Music Tree Notation for End-to-End Optical Music Recognition

Pau Torras Coloma

Abstract

The field of Optical Music Recognition (OMR) has seen a surge in performance thanks to the most recent advances in computer vision and deep learning as a whole. However, the application of these advances towards real-case scenarios for domains other than typeset music scores is rather complicated, due to limited availability of usable musical data. In this work we propose to tackle *OMR as a single-step process* from image to notation reconstruction, with the key objective to avoid intermediate targets and using existing transcriptions as output, provided that the image contents and the transcriptions can be aligned. We propose a *notation format based on a tree-like scheme* that can be inferred using sequence-to-sequence models, which has already demonstrated promising results in similar image transcription tasks. This tree-like nature could be further exploited by increasing the level of abstraction in the sequence from left to right, inducing the model to focus on actual primitives on the score first and then organising these primitives into higher-order compounds. Finally, we also contribute *a simple tough-to-beat baseline* on the proposed notation format using Sequence-to-Sequence and Transformer models with self-supervised pre-training.

Index Terms

Optical Music Recognition, Music Tree Notation, Sequence-to-Sequence, Transformers, Self-Supervision, Datasets

I. Introduction

"Music is the arithmetic of sounds as optics is the geometry of light."
Claude Debussy

WRITTEN music has been part of the cultural heritage of humankind for many centuries. From neumes as old as a thousand of years to current-time western notation scores, people have built methods to make the most ephemeral of arts persistent through the passing of time. Rivers of ink have been imprinted on libraries worth of paper in order to preserve music, and much of it has indeed survived until today.

It is therefore unsurprising that, given the sheer volume of notably interesting (and oftentimes, forgotten) works of art that have been endowed to our current generations, scholars have turned their attention to computers to aid them in the endeavour of preserving and analysing them. The aspect that concerns us in this work is Optical Music Recognition (OMR), which is the principal task for the computational analysis of written scores; converting images or scans of music into a defined format a computer can process.

While OMR has been studied for more than 4 decades, it still remains a very challenging task. Music notation is bidimensional by nature, with many symbols altering their meaning depending on both local and global context modifiers. The position of symbols in the score also modifies their underlying semantics, the prime example being notes. Furthermore, the syntax of Western Music Notation is very permissive, with many scores being possibly correct but hardly acceptable due to readability or ease of interpretation. Finally, and most critically, there is a severe lack of finely annotated datasets for many recognition contexts, which complicates the application of any pattern recognition method that requires learning.

As an application of Computer Vision, OMR has been subjected to most of the same breakthroughs as its parent field during the last decade. Deep Learning methods have shown to be a very powerful tool, thanks to which the state of the art for music recognition has improved substantially. Nevertheless, the incorporation of these methods in the OMR toolset has also brought a considerable divide within the community as a result of the many new perspectives from which to tackle the problem. Many promising approaches exist for specific scenarios, but there is no single method of addressing music recognition that accomodates all use cases. As a result of this, no such thing as a single commonly accepted framework of evaluation of OMR systems exists.

In this work we want to tackle some of OMR's shortcomings by proposing a new way of recognising scores, in which we can sidestep most of the current limitations in the field while using as many available resources as possible. We propose doing so by exploiting current sequence-based end-to-end models, but using a more expressive tree-like representation, with

Author: Pau Torras Coloma, ptorras@cvc.uab.cat
Advisor 1: Alicia Fornés, Computer Science Department, Universitat Autònoma de Barcelona
Advisor 2: Sanket Biswas, Computer Science Department, Universitat Autònoma de Barcelona
Thesis dissertation submitted: September 2022

which we skip the problem of notation reconstruction, we provide intrinsic induction biases to music semantics thanks to the increasing abstraction level, which is reminiscent of the idea of summarisation in NLP, and we enable the use of common music engraving formats as ground truth.

We now present the key insights of our proposal:

- **Address OMR end-to-end; no intermediate representations.**

We need to produce a recognition model that is able to produce a fully fledged reconstruction of the score directly. This is because:

(a) There are many more scores available for engraving purposes than there are for Deep Learning-specific tasks – e.g. object detection. The latter are also usually costlier to produce.
(b) By having a model learn an output format directly we avoid having to use heuristics, grammars or a second model for notation reconstruction, reducing the scope of mistakes that come as a result of it.

In particular, we want to be able to generate scores in common and established, as their ubiquity can be exploited to collect training data at little to no cost. The best candidates are the Music Encoding Initiative (MEI) format and MusicXML, which are XML-based formats designed to contain engrave-ready scores. Since the latter is used in some of the biggest free access repositories of music while also being supported in many engraving programs, we believe it is the safest starting point.

- **The format has to be expressive enough for most scores, yet simple enough for models to grasp it.**

Scores are probably best represented as a graph where symbols or primitives are nodes and edges represent semantic relationships between them. Nevertheless, there are practical concerns in their application for OMR, as the literature on image-to-graph models is, to the best of our knowledge, quite immature yet. On the other hand, image-to-sequence approaches are thriving not only on OMR but also in related fields such as Optical Character Recognition (OCR). Therefore, we propose the output to be a middle ground between both in order to exploit their strengths at the same time. This middle ground is representing music as a Tree. The rationale is the following:

(a) We can exploit mature image-to-sequence models already available in the literature because trees can be represented as a 1D sequence – to portray a simple example, one can express a tree in lisp-like form by adding parentheses delimiting levels.
(b) Relationships between musical objects can be modelled using two dimensions: sequence order within the same level for ordinal links and parent-child connections for abstraction (n primitives form an instance of a higher order object).
(c) Existing music engraving formats such as MusicXML are tree-based by nature, which helps ease the conversion step.
(d) The fact that at lower levels in the tree there are concrete visual elements and higher up lie the more abstract constructs might also be helpful for recognition, as well as being something that can be exploited in recognition architectures.

- **This method can be used for any use case scenario for which a transcript is available.**

An end goal for us is to be able to study historical or handwritten sources without being as heavily bound by data availability. This simplifies the problem of producing training data for these rarer use cases to an alignment one. The latter offers the advantage that it requires a much lower level of recognition detail than fully-fledged OMR would otherwise, with the possibility of using human knowledge heuristics for decent results.

Moreover, since methods of many different natures can target this notation format, it can also act as a *lingua franca* among OMR researchers to develop a much needed common evaluation framework. Other upsides of interpreting the output as either a sequence or a tree are that evaluation metrics that apply to both approaches are usable.

- **Since raw data is abundant, self-supervision methods are possible.**

One final remark is that many image-to-sequence tasks can benefit from pre-training in unsupervised scenarios [1]. Therefore, even if there is a limited supply of labelled data, there are chances of providing robust solutions to some of the more niche OMR use cases. Moreover, these pre-training methods can be a boost of performance in any other context.

With the ideas seen above, in this work we develop a tree-based notation format for end-to-end recognition of scores using Deep Learning Models. The format, which we have named (rather unimaginatively) Music Tree Notation (MTN), uses MusicXML files as origin. We create a dataset based on this notation and we test its suitability for recognition using two image-to-sequence architectures: the Attention-based Sequence-to-Sequence (Seq2Seq) model, which is strongly established in the OMR community, and the Transformer, a very promising architecture which is still very much untested in the field. We also attempt self-supervision in order to see its effect on Transformer-based models' performance.

This document is structured as follows. In Section II the field of OMR, its current trends and the current leading models and datasets are presented. In Section III a detailed explanation of MTN and the pipeline to produce the dataset is offered. Section IV is a detailed introduction to the models that are to be used to test the approach. Section V shows the rationale and design of experiments in order to assess the quality of MTN, whereas Section VI shows the results for each experiment. An assessment on the degree of accomplishment and the quality of the overall results is provided in Section VII, along with some closing words.

II. State of the art

"Without work, which is art, there is nothing."
Gabriel Fauré

A. Optical Music Recognition

OMR is a field of research that investigates how to computationally read music notation in documents [2]. Essentially, given a picture or a scan of a musical score, either handwritten or typeset, an OMR system aims to produce a symbolic representation of the said score that can be processed by a computer. The level of detail of such a representation very much depends on the intended application, but most common output formats span MusicXML or MEI when a full reconstruction of the score is required (that is, one would want to replicate the exact same input score) or MIDI when playback is the final objective. This definition is focused on the *offline* recognition scenario, the main focus of this work, but there are other works which tackle *online* OMR – recognition of scores where the input is the temporal sequence of strokes to produce them.

OMR is a field which is very closely related to others in the Computer Vision community. In particular, it is interesting to draw a parallel to Optical Character Recognition (OCR), which attempts to produce a symbolic representation of textual inputs. Nevertheless, some unique properties of music set both fields apart quite considerably:

- Music, unlike most widely used alphabets for natural language, is bidimensional by nature, as it encodes pitch and time of musical elements. As a result, the relative position of objects in both axes is relevant for successful recognition. Moreover, music can have multiple voices playing in parallel on the same staff, further complicating recognition.
- The syntax of music is less strict; scores may be engraved in many different equivalent ways, of which only a few are preferable mostly for readability reasons.
- Music is *not* a language in the proper sense of the word, as it does not form a proper system of signs [3]. There are no underlying common semantics in music, even if there are some reasonable parallelisms to be made considering elements such as motifs, phrases and the like. Context-based inference is therefore harder.

There are some other parallels to be made with Document Analysis and Understanding, as the interaction of objects within the score is reminiscent to regular document layout analysis as well as there being a motivation for layout analysis in music in the form of separation of staves from lyrics. Bosch-Campos *et al.* [4] delve into the matter by using established OMR architectures for layout recognition and classification.

In terms of the types of data that are currently under study in the field of OMR, two broad categories of documents are to be considered: typeset scores, both scanned and computer-generated, and handwritten scores. The former case is quite mature with low error rates, especially for MIDI-like outputs [5]. There are many available datasets for this sub-task: the Deepscores dataset [6], [7], with object-level annotations but no fully-reconstructive nor playback baselines, the DoReMi dataset [8], which provides examples in most available formats – MEI, MusicXML and MUSCIMA++ –, the PriMUS dataset [9], with output sequences as target and plenty of websites and repositories devoted to music – e.g. MuseScore or the IMSLP project – with limited amounts of accurately labelled data. Synthetically generated music is also used broadly [10].

Fig. 1. Example page from Bach's original manuscript of the Brandenburg Concerto. Many of the common paper artifacts and typical handwriting irregularities can be seen: ink stains, paper degradation, irregular symbols, etc.

Handwritten music, on the other hand, and especially when stored in historical documents, is a much harder task (see Figure 1 for an example). Aside from the irregular nature of handwritten scores and the possible degradation and artifacts present in the paper, there is a severe lack of properly annotated data to address the problem. The CVC-MUSCIMA [11] and the MUSCIMA++ [12] datasets are two of the most prominent for current western notation, the latter being fully annotated at a symbol level. For sequence-like outputs in historical scores, the Pau Llinás Dataset [10] is another small example. The IMSLP project and several other libraries throughout Europe contain some digitised documents, but they are generally not transcribed, which greatly limits their utility. Other datasets are aimed at notation formats other than western music notation such as mensural notation: the Capitan [13] or the SEILS [14] datasets.A comparison of the difficulty leap between typeset and handwritten scores may be found in [15].

Prior to the 2011 Deep Learning revolution, most OMR practitioners focused on incremental pipelines in which each step was independent from the rest. These pipelines usually follow a prototypical structure, as described in Rebelo *et al.*'s review on the field [16]:

- **Image Preprocessing:** This step includes standard document analysis preprocessing steps such as document binarisation, de-warping, de-noising and the like, but also contains some music-specific steps. Most notably, the step that has probably received most attention in the OMR literature is staff removal [17]–[20]. The importance of this step used to be paramount, as staff lines made detecting other musical objects on the page significantly harder [16], [21]. Other steps such as binarization have been investigated for specific types of documents such as handwritten old scores [22], since in this use case the documents are usually in a very rough state and a substantial amount of noise is propagated downstream in the pipeline, reducing the overall performance.
- **Recognition:** This step involves segmenting and classifying the musical symbols on the page. This is usually implemented with standard classifiers and can be done either at a primitive level [23], [24], which implies classifying the components of the musical objects separately (stems, noteheads, dots, etc.) and then joining them through heuristics into higher level constructs, or at a full object level, which uses a classifier with the entire connected component in question [25].
- **Reconstruction:** This step involves building the music semantics from the recognised objects. This step considers the relative position of elements in the page and interprets the relationships between all of them. Approaches such as Coüasnon *et al.*'s DMOS [26] provide ways to formalise these position-wise relationships in the form of "graphical" grammars. Other attempts use hard-coded heuristics or try to build a sequential output such that a regular grammar can be used to build the final notation [27], [28].
- **Musical Model:** Once the relevant relationships between objects are defined, the final step is to construct a model of the full score that can be used to rebuild or process it computationally. The target output formats are most usually MEI, MusicXML or MIDI, depending on the downstream application. As aforementioned, using grammars is one of the preferred methods to construct these models.

