# Practical End-to-End Optical Music Recognition for Pianoform Music

Jiří Mayer[0000−0001−6503−3442], Milan Straka[0000−0003−3295−5576], Jan Hajič jr.[0000−0002−9207−567X], and Pavel Pecina[0000−0002−1855−5931]

Charles University, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics, Prague, Czech Republic
{mayer,straka,hajicj,pecina}@ufal.mff.cuni.cz

**Abstract.** The majority of recent progress in Optical Music Recognition (OMR) has been achieved with Deep Learning methods, especially models following the end-to-end paradigm, reading input images and producing a linear sequence of tokens. Unfortunately, many music scores, especially piano music, cannot be easily converted to a linear sequence. This has led OMR researchers to use custom linearized encodings, instead of broadly accepted structured formats for music notation. Their diversity makes it difficult to compare the performance of OMR systems directly. To bring recent OMR model progress closer to useful results: (a) We define a sequential format called Linearized MusicXML, allowing to train an end-to-end model directly and maintaining close cohesion and compatibility with the industry-standard MusicXML format. (b) We create a dev and test set for benchmarking typeset OMR with MusicXML ground truth based on the OpenScore Lieder corpus. They contain 1,438 and 1,493 pianoform systems, each with an image from IMSLP. (c) We train and fine-tune an end-to-end model to serve as a baseline on the dataset and employ the TEDn metric to evaluate the model. We also test our model against the recently published synthetic pianoform dataset GrandStaff and surpass the state-of-the-art results.

**Keywords:** Optical Music Recognition · Evaluation · Datasets.

## 1 Introduction

Optical Music Recognition (OMR), is the field that investigates how to computationally read music notation in documents [8], is among the many sub-fields that have seen significant progress with end-to-end approaches to recognition [38,14,9,13,36].

This is straightforward for "monophonic" notation, where the encoded music has just one voice: its output thus consists of a single sequence, and it is analogous to text.[1] However, despite plenty of use cases for such monophonic OMR, a vast amount of music – and some of the most prominent repertoire in the world – is written for piano.

[1] The complexity introduced by the two-dimensional compositional nature of music notation, as opposed to most writing systems for natural languages, is no longer a significant issue for current deep learning methods.

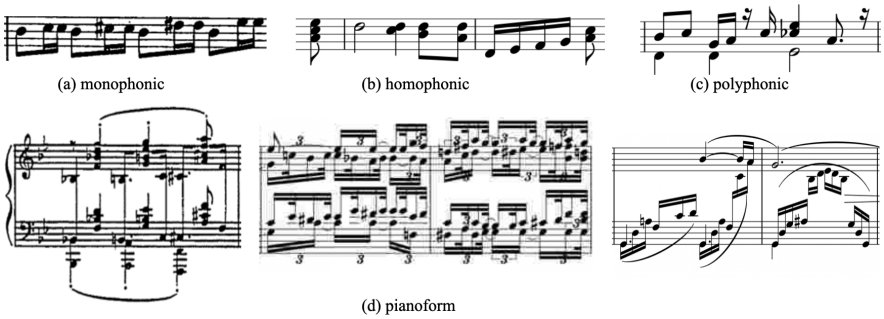


**Fig. 1.** Typology of music notation by complexity. Monophonic scores (a) are straightforwardly encoded as sequences; for (b) homophonic scores (chords allowed, but all simultaneous notes have the same length), advance coding has been used with promising results still with CTC objective [2]. For (c) polyphony, linearization becomes necessary, and (d) pianoform music adds interaction between staffs within one grand staff and generally contains the greatest density of objects.

In terms of notation complexity, pianoform music[2] in Common Western Music Notation (CWMN) is the "final frontier" for OMR models [8] (see Fig. 1), but it too has recently seen promising results with sequence-to-sequence models [37].

Specifically for sequence-to-sequence models, the fact that piano music contains multiple independent voices in parallel, which can arbitrarily appear and disappear in a composition (even for very short segments), introduces an extra layer of complexity, especially on the output side. The problem can perhaps be compared to trying to recognize an unknown amount of texts written over each other. Attention-based models [3], most prominently Transformers [40,37], can produce an arbitrary number of outputs for a single output and do not require a monotonous alignment to exist between the input and output as in Connectionist Temporal Classification (CTC) [21], but they do require the output to be a sequence. Therefore, the ground truth for piano music must be linearized to make training and evaluation possible. However, a model that outputs such a linearized representation is by itself not particularly useful beyond experiments. Formats that encode music notation in practice, such as MusicXML, **kern, MEI, LilyPond, or various open or proprietary formats used to represent notation in widely used editors (MuseScore, Finale, Sibelius, Dorico, etc.), strive to capture the multi-voice structure of the encoded music – their objective is not to be convenient for a particular class of models – and thus for a sequence-to-sequence OMR system to become useful in practice, its linearized target representation must be then followed up by a de-linearization step.[3] This

[2] The term encompasses not only piano music, but also music for organ, harpsichord, harp, vibraphone, possibly guitar, and other instruments. [3] The possible exception is **kern, which has a straightforward enough structure in text that it can be output directly by the OMR model as plain text, and converters exist to other formats. However, it is not as widely adopted as MusicXML, and converters are imperfect.

step introduces further complexity: especially, because there is inherent randomness in the trained model's output, it needs to be able to deal with sequences that do not necessarily lead to syntactically valid notation.

OMR also differs fundamentally from OCR, in that OMR users expect not just to recover the information about how the elements of music notation are arranged to encode a certain musical composition (what in OCR would be recovering the configuration of symbols on a page, termed "reprintability" [8]), but also decode the "musical semantics", the composition itself – which notes should be played at what time (termed "replayability", respectively [8]). All practical formats for representing music notation contain intertwined information about both these realms, and thus a model must decode the semantics such as pitches and durations of notes from the configurations of the graphical elements.

