DETR for Combined Object Detection and Notation Assembly in Optical Music Recognition

David O. Braae and Thomas R. Rusbjerg
Software, Group mi1014f21

Aalborg University
Department of Computer Science
Selma Lagerlöfs Vej 300
9220 Aalborg East, Denmark

***Abstract*—Optical Music Recognition is traditionally comprised of several steps in a pipeline that together decipher the contents of sheet music. These tasks include image preprocessing, object detection, notation assembly, and encoding. Deep learning has recently made its appearance within the field, simplifying the pipeline and combining tasks that were previously separate. However, most deep learning approaches still treat object detection and notation assembly as two disjoint tasks. R-CNN based models, which are state-of-the-art object detectors within general object detection, have been utilised to detect music objects and show promising results. Meanwhile, it has recently been shown that transformers can perform on par with R-CNN models in general object detection while serving as a more simple, end-to-end solution. Transformers inherently model relations between objects through their attention mechanism, making them suitable for notation assembly due to music notation's rich contextual nature. In this paper, we study how to utilise attention in transformers to detect handwritten music objects in common western music notation while simultaneously predicting object relationships. We thereby propose an end-to-end model which handles both object detection and notation assembly holistically.**

I. Introduction

Optical Music Recognition (OMR) is the field of research that investigates how to computationally read music notation in images and encode the musical content in a digital format [1]. Since there is an abundance of physical sheet music contained in various archives [2] [3] [4], the success of OMR systems would be of great value to the music community. The benefits of digitising the music scores in these archives are many, such as instant availability, ease of distribution, computational analysis, search, and editing. However, the process of manually digitising physical sheet music is tedious as well as expensive, and calls for either full or partial computational automation.

The traditional approach to OMR follows a pipeline consisting of four steps [1] [5] [6], see figure 1:

1) The image is pre-processed since many scores are in a deteriorated state or the image is noisy or skewed.
2) Object detection is performed, i.e. the class and location of each symbol are identified.
3) The musical semantics of the identified objects are analysed. This step is called *notation assembly* and requires a contextual analysis since the meaning of each music symbol depends on other symbols in the score.
4) The musical semantics are encoded into a standard music notation format, e.g. MIDI [7], MusicXML [8] or MEI [9].

The majority of work within OMR focuses on one of these steps, with the most recent work being targeted at object detection and notation assembly, as these steps are the most essential and challenging. Traditionally, these steps are themselves subdivided into subtasks, e.g. object detection has usually been divided into stave-line detection and removal, symbol segmentation, and symbol recognition [5]. However, since deep learning's recent appearance within OMR some of these steps have seen great improvement, while other steps have become obsolete or combined [1]. Object detection is no longer divided into three subtasks, since Convolutional Neural Networks (CNNs) have proven effective at dealing with the task holistically.

The level of detail required of an OMR system to extract from the score depends on the use case. For example, the objective could be to identify the genre or composer of the music, or to allow playback. At the highest level of comprehension lies *structured encoding*: capturing all of the information in the score and producing an identical digital version in a format such as MusicXML [1]. Having the score in this format makes it significantly more straightforward to conduct any task on the score, such as analysis, search, editing, or playback, and we therefore focus on this objective. However, while OMR has frequently been called "Optical Character Recognition (OCR) for music" [1], it is far more challenging in several regards. The difficulty lies in the fact that music notation is two-dimensional and its semantics rely heavily on context. For example, to determine the pitch of a note, one must look at its vertical position on the stave, identify the latest clef specifying which pitch correspond to which stave-line, and take into account any accidentals found in the score's key, the bar that surrounds the note, or any individual

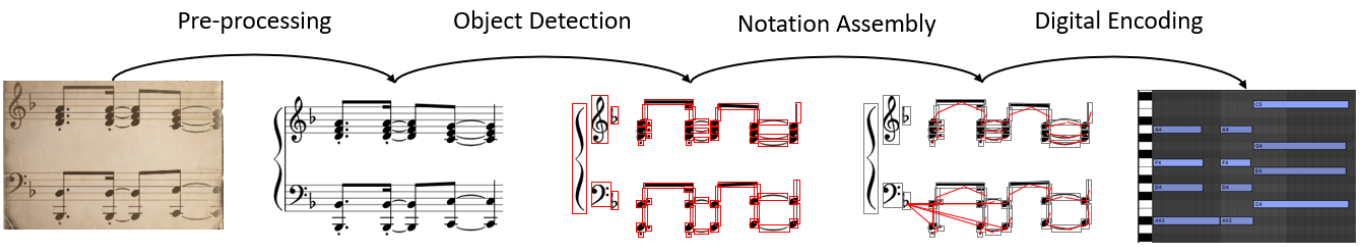


Fig. 1: The traditional OMR pipeline. In this example, the output musical encoding is MIDI, but the desired output encoding depends on the specific use case.

accidental next to the note. Failing to recognise any of this information will lead to an incorrect pitch. An example of the difficulties with pitch inference is given in figure 2.

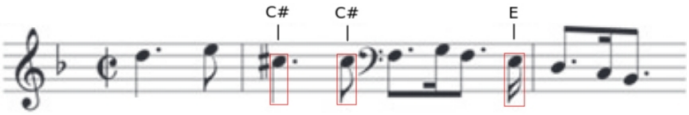


Fig. 2: A snippet from Henry Purcell's *The Epicure* illustrating the contextual information required for pitch inference. There are three notes placed in the second stave-space from the top, marked by red boxes. The first marked note is a C# since the G-clef at the start declares that stave-space to be a C, and because the note has an accidental in front of it (the # symbol, called a *sharp*). The second marked note is also a C# because the first accidental is still in effect until the end of the bar. However, the third marked note is an E, even though it is placed in the same stave-space and bar as the last two notes. This occurs because an F-clef is located right after the second marked note, changing the pitches of the stave-spaces.

Most research within OMR treats object detection and notation assembly as two separate steps. However, given the cohesive nature of objects in sheet music, it is appropriate to combine the two and utilise the rich level of contextual information to detect the objects. There are many cases in machine learning where combining related tasks, also known as *multi-task learning*, leads to better results [10]. This further motivates the combination of the two steps. In this paper, we therefore propose an approach that combines object detection with notation assembly. Our main novel contributions are as follows:

- We propose an OMR model that performs simultaneous object detection and notation assembly.
- We introduce transformers to OMR, a domain of images with many small and densely located objects.
- We present experiments that report the effectiveness of transformers and multi-task learning in combined music object detection and notation assembly.