With the revolution in the fields of Artificial Intelligence and Computer Vision birthed on 2011 with the introduction of AlexNet [29], the OMR community followed some of the new coming trends and started implementing parts of this pipeline using Neural Networks. Steps such as music symbol segmentation have employed Deep Object Detectors such as U-nets [30], [31], Faster-RCNN [7], [32], Deep Watershed [7], Fully Convolutional Networks [33], [34] or Single-Shot Detectors [33], [35], to name a few.

Nevertheless, recent trends are straying away more and more from this path, as with the advent of very powerful Deep Learning models during the latest decade it has been found that it is possible to use end-to-end approaches for OMR, in particular for historical handwritten scenarios. The main motivation is that, for these situations where no object-level segmentation is available, the model is able to infer the structure of the music primitives using only the ground truth transcription.

One of the first models to use this approach was borrowed from the OCR community (which is a frequent trend in OMR as both fields do have quite some overlap even if music is indeed a more complicated notation system). This model, pioneered in OMR by Calvo-Zaragoza *et al.* [36], consists on a Convolutional Neural Network (CNN) followed by a bidirectional Recurrent Neural Network (RNN) block implemented using Long Short-Term Memory units [37] that outputs a per-primitive representation of music in the form of a sequence. In particular, for every feature column vector in the convolutional output, a single output token is produced. The model is trained using the Connectionist Temporal Classification (CTC) loss [38], which merges any duplicate predictions that belong to the same symbol and are placed sequentially. This method reduces the OMR problem to a column-wise classification one essentially. This method was successfully used on handwritten scores in mensural notation and also for MIDI-like outputs [39].

In the same line, there is the Sequence to Sequence (Seq2Seq) family of models, which are closely related to CTC ones. Baró *et al.* [10] propose an attention-based Seq2Seq model based on [40] for OMR that is suitable for both typeset and handwritten scores in Western Music Notation. The model consists on a CNN feature extractor followed by an RNN-based encoder-decoder such that the input for each time step of the decoder is produced by a weighted sum of the encoder's output, obtained using the aforementioned attention mechanism. Music is yet again modelled as a sequence of arbitrary length by defining special column separation tokens and a reading order. A follow-up work which improves results on handwritten scores is [41], which incorporates a Language Model in order to introduce a grammar-only bias to the model which balances visual ambiguities.

In all cases in which the output is modeled as a sequence by defining a reading order there are strong limitations. For instance, reconstructing a polyphonic score unambiguously is impossible, because there is no telling of what sets of primitives belong to what voicings from the notation alone. The definition of the reading order is also ambiguous because top-to-bottom left-to-right relationships can only be safely assumed in strictly homophonic scores. Moreover, the notation reconstruction step is usually not implemented.

B. Transformers

The Transformer [42] is a model which was birthed in the context of Natural Language Processing (NLP) for translation tasks. It is the logical evolution from the Seq2Seq family of models, which have plenty of practical shortcomings:

- The existence of a recurring step makes inference inherently slow and computationally expensive, especially for very long sequences.
- Long sequences require recurrent units to keep "memory" of earlier elements for a large amount of steps, which in practice is difficult.

Transformers address these two problems by encoding the input sequence in a single step using *self-attention*. Broadly speaking, the representation of each element in the sequence is enhanced with successive linear combinations of projections of all elements in the sequence, such that a representation of the full context of the input is produced. The decoder is then tasked with producing tokens related to the entire encoded input and the previous predictions, therefore the model can be trained using a cross-entropy loss.

The model was highly successful, which spawned plenty of variations of the same concept (e.g. the Conformer [43], a Transformer that employs both convolutional and self-attention layers) and inspired applications in fields other than NLP, such as vision. The most widely known case of the latter is the Vision Transformer (ViT) [44], in which images are divided in a set of patches, each being projected into a common space using a feedforward layer, and then using them as the input sequence. This model also uses the extra token found in BERT-like models [45], a technique which was found highly effective for pre-training transformer models. Interesting variations of this model include the Cross-ViT [46] and the Swin Transformer [47], which incorporate multi-scale feature extraction improvements to the model.

Another relevant variation of the transformer for the task at hand is the Detection Transformer (DETR) [48], which tackles the problem of object detection using Transformers by incorporating a series of learnt object queries into the decoder and relying on cross-attention between these object queries and the encoded input to generate object predictions. The encoder may use any CNN as a previous feature extractor and then processes the resulting feature vectors. A bipartite graph matching algorithm is required to match predictions with the ground truth, which makes this model computationally expensive.

To the best of our knowledge, very few works [49], [50] thus far have attempted to work in OMR using Transformers, with rather underwhelming results. In the first one, the authors used a DETR to produce an OMR system based on object detection, but the intrinsic limitations from the model in the number of objects in the input image seem to have limited their results. The latter uses various combinations of Transformer modules, CNNs and CTC-based decoders on various datasets – PriMUS [9] (typeset), Capitan [13] (Handwritten), Fondo de Música Tradicional [1] (Handwritten) and SEILS [14] (Scan Typeset) – with regular 1D sequences as target.

Other fields which are interesting and related to OMR with relation to the output formats that are tackled (sequences or trees) are Document Understanding – in particular, the DoNUT model [51], which produces a JSON from an image of a document – Code Generation and Translation, in which Abstract Syntax Trees are used either as inputs or outputs [52], [53] or Natural Language Processing.

C. Self-Supervision

As models become more and more data hungry, the problem of having sufficient annotated data for successful convergence deepens. With this concern in mind, techniques have been developed in order to train models without requiring annotation by developing pretext tasks that only use the raw input data, a concept that is called Self-Supervision. With these pretext tasks models are forced to learn useful feature representations of the input data, thanks to which developing models for downstream tasks becomes both easier and less data-reliant.

Initial efforts in this direction consisted on the application of autoencoders [54] on noisy versions of images with the pretext task of de-noising them back to their original state. As Deep Learning models matured, more intricate techniques were developed, such as image colorisation [55], patch ordering [56] or weak classification [57] – learn to distinguish versions of the same image from other images.

With the recent surge in the use of Transformers, specific pre-training techniques have been devised for them. Masked Autoencoders [1] are a pre-training task in which a Transformer receives an image with 75% of input patches masked out and is set to reconstruct the original image. Variations on this idea include BEiT [58], in which instead of reproducing the input image the model is tasked with predicting a token that corresponds to each masked patch.

[1] https://musicatradicional.eu/

III. Proposed Datasets

"I am delighted to add another unplayable work to the repertoire."
Arnold Schönberg

A. The Music Tree Notation format

The basis for our approach to OMR as an end-to-end image-to-sequence task is our authored MTN, which represents music as an Abstract Syntax Tree (AST) that can be expressed a text sequence. Figure 2 shows the division of score elements into primitives and how from these primitives the notation format is constructed. It is designed to be closely related to MusicXML because of its ubiquity and its inherent dissociation between the visual representation of a given score and its playback. It is therefore possible to produce a "music-agnostic" notation format in which semantics – e.g. musical pitch or duration – are extracted by relationships of graphical primitives instead of requiring the model to learn them explicitly. We work under the hypothesis that the underlying graphical language in musical scores, based on proximity and contact of well-defined primitives, is better suited for recognition than high level musical concepts. This also has practical advantages due to the fact that no musical context of elements such as key or time are required for recognition, allowing the reconstruction of the final representation by reading self-contained subsets of the score.

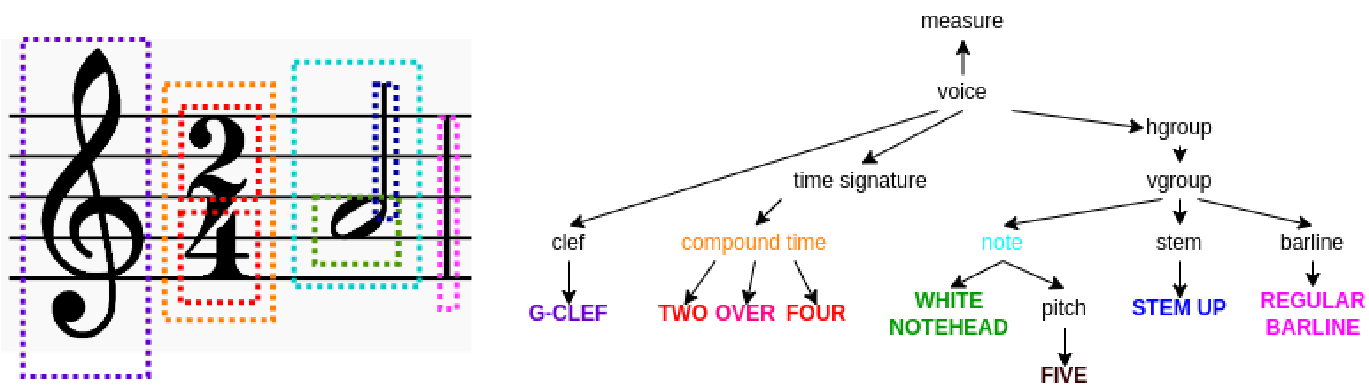

Fig. 2. Example showing the overall organisation of the dataset. Given a score, it shows the division into primitives and how these primitives relate into building the notation for a single measure.

Another key aspect of this system's design is the fact that a sequential model shall build the final notation tree in bottom-up fashion; starting from the leaves (which we interpret as terminals of a generative grammar) up to its root, navigating all intermediate nodes (which we interpret as non-terminals of a generative grammar). In other words, we expect the model to develop an internal grammar to be able to produce an AST from the graphical terminals it detects. This effectively sidesteps one of the most complex aspects of working with OMR as a whole, which is inferring the relationship between the various detected primitives to produce a playable score.

The core element of this notation format is the *musical primitive*, which is a terminal in the output AST. The set of musical primitives includes all graphical elements in a score that are self-contained and require no other symbols to convey meaning (this includes rests, clefs or time signatures), the set of letters and numbers in order to spell annotations and tempo indications and the set of graphical elements that compose notes (noteheads, stems, flags, dots, etc.).

The maximal element that our notation format is designed to represent is the musical measure at the single-staff level. The ideal scenario would be to predict entire lines, as they are almost always completely context-agnostic in properly engraved scores – all of the ongoing elements such as key, time signature and clef are repeated at the start of a line, with the only possible context-sensitive elements to account for being ties or slurs ongoing from an earlier line. Nevertheless, this poses some technical problems for transcription models since a significant amount of image detail is lost by having to include a larger slice of the score into a fixed-size model, while also having to predict much lengthier output sequences. Predicting measures solves this at the expense of requiring more complex pre and post-processing of scores; the input image needs to be cropped into measures for inference and the output notation is to be produced from combining and interpreting the outputs of many images.

The sequence of elements that represents the AST is built by reading the leftmost primitive in the score from the main voice (that is, the set of musical elements that spans the entire bar horizontally and amounts to the time signature number of beats) and then reading onward vertically. All elements pertaining to the same vertical position and voice will be grouped into a `vgroup` (vertical group) non-terminal, whereas all primitives that form a connected subgraph horizontally will be added

TABLE I
SUMMARY OF THE WORKS USED AS DATASETS AND SOME GLOBAL STATISTICS.

Name	Moniker	Composer	Samples	Length	Source
Brandenburg Concerto No. 3	Brandenburg	Bach	2014	102.57 ± 48.64	OpenScore
The Art of the Fugue	Fugue	Bach	6532	51.38 ± 30.26	OpenScore
9th Symphony	9th	Beethoven	44003	39.53 ± 27.69	OpenScore
Jupiter Symphony	Jupiter	Mozart	10275	35.01 ± 23.44	OpenScore

into an `hgroup` (horizontal group) non-terminal. In order to resolve ambiguities of elements pertaining to the same vertical element but to different objects within it, other non-terminals are added for notes, stems or any terminals that can appear repeated under the same object (such as dots or beams). Earlier drafts of the format included parentheses as tokens in order to group child elements in lisp-like fashion, but this was dropped due to the high level of context awareness needed to produce these parentheses correctly, as well as the extreme unbalance in the dataset they produced due to their ubiquity.

In order to represent elements that sound in unison but are not graphically tied to the main voicing, additional voice objects can be created to represent them. At the current specification of the format however there is no way to horizontally align the secondary voicings to the main one, as the assumption of beat adding up to the last known time signature value can be broken for these secondary voicings. Similarly, with double-staff pieces, the main voicing can move along two separate staves, which means measures read in staves separately may have an incomplete amount of beats. We solve this by using single-staff works for our experiments.

Notes are a special primitive in that its vertical placement in the score matters, while also having many symbols attached to them. Therefore they are modeled using a “note” non-terminal that includes a notehead, a pitch expressed as the number of positions to move from the first ledger line below the staff and optional elements such as accidentals, ties or slurs that depend on them. The pitch of the note is expressed using a non-terminal “pitch” with each individual numeral character terminal as children.

