BLSTMs and Attention-Based Seq2Seq Models for Handwritten Music Score Recognition: An Implementation Study

Nicola Bertocci, Stefania Cerboni
Università degli Studi di Firenze
nicola.bertocci@edu.unifi.it, stefania.cerboni@edu.unifi.it

October 28, 2023

Abstract

This work is part of the 9 ECTS credit course in Data and Document Mining, taught by Prof. Simone Marinai, for the Master's Degree in Computer Engineering at the University of Florence, for which it constitutes the final project.

The project begins with the analysis of the scientific paper *"Handwritten Historical Music Recognition by Sequence-to-Sequence with Attention Mechanism"* [1]. It then continues with the implementation of solutions similar to those proposed by the authors for the recognition of handwritten music scores, one of which considered a baseline and described in [2]. The work concludes with a comparison of the performance of our models with those presented in the paper.

Contents

1 Introduction

The scientific article from which this exercise was developed tackles the problem of recognizing historical handwritten musical scores through a sequence-to-sequence neural network model with an attention mechanism, curated by a group of researchers from the *Universitat Autònoma de Barcelona*.

Optical Music Recognition involves converting images of musical scores into a digital format, such as MIDI. Excellent performance has been achieved in the case of scans of printed musical scores, however, the recognition of handwritten scores is still a challenge due to additional difficulties, such as variability in the writing styles of composers, degradation of the paper medium on which they have been written for many centuries, a significant amount of touching elements and symbols in the images, the lack of a standard notation and, from the machine learning perspective, the scarcity of labeled datasets.

The motivations behind such a study are intuitive. The immense quantity of musical compositions in archives forces scholars to focus only on the most promising in terms of quality. To give some numbers: in a large enough European cathedral or church, it is easy to find more than 2000 compositions by various artists, which means tens of thousands of score pages for each church. The transcription of a musical score takes a lot of time if done manually, we are talking about one to three hours for each page, therefore the purpose of the research is to adapt existing Optical Music Recognition systems to the peculiarities of older scores, to facilitate their discovery, digital transcription, analysis, and dissemination of unknown compositions and composers.

2 Neural Networks

The authors in [1] compare the performance of a sequence-to-sequence neural network model with an attention mechanism to a BLSTM they themselves proposed as a baseline in a previous article [2].

2.1 Baseline Network

Here, we describe the key features of the neural network used as the baseline, in its revised version for the comparison detailed in [1].

1. **Input:** The musical score pages must have been previously segmented into lines; each image of a score line is binarized and then resized to have a standard height of 100 pixels, but the width-to-height ratio is retained: images in the same batch are padded so they all have the same width, while the width of images across different batches may vary.
2. **Convolutional Block:** It consists of three convolutional layers that mimic the first three levels of the ResNet18 network. The kernels are 3x3 in size, followed by batch normalization and an activation given by the Rectified Linear Unit (ReLU). Finally, a 2x1 max-pooling operation is applied, which is used to reduce the vertical dimension of the image while maintaining its original width.
3. **Recurrent Block:** This block uses bidirectional LSTM networks (BLSTM) to take advantage of the context during the recognition of each symbol; the LSTMs are capable of learning long-term dependencies, avoiding the gradient vanishing problem, and the bidirectionality provides additional context information from the entire image.
4. **Dense Layers:** After the recurrent block, there are two fully connected (FC) layers. In this way, two outputs are obtained: one for rhythm and one for pitch.
5. **Output:** The output from each of the two dense layers is a matrix whose columns are the class probabilities (respectively for rhythm and pitch) for each pixel column in the original image. Each matrix has the same width as the original image and a height of 80 possible classes for rhythm and 28 for pitch. By applying a threshold to these matrices, it is possible to determine which symbols appear in the musical scores; more than one symbol can appear at the same time.
6. **Conversion to Symbol Sequence:** The outputs are converted into an array, combining rhythm and pitch. The arrays are used to evaluate the model at the rhythm and pitch levels and also for the entire system, where both parts must be predicted correctly.