II. Related Work

This paper focuses on two separate tasks; object detection and notation assembly, and our work builds on prior research within both domains.

A. Object Detection

Being a fundamental and popular area within computer vision, object detection has been in the spotlight of research for several decades [11]. It is the task of locating and identifying object instances of pre-defined categories in images by predicting their class labels and rectangular bounding boxes that encapsulate them. In terms of accuracy, the field has been dominated by Region-based Convolutional Neural Networks (R-CNNs) since their introduction in 2014 [11] [12]. In these two-stage models, a Region Proposal Network (RPN) suggests candidate regions for where objects may be located in an image, and CNNs are used to extract features from those regions. After having filtered out overlapping and low-confidence regions using Non-Maximum Suppression (NMS), a number of classifiers and bounding box regressors use the final regions to make object predictions. The latest R-CNN model, Mask R-CNN, was proposed by He et al. [13] and predicts masks[1] in addition to classes and bounding boxes, thus improving its overall detection performance using multi-task learning. While R-CNN models provide the highest scores, other so-called one-stage models such as You Only Look Once (YOLO) [14] and Single Shot Detector (SSD) [15] trade off some accuracy for faster performance. This is useful when working in domains with little computation capabilities or strict real-time requirements. These more lightweight models use features from the entire image globally instead of generating region proposals to extract features from, which drops the RPN component entirely.

Recently, Carion et al. [16] proposed a new method, DEtection TRansformer (DETR) for object detection, drawing inspiration from other fields such as machine translation. They propose to simplify the pipeline of region-based networks by adopting an encoder-decoder architecture using transformers, effectively dropping components present in R-CNN models that require substantial fine-tuning and prior knowledge about the input, e.g. RPN and NMS. The accuracy of DETR is on par with R-CNN models, but transformers require large datasets and very long training time. DETR uses attention mechanisms to reason about the global image context and implicitly utilise relations between the objects as it predicts all

[1] A mask is the collection of pixels that make up the object.

objects at once in a single pass through the model. The work we present in this paper builds on DETR but differs from previous work since we augment DETR to explicitly model the relations between the detected objects. Furthermore, we evaluate DETR in the domain of OMR which contains challenges different from those in general datasets such as COCO [17], e.g. dense images with many small objects.

B. Object Detection in OMR

Traditionally, the object detection phase in OMR is subdivided into stave-line removal, symbol segmentation, and symbol classification [1] [6]. Since stave-lines often posed issues in symbol segmentation, they were removed first using various methods. Symbol segmentation is often approached using hierarchical decomposition, i.e. split by staves and then extract primitive symbols (noteheads, stems, etc.) [5]. The classification stage is sometimes part of the segmentation, and many methods have been used to classify music objects based on their segmentation, e.g. k-nearest neighbour, neural networks and hidden Markov models [6] [5].

In more recent work, however, there has been a shift towards deep learning methods that combine these stages [1] [6]. Pacha et al. [18] used Faster R-CNN [19] in a sliding-window approach on stave-wise images of handwritten music notation to detect the class and bounding box of every music object. The main downsides of their approach are long training times and sensitive hyperparameter tuning to fit the data.

Tuggener et al. [20] use a CNN to perform a watershed transformation for semantic segmentation of the entire score, achieving high performance on small objects, but struggling with overlapping symbols, accurate bounding boxes, and rare classes.

Hajič jr. et al. [21] utilise the U-Net architecture [22] to perform semantic segmentation, and then use a simple connected components detector to segment each object. Since this approach is based on connected components, it struggles with very dense scores with overlapping symbols.

Our approach uses an encoder-decoder transformer architecture which has not previously been applied to the field of OMR. Its primary difference from previous approaches is how it utilises attention mechanisms to reason about relationships between the objects when making predictions.

C. Notation Assembly

While several deep learning methods have been used to tackle the object detection stage of OMR, the same cannot be said for notation assembly. Instead, the majority of methods rely on grammars [23] [24] [25] [26] [27] [28] and heuristics about the specific music being recognised [29] [30] [31] [32]. These methods manually define syntactic rules that formalise how music can be written and uses these rules to reconstruct the musical information from the detected music objects. The drawback is that there exists a huge corpus of music symbols, and music notation is extremely complex [1] [33], making these rules hard to define. The result is that most suggested grammars only capture a tiny subset of music notation [25]. Furthermore, while it has been stated that it is possible to recognise music notation using an LL(k) grammar [34], there is no guarantee that any given music score will follow the rules of music notation. Instead, it is often the case that they are not followed [33] [35], and this becomes ever so more the case with handwritten music [6], which is the focus of this paper.

Some of the issues with grammatical and heuristical approaches can be circumvented using machine learning, as shown by Pacha et al. [36]. They encapsulate all of the syntactic information in the music score in a graph by modelling relationships between music objects (this graph is described in more detail in section V). They do this in a data-driven fashion by training a CNN to identify pairwise relationships between music objects, given cropped images centred on pairs of detected objects. However, they couple this with a rule-based approach by only giving the model training examples where the two objects are ”compatible”, i.e. when they are within close proximity, and when the classes are not incompatible, e.g. a notehead and a barline cannot be related. Therefore, their model does not learn to discern between incompatible and compatible music objects. Our approach differs greatly from theirs in this regard, in addition to using an entirely different type of neural network (transformers). Given an image of a music score, our model predicts the objects and their relationships simultaneously in a single pass, without having to centre the image on a pair of pre-detected objects which furthermore have been selected using a manually defined set of rules.

D. Scene Graph Generation

Exploring relationships between objects in images is not a task limited to OMR, and this issue is at the core of Scene Graph Generation (SGG), a popular field within computer vision [37]. SSG models predict the position and characteristics of objects in an image as well as their relations to other objects. The relations are usually expressed in a *subject - predicate - object* triplet manner, e.g. ”girl throwing ball”. While there are many different approaches to SGG [37], a significant number of proposed methods use the following graph based architecture [38] [39] [40] [41] [42]:

1) Use RPN or R-CNN to generate objects as vertices.
2) Generate an initial graph using a graph initialisation scheme.
3) Refine the graph using a Graph Neural Network (GNN).