In order to represent slurs, ties, glissandos, crescendos, diminuendos and other multiple-note-spanning elements, starting and ending elements are used to indicate their origin and end.

B. Generation of Training Data

Randomly generated synthetic scores are a good way of providing extra data to work with. However, for the task at hand it may not be the cleanest solution, as the intended output is in the form of a sequence and conditional probabilities between elements of said sequence matter. Therefore, we assume it is better to provide musically plausible training examples, as the model should be better able to identify musically plausible sequences. For instance, there are certain accidentals that are more likely to appear under certain keys than others – having a sharp 7th degree in a minor scale is an indication of a dominant major chord, which is a fairly common aesthetic resource.

For the experiments in this document we decided the best road was to use transcriptions in real scores produced in MusicXML. To the best of our knowledge, the only official dataset with such transcriptions is DoReMi, which we initially considered using. However, there are a few practical problems about it. The first one is that it does not have measure-level images, but rather line level ones, which implies there is a need for automation on the measure cutting. We implemented it exploiting the primitive-level annotations within the scores, but found it to be sometimes inconsistent with certain primitives such as double barlines. Furthermore, there were problems with some lines being split in multiple images, which made guaranteeing alignment of score and representation impractical overall.

Instead, we chose a catalogue of works from a similar period (late baroque and early romanticism, 18th century music). The idea is that these works have plenty of commonalities among themselves, they are in the public domain and they are relatively simple, in the sense that they do not contain too many virtuosistic movements nor highly elaborate playing indications. This solves another downside of using the DoReMi dataset for this task, which is that the catalogue of works within is of extreme complexity, which makes it a rather poor choice for a first proof of concept.

In Table I some generic statistics about number of images and sequence length may be found. The column “moniker” refers to the shortened name that shall be used to refer to each dataset in due context from here onward. In Figure 3 a histogram shows the number of samples within a certain length range for each dataset. Detailed numbers for each range and dataset are provided in Table II.

In order to produce the images for the dataset, a pipeline using various programs is orchestrated through a simple bash script. The overview of this pipeline may be found in Figure 4.

The main problem to overcome for dataset generation is that most pieces of software we are acquainted with do not produce measure level rasters of scores by default. Instead, we found Verovio to be largely compatible with MusicXML outside the box, even when it is designed primarily to engrave MEI scores. Moreover, its output format is SVG, which is based on XML and therefore easily and quickly modifiable without requiring image processing. This greatly simplifies the generation of the

TABLE II
COUNTS FOR EACH RANGE OF LENGTHS IN EACH DATASET.

	$<$ 25	$<$ 50	$<$ 75	$<$ 100	$<$ 125	$<$ 150	$<$ 175	$<$ 200	$<$ 225	$<$ 250	$<$ 275	$<$ 300	$<$ 325	$<$ 350	$<$ 375	$<$ 400
Brandenburg	54	174	288	571	442	155	173	62	10	72	11	2	0	0	0	0
Fugue	1179	2521	2125	1525	587	209	209	74	11	78	13	3	10	2	0	0
Jupiter	4676	6821	3422	2427	721	274	228	114	13	78	25	10	10	2	0	0
9th	20538	27204	8684	3948	1451	345	326	152	19	84	40	13	12	5	1	2

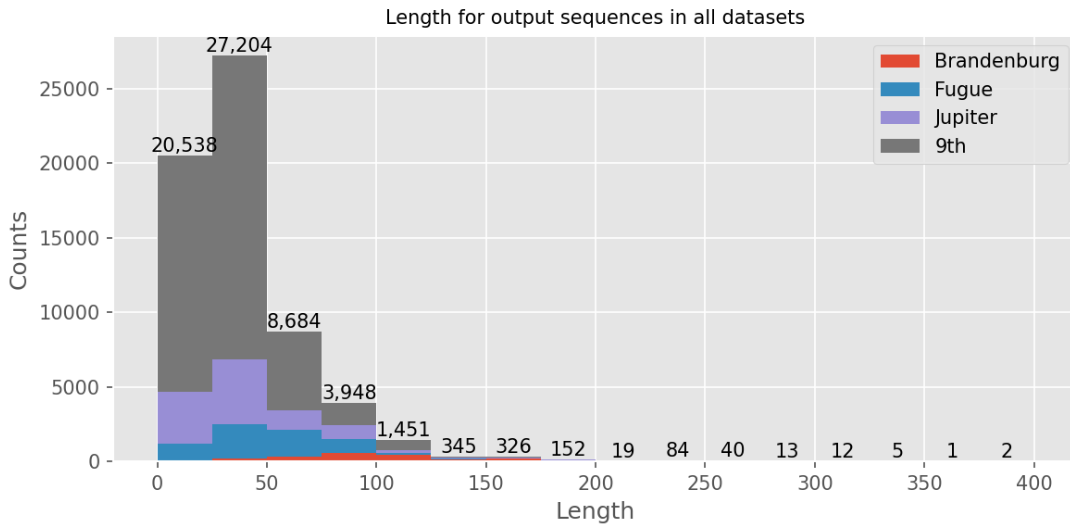

Fig. 3. Histogram with the number of samples in each dataset within a given length range in the X axis. The total sum of samples within each range is seen above each column.

dataset because some of the engraver's automated features (such as adding key, clef and tempo at the beginning whether or not it is actually in the MusicXML file) can be overridden by modifying an XML file, where identifying elements can be done by name and not by visual features.

The full pipeline works as follows. From the desired input MusicXML file, a Python script produces separate MusicXML files for each measure of the score. Since the running state of the score – alterations, clef and time signature – must be kept for later measures to be consistent, in all but the first measure an extra bar with the state of the score is added in the MusicXML file for Verovio to engrave the music properly.

Once these MusicXML files are produced, Verovio is used to generate an SVG file of the desired measure (and the aforementioned state measure before it). Using another Python script, the first measure of the image is found by exploiting XML identifiers in the SVG file. This first measure and all of its contents are removed from the image altogether.

The problem now is that the resulting SVG has a gap where the measure used to be, as all coordinates in the rest of the file are produced w.r.t. the top left corner as originally engraved. To solve this, the image is rasterised into PNG format using Inkscape and then the image is cropped to fit the contents using the convert tool from ImageMagick.

We shall now rationalise some of the decisions taken in the design of this pipeline. The decision of splitting the MusicXML file into smaller measure-level files came from the fact that Verovio is designed to engrave full pages, not single music measures. We could not find a reliable way of producing multiple-part scores measure-wise without having to deal with line breaks (which insert clefs and keys at the beginning which are not present in the notation), page breaks – Verovio renders each page separately, with similar issues as those from line breaks – and the other context-sensitive elements. The downsisde is that the system only produces one key, clef and time signature symbol at the beginning of each part, unless there are changes mid-piece.

The script that removes unneeded elements also performs some data-specific modifications. For "Fugue", the fifth part of the score is a two-staff element which also happens to be a reinterpretation in a piano-like form of the four preceding parts. Therefore, as we do not fully support multiple-staff scores yet and we do not want to introduce unwanted redundancy, we remove that specific part to avoid problems. Also, as many of the scores we use in this set of experiments are orchestral, which means there are extensive passages where some instruments will only have whole rests in their measures. As this would heavily unbalance the dataset and probably cause some unwanted overfitting effects, we stochastically remove these such measures with a probability of 95%.

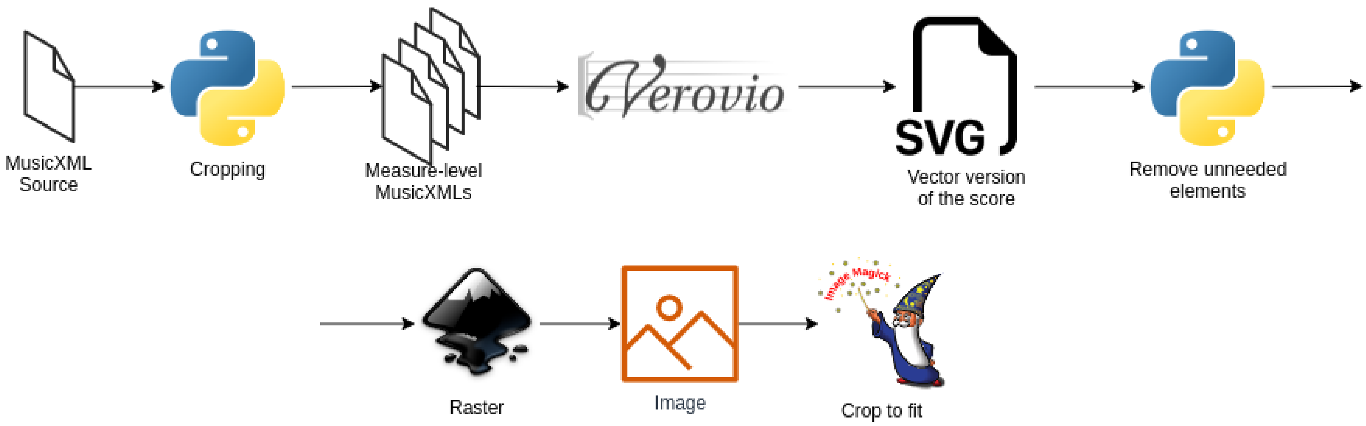

Fig. 4. Dataset generation pipeline. This only includes rastering the images using a score source in MusicXML. The notation underlying each measure can be produced independently.

IV. Models

"Essentially, all models are wrong, but some are useful."
George Box

A. Sequence to Sequence

This piece of research builds upon previous works in OMR that interpret music as an image-to-sequence translation task. Therefore, we implement an existing state-of-the-art model to see how they behave under our proposed notation format in order to establish a baseline.

The first model we shall introduce is the attention-based Seq2Seq model from [10], [40], [41], which is represented graphically in Figure 5. The core idea of this model is treating image transcription tasks as an image-to-text translation problem, which avoids the need of a fine-grained object-level dataset as the model can infer the relationships between objects in the image input and tokens in the sequence output. The downside is that there is less control in the inner workings of the model, as the single optimisation driving force in the model is the loss for the output tokens at the end, as well as the requirement for more intricate models capable of doing multiple tasks at once (feature extraction, representation learning, soft-alignment, etc).

This model is composed of three main components: a backbone CNN that extracts image visual features, an encoder that generates a global context-aware intermediate representation and a decoder that, given said representation, produces the output sequence.

The backbone CNN for this model is a VGG19 [59], a highly dense network that has proven to be quite performant along the years. It consists of 16 convolutional layers and 3 fully connected layers (which are discarded in this work alongside the last max pooling layer, as they are not needed). The output of this process is a $B \times C \times H/16 \times W/16$ tensor, where B is the size of the batch, C is the number of output channels (512 for the VGG19, as it is the depth of the last convolutional layer) and H and W are the height and width of the input images respectively. In order to interpret this output as a sequence, this tensor is reshaped into a $B \times W/16 \times (HC/16)$ one, where the width now represents its length and every vector along it represents the information of a vertical slice of the image.

The Encoder is a stack of N Gated Recurrent Units (GRU) [60], a variation of RNNs that tackles some of their intrinsic problems – vanishing/exploding gradients and fading memory. The goal for this encoder is to produce a context-aware hidden state from the visual features produced by the CNN. In other words, the goal is for the feature vectors in the hidden state to represent not only the contents of a certain area of the image, but rather incorporate information from all positions in order for the model to be able to construct higher-level semantic constructs and establish dependencies between objects throughout the input. Therefore, instead of using regular GRUs, which would produce an incrementally context-aware representation from left to right, bidirectional units are employed, so that for every position full-width context is available. The resulting tensor H is of shape $W \times D$, where D is the dimension of the GRU units – the tensors for both directions at each position are averaged.

The Decoder is the module that generates the actual output sequence. It is composed of a stack of unidirectional GRUs whose input is the result of an attention mechanism on the hidden vector. The final token output is produced from a linear layer atop the model with dimension N_{vocab}, after which a Softmax function is applied.