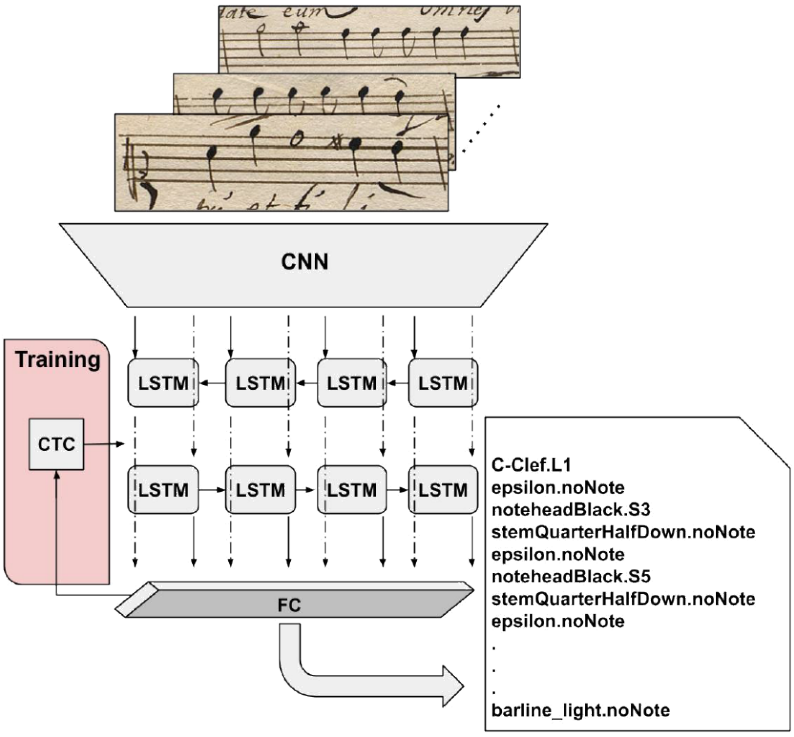


Figure 1: Baseline Network

7. **Loss Function:** Given the nature of the OMR (Optical Music Recognition) problem where multiple symbols can appear simultaneously, and the sequences can have varying lengths, the CTC (Connectionist Temporal Classification) loss function was chosen in [1], as more suitable for comparing sequences of different lengths, making it a more appropriate choice for this task over the previously used Smooth L1 loss in [2].

2.2 Seq2Seq Network

Musical scores are written on staves in a sequence, so the most recent approach proposed by the authors is also based on recurrent models. Specifically, the second method is grounded on the text recognition methodology sequence-to-sequence (seq2seq) and adapted to musical scores. This methodology employs an encoder-decoder framework based on an attention mechanism.

- **Encoder:** Given an input image, the encoder extracts high-level features by encoding the content of the image into an internal representation, termed the "hidden state", which is then used by the decoder to produce the output. The proposed encoder is implemented using a VGG-19-BN network without the last max-pooling layer and with weights pre-trained from ImageNet. Features extracted from the VGG are resized into a two-dimensional feature map which is then fed into a multi-layer BGRU for further positional information.
- **Attention Mechanism:** As an attention module, they chose to employ a position-based attention as proposed by Chorowski et.al.[3]. The attention mechanism is tasked with aligning the feature representations derived from the encoder with our decoding steps. At every step, the attention mechanism weighs the features encoded by the encoder and combines them with the current input of the decoder to assist in generating the next symbol, allowing the model to "focus" on different parts of the input as it produces the output. Instead of using only the last hidden state of the encoder, the decoder can thus weigh and use all hidden states to generate the output. This is especially useful when the input and output have different lengths, as in the case of musical recognition.
- **Decoder:** The decoder module employs a series of recurrent units, namely unidirectional and multi-layered GRUs (Gated Recurrent Units), to maintain a memory of the context as it produces the output. At each time step, the decoder receives the concatenation of its prior embedding vector