The primary distinction between their approaches and ours is twofold: firstly, we predict the objects and relationships simultaneously using a single model instead of building graphs based on existing regions or objects. Secondly, the relations between music objects do not follow the triplet format, since we are only interested in determining whether the relations exist or not.

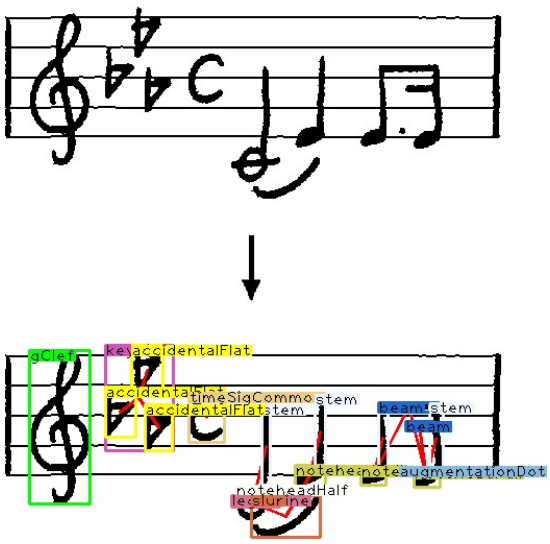


Fig. 3: An example of an input music score and the desired output with class labels, bounding boxes, and edges between related objects.

III. Problem Definition

In this paper, we focus on the task of simultaneous object detection and notation assembly in OMR. That is, we seek an algorithm that determines the position and class of each musical object in an image of a music score, as well as the relationships between the identified objects. The task is thus to create a function that given an image $\mathbf{I} \in \mathbb{R}^{3 \times H \times W}$ as input, produces a Music Notation Graph (MuNG) $G = (V, E)$ as output, where V is the set of vertices and E is the set of directed edges. A vertex $v = (c, b) \in V$ is a pair consisting of a class label c and a rectangular bounding box b that encapsulates an object. An example vertex is the *gClef* in figure 3 with its label and bounding box. The figure illustrates an example of the desired output given an image of sheet music, i.e. the label and bounding box of each object as well as their relationships with each other. The bounding box b is a 4-tuple (c_x, c_y, h, w) defining the centre coordinates of the box and its height and width. An edge e is defined as an ordered pair of vertices in V: $e \in E \quad \{(v_i, v_j) \mid (v_i, v_j) \in V^2$ and $v_i \neq v_j\}$. An edge signals that there is a syntactical relationship between two music objects. The figure illustrates several examples of edges, e.g. between an *accidentalFlat* and the *key*, or between a *noteheadHalf* and its *stem*. The output edges are given as an adjacency matrix $\mathbf{A} \in \{0, 1\}^{|V| \times |V|}$ where $A_{ij} = 1$ if $(v_i, v_j) \in E$ and $A_{ij} = 0$ if $(v_i, v_j) \notin E$.

To sum up, the output of a model solving the two-fold problem of object detection and notation assembly will consist of three parts: a class label and bounding box for each music object in the image, and an adjacency matrix describing the relationships between all detected objects.

IV. Method

We propose a new method for OMR in which object detection and notation assembly are combined in a single model, and we utilise the capabilities of the transformers in DETR [16] to achieve this.

A. Architecture

The architecture of our model follows that of DETR, and thus combines a CNN feature extractor with an encoder-decoder transformer, followed by Feed Forward Networks (FFNs) for each prediction branch, see figure 4. The CNN acts as a feature extractor, i.e. the input image is transformed through the CNN's layers to a latent representation of a lower dimension than that of the original input. The transformer encoder transforms the CNN's feature maps summed with a positional encoding. Through the attention mechanism, the encoder learns which areas of the image are related to each other and encodes this information in its output. The encoder's output, i.e. the transformed feature maps, is used in the cross-attention of the decoder as shown in figure 5. The decoder's input consists of a fixed number of learned positional embeddings referred to as *object queries*. These object queries learn to focus on specific parts of the encoder-transformed feature maps, i.e. the image representation. The decoder uses these queries in combination with the cross-attention weights to determine which part of the encoder's output to focus on. The decoder's output consists of a set of transformed object queries, each corresponding to a feature representation of a potential object in the image. Each of these transformed object queries is passed through FFNs to obtain the class and bounding box prediction. The relationships between objects are obtained by a dot product between each pair of the transformed object queries followed by an FFN.

B. Model Components

This section explains each model component in more detail.

1) CNN feature extractor: The first component of the model is the feature extractor, also referred to as a *backbone*. The backbone consists of a convolutional neural network with the purpose of reducing the input image to a compact feature representation. This reduces downstream computation which is especially important for transformer models due to the input quadratic self-attention. The chosen CNN architecture is ResNet-50 [43], which is a popular architecture utilising a deep CNN with residual connections. The model is pre-trained on the COCO dataset [17] in order to reduce training time by utilising transfer-learning. The input to the CNN is a set of images $img \in \mathbb{R}^{3 \times H \times W}$ consisting of 3 colour channels. H and W are the height and width of the largest image in the batch, respectively. Smaller images are adjusted with 0-padding such that all images are the same size. The CNN transforms the image into a set of feature maps, i.e. $fmap \in \mathbb{R}^{C \times H^0 \times W^0}$ with $C = 2048$, $H^0 = H/32$, and $W^0 = W/32$. The feature maps get further reduced by a 1x1