This model uses Bahdanau attention [61] as weighting function for the hidden state. This method leverages various sources of information within the model with the goal of obtaining an energy vector $\mathbf{e} \in \mathbb{R}^W$ such that $\sum_{i=1}^{W} \mathbf{e}_i = 1$. This energy

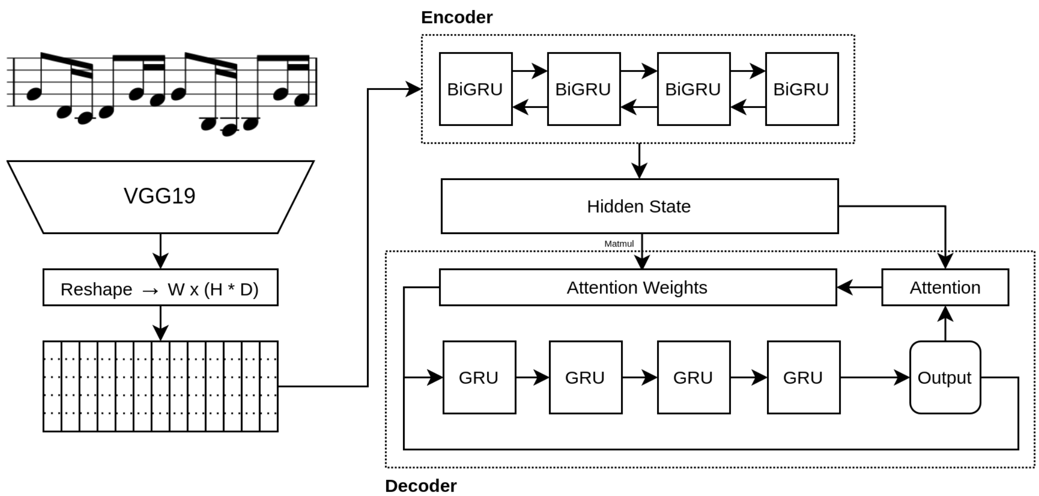

Fig. 5. Overview of the Seq2Seq model employed in our preliminary experiments.

vector is used to weight the relative importance of each vector in the hidden state for the current prediction step, computed as $\mathbf{h} = H^T \cdot \mathbf{e}$. In order to obtain the n-th value of $\mathbf{e}$ at inference step t, the model computes

$$\mathbf{e}_t n^* = \mathbf{w}^T \tanh\left(W_s \mathbf{s}_{t-1} + W_h \mathbf{h}_n + W_p \mathbf{f}_n + \mathbf{b}\right) \tag{1}$$

where W_s, W_h, W_p are parameter matrices, $\mathbf{w}$ and $\mathbf{b}$ are parameter vectors, $\mathbf{s}_{t-1}$ is the last hidden vector of the decoder at the previous inference step, $\mathbf{h}_n$ is the n-th vector in the global hidden state and f_n is the n-th vector of matrix F, a function of the previous attention weights computed as

$$F = Q * \mathbf{e}_{t-1} \tag{2}$$

where Q is another parameter matrix and $*$ denotes convolution. To enforce that the weights should add up to 1 then

$$\mathbf{e}_t n = \frac{\exp\left(\mathbf{e}_t^* n\right)}{\sum_{i=1}^{W} \exp\left(\mathbf{e}_t^* i\right)}. \tag{3}$$

The overall model is trained using cross-entropy loss on the final output of the decoder.

B. Transformers

RNN-based Seq2Seq models, as hinted in the state-of-the-art section, are quite troublesome to work with for practical reasons. The main and most relevant reason is the fact that they are recurrent and autoregressive, which makes them computationally ineffective for training: the dependence on the prior inference step makes these models rather unsuitable for parallel computations, as the data path limits the amount of compute to that of a single update which is usually lower than the compute capacity of a GPU.

The Transformer [42] is an encoder-decoder sequence-to-sequence model which tackles the shortcomings of RNN-based encoder-decoders by processing the input sequence in parallel relying on a self-attention mechanism exclusively. With regard to the overall architecture, Transformers maintain the Encoder - Hidden State - Decoder module structure, with the possibility of omitting the latter or extracting features with a separate CNN. The model we propose is a fairly standard iteration of this idea in which a Vision Transformer is used as the Encoder and the Decoder is a regular transformer decoder.

First, we shall describe global Transformer architecture elements, as both Encoders and Decoders are fairly symmetrical. These common elements are the scaled dot product attention operation, the self-attention layers and the input positional encoding.

The core element of the Transformer is the attention function, which produces a representation from matrices of query Q, key K and value V vectors. The scaled dot product operation is written as

$$F_{\text{Attention}}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V. \tag{4}$$

The softmax $\left(\frac{QK^T}{\sqrt{d_k}}\right)$ part of the operation is reminiscent of the computation of the $\mathbf{e}$ vector in the Bahdanau attention regime. Given a set of possible relevant positions within the vector (keys), one wants to find those that are relevant to the current inference step (queries). The dot product acts as a sort of "and" operator, a similarity metric between what is "searchable" and what is to be "searched". There are two further additions to the weight computation, which is the scaling by $\sqrt{d_k}$, the

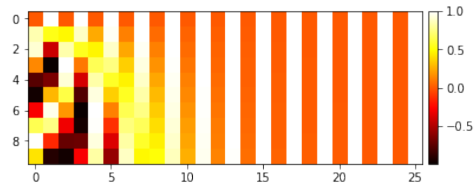

Fig. 6. Positional embedding mask example.

dimension of the key vectors, and the softmax function. The former is done to control the magnitude of the dot products, whereas the latter is applied in order to conform to the restriction that all attention weights in a vector should add up to 1.

Once the attention weights are obtained, they are multiplied by the set of value vectors to obtain the desired representation. Using once more the data retrieval abstraction, this would be analogous to extracting the all of the desired elements from a given database once the information from the structure of the database and the query performed against it are interpreted.

The matrices Q, K, and V are produced by projecting the desired input elements into a shared space using a linear transformation. For the Transformer encoder, these three matrices are projections of the same input sequence, which is called self-attention. The logic behind this operation is that the model successively generates attention-weighted versions of the same input sequence, which effectively produces highly context-aware representations. In the decoder self-attention is used alongside cross-attention, the latter being a variation in which the queries and keys are produced from the hidden vector at the end of the encoder and the values come from the sequence processed at the decoder.

The generic Transformer works the following way. Given an input sequence of length W, an embedding process generates a $W \times D$ matrix, where D is an arbitrary dimension number. Since all elements are to be processed in parallel, a mask is added to all vectors to encode the relative position of each element within the sequence for the model to be able to tell it apart. In the original paper, sine and cosine functions were used to emulate an intermittency effect (see Figure 6):

$$PE(p,d) = \begin{cases} \sin\left(p/10000^{2d/d_{\text{model}}}\right) & \text{if} d/2 = 0 \\ \cos\left(p/10000^{2d/d_{\text{model}}}\right) & \text{otherwise} \end{cases} \tag{5}$$

where d is the position along the embedding vector and p is the position in the sequence. The original paper remarks learnt positional embeddings produce similar results.

Each Transformer encoder layer is composed of multiple self-attention layers in parallel (self-attention heads), whose outputs are concatenated and projected back into the original dimension, and a simple feed-forward layer. There is a residual connection around both layers, combining both sources by addition and using layer normalisation [62] afterward.

Transformer decoder layers are the autoregressive part of the model. Their input is the sequence produced so far (properly embedded and positionally encoded), and need to be run as many times as elements in the output sequence. In practice, the input is as wide as the maximum output sequence, with elements not yet needed or produced masked by multiplication. This can be exploited to train in parallel all elements in the sequence by providing the ground truth sequence and masking all positions after the n-th inference step.

Functionally, the decoder is very similar to the encoder, with the main difference being an extra cross-attention layer after the regular self-attention one. This cross-attention layer uses the same attention mechanism described earlier, with queries and keys being projections of the encoded sequence and values being a projection of the previous layer in the decoder. It also has a residual layer around it and layer normalisation afterwards.

The final outputs are produced by linearly projecting the output logits for each element of the decoded sequence into a n_{tokens} dimension vector. Cross-entropy loss is used for training.

The Transformer model we propose for OMR is a fairly standard encoder-decoder in which the feature extractor is a Vision Transformer (ViT) [44] and the decoder is the original Transformer decoder with cross-attention and masked input sequence. The overall architecture can be seen in Figure 7.

The ViT is basically a regular Transformer encoder, with two key differences. The first one is the fact that it uses images as input instead of sentences. Images are fed into the Transformer by cropping them into patches and using a linear layer to embed them into a d-dimensional space. The second difference is the fact that the ViT is a BERT-like [45] model, which adds an extra input token which is used at the end once encoded for classification purposes. The same kind of 1D positional encoding is used, as in the original paper [44] it is stated they saw no significant changes using more elaborate encoding schemes. We use the `vit-pytorch` package [2] implementation.

C. Pre-Training

As the goal is to achieve good recognition of scores, it is rather important to make the most of the available data. Furthermore, as the models are trained with the output sequence as target only, initial convergence of visual features might either be slow or

[2]https://github.com/lucidrains/vit-pytorch

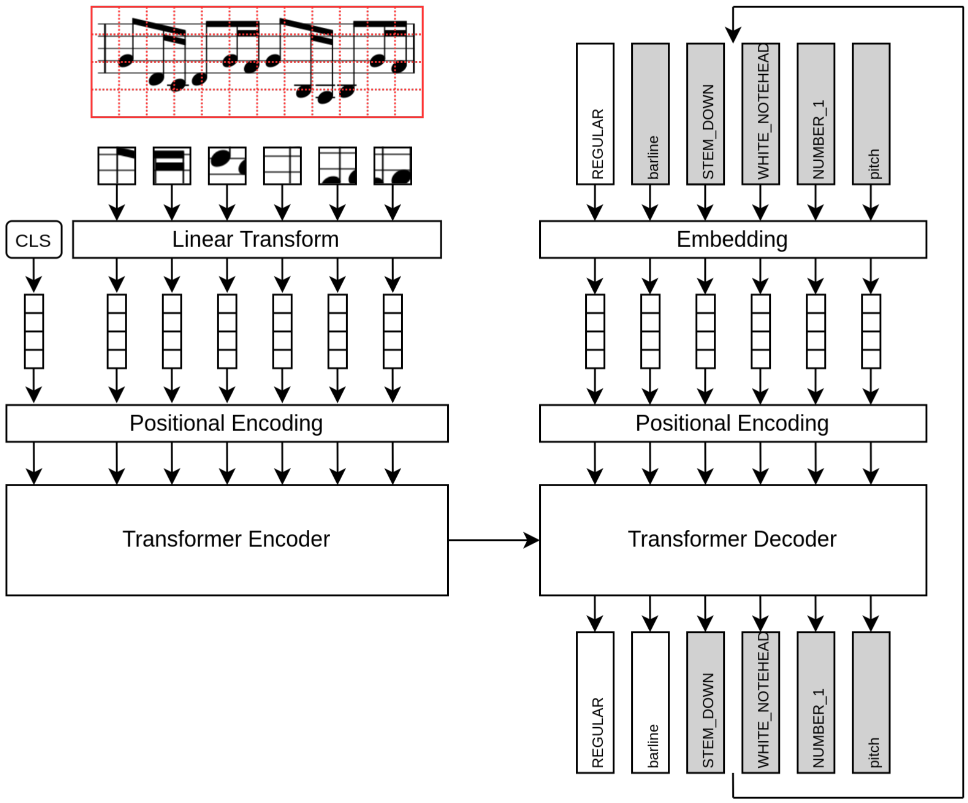

Fig. 7. Overview of the Transformer architecture proposed for OMR in this work.

downright ineffective. We explore the usage of some of the available OMR datasets in order to implement a robust pre-training strategy; in particular, we decided to use DeepScores v2's [7] large collection of typeset images for their variety, the different engraving fonts available outside the box and the sheer volume of scores packed within ($255,385$ images with 151 million objects).

The DeepScores v2 dataset is designed to perform OMR by tackling the problem as an object detection task at a page level, which is incompatible with our current design. We modify the dataset in order to obtain measure-level images. We do so by using the original annotations of stave objects to isolate music lines and by finding barlines in the score through mathematical morphology, as they are not part of the original repertoire of objects. The downside of this approach is we cannot guarantee inequivocally that we are cropping the entire dataset in a sensible manner, but our probing efforts demonstrated that failure cases were rather rare and harmless – we saw some cases of double end barlines being identified as a full measure by themselves, but their appearance was inconsistent. With this we produced $10,143,883$ non-annotated images.

The pre-training technique we use is the one proposed in [1]. They develop an image reconstruction proxy task in which the model has to generate the input image from a masked version of it. The full model consists of a ViT as an encoder and a lean Transformer decoder which will be scrapped once the model is pre-trained. The idea is to randomly mask 75% of the regular non-overlapping patches that form the input image; the ViT encoder receives the unmasked patches only, whereas the decoder receives a sequence of the same length as the initial number of patches with `masked` or `non-masked` tokens accordingly. From each element of the decoder's output sequence, a pixel-wise reconstruction of the input patch is produced. The model is trained to reduce the per-pixel Mean Squared Error.

D. Data Augmentation

Another relevant aspect of training these kinds of models is avoiding possible overfitting. A possible option to enrich the statistical variability of the input images without requiring more data is using data augmentation on the images. We propose a simple pipeline with the following transformations: A minor random affine transformation to change the angle between the elements of the image without changing whether or not the elements stay parallel and a random Kanungo noise function with $1/3$ probability of producing an aggressive augmentation, a normal augmentation or an identity function.