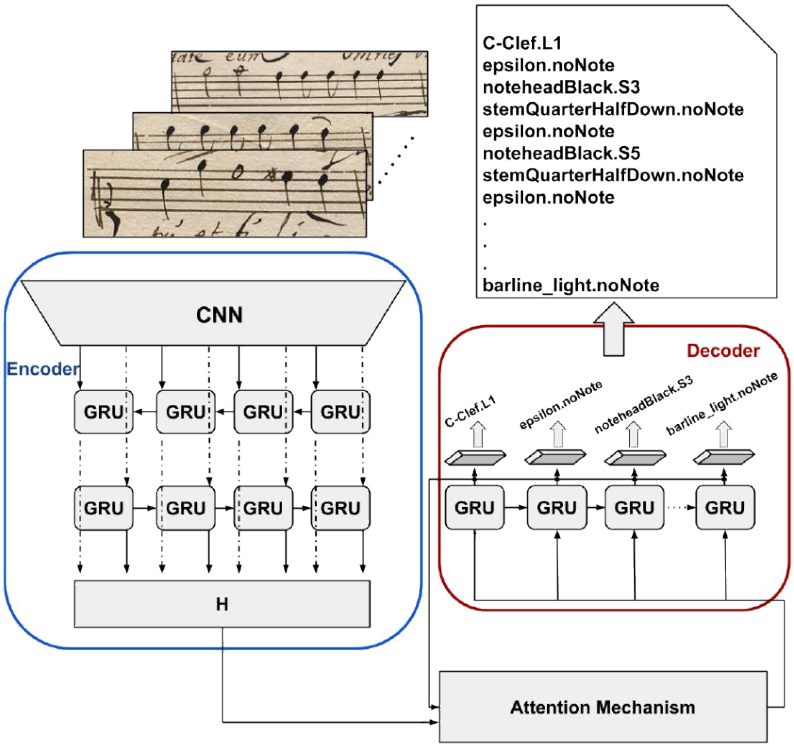


Figure 2: Seq2Seq Network

and the current context vector (defined by the features encoded by the encoder and the attention mechanism) in order to predict an output symbol. This is done by generating a probability vector for every possible single output symbol, from which the most probable one is chosen as the current step's output. The selected symbol also undergoes an embedding layer, converting it into a dense vector, ready to be fed back into the decoder for the prediction of the next symbol. The decoding process continues until an end-of-sequence symbol is generated or until the sequence length reaches a predefined maximum length.

GRUs are a variant of the LSTM (Long Short-Term Memory) units designed to capture long-term dependencies in a sequence without the vanishing gradient problem. GRUs are lighter than LSTMs in terms of parameters and computation but are just as effective in capturing temporal dependencies.

3 Dataset

To address the lack of necessary data, the authors explain that they used three datasets to train their models:

- A *real* dataset, made up of scans of genuine historical handwritten scores. This consists of a motet composed by the Catalan musician Pau Llinàs in the eighteenth century: 40 score lines were manually labeled, totaling 245 images of musical measures, an example of which can be seen in figure 3. These compositions have the property of being monophonic, meaning no multiple note symbols appear simultaneously.
- An "ancient" *synthetic* monophonic dataset, generated using the software *Lilypond* [1], which attempts to simulate the colors and degradation of the paper of historical scores by adding a background.
- A "modern" *synthetic* polyphonic dataset, generated in the same way, where multiple notes to be played simultaneously, like chords, are present. The goal is to allow the models to generalize to any type of ancient score.

[1] https://lilypond.org

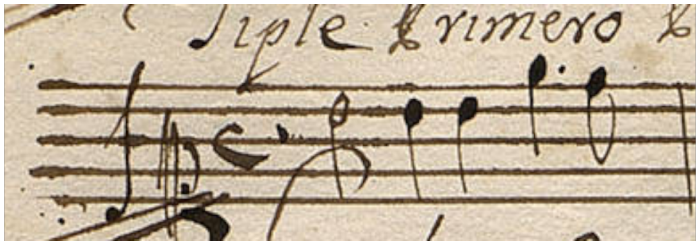
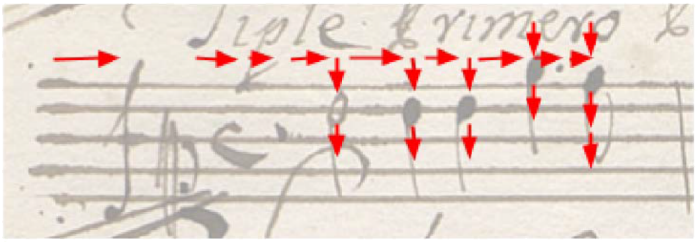


Figure 3: A music measure



C-Clef.L1~epsilon~timeSig_common~epsiln~halfRest~epsilon
~noteheadHalf.L4~stemDown~epsilon~noteheadBlack.L4~
stemDown~epsilon~noteheadBlack.L4~stemDown~epsilon~
noteheadBlack.S5~stemDown~epsilon~dot~epsilon~
noteheadBlack.L5~flag8thDown~epsilon~barline_light

Figure 4: The annotations

The same recurrent neural models that show good performance when applied to handwritten text recognition (HTR) need to be adapted to musical scores due to their intrinsic two-dimensional nature. As the authors point out, first and foremost, the notes on the stave are composed of rhythm and melody, and scores also feature additional symbols such as ornaments, ties, or annotations that provide further instructions. Essentially, there is the possibility that a group of symbols appears simultaneously in time and not just one at a time, as might be the case with letters in written text. For this reason, it was decided to approach the problem as a graph serialization task: as visible in the example in figure 4, an order among the symbols on the stave was defined that goes from left to right and from top to bottom. The scores are annotated at a primitive level (i.e., noteheads, stems, flags, rests, etc.) and distinctly for rhythm and melody. For instance, the symbol corresponding to the head (the dot) of the first note appearing on the stave is indicated as `noteheadHalf.L4`, where *noteheadHalf* indicates that it is an "empty" head, while *L4* indicates its position on the fourth line of the stave. The model's output will then be a one dimensional ordering of musical primitives of this kind.