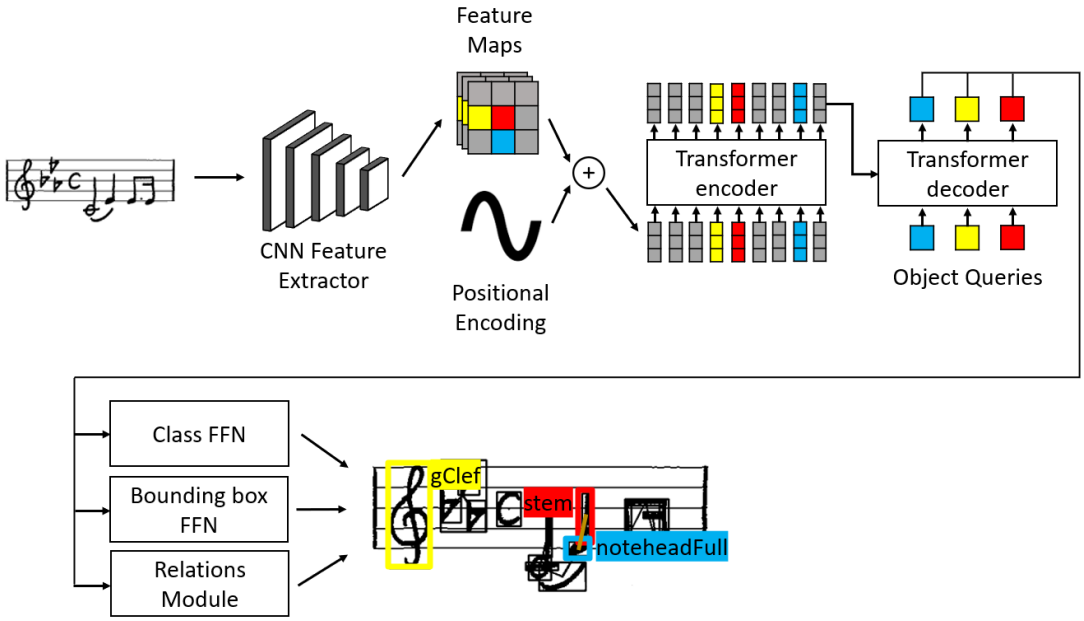


Fig. 4: The architecture of our proposed model from input to output. The CNN generates feature maps that are input as vectors to the encoder. The encoder transforms the extracted features and this is used in the decoder's cross-attention in combination with a set of learned object queries. Each object query is transformed and corresponds to an object prediction, as illustrated by the three colours. The three prediction branches process each query and outputs a class label, bounding box, and set of relationships between the objects, respectively.

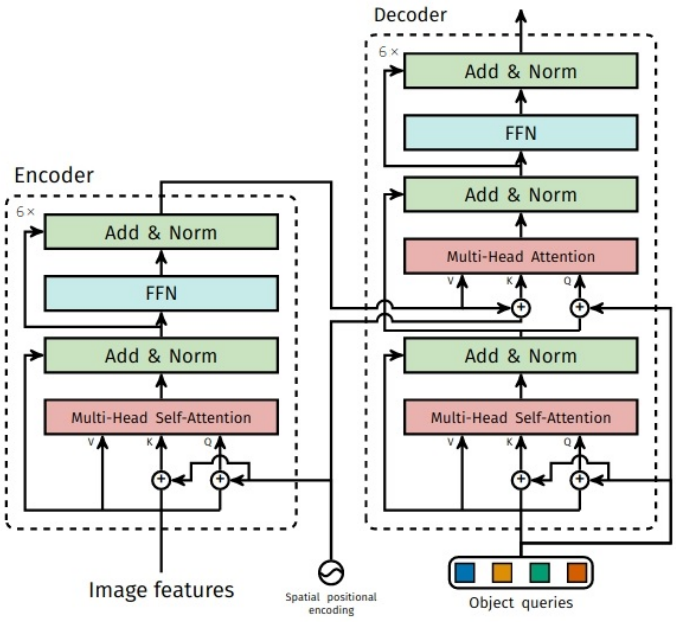


Fig. 5: The encoder and decoder components of DETR [16].

convolution which maps the C to a d-dimensional feature map $z \in \mathbb{R}^{d \times H^0 \times W^0}$, where $d = 512$.

2) Transformer: The next two components, the encoder and the decoder, are both *transformers*. A transformer is a sequence-to-sequence model that utilises *attention* to perform a transformation of an input. Attention is a mechanism that highlights the important parts of an input, i.e. which parts of the input to attend to, hence the name. The backbone CNN has extracted a number of useful features from the image, and the transformer's job is to further process those features, while enhancing the important parts and diminishing the rest. For example, musical dots are very small and may easily be mistaken for noise in the image. If the model can focus on whether the dot appears next to a note, it may help it to detect the dot. In general, the image features, encoder output, or object queries may contain many characteristics that the model can leverage when making predictions, and the goal of the attention is to enhance the useful characteristics. Since there may be many such characteristics, a single attention mechanism may become saturated and fail to capture them all. Because of this, the encoder and the decoder uses a form of attention called *multi-head* attention, as depicted in figure 5. In multi-head attention, the input is divided into d/h parts, where d is the feature dimension and h is the number of attention heads in the multi-head attention block. We use the same number of attention heads as the original DETR, i.e. $h = 8$. Each d/h slice of the input is processed by a single attention head and their outputs are concatenated afterwards. Intuitively, this allows each attention head to focus on its own characteristic in the input.

Encoder

The transformer encoder further transforms the feature maps by using 6 identical layers stacked on top of each other

containing multi-head self-attention followed by an FFN as proposed in [44], see figure 5. Since transformers are a sequence-based model, the encoder expects a sequence rather than an image. Therefore, the feature representation of the image, $d \quad H^0 \quad W^0$, is collapsed into a sequence of vectors, $d \quad H^0W^0$. Unlike other sequence-based models like LSTM which process their inputs sequentially, a transformer is able to process its inputs in parallel. While this allows significantly faster data processing, it also means the model is unable to capture order in the data, i.e. it is permutation-invariant. Because of this, the input is supplemented with a *positional encoding* that gives the model a sense of where each vector is located in the image [45]. The encoder transforms its input and therefore preserves the input's dimensions, i.e. $\mathsf{R}^{d \quad H^0W^0} \; ! \; \mathsf{R}^{d \quad H^0W^0}$.

Decoder

The decoder also follows the same architecture as in [44] containing 6 stacks of multi-head self-attention followed by cross-attention which performs multi-head attention over the output of the encoder stack, followed by an FFN, see figure 5. The decoder does not function in an autoregressive manner which means the input learned positional embeddings (object queries) of size d can be processed in parallel at each decoder layer. There is a total of N learned object queries which are added to each attention layer. The number of queries, N, must be significantly larger than the number of objects in the image. In the original DETR, N is set to 100 since they use the COCO dataset where there is an average of 7 objects per image [46]. In our case, the images contain many more instances and N is therefore required to be larger, i.e. $N = 500$, see section V-A. The N object queries are transformed by the decoder, and they are then used as input in the final prediction branches. By utilising self- and encoder-decoder attention, the model is able to globally reason about objects and their pair-wise relations with the entire image as context. Like the encoder, the decoder transforms its input and therefore keeps the input dimensions of the object queries i.e. $\mathsf{R}^{d \quad N} \; ! \; \mathbf{X} \; 2 \; \mathsf{R}^{d \quad N}$.