As the input is binarised and the full image is required for proper inference, no aggressive cropping nor color changes are used. Moreover, Kanungo noise is quite similar to salt and pepper noise while being more canonical with documents, which is why the former is not used either. Some examples of augmentations may be found in Figure 8.

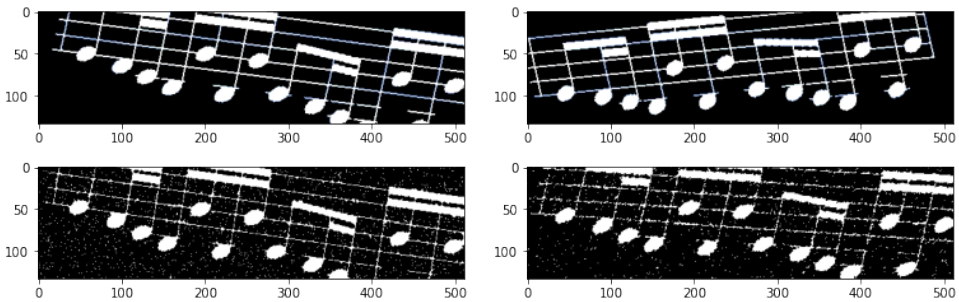

Fig. 8. Examples of the data augmentation pipeline, which is stocastic in nature so as to provide input image variety to the training examples.

V. Experiments

"The first principle is that you must not fool yourself, and you are the easiest person to fool."
Richard Feynman

In this section we shall describe the experiments we plan to perform in order to assess the quality of our proposed OMR method.

For most experiments, we use the combination of the Fugue, 9th and Jupiter datasets as the training partition and the Brandenburg dataset as the validation partition. The motivation is twofold: the training datasets are the largest available to us, while the Brandenburg dataset is well-aligned to all of the other ones in terms of the distribution of tokens. Having a smaller validation dataset also has the added benefit of reducing the time required for Transformers to do inference, a process which takes orders of magnitude longer to perform than regular training owing to the Transformers' autoregressive nature.

A. Exploratory search

We first run the model with diverse hyperparameter and layer configurations in order to understand their effect on the output's quality. For these experiments we use the full sequence length of the datasets.

B. Sequence to Sequence

A first experiment to build upon the previous state-of-the-art and motivate the move to transformers is using RNN-based Seq2Seq models. We experiment using our new notation format using old models and assess their results with relation to those experiments with transformers.

C. Transformer Pre-Training

We assess whether using a pre-trained ViT feature extractor improves recognition performance. The parameters that we fixed for the encoder are standard, with the input image size being a slight adjustment on the default ViT parameters in order to adjust to the aspect ratio of measures while keeping the number of input patches equal. The parameters are shown on Table III. We also assert whether it is better to keep the encoder weights frozen while training the full model.

TABLE III
Hyperparameters for all encoders in the experiments.

Layers	6	**Heads**	8	**Input Size**	512×128	**Patch Size**	16

D. Loss Weighting

The datasets we are using are quite unbalanced in terms of frequency of appearance of certain tokens. We design experiments to test whether adding a weight factor into the loss changes either the training process or the overall result to be better in any way. We attempt using the straightforward interpretation of each token prediction as an independent random variable, with the goal of forcing the expected value of the loss to be equivalent to randomly guessing between n_{tokens} elements. Therefore, given the cross-entropy loss function

$$\ell = -\sum_{t=1}^{N} k_t \log p(x_t) \tag{6}$$

where k_t is a weighting factor, $p(x_t)$ is the probability distribution of the output tokens as provided by the model and N is the size of the vocabulary, we want to force the expected value of this loss to equal

$$\mathbb{E}\left[\ell\right] = \frac{N-1}{N}. \tag{7}$$

The way to compute this is by setting weights $k_t = \frac{\max p(x)}{p(x=t)}$ and normalising back to the range of $[0...1]$. The weights therefore oscillate between 1.0 and roughly $1.0 \cdot 10^{-5}$. For supporting tokens such as end of sequence, we keep the 1.0 weight value due to their importance.

E. Changing the output length

As can be seen in Figure 3, the lengths of output sequences reach up to the 400 mark. Nevertheless, the overall length distribution is rather long tailed, with sequences of less than 125 elements representing more than 95% of the samples. Lengthy sequences are known to be an issue with most Seq2Seq and Transformer models [63], and thus we provide experiments with reduced length versions of all datasets.

Since the model should produce all instances of the validation dataset, we perform the same length reduction to both the training and the validation datasets, as downward changes in the performance could be justified by the model's inability to produce lengthier outputs. This of course causes the problem of not being able to directly compare results between runs of differing length, but does provide valuable insights on models trained for same-length datasets or in order to study outputs qualitatively.

F. Cross-Validation

In order to ensure the choice of the dataset partitions is not forcing a wrong picture of the model's capacity to recognise scores, we cross-validate the model by alternating the validation dataset between the Brandenburg, Fugue and Jupiter works. We avoid using the 9th symphony for this purpose as this work comprises roughly 40k images, depriving the model of the majority of the available training data and needlessly extending validation runs.

VI. Results

"Forty-Two"
Deep Thought

We shall now present the results of the experiments we conducted to test our proposed MTN notation and the two discussed architectures.

A. Evaluation

The Symbol Error Rate (SER) metric is used to evaluate the performance of the models, shown in tables in percentual points. The SER is a metric that summarises the number of edits (substitutions, insertions or deletions) of tokens required to obtain the ground truth sequence from the predicted sequence. The value is then normalised using the length of the output sequence. In mathematical notation,

$$SER_{\%}(\hat{y}, y) = \frac{I, R, S}{\text{length}(y)} \cdot 100, \tag{8}$$

where I, R, S are the aforementioned insertions, removals and substitutions obtained from the optimal edit path generated by Levenshtein's algorithm [64] and $\hat{y}$, y are the predicted and ground truth sequences respectively.

B. Sequence to Sequence

For Seq2Seq models we found ourselves unable to make the model converge into anything sensible with full-length sequences, as can be seen in Table IV. In all cases the model converges into a trivial solution state in which the produced sequence is always the same (see Figure 9). We suspect this is due to both the average length of the training samples and the relative presence of `vgroup` tokens – 3 per image on average. The other explanation might be the length of the sequences being too high, with updates for each time step smoothing out and vanishing when backpropagating towards the first elements of the sequence.

TABLE IV
RESULTS OF RUNNING THE SEQ2SEQ MODEL WITH FULL LENGTH DATASETS. COLUMNS FROM LEFT TO RIGHT: THE AMOUNT OF LABEL SMOOTHING, DROPOUT AND WHETHER TO WARM UP THE LEARNING RATE AS HYPERPARAMETERS, AND THEN THE SYMBOL ERROR RATE IN PERCENTUAL POINTS AND LOSS FOR BOTH TRAINING AND VALIDATION. USING BRANDENBURG FOR VALIDATION AND THE OTHER DATASETS AS TRAINING.

Label Smoothing	Dropout	Warmup	Learning Rate	Train. SER(%)	Valid. SER(%)	Loss (Train.)	Loss (Valid.)
0.1	0.25	$False$	$5.0e-06$	196.5 ± 132.5	97.69 ± 26.95	3.111	3.037
0.05	0.1	$False$	$1.5e-05$	196.5 ± 132.5	97.69 ± 26.95	3.236	3.09
0.1	0.5	$False$	$1.0e-05$	196.5 ± 132.5	97.69 ± 26.95	3.058	3.035
0.1	0.25	$True$	$3.0e-04$	196.5 ± 132.5	97.69 ± 26.95	3.073	3.032

Prediction

vgroup vgroup

Fig. 9. Output from a validation sample of a Seq2Seq experiment.

C. Encoder Pre-Training

Due to the volume of data required and the limited amount of time and resources we had, we could only train one instance of the ViT, which took around two weeks for a total of 17 epochs with a final pixel average MSE of roughly 12%.

Figure 10 shows an example of the pre-training mechanism in one of the samples from the DeepScores 2 dataset. This example is particularly tricky because a part of staff below overlaps with the current measure. Moreover, there is a very high amount of objects of highly differing nature, a considerable amount of which that are almost completely masked away. The reconstruction the model makes is extremely context-aware, as can be inferred from the insertion of dots in all notes of the chord in the central part of the image or the reconstruction of the G clef with just two curved segments. There is an image smoothing effect which comes as a result of using a quadratic loss and having uncertainty in some parts of the image: the key is almost completely masked away, with no real way of foretelling the position of the remaining alterations aside from using the relative position of the sharps on the second staff.

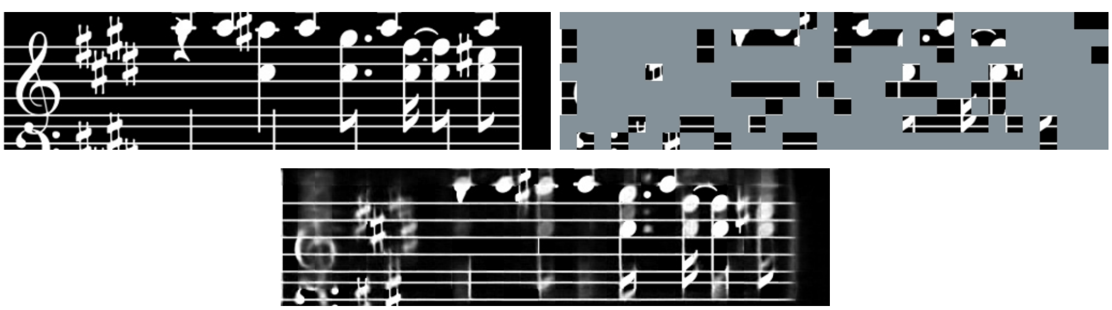

Fig. 10. Example of the pre-training procedure. Top-left is the clean input image as given to the model, top-right is the same image after masking randomly 75% of the images' patches and at the bottom the predicted reconstruction of the model

D. Preliminary Parameter Search

Table V shows all long-running experiments in which we tried to find the best hyperparameter combination for training. In these experiments we used a pre-trained unfrozen decoder to hasten the training process. As the first row shows, we quickly identified some very clear signs of overfitting when training the model from scratch, as both the validation loss and symbol error rate skyrocket when not using any kind of label smoothing or dropout. In short-running experiments (less than 1h) we also identified convergence issues when using too large learning rates, hence our use of either warmup for the first 1000 batches or lowering the learning rate for an order of magnitude overall.

We also tested weight decay, but we discarded it early due to the highly polarised nature of its effect; either it was inconsequential or it caused early layer gradients to vanish, severely crippling the model's ability to learn.

We first assess the distribution of the error in the best performing model to understand whether having high error rates is the rule or the exception. Figure 11 (left) shows the histogram for validation images with a given error value. As can be seen, most examples seem to befall the range below 20% error rates, with a very long tail toward the right for some very particular cases.

We also study the effect of sequence length on the output error in the best performing experiment to try to identify a cause for the observed error distribution, with the conclusion that there does not seem to be a strong correlation between both

TABLE V
SAMPLE OF PRELIMINARY EXPERIMENTS ON INCORPORATING LABEL SMOOTHING AND DROPOUT WITH CONSIDERATION TO LEARNING RATE. COLUMNS LEFT TO RIGHT, THE AMOUNT OF LABEL SMOOTHING AND DROPOUT, WHETHER THE LEARNING RATE WILL BE AN ORDER OF MAGNITUDE LOWER AT THE BEGINNING OF TRAINING, THE LEARNING RATE VALUE, THE SYMBOL ERROR RATE IN PERCENTUAL POINTS AND ITS STANDARD DEVIATION FOR BOTH TRAINING AND VALIDATION SPLITS AND THE LOSS AT EACH SPLIT. USING BRANDENBURG FOR VALIDATION AND THE OTHER DATASETS AS TRAINING. USING AN UNFROZEN PRE-TRAINED BACKBONE.

Label Smoothing	Dropout	Warmup	Learning Rate	Train. SER(%)	Valid. SER(%)	Loss (Train.)	Loss (Valid.)
0	0	$False$	$1.5e-05$	14.24 ± 6.224	88.98 ± 7.61	0.2982	60.71
0.25	0	$True$	$3.0e-04$	1560 ± 1052	500 ± 359.2	2.586	3.093
0.25	0	$True$	$3.0e-04$	1560 ± 1053	500.6 ± 357.3	2.691	3.168
0.25	0.25	$True$	$3.0e-04$	323.8 ± 423.4	506.6 ± 356.6	2.159	5.85
0.1	0.1	$False$	$1.5e-05$	0.8037 ± 1.98	8.228 ± 11.86	0.1534	3.914
0.2	0.25	$False$	$1.5e-05$	1.605 ± 2.831	10.73 ± 10.93	0.2462	3.611

magnitudes. As can be seen in Figure 11 (right), both axes are fairly uniformly distributed, with some specific outliers at the edge of the error and the only noticeable decrease of performance above 150 length. The Pearson correlation between both magnitudes stands at 0.327.