3.1 Synthetic Dataset

As mentioned, due to the limited availability of data, the article's authors generated two synthetic datasets using the *Lilypond* software for training their neural networks. LilyPond is a music notation system designed for composing and arranging scores. It uses a text-based markup language to represent music, allowing users to produce scores in a readable format and obtain high-quality, typographically accurate outputs. Below is an example of a *C Minor* scale in Lilypond syntax. Figure 5 displays the image of this scale, converted from the PDF produced by the software (the conversion process will be detailed later).

```
\version "2.24.2"
\paper {
oddFooterMarkup=###f
}
\relative c' {
\key c \minor
c d es f g as bes c2 |
```

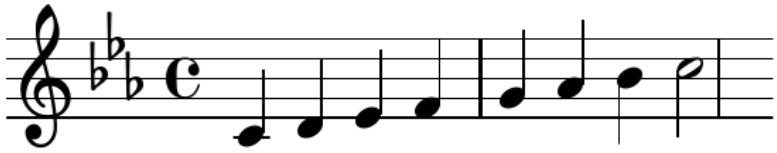


Figure 5: C Minor Scale generated with Lilypond

```
}
```

The example specifies the LilyPond software version used. The `\paper` block defines certain layout settings for the score, in this instance, simply removing the footer. The notation `\relative c'` sets the starting point for the notes that follow, and the subsequent block denotes the C minor scale. The sequence `c d es f g as bes c2` represents the scale's notes.

3.1.1 The Original Annotations

Given that the synthetic datasets utilized by the authors aren't publicly available – neither the one mimicking the appearance of ancient scores nor the modern-looking one with complex symbols – we chose to emulate their approach using Lilypond. This would allow us to generate an automatically annotated dataset. Our main challenge was the annotation format of the existing dataset, which couldn't be precisely replicated using Lilypond's syntax. The core issue lay with the primitive level of annotation in the historical dataset: the authors defined classes such as full or empty noteheads, stems with their orientation, and additional symbols like sharps, flats, or dots that extend the note's duration by 50%. Due to our limited familiarity with Lilypond's syntax, controlling note orientation in such detail seemed unfeasible. Hence, we introduced new symbol classes and mapped the original annotations to our definitions. It's worth noting that each symbol combines rhythm and pitch. Below are the lists of rhythm and pitch classes from the historical dataset:

```
barline_light, barline_light-light, beam8thDown, beam8thUp, beamDownEnd
    ↪ , beamDownStart, beamUpEnd, beamUpStart, C-Clef, dot, eighthRest,
    ↪  endSlur, flag16thDown, flag8thDown, flag8thUp, flat, halfRest,
    ↪ mmrSymbol_1, mmrSymbol_2, mmrSymbol_3, natural, noteheadBlack,
    ↪ noteheadHalf, noteheadWhole, quarterRest, sharp, startSlur,
    ↪ steamQuarterHalfDown, steamQuarterHalfUp, timeSig_common, 32
    ↪ thRest, epsilon
```

```
noNote, L1, L2, L3, L4, L5, L6, S1, S2, S3, S4, S5, S6, epsilon
```

The list of rhythm classes is evidently incomplete due to the scarcity of labeled images. We notice two types of `barlines` representing bars separating measures, `beams` connect consecutive eighth notes, `C-Clef` signifies a clef, `dot` is the dot that increases a note's duration, `rests` and `slurs` are obvious, `flags` are the characteristic flags of eighth or sixteenth notes, `flat`, `sharp` and `natural` are accidentals symbols, `mmrSymbols` are longer rests, the `noteheads` are the various note heads, the `steams` are the stems of the quarter or half notes, `timeSig_common` is the 4/4 time signature indication (represented by the letter `C`), and finally, `epsilon` is the spacing symbol separating two distinct symbols on the stave. Each of these symbols is associated with one of the pitch classes, which indicate its position on the staff, if relevant, as in the case of notes, or if it is not a note. Symbols from `L1` to `L6` are placed on the lines (starting from the lowest one), and those from `S1` to `S6` are in the spaces.