This added complexity leads to another issue in turning the advances in OMR models into palpable progress: evaluation. OMR evaluation is a difficult issue on its own [4,6,7,23]. The natural evaluation metric on sequences of tokens, Symbol Error Rate (SER), has unclear interpretation outside of the specific encoding used by that particular system, and does not allow for direct comparisons between different linearizations. Because music notation is a writing system that tends to have an exception to every basic rule, especially in piano music [6], it is tempting to preprocess data so that symbols and situations that appear peripheral are left out (especially slurs and other symbols that do not directly affect how the encoded music would be exported to MIDI), or that a priori "easier" datasets are assembled in the first place. However, a transparent evaluation of a system's usefulness (as opposed to measuring just the ability of a model to learn what is required of it) should be performed directly on the ground truth files, also in order to show how much of the original score was discarded in this process of "trimming down" to some "core" subset of music notation. While such metrics have previously been suggested, which also attempt to be more informative than SER [25], these have not seen broader adoption.

In order to design, build, and evaluate OMR systems, so that the considerable advances in the field made in recent years thanks to end-to-end models can reach the many potential users, we tackle these challenges. The main contributions of this paper are therefore:[4]

- We propose and implement a direct linearization and de-linearization procedure for MusicXML, the most widely adopted machine-readable music notation interchange format (Sec. 2),
- collect a "difficult" dataset (OLiMPiC) of pianoform music notation from the OpenScore Lieder Corpus [19,20] with synthetic training images, but dev and test sets with real-world images from public IMSLP scans (Sec. 3),
- establish an evaluation comparing MusicXML files directly with an implementation of Tree Edit Distance (TEDn), which better correlates with the preferences of human editors [25] (Sec. 4), and
- achieve state-of-the-art performance on pianoform music (Sec. 6).

[4] The related work for each of the contributions is discussed in their respective sections.

While this work does present a near-complete[5] OMR system for pianoform music with state-of-the-art performance, we have little doubt that better models will soon follow. The main value of our contributions is significant improvements of OMR infrastructure that, taken together, bring considerable progress in the field much closer to application.

## 2 Linearized MusicXML Encoding

In an ideal world, a recognition model will output a well-known standardized format. Out of the available formats, we consider MusicXML to be the most practical choice for machine-readable music notation today, as it retains broad support among all popular notation editors and further tooling (such as the Music21 library), and while the successor MNX format[6] is being developed in the W3C Music Notation Community Group, no obsolescence for MusicXML is planned. Importantly, MusicXML is also the preferred interoperability format for the MuseScore open-source notation editor.[7] Thanks to this broad support, nearly all of the music stored in a computer-readable format can be expected to have a way of being exported to MusicXML with relatively little information lost due to priority support for MusicXML conversion. We therefore view the ability to use MusicXML files for training OMR systems, and producing results in MusicXML, as a major step towards shortening the journey from improved OMR models to improved results for users.

However, while it is technically possible to train a model to output MusicXML files directly, this is not an optimal choice. XML-based formats are tree-based and often excessively verbose. They often contain lots of additional information that cannot be leveraged for training: for example, unique element IDs, other metadata, or pixel-perfect formatting of the score. While it would technically be possible to train a model to output MusicXML strings directly despite these disadvantages, MusicXML (as do all such structured formats) has strict rules on syntax and validity. Thus, a single recognition mistake on a page can make the output document completely invalid, and not even processable by GUI tools for post-correction that assume a valid MusicXML file on input. One would therefore anyway have to implement some post-processing step that would handle the inevitable errors against XML syntax or MusicXML specification. Since a postprocessing step is thus necessary anyway (and in practice preprocessing as well, at least to discard unneeded information and metadata that are not even visible on the page, plus further standardization), we can instead implement these steps by designing a linearization of MusicXML and implementing conversion procedures. Aside from the pre- and post-processing requirements, this allows us to reformat the input data in a way that is much more amenable to sequence-to-sequence learning.

Similar encoding approaches have already been applied for monophonic music, most notably in the PrIMuS dataset [11]. Recently, sequential encoding has

[5] The only remaining step is on the input side: detect where on the page the notation is and split it into systems. [6] https://github.com/w3c/mnx
[7] https://musescore.org/en/node/82366#comment-363536

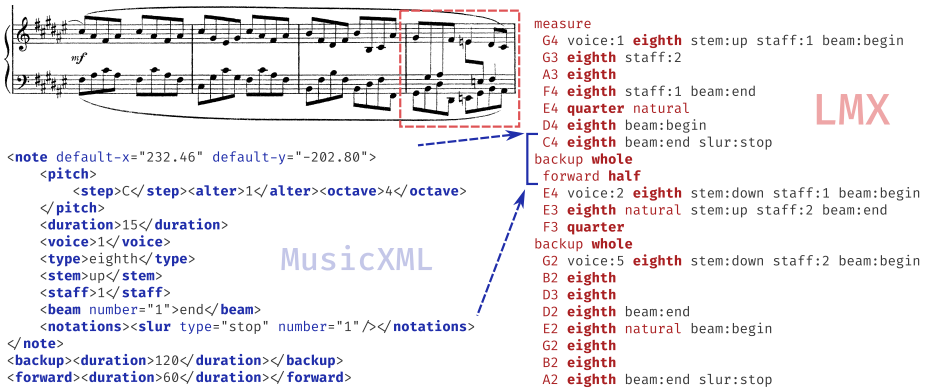


**Fig. 2.** One measure – 246 lines of MusicXML represented only by 96 tokens of Linearized MusicXML (formatting and indentation is present only for better readability).

been tried on homophonic [2] and pianoform music [37]. Building on this work, we propose the LMX format: Linearized MusicXML.

Thus, we propose the LMX format: Linearized MusicXML. The initial idea is to take the XML tree, perform a depth-first walk over its nodes, and convert each element to a corresponding token. We leverage the fact that MusicXML defines the element order, which in turn defines the order of our sequence tokens. We also utilize existing element (and value) names and use them during the naming of our tokens. Modifications are done to reduce verbosity (without loss of information). We encode some elements only via their values, for example, the note `<type>` element is represented by tokens `whole`, `quarter`, `16th`. We do not need to have an explicit `type` token since it would convey no additional meaning. We only use the values as tokens in most other situations (`G4` for pitch), often adding the type explicitly to aid LMX readability (`voice:1`, `tied:start`, `clef:F2`). We also reduce the number of tokens by only encoding state changes instead of absolutes for certain note properties. These are the voice number, staff number (staff within the grandstaff), and stem orientation. All of these are "forgotten" and re-emitted with each measure and with each voice change. We also omit the `beam:continue` token, since it can be inferred.