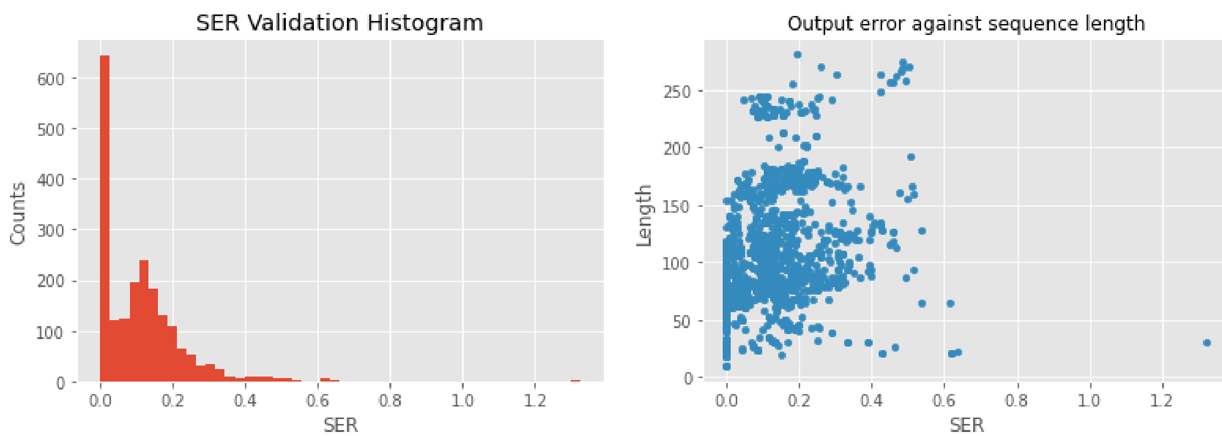

Fig. 11. On the left, histogram depicting the number of images within a set SER range in the best-performing preliminary experiment for the validation partition. On the right, the same experiment results plotted against the length of the ground truth sequence.

We qualitatively investigate the cases where the model seems to be underperforming by manually searching high-error cases and drawing parallels among them. Figure 12 shows the input 10 images where the model is performing worst – all cases around 60% of SER. The commonalities between these examples are the presence of a clef or some engraving errors, either a missing symbol or illogical grouping of elements.

We delve further into the possible causes for the case of clef-containing measures. In Figure 13 we show the first measure of the validation dataset and the output from the best performing model, in which the error rate is the highest. There are very notable additions in the output sequence that do not really appear anywhere in the input image. In particular, there are duplicated notes in the group and a final white notehead that is nowhere to be seen in the initial image. By inspecting the training datasets we suspect this is because the co-occurrence of white noteheads with clefs on starting measures (38.84%) is significantly higher than those without (24.34%). The fact that clefs are overall a “rare” token might be cause inducing this behaviour. We ensure the effect is not a result of memorisation of a particular sequence by checking the most similar sequence available in the training datasets, which has a 26.98% of SER w.r.t. the queried output.

In terms of the duplicated notes, we speculate that the presence of text above the measure might be causing issues in the detection, but cannot provide a conclusive answer. As we do not have any further supporting evidence and we do not have a dataset with a higher amount of clefs available, we cannot further investigate the issue, but given that the effect is consistent in all experiments we believe it is a very likely cause.

A recurring observation is the fact that the validation loss is consistently an order of magnitude above the training loss. This leads to the intuition that the model is much less confident in its predictions on validation than it is on training, which can be an indication of either overfitting or a rift in the distribution on training w.r.t. validation. With the observations made thus far we can be quite confident that overfitting seems not to be the issue.

We also ensured the training samples are sufficiently distinct from the validation samples by computing the SER between each training and validation sample. The results are summarised in Figure 14. The only 32 identical samples are full-rest measures, which are statistically relevant to the task at hand and therefore expected. Otherwise we deem the sequences sufficiently distinct.

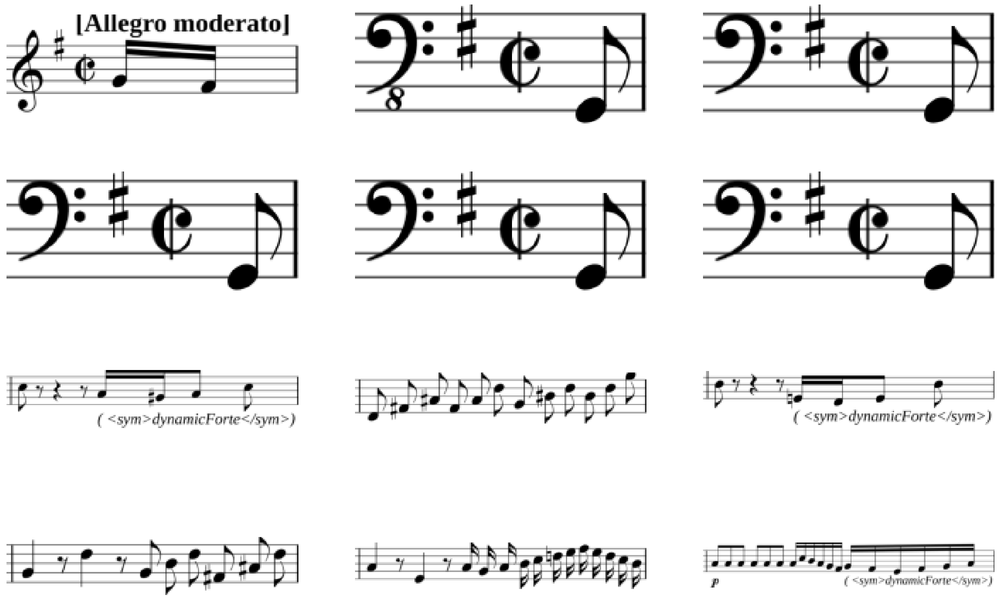

Fig. 12. Validation images on which the model is performing worst.

Ground Truth

REGULAR barline SHARP key TIMESIG_CUT time_signature G_CLEF clef STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_4 pitch note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_3 pitch note vgroup hgroup REGULAR barline voice measure

Prediction (132.258 SER(%))

REGULAR barline TIMESIG_C time_signature (TIMESIG_CUT) (time_signature) G_CLEF clef STEM_UP stem BLACK_NOTEHEAD NUMBER_4 pitch START_TIE ties note vgroup hgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_4 pitch END_TIE ties note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_4 pitch note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_3 pitch note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_3 pitch note vgroup hgroup STEM_UP stem WHITE_NOTEHEAD NUMBER_3 pitch note vgroup hgroup REGULAR barline voice measure

Fig. 13. Output of the model for the first measure of the validation dataset in the best performing preliminary experiment. In blue, tokens that should be substituted; in red, tokens that should be removed; in orange and between parentheses, tokens that are missing and should be included. Below on the left, the ground truth image and on the right a replica of the produced output.

E. Length, Pre-training and Weighting

In Table VI we summarise the entire set of experiments in which we ablate with various training parameters in order to try to improve results and identify possible issues. All models are trained fixing a set of hyperparameters as seen in Table VII. Note that for these experiments we used a bigger learning rate, as in tentative runs we found it much easier to find convergence on smaller sequences and higher learning rates than with full-length sequences.

The main takeaways from Table VI can be summarised in the following points:

- When using an unfrozen untrained backbone, the model tends to overfit heavily. This is indicated by the substantial difference in training and validation SER.
- When using a pre-trained backbone, freezing it during training makes the output seemingly more stable (less uncertainty in the output error; overall better results).
- Adding a weighting term in the loss does not work well the way proposed in this work. Since we did not modify the support tokens' weights (start and end of sequence), the model produces trivial solutions in which the output is the empty sequence. We have also observed the presence of many rare tokens in no particular structure, which are favored by the loss.
- Models trained on longer sequences seem to be ill-behaved – surges in the uncertainty. From what we learnt in preliminary

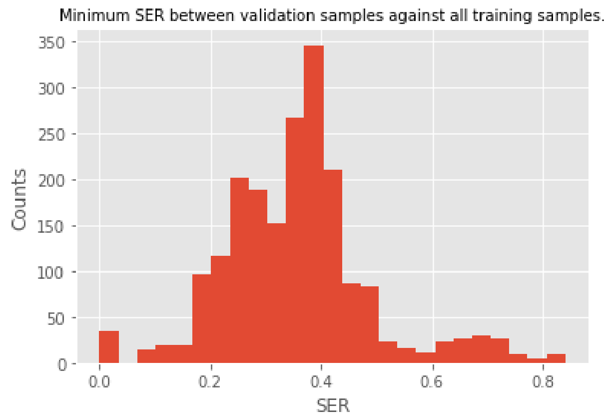

Fig. 14. Histogram with the counts of samples of the validation dataset within a minimum set SER from any other sample of the training dataset.

TABLE VI

DETAILED RESULTS FOR MOST RELEVANT EXPERIMENTS CHANGING OUTPUT LENGTH FOR THE MODEL. FROM LEFT TO RIGHT, COLUMNS INDICATE 1) WHETHER A PRE-TRAINED BACKBONE IS BEING USED, 2) WHETHER THIS BACKBONE IS FROZEN DURING TRAINING, 3) WHETHER A WEIGHTING FACTOR IS USED ON THE LOSS FUNCTION, 4) THE INPUT SEQUENCE LENGTHS, 5) THE SYMBOL ERROR RATE OF THE LAST TRAINING EPOCH (IN PERCENTAGE POINTS, WITH THE STANDARD DEVIATION PRECEDED BY $\pm$ ALSO IN ABSOLUTE PERCENTAGE POINTS), 6) THE VALIDATION SYMBOL ERROR RATE OF THE LAST VALIDATION EPOCH, 7) THE LOSS OF THE LAST TRAINING EPOCH AND 8) THE LOSS OF THE LAST VALIDATION EPOCH. USING 0.1 FOR BOTH DROPOUT AND LABEL SMOOTHING.

Pre-Trained	Frozen	Weighted	Length	Train. SER (%)	Valid. SER (%)	Loss (Train)	Loss (Valid)
			64	3.563 ± 4.569	22.48 ± 22.42	0.530	4.460
			128	5.023 ± 5.184	31.61 ± 18.47	0.432	6.179
			256	5.846 ± 5.502	33.19 ± 18.97	0.320	6.915
✓			64	1.886 ± 9.320	5.382 ± 11.40	0.494	2.156
✓			128	2.691 ± 13.59	11.53 ± 9.539	0.343	3.524
✓			256	16.37 ± 89.41	31.11 ± 50.75	0.435	4.685
✓	✓		64	1.578 ± 3.198	6.326 ± 1.261	0.458	2.000
✓	✓		128	1.783 ± 3.376	4.828 ± 9.245	0.346	2.274
✓	✓		256	3.431 ± 32.54	8.843 ± 10.93	0.247	3.263
✓	✓	✓	64	99.42 ± 4.294	179.6 ± 66.66	0.429	6.063
✓	✓	✓	128	100.2 ± 6.113	145.9 ± 60.58	0.526	4.324
✓	✓	✓	256	100.8 ± 15.79	100.0 ± 0.101	0.396	0.988

runs with full-length sequences, we attribute this to the increase in Learning Rate in comparison to earlier experiments, as we found similar phenomena in incomplete runs.

We shall now address the last observed effect. By inspecting the training sequences with error above $\mu + \sigma$ we have isolated a set of examples that are explanatory of the issue. Figure 15 shows one of these examples, in which the first half of the sequence is properly produced, but then a series of random tokens are generated until the full-length output is obtained. The explanation is the model has missed the prediction of the end token, after which the model receives padding tokens as input. Nevertheless, these tokens are ignored in the training loss, therefore causing the model to be unable to generate a sensible output.

We checked the results for the Pre-Trained + Frozen + Unweighted experiments using only the length 64 validation samples. The resulting symbol error rates for models trained on lengths 64, 128 and 256 were 6.326 ± 12.60, 4.754 ± 11.93 and 4.774 ± 12.27. What this indicates is that the model overall seems to benefit from the extra samples from the 128-length dataset, but stagnates when incorporating up to 256. Considering the added training ease and the distribution of samples of our datasets in a real scenario it would be best to work with 128-length samples.

We also checked the first validation measure as in Figure 13, with a similar result over 80% SER in most cases.

Finally, we checked what the most edited tokens were in experiments with Pre-Trained + Frozen + Unweighted hyperparameters, and concluded that the edit distribution is not very informative as it closely correlates with the validation dataset's distribution (see Table IX). Figure 16 shows a plot with the 25 most frequent edits in the aforementioned scenario. Worthy of comment is the fact that it is not guaranteed that the token edits represent the most sensible way of producing the ground truth sequence, as there may be more than one optimal edit path. Therefore some edits might not reflect the tokens that are actually incorrectly produced (e.g. there might be cases where insertions and removals are commutative).