3) Prediction: As previously mentioned, the output of the model consists of three parts: classifications, bounding boxes, and relationships between objects. Each of these outputs is obtained separately in parallel from the transformed object queries, i.e. the decoder output $\mathbf{X}$. The classification is predicted by a linear transformation of the decoder output $\mathsf{R}^{N \quad d} \; ! \; \mathsf{R}^{N \quad C}$ where C is the number of unique classes plus one to account for predictions that should be labelled as Ø (no object). The no object class is required since there are more object queries than objects in the image.

Bounding boxes are obtained using a three layer Multilayer Perceptron (MLP) of hidden dimension d with a ReLU activation function followed by a linear projection layer $\mathsf{R}^{N \quad d} \; ! \; \mathsf{R}^{N \quad 4}$. The four values are the normalised centre coordinates, width, and height of the bounding box w.r.t. the input image. The relationships between objects are obtained as an adjacency matrix $\mathbf{A}$ by taking the dot product between all pairs of transformed object queries $\mathbf{X}$ (the decoder output) followed by a linear transformation and a sigmoid activation function as seen in equation (1).

$$\mathbf{A} = \sigma((\mathbf{X}\,\mathbf{X}^T)\,\mathbf{W} + b), \tag{1}$$

where $\mathbf{W}$ is the linear transformation's weight matrix, b is its bias, and σ is the sigmoid function. We use the dot product since it is often used in similarity measures, e.g. in transformers or as in word2vec [47], and it is therefore capable of creating latent spaces where objects with similar properties are embedded closely together. Since the output of the dot product contributes significantly to the model's training loss, it learns to generate similar object embeddings for related objects. The linear transformation is used to add learnable parameters the model can utilise to predict relations. The sigmoid activation function is used to ensure the model's prediction is between 0 and 1, since predicting relationships is a binary classification task.

C. Loss function

We denote the ground truth set of objects as y, and the model's output predictions as $\hat{y}$. The output consists of a fixed-size set of N predictions, where N is determined by the number of object queries in the decoder, i.e. $\hat{y} = f\hat{y}_i g_{i=1}^N$. Each object in y and $\hat{y}$ consists of two elements: a class $c \; 2 \; C$ and a bounding box b. The ground truth relationships between the objects are given by an adjacency matrix $\mathbf{A}$, and the predicted relationships in $\hat{\mathbf{A}}$. The model is optimised using a set-based loss function. The loss function produces an optimal bipartite matching between the predicted and ground truth objects and then optimises individual class, bounding box, and relationship losses. That is, the model's loss is calculated on the optimally matched pairs. As a consequence of using a bipartite matching loss the model learns to not predict the same objects multiple times, since only one prediction can be matched to one ground truth object. As previously mentioned, the number of predicted objects N is greater than the number of objects in any of the images of the dataset. Therefore, the labels y are padded with Ø (no object) as a means to discard excess predictions. The matching between predictions and labels is performed by the Hungarian Algorithm which is a combinatorial optimisation algorithm that solves the assignment problem in polynomial time [48]. An optimal matching $\hat{\sigma}$ is found by searching for a permutation of the N predictions $\sigma \; 2 \; \mathfrak{S}_N$ with the lowest cost:

$$\hat{\sigma} = \underset{\sigma 2 \mathfrak{S}_N}{\arg\min} \sum_{i=1}^{N} L_{\text{match}}(y_i, \hat{y}_{\sigma(i)}), \tag{2}$$

where $L_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$ is the matching cost between ground truth y_i and $\hat{y}_{\sigma(i)}$ with $\sigma(i)$ being the i-th index of permutation σ. The matching cost used to find the optimal matching is calculated based on the class and bounding box cost between pairs of matched objects, see equation (3):

$$\begin{aligned} L_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) = \quad & 1_{f c_i \in} \; g\hat{p}_{\sigma(i)}(c_i) + \\ & 1_{f c_i \in} \; g L_{\text{box}}(b_i, \hat{b}_{\sigma(i)}) \,, \end{aligned} \tag{3}$$

where c_i is the class label of ground truth object y_i and b_i is its relative bounding box coordinates. For the prediction with index $\sigma(i)$, the predicted probability of class c_i is denoted $\hat{p}_{\sigma(i)}(c_i)$, and $\hat{b}_{\sigma(i)}$ denotes the predicted bounding box coordinates. $\mathbb{1}_{\{c_i \neq \varnothing\}}$ is a set containing ones unless the class is Ø in which case it resolves to 0. This ensures that excess predictions matched with the no object label Ø are not used in the matching loss. Optimising the matching between predictions and Ø does not affect the matching cost between predictions and ground truth objects, which is the only matching cost we are interested in. The class term $\mathbb{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i)$ is negative since it describes a cost, and minimising the cost will maximise the predicted probability for the ground truth class. $L_{\text{box}}(b_i, \hat{b}_{\sigma(i)})$ is the bounding box cost.

After the optimal bipartite matching has been determined, the cost incurred by this matching is used to optimise the model. Additionally, the cost introduced by the relationship predictions $\hat{\mathbf{A}}$ is also taken into account. This loss is defined as follows:

$$L_{\text{Hungarian}}(y, \hat{y}) = L_{\text{relation}}(\mathbf{A}, \hat{\mathbf{A}}) + \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} L_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}(i)}) \right] \tag{4}$$

The loss consists of three main components: L_{relation} for the loss of the relations, the negative log-likelihood for the classes, and L_{box} for the bounding box loss. The logarithm is used for the class term since it provides larger penalties for poor predictions. Furthermore, the excess predictions that could not be matched to a ground truth object also contribute to the loss, since $\mathbb{1}_{\{c_i \neq \varnothing\}}$ is absent from the class term. This will teach the model to make empty predictions (Ø), when there are no objects to predict. However, since the number of predictions (object queries) is significantly larger than the number of ground truth objects, there will be a class imbalance where the no object class Ø will dominate. The model will therefore learn to predict Ø more often than it should. For this reason, the loss is decreased by a factor of 10 when the ground truth class is Ø.