MusicXML is designed not only for capturing notation visually but also for music replayability [8]. This means that visual information is often separated from audible information. A great example is the `<tie>` and `<tied>` elements. The first one specifies that the note's duration is extended to blend with the following note when played, whereas the second one states that there is a graphical tie present in the score. In this case, we only encode the visual `<tied>` element in LMX and reconstruct both of them during de-linearization. Similarly, we do not encode the pitch `<alter>` token, instead we encode `<accidental>` and key signatures and later infer the alteration during decoding.

For replayability especially, MusicXML contains `<duration>` elements, that encode time information in the number of `<divisions>` (specified for the whole

document). This duration can be reconstructed from the note's `<type>`, duration dots, and `<time-modification>` so we completely omit this information. The only problem is with `<forward>` and `<backup>` elements (note-like objects that only move the internal clock). These lack the `<type>` element so the inference cannot be done here. Instead, we encode the duration as a combination of `<type>` values, that together resolve to the same duration. This also unifies the representation of notes and rests, with these note-like elements and makes the `<type>` token the root of any note-like object.

From this discussion, the design principles behind LMX can be summarized: (1) minimize originality – stick to MusicXML as much as possible; (2) reduce excessive verbosity – represent state changes, instead of state; (3) focus on the visual aspect of music notation – suppress semantics and ignore sound, layout, and metadata information.

Why MusicXML, and not some other format? A seemingly good linear representation is LilyPond, but its purpose is to typeset music, not to describe music that has already *been* typeset. This makes it a programming language rather than a data format and it drives certain decisions, such as that one can force an accidental to appear by writing `!`, but cannot represent an explicitly missing accidental. The Humdrum **kern is also a good option and it is used in the GradStaff dataset [37], but it cannot represent certain situations, such as a voice changing staves in the middle of a beamed group - a situation that should not be neglected in piano music. The remaining MEI and MusicXML are both mature, standardized, and well-known formats. We chose MusicXML because it is well supported by the open-source notation editor MuseScore and the same editor was chosen by Gotham et al. [20] for the creation of the OpenScore Lieder corpus, signifying its importance.

MuseScore also serves an important role in our setup: MusicXML canonicalization. While the standard defines a lot of properties, it leaves some to the user. These include within-chord note ordering, voice numbering (and ordering), or the specifics of linearization of multiple voices. This also includes hacks that get around MusicXML limitations (usually from converging voices on one note/chord) and various bugs and oddities of MuseScore (first voice rests cannot be deleted, only made invisible). It is important to state that we use MuseScore 3.6.2 and that a dedicated canonicalization module should be added in the future to get rid of the tight coupling with MuseScore.

While MusicXML allows for arbitrary time-travel with the `<forward>` and `<backup>` elements, and for arbitrary note-voice assignment with the `<voice>` element, it is more efficient for the format to represent voices one after each other within a single measure (though not the only option, **kern orders notes onset-wise, not voice-wise). MuseScore outputs exactly this variant, using the `<backup>` command as a jump to the start of the measure and the next voice. It never uses it to jump smaller distances. If a voice starts or terminates inside a measure, `<forward>` is added around the notes so that the voice takes up exactly one measure worth of duration. MuseScore also defines a maximum of 4 voices per staff and they are labeled 1–4 and 5–8 for two staves. We also adopt this approach.

One final interesting aspect is the encoding of tuplets. We encode the visual grouping of tuplets via `tuplet:start` and `tuplet:stop` and the duration change via an `XinY` token derived from the `<time-modification>` element. This token modifies the `type` token so an eighth-note sextuplet has duration `eighth 6in4`. We do not cover nested tuplets. The LMX format supports 224 unique tokens. Our current implementation discards dynamics markings, barline styles, pedal symbols, and other symbols with no effect on musical semantics; this accounts for about 4 % of MusicXML nodes (Tab. 3). The complete documentation of how LMX linearization works can be found in our GitHub repository.[8]

## 3 Datasets

Existing OMR datasets fall roughly into two categories based on their purpose: object detection and end-to-end recognition. Object detection datasets contain very little pianoform music (only 3 pages in MUSCIMA++ [26,17]), and the end-to-end datasets have focused mostly on monophonic or homophonic scores, such as PrIMuS [11,10] and Alfaro-Contreras dataset [2]. The GrandStaff dataset is the first one targeting pianoform music [37]. Another large set of manually encoded music that involves piano is the OpenScore Lieder corpus [19,20].

**GrandStaff-LMX** The GrandStaff dataset is a recently published, synthetic, and the first available pianoform dataset intended for end-to-end OMR [37]. It is based on the KernScores corpus[9] – a collection of music scores in the Humdrum **kern format. It contains 474 full-length scores by 6 composers, that were transposed to 3 additional key signatures and sliced up into 3–6 measure segments. These segments emulate individual systems[10] of music on a page. One such segment represents one training sample for the end-to-end recognition model. The resulting dataset contains 53,882 data samples. Each sample is accompanied by a synthetic JPG image of the music (rendered by Verovio [35]) and its distorted variant. We refer to these distorted images as the Camera-GrandStaff dataset.

The authors of GrandStaff purposefully removed dynamics markings, slurs, lyrics, and non-graphic information **kern tokens. Each sample (system) also starts with clefs and key signature (as is usual in music notation), but also with time signature, which is not usually done in printed music. This may artificially help the model in the recognition of tuplets.

For the purpose of the experiments presented in this work, we convert the original GrandStaff encoding into MusicXML by the Music21 library and then into LMX (see Sec. 2). After linearization, the produced LMX files contain 133 unique tokens, omitting slurs, fermatas, tremolos, and many ornaments (staccato, arpeggios, accents). The resulting files are available for download at http://hdl.handle.net/11234/1-5423.

[8] https://github.com/ufal/olimpic-icdar24 [9] http://kern.ccarh.org/ [10] A system in music notation means one line of music, containing all the voices and instruments. It equals one grandstaff in this case.

**OLiMPiC** The OpenScore Lieder corpus [19,20] is a collection of 19th-century German and French songs manually transcribed via MuseScore and made available in its MSCX format. We work with the corpus snapshot from Oct 30, 2023, which includes 1,356 scores (songs), coming from 253 sets[11] by 107 composers, making it very diverse. Almost all scores have one voice part and an accompanying piano part.