TABLE VII
HYPERPARAMETERS IN USE DURING THE ABLATION EXPERIMENT.

Encoder									
Layers	6	**Heads**	8	**Input Size**	512×128	**Patch Size**	16	**Dimension**	768

Decoder									
Layers	6	**Heads**	8	**Dimension**	768	**Emb. Dropout**	0.0	**Dropout**	0.1

Overall									
Learn. Rate	$3.0 \cdot 10^{-4}$	**L. Smoothing**	0.1	**Weight Decay**	0.0	**Eval. every**	3 Epochs	**Data Aug.**	Yes

Ground Truth

REGULAR barline STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_2 pitch END_TIE ties note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_3 pitch note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_4 pitch note vgroup hgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_5 pitch note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_6 pitch note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_5 pitch note vgroup hgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_4 pitch note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_5 pitch note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_4 pitch note vgroup hgroup REGULAR barline voice measure

Prediction (188.0 SER(%))

REGULAR barline STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_5 pitch note ties note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_3 pitch note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_4 pitch note vgroup STEM_UP (STEM_UP) STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_3 pitch note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_4 pitch note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_6 pitch note vgroup hgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_6 pitch note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_5 pitch note vgroup STEM_UP BEAM BEAM stem BLACK_NOTEHEAD NUMBER_4 pitch note vgroup hgroup REGULAR barline voice measure rest rest hgroup vgroup hgroup hgroup vgroup hgroup hgroup hgroup vgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup vgroup hgroup measure hgroup hgroup hgroup hgroup measure hgroup hgroup hgroup hgroup hgroup vgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup vgroup hgroup hgroup vgroup hgroup hgroup hgroup hgroup vgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup BEAM hgroup hgroup hgroup hgroup hgroup hgroup hgroup vgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup vgroup hgroup hgroup vgroup hgroup hgroup vgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup vgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup vgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup hgroup vgroup

Fig. 15. Example of malformed output during training in high error training cases

F. Cross-Validation Experiment

TABLE VIII
CROSS-VALIDATION EXPERIMENT RESULTS. THE FIRST COLUMN PRESENTS THE DATASET USED FOR VALIDATION; SUCCESSIVE COLUMNS SHOW THE SYMBOL ERROR RATE FOR TRAINING AND VALIDATION WITH THEIR STANDARD DEVIATION AND THE LOSS IN TRAINING AND VALIDATION.

Valid. Dataset	Train. SER (%)	Valid. SER(%)	Loss (Train)	Loss (Valid)
Jupiter	2.123 ± 3.766	5.215 ± 9.336	0.407	1.891
Fugue	1.731 ± 3.108	3.967 ± 7.783	0.338	3.417
Brandenburg	1.894 ± 3.495	6.045 ± 10.96	0.369	2.366

Table VIII shows the results of the Cross-Validation experiment employing 128-length datasets. All models are performing similarly, with differences caused by the uneven natures of each dataset. This confirms the results obtained so far are sensible and uncorrelated with the input data used.

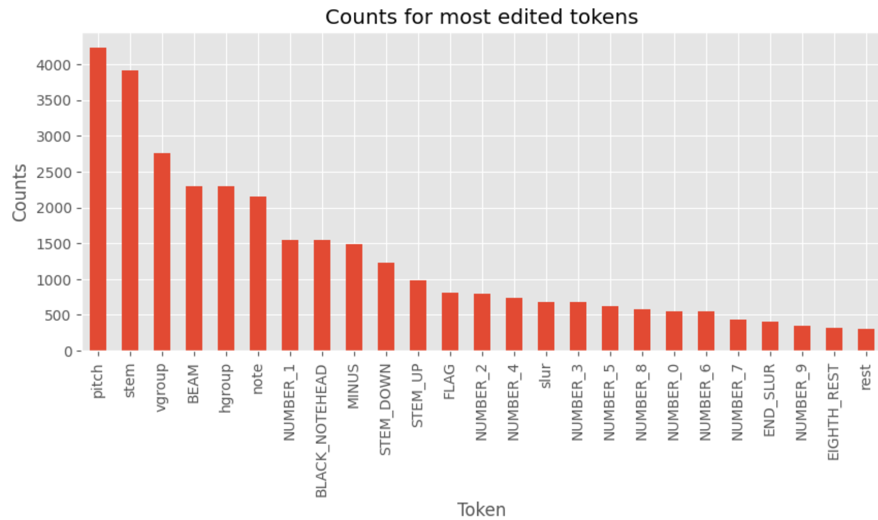

Fig. 16. Most frequent token edits in validation for the Pre-Trained + Frozen + Unweighted set of experiments (all lengths)

VII. Conclusions

"It is only afterward that a new idea seems reasonable. To begin with, it usually seems unreasonable."
Isaac Asimov

In this work we have proposed a new tree-like scheme notation format for Optical Music Recognition with the purpose of tackling OMR end-to-end without requiring any intermediate representations and making possible the production of engraving-ready scores. We have developed a series of new datasets that employ this format and tested its performance on Sequence to Sequence and Transformer-based models, on which we also tested some pre-training techniques. Overall, we can summarise our findings in the following points:

- **We have proposed a new notation format aimed at end-to-end music recognition, but usable in any other context.**

The MTN format is expressive and provides a way of producing engraving-ready scores, as well as being suitable for many OMR setups, not only end-to-end applications. Moreover, it can provide a *lingua franca* for OMR researchers and practitioners through which common evaluation frameworks can be developed. The fact that it is treatable as both a graph and a sequence allows for a wider array of possible analysis and evaluation mechanisms to be developed. We have used SER, which is a standard metric when working with sequences, but graph edit distances or variations of them are also usable.

There are still some issues that may be addressed in the future, such as adding support for multiple-staff parts and polishing elements such as the alignment of additional voicings. The specification does work well for any kind of single-staff homophonic score, with limited support for full polyphony.

- **We have built a 60k sample typeset OMR dataset.**

We have developed a dataset to do a proof-of-concept of the notation system on OMR systems and written tools to work with with new sources of data. We used OpenScores transcriptions due to their status as public domain works, quality and historical relevance.

- **The proposed notation format works well for recognition.**

From the recognition experiments we have conducted we can conclude MTN can be employed successfully in end-to-end models. Excluding the unsuccessful Seq2Seq experiments, we have seen models being able to fully grasp the syntax of the notation and obtain quality results from them.

- **The proposed Transformer-based model is very promising.**

The Transformer model offers a great many deal of advantages when compared against RNN Seq2Seq models. These by-design advantages come with some downsides, such as their higher data requirements and their ease for overfitting. We did have to very carefully tune hyperparameters and impose some sequence length restrictions in order to make them converge consistently, but successive data analysis proved models to be performant on most real use-case scenarios with down to 4.828% SER in 128-length sequences.

Some other trials we have considered but we left as future work are iterations on the models' structure. One of the issues we have found is that having too long input sequences makes training the models very unstable, aside from a dramatically increased cost of inference during validation. A possible solution would be to use models designed for long sequences such as sparse Transformers [65] or BigBird [63].

- **Pre-training improves results substantially.**

When incorporating a pre-trained set of weights in the encoder, the model always produced better results in validation than otherwise. The model seems better fit for generalisation when training with such an encoder with frozen weights than training from scratch using only the annotated data and the cross-entropy loss. This is also the first time this has been tried for music to the best of our knowledge.

This opens the door for a very interesting road of research towards self-supervision in music recognition, in which an overwhelming abundance of non-labeled data is available. In particular, self-supervision is an ideal road to tackle problems such as handwritten music recognition, for which a great corpus of unlabeled data is available.

- **There is a very extensive library of scores that can be used.**

We have developed tools to generate MTN files from MusicXML files. Given the ubiquity of the format, through this approach we believe we can palliate some of the input data problems the OMR community has had for many years, provided the MusicXML file can be aligned to the source material.

At the same time, we also found some ground to cover as future work.

- **The matter of testing the approach on real handwritten scores remains.**

We did not have time to delve into this matter with the attention the problem deserves. Tackling automatic cropping and alignment of handwritten scores to any notation system is probably worth an entire thesis of work, hence our hesitancy to attempt it on the first place. Nevertheless, it is a very logical step forward in which limited size attempts can be made; for instance, manually annotating a single piece and using a mixture of synthetic and real data for training (as seen in [10] or [41]).

In particular, given that the notation format is independent of the representation of the score, another possible way to tackle handwritten scores is style transfer. By generating samples from synthetic scores with a handwritten look, the problem of having annotated training data is solved.

All in all, we consider our initial goals accomplished. The obtained results are very promising, while still a little bit behind the current state of the art for typeset scores. Notably, the current established methods are very mature, whereas this work is exploratory in nature, setting ground work for further developments going forward in time.

ACKNOWLEDGMENTS

There are many people to whom I am greatly indebted for their help and support throughout this work. First, I must thank both my co-supervisors, Alicia and Sanket, for their colossal task in guiding me through this thesis and the opportunities and resources they have provided me with during this time. I must also thank Pau Riba, who was initialy my co-supervisor, for his insightful ideas and for being such an inspiring figure within the Computer Vision Centre as a whole.

I have also had the pleasure of sharing discussions with many people within the CVC, many of which have been extremely inspiring and helpful for the outcome of this work. In no particular order, I want to thank Ali Furkan Biten, Dimosthenis Karatzas, Sergi Masip, Ruben Pérez, Mohamed Ali Souibgui and all other people within the center who have shared words of support and wisdom with me.

Finally, I would like to give some words of appreciation to my significant other, Laura, for her support and patience with the most tired and utterly stressed version of myself.

APPENDIX A
DATASET FREQUENCIES

TABLE IX
DETAILED DATASET ABSOLUTE (f) AND RELATIVE (p) FREQUENCY OF APPEARANCE OF TOKENS.