3.1.2 Our Annotations

We retained the separation of classification into rhythm and pitch, but defined for the rhythm classes that do not consider the orientation of the notes or the accessory symbols that modify their meaning. For example, with the old classes, it was possible to find the following sequence of annotations: `~sharp.noNote~epsilon~noteheadBlack.L3~flag8thDown.noNote~epsilon~dot.noNote~`, which corresponded to an eighth note with the stem facing down, positioned on the third line, with a sharp alteration and a

dot to extend its duration. Even excluding the spacing given by the `epsilon`, it is still four symbols to indicate one note. Therefore, we defined new classes that somehow collect and combine these possibilities into unique names for even different combinations of symbols. The following are the lists we defined:

```
blank, epsilon, barline, C-Clef, startSlur, endSlur, timeSig_common, 32
    ↪ thRest, 16thRest, eighthRest, quarterRest, halfRest, wholeRest,
    ↪ doubleWholeRest, quadrupleWholeRest, 32thNote, 16thNote,
    ↪ eighthNote, quarterNote, halfNote, wholeNote, dotted32thNote,
    ↪ dotted16thNote, dottedEighthNote, dottedQuarterNote,
    ↪ dottedHalfNote, dottedWholeNote, flat32thNote, flat16thNote,
    ↪ flatEighthNote, flatQuarterNote, flatHalfNote, flatWholeNote,
    ↪ sharp32thNote, sharp16thNote, sharpEighthNote, sharpQuarterNote,
    ↪ sharpHalfNote, sharpWholeNote, natural32thNote, natural16thNote,
    ↪ naturalEighthNote, naturalQuarterNote, naturalHalfNote,
    ↪ naturalWholeNote, dottedFlat32thNote, dottedFlat16thNote,
    ↪ dottedFlatEighthNote, dottedFlatQuarterNote, dottedFlatHalfNote,
    ↪ dottedFlatWholeNote, dottedSharp32thNote, dottedSharp16thNote,
    ↪ dottedSharpEighthNote, dottedSharpQuarterNote,
    ↪ dottedSharpHalfNote, dottedSharpWholeNote, dottedNatural32thNote,
    ↪  dottedNatural16thNote, dottedNaturalEighthNote,
    ↪ dottedNaturalQuarterNote, dottedNaturalHalfNote,
    ↪ dottedNaturalWholeNote
```

```
blank, epsilon, noNote, L0, L1, L2, L3, L4, L5, L6, S0, S1, S2, S3, S4,
    ↪  S5, S6
```

We kept the spacing symbol `epsilon`, as well as the key `C-Clef`, the slurs, and the rhythm indication `timeSig_common`. We merged the `barline` into a single symbol, defined rests from a duration of 1/32 to 16/4, and defined standard classes for notes: `32thNote`, `16thNote`, `eighthNote`, `quarterNote`, `halfNote`, `wholeNote`. Additionally, notes can also have the prefix `flat`, `sharp`, and `natural` for alterations and `dotted` in case they are followed by a dot that increases their duration. Regarding the pitch, we effectively kept the same ones, adding `L0` and `S0` for notes that are sometimes written below the staff (for example, C and D in treble clef).

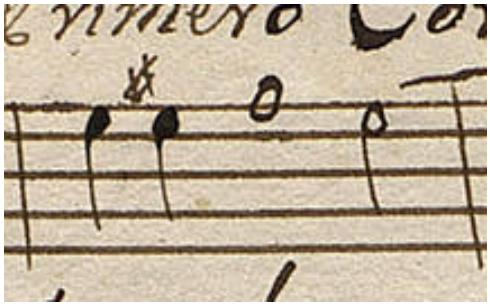
In Figure 6, there is an example of a musical measure annotated in both the old and the new styles. It's immediately clear how the new format is much more concise.

3.1.3 Dataset Generation

We generated a dataset to address the OMR problem. Specifically, our aim was to create a dataset compatible as input for our pre-existing neural networks. To ensure that our generated images resembled the handwritten style, we made certain that each musical segment precisely emulated a single measure. Like the handwritten scores, our dataset is monofonic. This process involved creating annotations in a text file and simultaneously producing image files in PNG format of the associated scores. In detail, we defined a function that, for every image to be generated:

1. Creates a random annotation of a length between 1 and 8 symbols, excluding barlines or musical clefs that alternate in the first position.
2. Converts the notation into Lilypond syntax and writes it to a .ly file.
3. Converts the .ly file to a PDF using Lilypond.
4. Converts the PDF to a PNG image, then crops it to its content with a slight margin, making it resemble scans of handwritten measures.