We first used MuseScore 3.6.2 to convert the corpus to MusicXML and to generate PNG and SVG images. We used the SVG output to detect the piano brace shape and match it with the corresponding stafflines. This let us slice the PNG files into individual systems, which would then be paired with the structured representations. Not all scores could be processed in this way: some contain no piano part, some contain the brace symbol for non-piano parts, and others are problematic in many different unique ways. We had to skip 52, giving us 1,295 scores.[12] We then extracted piano parts from the MusicXML and sliced them into pages and systems. We made sure each system starts with clefs and key signature (as it should). Finally, we used our linearizer to produce LMX annotations for each system. We release this processed subset as the OLiMPiC (**O**penScore **Li**eder **Li**nearized **M**usicXML **Pi**ano **C**orpus) dataset – the synthetic variant. It contains 17,945 samples (music systems) with 182 unique LMX tokens.

The scores in OpenScore Lieder also contain a reference to the original IMSLP document they were transcribed from. Thanks to the strict transcription rules set for OpenScore, these IMSLP documents have an identical score layout as the transcriptions (measures per system, systems per page). We manually annotated system bounding boxes for 200 scores in their original IMSLP documents to acquire real-world scanned images (dev and test set).

We extracted the PNG images from IMSLP PDFs using the `pdfimages` tool[13]. This sidestepped any rasterization losses but caused annotation issues because some images were extracted B/W inverted, some were rotated, and some were split to multiple layers making them unusable as-is. Also, sometimes MuseScore wraps a system too early, creating an additional system and desynchronizing the layout for the score. This is not an issue in synthetic images, but here we decided to skip these cases instead of painfully implementing a fix. This made us skip 60 scores before we were able to annotate the desired 200 scores. Given the reasons for skipping, we believe this did not introduce any bias to the test set. We release this dataset as OLiMPiC – the scanned variant.

Both the scanned and synthetic OLiMPiC variants come with the same train/dev/test splits. These splits are defined by score IDs and the test split is also set-independent, meaning if a score appears in the test set, no other score from the same set is allowed to appear in the train/dev sets. The sizes of training, dev, and test sets are shown in Tab. 1. The complete dataset is available for download at http://hdl.handle.net/11234/1-5419 under the CC BY-SA license.

[11] A set is the extended work a song belongs to; e.g. a print edition. [12] To ensure train-test set-independence, 9 more scores are ignored. [13] Available from https://www.xpdfreader.com/

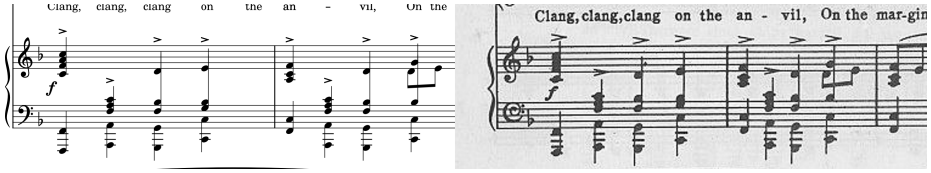


**Fig. 3.** Comparison of a synthetic and scanned sample. Notice the different bass clef style and measure width.

**Table 1.** The OLiMPiC dataset statistics.

| Partition | Synthetic | Scanned | #Sets | #Scores | #Samples | #Tokens |
|---|---|---|---|---|---|---|
| Train | ✓ | ✗ | 206 | 1,095 | 15,014 | 4,107,597 |
| Dev | ✓ | ✓ | 71 | 100 | 1,438 | 378,119 |
| Test | ✓ | ✓ | 33 | 100 | 1,493 | 405,104 |

## 4 Evaluation

Evaluation of OMR is an open problem with few solutions in sight, much less in practice [4,5,6,25,23,8]. There is no one overall best way to evaluate OMR systems because user needs differ significantly among use cases. Our focus in this work is on effort-to-correct (also known as recognition gain [4]): How much work would it be for a user to post-process the output to match the desired music? (Again, this can be approximated just very roughly.)

For sequence-to-sequence models, the go-to class of evaluation metrics is the Symbol Error Rate (SER) which counts the proportion of correctly predicted symbols, and the more stringent Line Error Rate (LER) that counts the proportion of error-free lines. While this is a natural choice, especially during development, when an automated metric is necessary, interpreting the SER numbers is not straightforward. First, symbols in an encoding might have vastly different importance to the result (even before a user is considered) [4]: some may influence the semantics of just one note, others may influence multiple (clef and key signature errors, notoriously, or tuples), others may be negligible (such as the presence or absence of articulation marks). While the specific weights assigned to error classes should take specific use cases into account, which can hardly be done in the course of basic research, we can at least broadly say that errors that influence the musical semantics of the recognition output – pitches, durations, and ordering of notes (or the absence of one, or the presence of a spurious note) – should perhaps have more weight. Second, the choice of linearization introduces artifacts, such as "advance" characters after every note [1] or dots for empty positions on unused spines in **kern [37]). The presence of such artifacts further complicates the comparison between systems that use different linearizations.

A practical comparison of OMR systems should compare "apples to apples" [4,6]. Despite the advantages of LMX for linearization, it is unavoidable that other design criteria will lead developers to use different encodings, and anyway,

ideally the comparison of systems should not depend on the choice of linearization (which is merely a necessity dictated only by the currently best-performing class of models), so this kind of representation is in principle not suitable. Such an "apples to apples" comparison can be done at the level of correctly recovering the finite information visually encoded on the page [23], as exemplified mostly by correct symbol counting [16,4], but this approach cannot be used with end-to-end systems that do *not* recover explicit information about the placement of individual notation symbols.

The other option that avoids polluting evaluation metrics with artifacts of the specific methods used, and possibly more informative for a user, is to compare directly the true endpoint of the OMR process – the encoding of the output in a broadly usable format [8]. MusicXML is a natural choice for this purpose. Evaluating OMR by comparing MusicXML representations has in fact been proposed for this purpose [32,34].[14] So far, however, there is little consensus on *how* to compare two MusiXML files (Padilla et al. write that they "align the OMR output to the ground truth", with no further details provided [34]).