	Brandenburg		Fugue		Jupiter		9th	
	f	p	f	p	f	p	f	p
BEAM	26947	0.130435	24863	0.074081	20587	0.050684	63339	0.041111
vgroup	21558	0.104350	32627	0.097214	39009	0.096038	143180	0.092932
pitch	19814	0.095908	30540	0.090996	32803	0.080760	114177	0.074108
stem	19814	0.095908	30135	0.089789	31159	0.076712	110174	0.071510
note	19814	0.095908	30540	0.090996	32803	0.080760	114177	0.074108
BLACK_NOTEHEAD	19782	0.095753	27536	0.082045	27537	0.067795	102646	0.066623
STEM_DOWN	12083	0.058487	23312	0.069460	20992	0.051681	75470	0.048985
hgroup	9323	0.045127	19081	0.056853	28421	0.069971	114311	0.074195
STEM_UP	7731	0.037421	6823	0.020330	10167	0.025031	34704	0.022525
NUMBER_1	7047	0.034111	13103	0.039041	15659	0.038552	52687	0.034197
barline	4028	0.019497	13064	0.038925	20550	0.050593	88006	0.057121
REGULAR	3984	0.019284	12932	0.038532	20312	0.050007	85306	0.055369
FLAG	2734	0.013234	835	0.002488	1936	0.004766	12402	0.008050
NUMBER_3	2048	0.009913	2413	0.007190	3180	0.007829	12744	0.008272
NUMBER_8	2037	0.009860	3699	0.011021	2934	0.007223	11722	0.007608
voice	2014	0.009749	6598	0.019659	10316	0.025398	44221	0.028702
measure	2014	0.009749	6532	0.019462	10275	0.025297	44003	0.028561
NUMBER_2	2007	0.009715	2573	0.007666	3961	0.009752	11713	0.007602
NUMBER_0	1989	0.009628	3753	0.011182	3515	0.008654	13380	0.008684
NUMBER_7	1925	0.009318	3440	0.010250	3065	0.007546	11063	0.007181
NUMBER_6	1875	0.009076	3114	0.009278	2010	0.004949	9996	0.006488
NUMBER_4	1808	0.008752	2230	0.006644	4412	0.010862	12736	0.008266
NUMBER_5	1780	0.008616	2318	0.006907	2686	0.006613	9819	0.006373
rest	1744	0.008442	2089	0.006224	6764	0.016653	31668	0.020554
NUMBER_9	1699	0.008224	3855	0.011486	3194	0.007863	9970	0.006471
MINUS	1564	0.007570	141	0.000420	968	0.002383	4472	0.002903
accidental	1382	0.006689	4318	0.012866	4104	0.010104	8662	0.005622
slur	1126	0.005450	166	0.000495	5563	0.013696	17013	0.011042
EIGHTH_REST	1096	0.005305	404	0.001204	1421	0.003498	9362	0.006076
SHARP	965	0.004671	2066	0.006156	1338	0.003294	5066	0.003288
START_SLUR	599	0.002899	84	0.000250	2784	0.006854	8799	0.005711
END_SLUR	599	0.002899	84	0.000250	2784	0.006854	8799	0.005711
QUARTER_REST	540	0.002614	942	0.002807	3330	0.008198	11206	0.007273
NATURAL	301	0.001457	1498	0.004463	1933	0.004759	2959	0.001921
FLAT	138	0.000668	850	0.002533	854	0.002103	3573	0.002319
DOT	130	0.000629	1860	0.005542	1876	0.004619	13699	0.008891
dots	130	0.000629	1860	0.005542	1863	0.004587	13263	0.008608
HALF_REST	87	0.000421	336	0.001001	1494	0.003678	1516	0.000984
articulations	40	0.000194	80	0.000238	3294	0.008110	20343	0.013204
time_signature	33	0.000160	96	0.000286	68	0.000167	1450	0.000941
WHITE_NOTEHEAD	32	0.000155	3004	0.008951	5266	0.012965	11531	0.007484
LIGHT-HEAVY	22	0.000106	100	0.000298	170	0.000419	400	0.000260
key	22	0.000106	96	0.000286	21	0.000052	1407	0.000913
FERMATA	22	0.000106	62	0.000185	30	0.000074	140	0.000091
STACCATISSIMO	18	0.000087	0	0.000000	471	0.001160	0	0.000000
clef	12	0.000058	156	0.000465	27	0.000066	59	0.000038
ties	12	0.000058	4218	0.012568	1619	0.003986	11924	0.007739
16TH_REST	12	0.000058	105	0.000313	109	0.000268	1519	0.000986
LIGHT-LIGHT	11	0.000053	16	0.000048	0	0.000000	2050	0.001331
TIMESIG_C	11	0.000053	38	0.000113	17	0.000042	350	0.000227
TIMESIG_CUT	11	0.000053	48	0.000143	17	0.000042	300	0.000195
OVER	11	0.000053	10	0.000030	34	0.000084	800	0.000519
compound_time_signature	11	0.000053	10	0.000030	34	0.000084	800	0.000519
HEAVY-LIGHT	11	0.000053	16	0.000048	68	0.000167	250	0.000162
WHOLE_REST	9	0.000044	283	0.000843	389	0.000958	8001	0.005193
START_TIE	6	0.000029	2158	0.006430	880	0.002167	6864	0.004455
END_TIE	6	0.000029	2157	0.006427	880	0.002167	6864	0.004455
F_CLEF	6	0.000029	33	0.000098	10	0.000025	14	0.000009
ornaments	6	0.000029	80	0.000238	693	0.001706	1653	0.001073
TRILL-MARK	6	0.000029	35	0.000104	104	0.000256	154	0.000100
C_CLEF_3	3	0.000015	45	0.000134	1	0.000002	6	0.000004
G_CLEF	3	0.000015	7	0.000021	14	0.000034	38	0.000025
CAESURA	1	0.000005	0	0.000000	0	0.000000	0	0.000000
C_CLEF_1	0	0.000000	47	0.000140	0	0.000000	0	0.000000
C_CLEF_4	0	0.000000	24	0.000072	2	0.000005	1	0.000001
MORDENT	0	0.000000	22	0.000066	0	0.000000	0	0.000000
INVERTED-MORDENT	0	0.000000	22	0.000066	0	0.000000	0	0.000000
32ND_REST	0	0.000000	19	0.000057	21	0.000052	64	0.000042
STACCATO	0	0.000000	18	0.000054	2793	0.006876	20154	0.013081
TURN	0	0.000000	1	0.000003	2	0.000005	0	0.000000
TREMOLO	0	0.000000	0	0.000000	586	0.001443	1497	0.000972
WAVY-LINE	0	0.000000	0	0.000000	2	0.000005	4	0.000003
ACCENT	0	0.000000	0	0.000000	0	0.000000	49	0.000032
FLAT-FLAT	0	0.000000	0	0.000000	0	0.000000	1	0.000001

References

[1] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 000–16 009.

[2] J. Calvo-Zaragoza, J. H. Jr, and A. Pacha, "Understanding optical music recognition," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–35, 2020.

[3] T. W. Adorno and S. Gillespie, "Music, language, and composition," *The Musical Quarterly*, vol. 77, no. 3, pp. 401–414, 1993.

[4] V. Bosch Campos, J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal Ruiz, "Sheet music statistical layout analysis," in *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016, pp. 313–318.

[5] C. Li, J. Zhao, J. Cai, H. Wang, and H. Du, "Optical music notes recognition for printed music score," in *2018 11th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 1. IEEE, 2018, pp. 285–288.

[6] L. Tuggener, I. Elezi, J. Schmidhuber, M. Pelillo, and T. Stadelmann, "Deepscores-a dataset for segmentation, detection and classification of tiny objects," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 3704–3709.

[7] L. Tuggener, Y. P. Satyawan, A. Pacha, J. Schmidhuber, and T. Stadelmann, "The deepscoresv2 dataset and benchmark for music object detection," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 9188–9195.

[8] E. Shatri and G. Fazekas, "Doremi: First glance at a universal omr dataset," *arXiv preprint arXiv:2107.07786*, 2021.

[9] J. Calvo-Zaragoza and D. Rizo, "End-to-end neural optical music recognition of monophonic scores," *Applied Sciences*, vol. 8, no. 4, 2018. [Online]. Available: https://www.mdpi.com/2076-3417/8/4/606

[10] A. Baró, C. Badal, and A. Fornés, "Handwritten historical music recognition by sequence-to-sequence with attention mechanism," in *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 2020, pp. 205–210.

[11] A. Fornés, A. Dutta, A. Gordo, and J. Lladós, "Cvc-muscima: a ground truth of handwritten music score images for writer identification and staff removal," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 15, no. 3, pp. 243–251, 2012.

[12] J. Hajič and P. Pecina, "The muscima++ dataset for handwritten optical music recognition," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 39–46.

[13] J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal, "Handwritten music recognition for mensural notation: Formulation, data and baseline results," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 1081–1086.

[14] E. Parada-Cabaleiro, A. Batliner, A. Baird, and B. Schuller, "The seils dataset: Symbolically encoded scores in modern-early notation for computational musicology," 2017.

[15] A. Baró, P. Riba, J. Calvo-Zaragoza, and A. Fornés, "From optical music recognition to handwritten music recognition: A baseline," *Pattern Recognition Letters*, vol. 123, pp. 1–8, 2019.

[16] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. R. Marcal, C. Guedes, and J. S. Cardoso, "Optical music recognition: state-of-the-art and open issues," *International Journal of Multimedia Information Retrieval*, vol. 1, no. 3, pp. 173–190, 2012.

[17] C. Dalitz, M. Droettboom, B. Pranzas, and I. Fujinaga, "A comparative study of staff removal algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 753–766, 2008.

[18] A. Dutta, U. Pal, A. Fornes, and J. Llados, "An efficient staff removal approach from printed musical documents," in *2010 20th International Conference on Pattern Recognition*. IEEE, 2010, pp. 1965–1968.

[19] I. Fujinaga, "Staff detection and removal," in *Visual Perception of Music Notation: On-Line and Off Line Recognition*. IGI Global, 2004, pp. 1–39.

[20] J. dos Santos Cardoso, A. Capela, A. Rebelo, C. Guedes, and J. P. da Costa, "Staff detection with stable paths," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 6, pp. 1134–1139, 2009.

[21] J. Calvo-Zaragoza, I. Barbancho, L. J. Tardón, and A. M. Barbancho, "Avoiding staff removal stage in optical music recognition: application to scores written in white mensural notation," *Pattern Analysis and Applications*, vol. 18, no. 4, pp. 933–943, 2015.

[22] J. Calvo-Zaragoza, G. Vigliensoni, and I. Fujinaga, "Pixel-wise binarization of musical documents with convolutional neural networks," in *2017 Fifteenth IAPR international conference on machine vision applications (MVA)*. IEEE, 2017, pp. 362–365.

[23] P. Bellini, I. Bruno, and P. Nesi, "Optical music sheet segmentation," in *Proceedings First International Conference on WEB Delivering of Music. WEDELMUSIC 2001*. IEEE, 2001, pp. 183–190.

[24] F. Rossant and I. Bloch, "Robust and adaptive omr system including fuzzy modeling, fusion of musical rules, and possible error detection," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, pp. 1–25, 2006.

[25] A. Rebelo, G. Capela, and J. S. Cardoso, "Optical recognition of music symbols," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 13, no. 1, pp. 19–31, 2010.

[26] B. Coüasnon, "Dmos, a generic document recognition method: Application to table structure analysis in a general and in a specific way," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 8, no. 2, pp. 111–122, 2006.

[27] D. Bainbridge and T. Bell, "A music notation construction engine for optical music recognition," *Software: Practice and Experience*, vol. 33, no. 2, pp. 173–200, 2003.

[28] B. Coüasnon, P. Brisset, I. Stéphan, and C. P. Brisset, "Using logic programming languages for optical music recognition," in *In Proceedings of the Third International Conference on The Practical Application of Prolog*. Citeseer, 1995.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[30] A. Pacha, J. Hajič Jr, and J. Calvo-Zaragoza, "A baseline for general music object detection with deep learning," *Applied Sciences*, vol. 8, no. 9, p. 1488, 2018.

[31] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[32] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[33] A. Pacha, K.-Y. Choi, B. Coüasnon, Y. Ricquebourg, R. Zanibbi, and H. Eidenberger, "Handwritten music object detection: Open issues and baseline results," in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE, 2018, pp. 163–168.

[34] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[35] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[36] J. Calvo-Zaragoza, J. J. Valero-Mas, and A. Pertusa, "End-to-end optical music recognition using neural networks," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR*, 2017, pp. 23–27.

[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[38] M. Liwicki, A. Graves, S. Fernàndez, H. Bunke, and J. Schmidhuber, "A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks," in *Proceedings of the 9th International Conference on Document Analysis and Recognition, ICDAR 2007*, 2007.

[39] A. Baró, P. Riba, J. Calvo-Zaragoza, and A. Fornés, "Optical music recognition by long short-term memory networks," in *International Workshop on Graphics Recognition*. Springer, 2017, pp. 81–95.

[40] L. Kang, J. I. Toledo, P. Riba, M. Villegas, A. Fornés, and M. Rusinol, "Convolve, attend and spell: An attention-based sequence-to-sequence model for handwritten word recognition," in *German Conference on Pattern Recognition*. Springer, 2018, pp. 459–472.

[41] P. Torras, A. Baró, L. Kang, and A. Fornés, "On the integration of language models into sequence to sequence architectures for handwritten music recognition." in *ISMIR*, 2021, pp. 690–696.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[43] A. Gulati, C.-C. Chiu, J. Qin, J. Yu, N. Parmar, R. Pang, S. Wang, W. Han, Y. Wu, Y. Zhang, and Z. Zhang, Eds., *Conformer: Convolution-augmented Transformer for Speech Recognition*, 2020.

[44] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[45] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[46] C.-F. R. Chen, Q. Fan, and R. Panda, "Crossvit: Cross-attention multi-scale vision transformer for image classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 357–366.

[47] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 10 012–10 022.

[48] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," *CoRR*, vol. abs/2005.12872, 2020. [Online]. Available: https://arxiv.org/abs/2005.12872

[49] D. Braae and T. Rusbjerg, "Detr for combined object detection and notation assembly in optical music recognition."

[50] A. Ríos-Vila, J. M. Iñesta, and J. Calvo-Zaragoza, "On the use of transformers for end-to-end optical music recognition," in *Pattern Recognition and Image Analysis*, A. J. Pinho, P. Georgieva, L. F. Teixeira, and J. A. Sánchez, Eds. Cham: Springer International Publishing, 2022, pp. 470–481.

[51] G. Kim, T. Hong, M. Yim, J. Park, J. Yim, W. Hwang, S. Yun, D. Han, and S. Park, "Donut: Document understanding transformer without ocr," 2021. [Online]. Available: https://arxiv.org/abs/2111.15664

[52] V. Shiv and C. Quirk, "Novel positional encodings to enable tree-based transformers," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/6e0917469214d8fbd8c517dcdc6b8dcf-Paper.pdf

[53] Z. Sun, Q. Zhu, Y. Xiong, Y. Sun, L. Mou, and L. Zhang, "Treegen: A tree-based transformer architecture for code generation," 2019. [Online]. Available: https://arxiv.org/abs/1911.09983

[54] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[55] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European conference on computer vision*. Springer, 2016, pp. 649–666.

[56] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1422–1430.

[57] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[58] H. Bao, L. Dong, and F. Wei, "Beit: Bert pre-training of image transformers," *arXiv preprint arXiv:2106.08254*, 2021.

[59] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[60] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[61] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 4945–4949.

[62] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[63] M. Zaheer, G. Guruganesh, A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed, "Big bird: Transformers for longer sequences," 2020. [Online]. Available: https://arxiv.org/abs/2007.14062

[64] V. I. Levenshtein *et al.*, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8. Soviet Union, 1966, pp. 707–710.

[65] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.