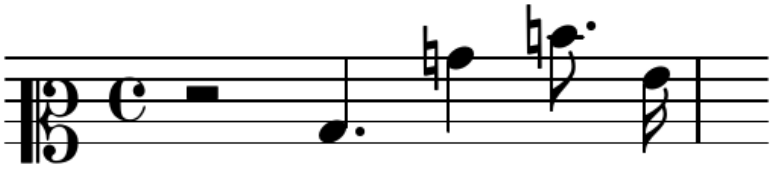
While a musician might discern in figure 7 that the total values of the depicted notes don't match the 4/4 time signature, within the scope of symbol recognition, adhering to musical metrics isn't crucial. In our generated dataset, each symbol, barring the clefs but including combination of signs on the staff that, when combined, produce a note, has an equal likelihood of appearing in any sequence. For completeness,



```
01-1$166|barline_light.noNote~epsilon~noteheadBlack.S4~
    ↪ steamQuarterHalfDown.noNote~epsilon~sharp.S4~epsilon~
    ↪ noteheadBlack.S4~steamQuarterHalfDown.noNote~epsilon~
    ↪ noteheadWhole.L5~epsilon~startSlur.noNote~epsilon~
    ↪ noteheadHalf.S4~steamQuarterHalfDown.noNote~epsilon~
    ↪ barline_light.noNote
```

```
01-1$166|barline.noNote~epsilon~quarterNote.S4~epsilon~
    ↪ sharpQuarterNote.S4~epsilon~wholeNote.L5~epsilon~
    ↪ startSlur.noNote~epsilon~halfNote.S4~epsilon~barline.
    ↪ noNote
```

Figure 6: Example of Annotation Update



```
00_11$166|C-Clef.L2~epsilon~timeSig_common.noNote~epsilon~
    ↪ halfRest.noNote~epsilon~dottedQuarterNote.S1~epsilon~
    ↪ naturalQuarterNote.L5~epsilon~dottedNaturalEighthNote.L6~
    ↪ epsilon~16thNote.L4~epsilon~barline.noNote
```

Figure 7: A musical measure automatically generated by our software and its corresponding annotation

in Figure 8, we present the entire Lilypond syntax used to generate a PDF containing only the musical score, without any additional annotations or automatic symbols. The displayed syntax corresponds to the example shown in Figure 7. Using this approach, we built a synthetic dataset five times larger than the handwritten one.

4 Experimentations

The implementations of the two neural networks were developed in the Python language using the Pytorch library[2]. The code produced over the course of this study is publicly available on GitHub[3].

Both the handwritten dataset and the dataset we generated with Lilypond have been divided into distinct training, validation, and test sets, in the proportions of 60%, 20%, and 20%, respectively. Specifically:

- The handwritten dataset consists of 245 images; 147 were used for training, 49 for validation, and 49 for testing.

[2] https://pytorch.org/
[3] https://github.com/stefaniacerboni/DDM_HandwrittenMusicRecognition

```
\version "2.24.2"
\paper {
    line-width = 210\mm
    oddFooterMarkup=##f
    oddHeaderMarkup=##f
    bookTitleMarkup = ##f
    scoreTitleMarkup = ##f
    ragged-bottom = ##t
    ragged-last-bottom = ##t
    ragged-right = ##t
    footer = ##f
}
{
    \override Staff.Clef.stencil = ##f
    \override Staff.TimeSignature.stencil = ##f
    \cadenzaOn
    \revert Staff.Clef.stencil \clef "mezzosoprano" \revert Staff.
        ↪ TimeSignature.stencil \time 4/4
    r2  b4.  b'!4  d''!8.  g'16  \bar "|"
    \cadenzaOff
    \once
    \override Score.TimeSignature.transparent = ##t
    \time 1/16
    s32
}
```

Figure 8: Lilypond syntax

- The synthetic dataset comprises 1225 images; 735 were allocated for training, 245 for validation, and 245 for testing.

We chose to train the implemented neural networks using different combinations of the available datasets and observe the variations in performance. As an error metric, we used the Symbol Error Rate (SER), as defined by the authors of the article:

$$SER = \frac{S + D + I}{N} \tag{1}$$

where the sum of the number of substitutions (S), deletions (D), and insertions (I) of predicted symbols is divided by the number of symbols actually present in the ground truth (N). Below is the list of experiments conducted:

1. Training, validation, and testing of BLSTM and Seq2Seq on the handwritten dataset, with old and new annotations.
2. Training of BLSTM and Seq2Seq on the synthetic dataset, validation and testing on the handwritten one.
3. Training and validation of BLSTM and Seq2Seq on both mixed datasets, testing on the handwritten one.
4. Training and validation of BLSTM and Seq2Seq using curriculum learning on both datasets, testing on the handwritten one.
5. **Bonus.** Training, validation, and testing of BLSTM and Seq2Seq on the synthetic dataset.