XML files are organized as trees, so Tree Edit Distance (TED) would be a natural choice. Polynomial-time algorithms are known, esp. the Zhang-Shasha algorithm that runs in $O(m^2n^2)$ time and only requires $O(mn)$ memory complexity [41] which also has a Python implementation `zss`. Zhang-Shasha relies on the ordering of child nodes, but fortunately, MusicXML does have an ordering of child nodes defined, and the remaining ambiguity is handled by MusicXML canonization described in Hajič jr. et al. [25] proposes TEDn, a modified TED to estimate replacement costs for differences in musical semantics on notes specifically (which naive TED would over-estimate because of how MusicXML decomposes this information into nodes), and, importantly, provides evidence that TEDn correlates with human editors' expectations of how much effort it would take to modify one file to fit the other using a WYSIWYG editor like MuseScore better than the (few) alternatives.

Therefore, we created a new TEDn implementation and use it as an evaluation metric in this work. One disadvantage of TEDn is that its behavior with respect to SER is not understood, as TEDn has not yet in fact been used to evaluate experiments (because the end-to-end models have not yet been producing MusicXML outputs). Therefore, we report SER as well, which also allows us to directly compare to previous work, and also to have a more direct comparison of the GrandStaff and OLiMPiC datasets.

## 5 Experiments

In the neural network era, optical character recognition (OCR) has been commonly approached by using the convolutional recurrent neural network (CRNN) model [38]. In this model, an image with a line of text is first passed through

[14] For applications focused on the "musical semantics" of notes only, without regard for what elements of music notation were used to encode them, the natural endpoint would be MIDI, such as in [24].

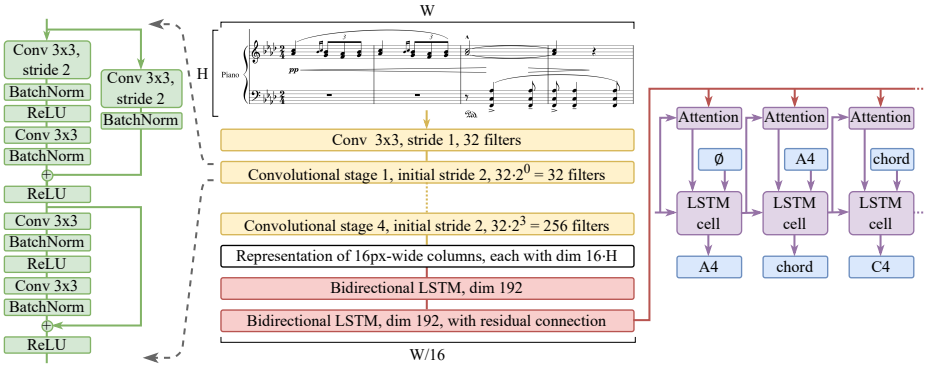


**Fig. 4.** Architecture of our model.

several convolutional layers, then processed with bidirectional recurrent neural networks [22], most commonly LSTMS [29,18], and then a prediction is performed using the CTC layer [21]. The CTC layer enables efficient training and inference when an output sequence (characters in case of OCR) should be generated from an input sequence (representation of fixed-with columns on the input image) without an explicit alignment; however, it requires the order of elements to be the same in both input and output sequences.

The CRNN model can be extended to OMR, but the requirement of the same ordering in the input and output sequences limits it only to homophonic scores [11,12,1]. To approach optical music recognition of polyphonic music, we exchange the CTC layer of the CRNN model with a sequence-to-sequence architecture [39,15], namely an LSTM decoder with Bahdanou attention [3].

The architecture of our model dubbed Zeus is detailed in Fig. 4. The input fixed-height image is first passed through a single $3 \times 3$ convolution, and then through four convolutional stages. In each stage, two ResNet-like blocks [28] with batch normalization [30] are employed, with first convolution in a stage having a stride 2 and since stage 2 doubling the number of filters. Afterward, the values in a single image column are concatenated, obtaining representations of fixed-width columns in the input image. These representations are contextualized by two layers of bidirectional LSTM [22], the second with a residual connection, and finally used as input to an LSTM decoder with Bahdanau attention [3].

We deliberately do not use the Transformer architecture [40], neither as the image encoder nor the sequence decoder. While it is capable of delivering unrivaled performance, it requires a substantial amount of data (even the data-efficient masked autoencoders [27] employ a million images), and we surmise both the sequence encoder and decoder benefit from the inductive locality bias of LSTMs. In Section 6, we validate our approach by showing that our model delivers a 50% relative error reduction compared to existing RNN- and Transformer-based models.

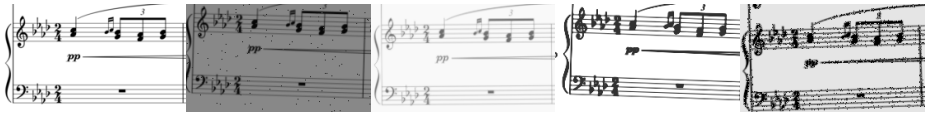

**Fig. 5.** An exemplary part of a system and its four random augmentations.

We train the model on a single 40GB A100 GPU using the Adam optimizer [31] and a learning rate of 1e-3 with a cosine-decay [33] for 500 epochs with a batch size of 64. The input image is rescaled to height 192, the dimensionality of the LSTM cells is set to 192, and we use dropout of 0.2 before and after the bidirectional LSTM layers.

**Augmentations** Given that the training data is synthetic and our goal is to process scanned images, we optionally apply augmentation operations to the synthetic training data. For every image, we consider the following operations in the given order and apply each with 50% chance and randomly chosen magnitude:

- horizontal shift by at most 8 pixels,
- rotation by at most 1 degree,
- vertical shift by at most 4 pixels,
- dilatation/erosion in a random direction on an ellipse with x semi-axis 1 and y semi-axis 0.5,
- for a random probability of up to 20%, negate pixels whose value and value of their 8 neighbors are not uniformly white or uniformly black,
- for a random probability of up to 1%, negate every pixel,
- adjust contrast by a factor with random base-2 logarithm in $[-1, 1]$ range,
- adjust brightness by a random factor in $[-0.5, 0.2]$ range.

Four random augmentations of a part of a system are displayed in Fig. 5. For more details, see the source code of the implementation.