The bounding box loss is defined in equation (5).

$$L_{\text{box}}(b_i, \hat{b}_{\sigma(i)}) = \lambda_{\text{iou}} L_{\text{iou}}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{\text{L1}} ||b_i - \hat{b}_{\sigma(i)}||_1 , \tag{5}$$

where λ_{iou} and λ_{L1} are hyperparameters determining the weight of the $L_{\text{iou}}(b_i, \hat{b}_{\sigma(i)})$ and $L1$ terms, respectively. $L_{\text{iou}}(b_i, \hat{b}_{\sigma(i)})$ is a measure of overlap between the ground truth and predicted bounding box called *Generalised Intersection over Union* (GIoU) [49], and is defined as follows:

$$L_{\text{iou}}(b_i, \hat{b}_{\sigma(i)}) = 1 - \left(\frac{|b_i \cap \hat{b}_{\sigma(i)}|}{|b_i \cup \hat{b}_{\sigma(i)}|} - \frac{|B(b_i, \hat{b}_{\sigma(i)}) \setminus b_i \cup \hat{b}_{\sigma(i)}|}{|B(b_i, \hat{b}_{\sigma(i)})|} \right), \tag{6}$$

where $|\ |$ denotes the area function of a bounding box, and $B(b_i, \hat{b}_{\sigma(i)})$ is the smallest convex hull containing both b_i and $\hat{b}_{\sigma(i)}$.

The relationship loss term in equation (4) is a binary cross-entropy loss for relations between all pairs of valid objects, and is defined in equation (7).

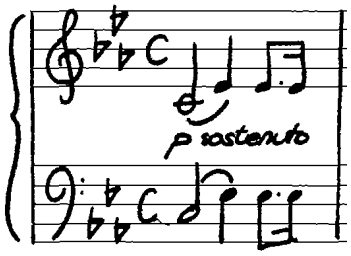
$$L_{\text{relation}}(\mathbf{A}, \hat{\mathbf{A}}) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left[\mathbb{1}_{\{c_i \neq \varnothing \ \text{or} \ c_j \neq \varnothing\}} \left(-\mathbf{A}_{ij} \log(\hat{\mathbf{A}}_{ij}) + (1 - \mathbf{A}_{ij}) \log(1 - \hat{\mathbf{A}}_{ij}) \right) \right] \tag{7}$$

V. Experiments

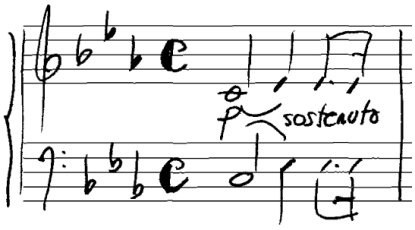
This section aims to evaluate if our augmented DETR alone can be used to perform both object detection and notation assembly in OMR. We therefore evaluate the model on an OMR dataset while comparing its results with the state-of-the-art within object detection as well as notation assembly. Furthermore, we compare the object detection score of our Relation-DETR to that of the original DETR, to investigate the effects of multi-task learning caused by predicting relations in addition to objects.

A. Dataset

We use MUSCIMA++ [50] to conduct the experiments, a dataset consisting of 140 manually annotated images of handwritten sheet music. At this point in time, MUSCIMA++ is the only OMR dataset that has annotations for both objects and their relationships. The dataset contains 91254 annotated symbols as well as 82247 relationships. It is based on the CVC-MUSCIMA dataset [51], which was originally designed for writer identification and stave-removal. There are 50 different writers represented in MUSCIMA++, with 2-3 pages from each writer, and the writers exhibit vastly different styles of writing as shown in figure 6. The dataset contains a total of 115 different classes of music symbols, and these vary greatly in both size and frequency. The 140 images are split such that 100 are used for training, and 20 are used for validation and testing, respectively. To test the model's ability to generalise, we use a strict test-split facilitated by the authors of MUSCIMA++ [50] which ensures that the test set only contains scores by writers not appearing in the training set. The images are cropped such that each sub-image contains one stave each, thus reducing the number of objects and relations in each image significantly. A full page in the dataset contains 735 music symbols on average with a standard deviation of 123. In contrast, the average number of symbols in a stave is 142 with a standard deviation of 56. However, even after cropping to staves, the



(a) Writer 1 has handwriting very similar to typeset.



(b) Writer 22 draws notes disjointly and some symbols such as the clefs and noteheads are deformed.

Fig. 6: The same bar written by two different writers to illustrate the diversity in writing styles present in MUSCIMA++.

number of objects and relationships in the images is still very high compared to other popular image datasets, making the task of object detection in OMR more challenging. To put things in perspective, the popular COCO dataset [17] contains an average of 7 instances per image [46].

B. Evaluation Metrics

Object detection and notation assembly are both evaluated using classification metrics, even though specifically predicting the bounding boxes is a regression task. In classification, we are interested in predicting True Positives (TP) and True Negatives (TN), while minimising False Positives (FP) and False Negatives (FN). For example, given an image of a half-note, a true positive would be to predict that there is an empty notehead. Similarly, when predicting its bounding box, a true positive would be a bounding box that overlaps significantly with the ground truth bounding box. As for relationships, a true negative would be to predict that there is no relationship between two given objects if no such relationship exists. These metrics are described by two overall metrics: precision and recall. *Precision* is the accuracy of the positive classifications, i.e. how often the positive classifications are correct, see equation (8).

$$P = \frac{TP}{TP + FP} \tag{8}$$

Only measuring precision can give a false representation of the model's performance, and we therefore also measure the overall ratio of true positive classifications. This is called the model's *recall*, and is defined in equation (9).

$$R = \frac{TP}{TP + FN} \tag{9}$$

Since the performance of the model is evaluated using both metrics, they are often combined in a single metric called the F_1 *score*, see equation (10).

$$F_1 = 2\frac{P \quad R}{P + R} \tag{10}$$

The F_1 score can be used to evaluate the model's overall classification score. Another way to collectively report precision and recall is to plot precision and recall as a graph, called the *Precision-Recall* graph, see figure 7. The ideal model maximises precision and recall, and this leads to the greatest area under the graph. This area is called *Average Precision* (AP)[2], and it is the most commonly used evaluation metric in object detection [11]. When the classification task includes more than two classes, e.g. 115 classes in MUSCIMA++, the AP is reported as a mean across all classes. This is called the *mean Average Precision* (mAP) but is often referred to simply as AP, since it is implied to be across all classes.