In all cases, we trained the neural networks for an indeterminate number of epochs; training would halt when the prediction performance on the validation set worsened for two consecutive epochs compared to the previous one. The performance of each combination of neural network and training dataset was evaluated on the test set of the handwritten dataset, as the research aim is the recognition of handwritten musical scores. As a "bonus" experiment, we assessed the performance of the same networks when trained and tested on the synthetically generated dataset; given that these images are more "straightforward," we expected better performance.

4.1 Experiment 1

Neural Network	Annotations	Images per Epoch	Epochs Reached	SER on Test Set (%)	Paper Test SER (%)
BLSTM	old	245	80	78.29	56.20
Seq2Seq	old	245	90	2.83	40.39
BLSTM	new	245	90	101.20	-
Seq2Seq	new	245	90	5.16	-

Table 1: Training, validation and testing on handwritten dataset.

In the first experiment, we sought to replicate the training, validation, and testing procedures on a handwritten dataset as detailed in the referenced article. While we had access to the original handwritten dataset, it's important to note that we did not possess all the possible original mappings. Instead, our mapping was limited to notations present in the dataset splits (train, validation, test) available to us. Notably, when using the original mappings, our BLSTM model exhibited a slight performance advantage with respect to new annotations, possibly due to the smaller set of mappings employed. Both models exhibited performance that was worse than the results reported by the authors. In contrast, our Seq2Seq model demonstrated a substantial improvement in performance, achieving a test Symbol Error Rate (SER) of 2.83%, a significant enhancement compared to the BLSTM model and the paper results too. We also investigated whether transitioning from the old mappings to our own mapping adversely affected model performance. This transition did indeed lead to a decline in performance for the BLSTM, possibly because the handwritten dataset was insufficient to encompass all possible mappings. However, our Seq2Seq model continued to deliver consistent performance.

4.2 Experiment 2

Neural Network	Images per Epoch	Epochs Reached	SER on Test Set (%)	Paper Test SER (%)
BLSTM	1225	250	137.39	96.20
Seq2Seq	1225	70	0.00	83.80

Table 2: Training on synthetic dataset, validation and testing on handwritten dataset.

In our second experiment, we trained our models using a synthetic dataset and conducted validation and testing on a handwritten dataset. Notably, our BLSTM model's performance lagged behind the results presented in the reference article. This divergence in performance could be attributed to the BLSTM model's challenges in generalizing the knowledge gained during training on synthetic data and applying it effectively to the handwritten dataset. It's worth highlighting that the reference article utilized two distinct synthetic datasets: old and new, and it's conceivable that the inclusion of the old synthetic dataset played a pivotal role in aiding model learning. In stark contrast, our Seq2Seq model surpassed the performance of the model described in the article, achieving an impeccable 0% test SER on the handwritten dataset.

4.3 Experiment 3

Neural Network	Images per Epoch	Epochs Reached	SER on Test Set (%)
BLSTM	1470	200	59.94
Seq2Seq	1470	100	0.00

Table 3: Training and validation on both mixed datasets, testing on handwritten dataset.

In the third experiment, we noted a substantial enhancement in the BLSTM model's performance when incorporating handwritten data into the training process. This led to a notable reduction in the test

SER, effectively halving it compared to the results from experiment 2. The performance of the BLSTM model, in this case, surpasses that of the same model when trained solely on the handwritten dataset. In contrast, the Seq2Seq model continued to exhibit exceptional performance, consistently maintaining a 0% test SER.

4.4 Experiment 4

Neural Network	Images per Epoch	Epochs Reached	SER on Test Set (%)	Paper Test SER (%)
BLSTM	245	400	104.59	-
Seq2Seq	245	100	20.64	31.79

Table 4: Curriculum learning on both datasets, testing on the handwritten dataset.

In the fourth experiment, we extended the curriculum learning approach applied in the article to both of our models. The article exclusively employed this approach with the Seq2Seq model, whereas we adapted it for our BLSTM model as well.