## 6 Results

**GrandStaff** We first show that our model surpasses existing models for optical pianoform music recognition, giving credibility to the later results on the OLiMPiC dataset. We compare our model to the three architectures proposed in Rios-Vila et al. [37] on the GrandStaff and Camera Grandstaff datasets. The evaluation is performed using Character Error Rate (CER), Symbol Error Rate (SER), and Line Error Rate (LER).[15]

The results are presented in Table 2. When trained on GrandStaff and also on Camera GrandStaff, our approach reduces the errors by at least 50% relative compared to all other models, including the Transformer one.

[15] The Character Error Rate originally computed in [37] contained a bug that has since been fixed in https://github.com/multiscore/e2e-pianoform. We report this $\text{CER}_{\text{bug}}$ to compare directly to [37], but we also report the correct CER for future comparison.

**Table 2.** Evaluation of our model on the GrandStaff dataset [37].[15]

| Model | GrandStaff [%] | | | | Camera-GrandStaff [%] | | | |
|---|---|---|---|---|---|---|---|---|
| | $CER_{bug}$ | CER | SER | LER | $CER_{bug}$ | CER | SER | LER |
| Encoder-only CNN [37] | 6.4 | — | 11.3 | 29.8 | 11.9 | — | 22.5 | 58.3 |
| CNN, RNN decoder [37] | 5.0 | — | 7.3 | 23.2 | 7.2 | — | 9.9 | 29.5 |
| CNN, Transformer decoder [37] | 3.9 | — | 5.8 | 16.3 | 4.6 | — | 6.5 | 17.5 |
| Zeus | **1.68** | **2.30** | **2.77** | **8.19** | **1.91** | **2.54** | **3.03** | **8.49** |

**Table 3.** The results on the OLiMPiC and GrandStaff-LMX datasets.

| Dataset | Augmented | SER full [%] | SER w/o tuplets [%] | TEDn full [%] | TEDn lmx [%] |
|---|---|---|---|---|---|
| OLiMPiC Synthetic | ✗ | 11.29 | 9.89 | 13.74 | 9.89 |
| OLiMPiC Synthetic | ✓ | 12.04 | 10.48 | 14.41 | 10.57 |
| OLiMPiC Scanned | ✗ | 59.90 | 58.11 | 44.41 | 42.45 |
| OLiMPiC Scanned | ✓ | 17.72 | 16.11 | 18.40 | 14.85 |
| GrandStaff-LMX | ✗ | 1.78 | 1.70 | 1.60 | 1.56 |
| Camera GrandStaff-LMX | ✗ | 1.99 | 1.92 | 1.77 | 1.73 |

**OLiMPiC** The performance of our model on the OLiMPiC dataset is quantified in Table 3. We train two models – without the training data augmentations and with them and evaluate on both the synthetic and scanned test sets. We report SER on the full Linearized MusicXML, and also the TEDn metric using both the full MusicXML and only the subset captured by Linearized MusicXML. Out of these alternatives, only the full TEDn metric is independent on the encoding selected and capable of comparing dissimilar models.

Considering first the model without augmentations, it achieves 11.3% SER on the synthetic dataset. The TEDn metric evaluated on the full MusicXML is 13.7% and decreases by nearly 4 percent points when considering only the subset captured by Linearized MusicXML. Unsurprisingly, when the model is applied to the scanned images, it performs poorly with 44.4% full TEDn.

The model with augmentations performs slightly worse on the synthetic dataset – by less than a percent point absolute. However, its performance on the scanned images improves considerably to 18.4% full TEDn (one-third more errors compared to the synthetic dataset) and 14.85% Linearized-MusicXML-specific TEDn (one-half more errors). This setting, in our view, starts to provide meaningful numbers in measuring OMR performance overall. Given the inherent limitations of manually annotating real-world images, a user is likely to bring out-of-domain images. The scanned test images simulate this expected out-of-domain nature of production scenarios, at least for IMSLP-style repositories of printed music PDFs, because the OLiMPiC test set comprises flatbed scans with little to no unevenness in lighting and 3D deformation and thus does not provide yet a good model for images taken with phones.

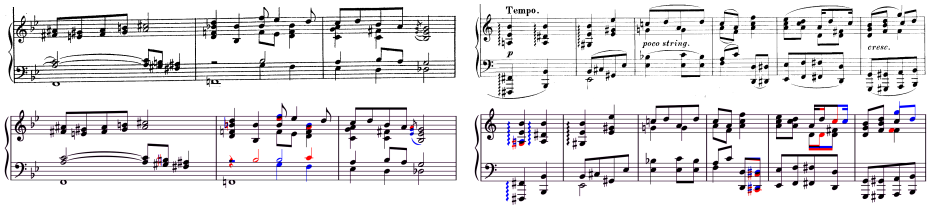


**Fig. 6.** A median-error recognition result on OLiMPiC scanned. Blue is ground truth.

In Fig. 6, we show two example systems from the OLiMPiC scanned dataset and visualization of their recognition. We chose systems containing the median number of recognition errors.

**GrandStaff-LMX** Table 3 includes also the results on the (Camera) GrandStaff-LMX datasets. Both the SER and TEDn metrics on these datasets are less than one-fifth compared to the OLiMPiC dataset, supporting our claim that the music itself in OLiMPiC is a significantly harder pianoform OMR challenge. At the same time, being based on the real-world nature of the OpenScore Lieder Corpus, we believe the measurements on OLiMPiC to be a more accurate reflection of how state-of-the-art end-to-end OMR systems actually perform from a hypothetical user's perspective.

## 7 Discussion and Conclusions

Our work enables applying state-of-the-art sequence-to-sequence models to process pianoform music and output MusicXML, with only MusicXML representations of training data required. Thus, it is now possible to directly develop OMR models for this practical, broadly supported format for representing music notation. Additionally, all sheet music already available in MusicXML (for instance via the MuseScore community) has been "unlocked" for OMR training. Because of limitations of the existing GrandStaff dataset, chief among them being rather "easy" (as evidenced in Tab. 3), we derived the OLiMPiC dataset from the OpenScore Lieder corpus, which can serve as a sufficiently difficult benchmark for comparing further pianoform OMR. While our datasets are pianoform, we worked on this music as it is the most complex, and thus the tools for handling it are as general as necessary to process other kinds of CWMN. Note also that while our experiments are done on individual systems, not entire pages, that is a property only of the experimental setup – nothing in the (de)linearization or evaluation procedures requires splitting the page into systems.