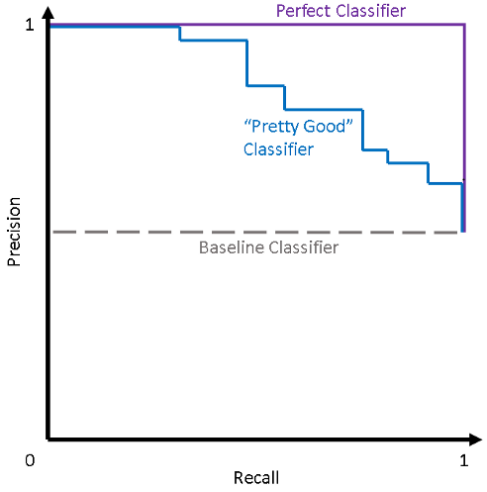


Fig. 7: Three precision-recall graphs. A good classifier will keep both precision and recall as high as possible, thus maximising the area under its graph. Image source: [53].

The method for evaluating how much a predicted bounding box overlaps with its ground truth counterpart is a simpler version of the loss function described in equation (6). It is defined in equation (11) where $\hat{b}$ is the predicted bounding box and b is the ground truth bounding box. The perfect IoU score is 1, i.e. when the area of the two boxes' intersection is as large as their areas combined. The IoU of two bounding boxes is illustrated in figure 8.

$$IoU = \frac{j\hat{b} \setminus bj}{j\hat{b} \; [\; bj} \tag{11}$$

Usually, a bounding box prediction is considered a true positive when its IoU with the ground truth bounding box is 0.5

[2]A more in-depth explanation of AP and mAP can be found in [52].

or more. We use the popular COCO evaluation metric [54] that is an average of AP at different IoU thresholds, i.e. AP at IoU = .50:.05:.95. This means we measure the model's AP at an IoU of 0.5, and then incrementally at steps of 0.05 until the final IoU of 0.95. In addition to the primary AP, the score is also commonly reported for different IoUs thresholds and object sizes. AP50 and AP75 report the AP at IoU thresholds 0.5 and 0.75, respectively, and APs, APm, and APl report the AP for small, medium, and large objects, where the sizes are based on the object's area. Objects smaller than 32^2 pixels are categorised as small, medium objects are between 32^2 and 96^2 pixels, and large objects are larger than 96^2 pixels.

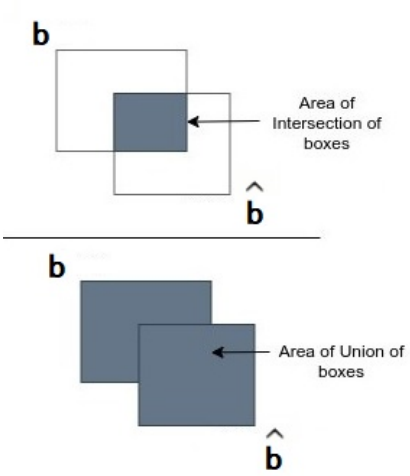


Fig. 8: The IoU of two boxes illustrated. The closer the area of intersection is to the area of union, the better the overlap is. Image source: [55].

C. Results

1) Training: The model has been trained using a transfer learning strategy, i.e. the model has been pre-trained on a different dataset before being adapted to our dataset. The model was initialised with the weights of the original DETR model which have been trained on the COCO object detection dataset. The model was then trained on the MUSCIMA++ dataset for 1.108.800 iterations corresponding to four full training days using a Google Cloud instance with 4 vCPUs and a single NVIDIA V100 GPU with a batch size of 8. Following this, the model's relations-module was added and further trained for 55.439 iterations over 2 days on our own hardware with a batch size of 1. The model was trained using AdamW optimiser with an initial learning rate of 10^{-5} and a weight decay of 10^{-4}.

2) Object Detection: The state-of-the-art object detector within OMR is MusicObjectDetector [18] (Faster R-CNN), which was later slightly refined to process stave-wise images directly [36]. In addition to comparing our model with the state-of-the-art, we also compare with the original DETR model to evaluate whether or not object detection results are improved by explicitly learning relations between the predicted objects. The results of the object detection experiments are given in table I. As the table illustrates, the performance of DETR and our model is considerably lower than MusicObject-Detector. The difference between the scores of APs, APm, and APl show that DETR struggles with detecting small objects and is significantly better at detecting larger objects. Since the majority of objects in sheet music are small, the overall AP is greatly reduced by its low score on small objects. It is furthermore evident that the relations module did not improve the performance, given the slightly lower score of our model compared to the original DETR. The MUSCIMA++ dataset contains a wide array of 115 symbols, and it is therefore of interest to inspect how the model performs on the different object classes. These results are illustrated in table II which shows the average precision on the 10 most common symbols. The per-class AP scores also show that DETR is better at detecting large objects, e.g. *slur* and *beam*, and struggles with small objects, e.g. *legerline* and *augmentation dot*. Given that the two most frequent classes, *noteheadFull* and *stem*, outnumber the other instances significantly, we would expect the model to excel at detecting these. Instead, the model fails to reach a high score on them because of their small size. This is also evident in table III which shows the 10 classes with the highest per-class AP. These objects are among the largest objects in the dataset, and even though there are very few examples of some of these classes, the model still performs well due to the objects being large.

Symbol	#Examples (training-set)	Relation-DETR	DETR
noteheadFull	19288	1.916	2.355
stem	16597	0.825	0.956
legerLine	8733	1.007	1.072
beam	6169	8.282	8.829
barline	3660	5.277	4.874
measureSeparator	3142	3.414	4.130
slur	2466	15.975	16.716
augmentationDot	2035	0.000	0.000
accidentalSharp	1578	7.106	7.950
articulationStaccato	1506	0.000	0.000

TABLE II: The per class AP performance of Relation-DETR and DETR as evaluated on the test set for the 10 most common symbols.

Model	AP	AP50	AP75	APs	APm	APl
MusicObjectDetector [36]	69.5*	-	-	-	-	-
DETR [16]	6.9719	19.2104	3.6496	4.1903	8.3341	28.1347
Relation-DETR	6.9463	18.7687	3.7223	4.3319	7.4160	27.5300

TABLE I: The performance of the state-of-the-art OMR object detector compared to our model and the original DETR. * indicates that the score was reported by the authors in their paper.

