there are many new possibilities for obtaining development data. We view this aspect as the largest advantage the proposed method has over segmented models, as the acquisition of quality training data can be a limiting factor. The proposed model shows that it is robust to noisy input, an important quality for any OMR model. Additionally, the experiments show that it can deal with the large scale Elastic Transformations that essentially change the musical font. In future research, this aspect could be expanded to include handwritten notation.

A weakness of the model is pitch classification. Pooling operations introduce a degree of translation invariance, we hypothesize this invariance reduces the pitch recognition accuracy by discarding information about symbol position. However, omitting pooling operations from the model would greatly reduce the dimensionality reduction performed by the CNN. We propose incorporating a combination of convolutional layers and fully connected layers as a possible solution.

Furthermore, on more complex scores the model performs significantly worse. Both the number of sharps or flats in the key signature and the note density in the score fragment play a large role in the prediction accuracy. In future work, these problems could be addressed in multiple ways. A separate key signature recognition could be performed, and given as additional information to the model. This would take away some of the long range computations the key signature introduces and could improve the results on more complex scores.

The difficulty of translating long sequences with Sequence-to-Sequence models is a well studied problem [4, 16]. For longer sequences, the model needs to encode more information in the same fixed size representation, reducing the amount of storage available per note. A possible solution for this difficulty is proposed by Bahnadau et al. [2]: they replace the fixed size representation between encoder and decoder with an attention mechanism, a method that essentially performs a search function between the two networks. This mechanism has shown improvements to Sequence-to-Sequence models in neural machine translation, and could be used in the proposed method to alleviate some of the problems introduced with long sequences and long range dependencies.

The experiments performed in this research are exclusively on monophonic scores. The current representation of (pitch, duration) pairs does not allow for polyphonic note sequences, and in order to apply the model to polyphonic OMR tasks this representation needs to be adapted. A possible representation could be produced by using a method close like the MIDI-standard, which only indicates the starts and ends of pitches. An in-depth description of this method can be found in Appendix 8.1.

Finally, we propose that the Convolutional Sequence-to-Sequence model could be applied to tasks outside of OMR that translate a spatial sequential representation to a sequence of labels. Within MIR, tasks like Automatic Music Transcription can be considered as such a task, where a representation of an audio signal is converted to a sequence of pitches and durations. Outside of MIR, tasks like video tagging or Optical Character Recognition are similar examples.

7 Acknowledgements

Anonymized

8 Appendices

8.1 Polyphonic Representation

The representation as presented in Section 3.1 is only applicable to monophonic scores. In order to extend the method to polyphonic notation, we propose a method close to a piano roll or MIDI format. In MIDI representation, a sequence is divided into elementary time elements called ticks. Each tick can contain events about the starts or ends of notes. We propose a polyphonic representation using a similar division, for which four types of events are required: a tick start event e, a tick stop event s, a sequence end token S, and an event to indicate a pitch. Each s and e pair indicates a tick of pre-defined length. Between each s and e symbol, starts and ends of pitches are defined. As an example, with a tick duration of an 8th note, a C major triad of one quarternote, followed by a C5 8th note would be represented as follows: *(s, C4, E4, G4, e, s, C4, E4, G4, e, s, C5 e, S)*.

An important difference between this representation and the monophonic representation is that the lengh of the polyphonic representation is significantly longer. For this reason, we suggest a dynamic deep learning framework such as Torch [5] or Chainer [17].

9 References

[1] Musescore. `https://musescore.org/`.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Donald Byrd and Megan Schindele. Prospects for improving omr with multiple recognizers. In *ISMIR*, pages 41–46, 2006.

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[5] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Anil K Jain and Douglas Zongker. Representation and recognition of handwritten digits using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1386–1390, 1997.

[8] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[9] Ken Perlin. Improving noise. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 681–682. ACM, 2002.

[10] Laurent Pugin. Optical music recognitoin of early typographic prints using hidden markov models. In *ISMIR*, pages 53–56, 2006.

[11] Laurent Pugin, John Ashley Burgoyne, and Ichiro Fujinaga. Map adaptation to improve optical music recognition of early music documents using hidden markov models. In *ISMIR*, pages 513–516, 2007.

[12] Ana Rebelo, G Capela, and Jaime S Cardoso. Optical recognition of music symbols. *International journal on document analysis and recognition*, 13(1):19–31, 2010.

[13] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre RS Marcal, Carlos Guedes, and Jaime S Cardoso. Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, 1(3):173–190, 2012.

[14] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[15] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, volume 3, pages 958–962. Citeseer, 2003.

[16] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[17] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*, 2015.

[18] Cuihong Wen, Ana Rebelo, Jing Zhang, and Jaime Cardoso. A new optical music recognition system based on combined neural network. *Pattern Recognition Letters*, 58:1–7, 2015.