The TEDn evaluation metric then allows for an apples-to-apples comparison directly on MusicXML files, regardless of the linearization or other intermediate representations used within competing systems.

We therefore believe we are now significantly closer to establishing an objective methodology for comparing different OMR systems – again, directly in a

broadly adopted interchange format. While more needs to be done to understand the interactions between the ZSS algorithm and various weighing schemes for elements of MusicXML, there are at least results that show it also correlates with human editors' preferences [25], and thus represents the best available metric.

Finally, our experiments achieve state-of-the-art results on piano music, even with less resource-intensive attention model than the Transformer architecture. They demonstrate that the LMX-based pipeline introduces no new risk for training the sequence-to-sequence models at the core of improvements in OMR performance, while presenting significant advantages in practicality. Also, these results can serve as a baseline for further development and improvements in OMR models – perhaps challenging, but hopefully not be too hard to overtake. If there is a model next month that performs better on the OLiMPiC dataset using LMX and reporting on TEDn, we will consider this work successful than if we retain "top score" for longer.

**Limitations and Future work** One serious limitation of our work is that we rely on MuseScore 3.6.2 for MusicXML canonization, and thus we have a critical external – albeit open-source – dependency. While not an immediate issue, it will take significant development effort to implement MusicXML canonization ourselves, so that LMX becomes a truly standalone, transparent toolchain.

For the recognition model, tuplets remain the most serious issue, with the `2in3` symbol accounting for about 1.5% of SER. This is due to "implicit" triplets: note groups beamed in groups of 3 that should be played as triplets (obviously to human players) but are not explicitly marked as such.

Within the LMX encoding, aside from the roughly 4% of TEDn performance due to LMX not covering certain symbols (see Tab. 3), sets of slurs that reach across multiple measures in parallel are not delinearized in the correct order; we expect that with broader adoption and new datasets, such bugs and edge cases will be discovered. The open-source licensing of our code fortunately allows addressing such limitations as they are encountered by the community.

Finally, we have been glossing over the distinction between OMR for reprintability and replayability, and the different purposes for which OMR can be used [8]. Each of these indeed imposes different evaluation criteria: for instance, focusing on retrieval based on melodies alone does not much care for the correctness of articulation marks, how notes are assigned to voices, or whether other notes than the melody are recognized at all. However, the MusicXML format that we selected with practicality in mind requires the OMR system to recover both the musical semantics, and (most of) the elements of music notation used to encode this music. In any case, adaptations that reduce the vocabulary of LMX for users with such more specific needs are straightforward to implement.

The contributions that we present here should also finally make it possible to move towards one of the major goals of the OMR community, as stated by Calvo-Zaragoza, Hajič jr. and Pacha after a seminal community meeting at GREC/ICDAR 2017 [7]: greater interoperability. Perhaps the long-sought goal of creating an OMR benchmark [6] that communicates meaningful answers to the question: "Does OMR work?" is now within reach.

## References


1. Alfaro-Contreras, M., Calvo-Zaragoza, J., Iñesta, J.M.: Approaching end-to-end optical music recognition for homophonic scores. In: Morales, A., Fierrez, J., Sánchez, J.S., Ribeiro, B. (eds.) Pattern Recognition and Image Analysis. pp. 147–158. Springer International Publishing, Cham (2019)
2. Alfaro-Contreras, M., Iñesta, J.M., Calvo-Zaragoza, J.: Optical music recognition for homophonic scores with neural networks and synthetic music generation. International Journal of Multimedia Information Retrieval **12**(1) (May 2023). https://doi.org/10.1007/s13735-023-00278-5, http://dx.doi.org/10.1007/s13735-023-00278-5
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)
4. Bellini, P., Bruno, I., Nesi, P.: Assessing optical music recognition tools. Computer Music Journal **31**(1), 68–93 (2007). https://doi.org/10.1162/comj.2007.31.1.68
5. Byrd, D., Guerin, W., Schindele, M., Knopke, I.: OMR evaluation and prospects for improved OMR via multiple recognizers. Tech. rep., Indiana University, Bloomington, IN, USA (2010), http://homes.soic.indiana.edu/donbyrd/MROMR2010Pap/OMREvaluation+Prospects4MROMR.doc
6. Byrd, D., Simonsen, J.G.: Towards a standard testbed for optical music recognition: Definitions, metrics, and page images. Journal of New Music Research **44**(3), 169–195 (2015). https://doi.org/10.1080/09298215.2015.1045424
7. Calvo-Zaragoza, J., Hajič jr., J., Pacha, A.: Discussion group summary: Optical music recognition. In: Fornés, A., Bart, L. (eds.) Graphics Recognition, Current Trends and Evolutions. pp. 152–157. Lecture Notes in Computer Science, Springer International Publishing (2018). https://doi.org/10.1007/978-3-030-02284-6_12
8. Calvo-Zaragoza, J., Hajič jr., J., Pacha, A.: Understanding optical music recognition. ACM Comput. Surv. **53**(4) (jul 2020). https://doi.org/10.1145/3397499
9. Calvo-Zaragoza, J., Rizo, D.: Camera-primus: Neural end-to-end optical music recognition on realistic monophonic scores. In: 19th International Society for Music Information Retrieval Conference. pp. 248–255. Paris, France (2018), http://ismir2018.ircam.fr/doc/pdfs/33_Paper.pdf
10. Calvo-Zaragoza, J., Rizo, D.: Camera-primus: Neural end-to-end optical music recognition on realistic monophonic scores. In: Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR). pp. 248–255. Paris, France (2018)
11. Calvo-Zaragoza, J., Rizo, D.: End-to-End Neural Optical Music Recognition of Monophonic Scores. Applied Sciences **8**(4) (2018). https://doi.org/10.3390/app8040606
12. Calvo-Zaragoza, J., Toselli, A.H., Vidal, E.: Handwritten music recognition for mensural notation with convolutional recurrent neural networks. Pattern Recognition Letters **128**, 115–121 (2019). https://doi.org/https://doi.org/10.1016/j.patrec.2019.08.021
13. Calvo-Zaragoza, J., Toselli, A.H., Vidal, E.: Hybrid hidden markov models and artificial neural networks for handwritten music recognition in mensural notation. Pattern Analysis and Applications (Mar 2019). https://doi.org/10.1007/s10044-019-00807-1