The curriculum learning method employed in the article involved changing the proportion between handwritten and synthetic datasets every 10 epochs until reaching 100 epochs. However, this method might be limiting for BLSTM models, especially considering their extended training times observed in our previous examples. Given the BLSTM's tendency to require more than 100 epochs for effective training, we adjusted the curriculum learning process, only for the BLSTM.

Instead of fixed intervals, we dynamically altered the proportions between handwritten and synthetic datasets each time the validation Symbol Error Rate (SER) increased. This adaptation resulted in a total of 400 epochs for the curriculum training of the BLSTM.

Notably, the BLSTM model struggled to effectively adapt to this training strategy, resulting in a significantly high test SER. It is reasonable to assume that the model in the article might have encountered similar difficulties, potentially yielding comparable outcomes.

In contrast, our Seq2Seq model demonstrated an improved performance compared to the model in the article. It achieved a lower test SER of 20.64%, surpassing the 31.79% reported in the article. This performance boost is a promising indication of Seq2Seq's adaptability and potential for more effective training in such scenarios.

However, we observed that the Seq2Seq model's performance was inferior to that in experiment 3 with the mixed dataset. We speculate that the challenge lies in the number of images per epoch: as the handwritten dataset consists of 245 samples, implementing curriculum learning necessitates the synthetic dataset (which is considerably larger) to adapt and maintain the appropriate proportions between handwritten and synthetic images.

4.5 Bonus Experiment

Lastly, intrigued by the possibility that learning and recognizing symbols from the synthetic dataset alone might be easier than doing so on the handwritten dataset, we used only the images generated with Lilypond for both training and validation sets, ultimately testing recognition on similar images as well.

Neural Network	Images per Epoch	Epochs Reached	SER on Test Set (%)
BLSTM	1225	250	84.23
Seq2Seq	1225	90	3.83

Table 5: Training, validation and testing on synthetic dataset.

As a bonus experiment, we sought to determine whether learning and recognizing symbols from the synthetic dataset alone was a more straightforward task compared to the handwritten dataset. This additional experiment complements the insights gained from our prior experiments.

In this test, we found that the BLSTM model struggled when trained and tested exclusively on the synthetic dataset, resulting in a high test SER of 84.23. This aligns with the challenges the model encountered in adapting to curriculum learning and suggests that recognizing symbols in a purely synthetic data context may pose significant difficulties.

Conversely, the Seq2Seq model performed significantly better under the same conditions, achieving a much lower test SER of 3.83. This outcome underscores the Seq2Seq model's robustness and adaptability, even when learning and recognizing symbols exclusively from a synthetic dataset. These results, in conjunction with the Seq2Seq model's consistent performance in prior experiments, emphasize its potential and versatility in OMR tasks, regardless of the dataset type.

5 Conclusions

The objective of this project was to reproduce and analyze existing methods in the field of Optical Music Recognition (OMR), with a particular focus on the implementation of two types of neural networks: BLSTM (Bidirectional Long Short-Term Memory) and Seq2Seq (Sequence to Sequence). The findings from this study provide valuable insights into the relative strengths and weaknesses of these models in different experimental settings.

We used two distinct datasets to train the neural models. The first is a handwritten dataset provided by the article whose results we have replicated, and the other was synthetically generated by us using the Lilypond software. Both were divided into training, validation, and test sets. Additionally, due to limitations in using Lilypond for the automatic annotation of the generated dataset, we had to redefine the list of class symbols.

Finally, we conducted a series of experiments to evaluate the performance of the models, following the methodologies proposed in the reference article. The Seq2Seq model showed remarkable robustness, with a Symbol Error Rate (SER) of 0% in some scenarios, thus confirming its effectiveness and potential in this field. The BLSTM model, on the other hand, showed a wider variation in its performance. The inclusion of handwritten data in its training process led to a significant improvement in performance, consistent with the original results.

References

[1] Arnau Baro, Carles Badal, and Alicia Fornés. "Handwritten Historical Music Recognition by Sequence-to-Sequence with Attention Mechanism". In: Sept. 2020, pp. 205–210. DOI: 10.1109/ICFHR2020.2020.00046.

[2] Arnau Baró et al. "From Optical Music Recognition to Handwritten Music Recognition: A baseline". In: *Pattern Recognition Letters* 123 (2019), pp. 1–8. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2019.02.029. URL: https://www.sciencedirect.com/science/article/pii/S0167865518303386.

[3] Jan Chorowski et al. "Attention-Based Models for Speech Recognition". In: *CoRR* abs/1506.07503 (2015). arXiv: 1506.07503. URL: http://arxiv.org/abs/1506.07503.