14. Calvo-Zaragoza, J., Valero-Mas, J.J., Pertusa, A.: End-to-end optical music recognition using neural networks. In: 18th International Society for Music Information Retrieval Conference. Suzhou, China (2017), https://ismir2017.smcnus.org/wp-content/uploads/2017/10/34_Paper.pdf
15. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: Moschitti, A., Pang, B., Daelemans, W. (eds.) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar (Oct 2014). https://doi.org/10.3115/v1/D14-1179, https://aclanthology.org/D14-1179
16. Droettboom, M., Fujinaga, I.: Symbol-level groundtruthing environment for OMR. In: 5th International Conference on Music Information Retrieval. pp. 497–500 (2004), http://ismir2004.ismir.net/proceedings/p090-page-497-paper117.pdf
17. Fornés, A., Dutta, A., Gordo, A., Lladós, J.: CVC-MUSCIMA: A ground truth of handwritten music score images for writer identification and staff removal. International Journal on Document Analysis and Recognition **15**, 243–251 (2011)
18. Gers, F.A., Schmidhuber, J., Cummins, F.: Continual Prediction using LSTM with Forget Gates. In: Marinaro, M., Tagliaferri, R. (eds.) Neural Nets WIRN Vietri-99. pp. 133–138. Springer London, London (1999)
19. Gotham, M., Jonas, P., Bower, B., Bosworth, W., Rootham, D., VanHandel, L.: Scores of scores: an openscore project to encode and share sheet music. In: Proceedings of the 5th International Conference on Digital Libraries for Musicology. p. 87–95. DLfM '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3273024.3273026
20. Gotham, M.R.H., Jonas, P.: The OpenScore Lieder Corpus. In: Münnich, S., Rizo, D. (eds.) Music Encoding Conference Proceedings 2021. pp. 131–136. Humanities Commons (2022). https://doi.org/10.17613/1my2-dm23
21. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine Learning. p. 369–376. ICML '06, Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1143844.1143891
22. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks **18**(5), 602–610 (2005). https://doi.org/https://doi.org/10.1016/j.neunet.2005.06.042, iJCNN 2005
23. Hajič jr., J.: A case for intrinsic evaluation of optical music recognition. In: Calvo-Zaragoza, J., Hajič jr., J., Pacha, A. (eds.) 1st International Workshop on Reading Music Systems. pp. 15–16. Paris, France (2018), https://sites.google.com/view/worms2018/proceedings
24. Hajič jr., J., Dorfer, M., Widmer, G., Pecina, P.: Towards full-pipeline handwritten OMR with musical symbol detection by u-nets. In: 19th International Society for Music Information Retrieval Conference. pp. 225–232. Paris, France (2018), http://ismir2018.ircam.fr/doc/pdfs/175_Paper.pdf
25. Hajič jr., J., Novotný, J., Pecina, P., Pokorný, J.: Further steps towards a standard testbed for optical music recognition. In: Mandel, M., Devaney, J., Turnbull, D., Tzanetakis, G. (eds.) 17th International Society for Music Information Retrieval Conference. pp. 157–163. New York University, New York University, New York, USA (2016), https://wp.nyu.edu/ismir2016/event/proceedings/

26. Hajič, jr., J., Pecina, P.: The MUSCIMA++ dataset for handwritten optical music recognition. In: 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). pp. 39–46. Kyoto, Japan (2017)
27. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked Autoencoders Are Scalable Vision Learners. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 15979–15988 (2022). https://doi.org/10.1109/CVPR52688.2022.01553
28. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
29. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (Nov 1997). https://doi.org/10.1162/neco.1997.9.8.1735
30. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. p. 448–456. ICML'15, JMLR.org (2015)
31. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)
32. Knopke, I., Byrd, D.: Towards musicdiff : A foundation for improved optical music recognition using multiple recognizers. In: 8th International Conference on Music Information Retrieval. pp. 123–126. Vienna, Austria (2007), http://homes.sice.indiana.edu/donbyrd/Papers/ismir_2007_omr.pdf
33. Loshchilov, I., Hutter, F.: SGDR: Stochastic Gradient Descent with Warm Restarts. In: International Conference on Learning Representations (2017), https://openreview.net/forum?id=Skq89Scxx
34. Padilla, V., Marsden, A., McLean, A., Ng, K.: Improving omr for digital music libraries with multiple recognisers and multiple sources. In: 1st International Workshop on Digital Libraries for Musicology. pp. 1–8. ACM, London, United Kingdom (2014). https://doi.org/10.1145/2660168.2660175
35. Pugin, L., Zitellini, R., Roland, P.: Verovio: A library for engraving MEI music notation into SVG. In: Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014. p. 107–112. International Society for Music Information Retrieval (2014), https://archives.ismir.net/ismir2014/paper/000221.pdf
36. Ríos-Vila, A., Iñesta, J.M., Calvo-Zaragoza, J.: End-to-end full-page optical music recognition of monophonic documents via score unfolding. In: Calvo-Zaragoza, J., Pacha, A., Shatri, E. (eds.) Proceedings of the 4th International Workshop on Reading Music Systems. pp. 20–24. Online (2022), https://sites.google.com/view/worms2022/proceedings
37. Ríos-Vila, A., Rizo, D., Iñesta, J.M., Calvo-Zaragoza, J.: End-to-end optical music recognition for pianoform sheet music. International Journal on Document Analysis and Recognition (IJDAR) **26**(3), 347–362 (Sep 2023). https://doi.org/10.1007/s10032-023-00432-z
38. Shi, B., Bai, X., Yao, C.: An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. IEEE Transactions on Pattern Analysis & Machine Intelligence **39**(11), 2298–2304 (nov 2017). https://doi.org/10.1109/TPAMI.2016.2646371

39. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to Sequence Learning with Neural Networks. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. p. 3104–3112. NIPS'14, MIT Press, Cambridge, MA, USA (2014)
40. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is All you Need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
41. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal on Computing **18**(6), 1245–1262 (1989). https://doi.org/10.1137/0218082