Symbol	#Examples (training-set)	Relation-DETR	DETR
staffSpace	4910	77.776	77.389
gClef	294	30.674	33.985
fClef	208	30.951	32.254
volta	8	43.498	29.191
repeat1Bar	12	20.152	28.556
cClef	131	24.583	27.051
dynamicDH	365	26.541	25.134
tupleBracket	65	18.431	25.118
restWhole	113	12.273	23.487
staffLine	3195	14.627	18.085

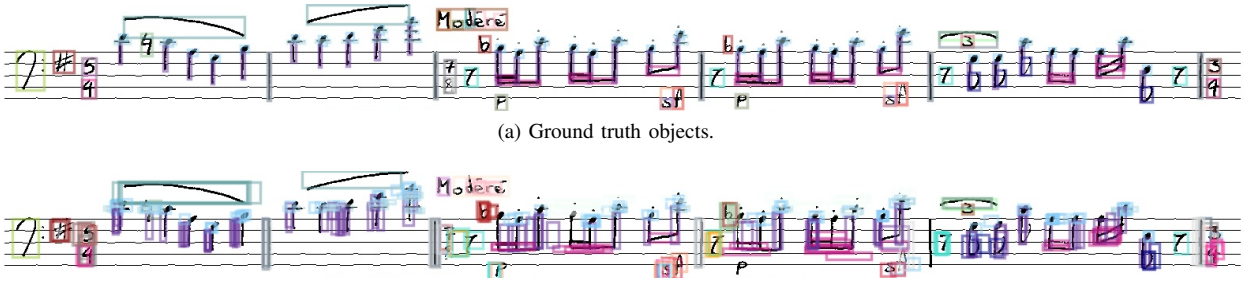
TABLE III: The 10 highest per-class AP scores of Relation-DETR and DETR as evaluated on the test set.

3) Notation Assembly: We compare our model's ability to capture relationships between music objects with the performance of Pacha et al. [36] and their MungLinker model (CNN). The relationship scores are based on binary classification metrics since we are only interested in whether or not a relation exists. Our model only predicts the relationships between identified objects, and a relationship prediction is therefore only correct if it is between two objects the model correctly identified. However, the relationships between objects the model failed to predict are still counted as errors. The notation assembly results for both models are given in table IV. When comparing the performance of the two, it is important to take into account their differences. MungLinker is, as mentioned earlier, trained and evaluated sequentially only on selected pairs of detected objects using a set of rules, i.e. how close the two objects are and whether or not they are musically compatible. The input to the model also only contains the smallest crop of the image needed to encapsulate both objects. In contrast, our model predicts the relationships between all pairs of objects, without the help of proximity and musical rules. Furthermore, it does so simultaneously in a single pass on the image while detecting the objects. This inevitably means that our model tackles a more challenging task, which consequently decreases its performance. Regardless of this, however, the results clearly show that our model is far from adequate for the task, and in no way compares to MungLinker.

Model	Precision (%)	Recall (%)	F_1-Score (%)
MungLinker	93.2	91.5	92.3
Relation-DETR	0.0024	0.4593	0.0047

TABLE IV: The relations scores between MungLinker and our Relation-DETR.

4) Qualitative Results: To get better insight into our model's behaviour we inspect its predictions directly on images from the test data. The model's predictions are illustrated and compared to the ground truth in figure 9. The figure shows that the model is able to locate almost all of the objects in the image, but it also illustrates the model's flaws. The model makes too many predictions and often predicts multiple overlapping objects, and the predicted bounding boxes are often slightly misplaced and do not always accurately enclose the objects. Another way to examine the model's behaviour is to look at its learned attention weights. The cross-attention weights in the decoder illustrate which parts of the image the decoder focuses on when making predictions, and each object query is associated with its own set of attention weights in the decoder's cross-attention. The attention weights of two object queries are shown in figure 10. While we only show attention weights for two queries, the same pattern is exhibited by all



(a) Ground truth objects.

(b) Relation-DETR object predictions.

Fig. 9: An example of our model's object detection on an image from the test set compared with the ground truth. Each object class is represented by a single colour.

Fig. 10: The cross-attention weights of the decoder (top) for two given object queries when making a prediction on the test set, where the attention is largest in the dark blue areas. The objects predicted by the object queries are marked in blue on the original figure (bottom).

object queries. As the figure shows, the attention for each query is focused on the area where it makes its prediction. The model also focuses slightly on similar object groupings other places in the stave, indicating that the model has learned to match object queries with certain patterns in the data. If the model had successfully learned to model relationships between objects, we would expect to see more attention to related objects, e.g. slurs when detecting noteheads.

VI. Conclusion

In this paper, we studied how to utilise the attention mechanisms in transformers to detect objects as well as their pairwise relationships in music scores. The model we propose is an augmented DETR with the purpose of tackling object detection and notation assembly simultaneously, the two most challenging tasks in OMR. We furthermore study how the combination of these two tasks can affect the performance of both tasks using multi-task learning. Unlike previous work that explores the utilisation of deep learning methods in notation assembly, our approach does not involve explicitly defined musical rules when detecting object relationships. However, the results of our experimental studies show that our proposed model fails to compete with the state-of-the-art within OMR object detection and notation assembly. It is evident that the model needs further refinement to perform on par with the state-of-the-art. One such refinement would be to optimise the models ability to detect smaller objects. A common method in computer vision is to increase the resolution of the feature maps, giving the model more fine-grained information about the objects. This is achieved by modifying the CNN backbone in the model, such as lowering its stride or using the output of one of the hidden layers which have higher resolution. However, larger feature maps lead to even longer training times because of the quadratic computation incurred by the attention mechanisms in transformers. It would therefore be required to reduce the model's computational complexity, e.g. by using local attention instead of global attention. This would allow the model to focus on relationships between objects that are close to one another which is appropriate for sheet music since the majority of relationships occur between objects within close proximity, e.g. a beamed group of notes. We hope that by proposing a new method for combining object detection and notation assembly in OMR, we inspire more research on combined OMR tasks and how transformers can be utilised to perform both simultaneously.

Acknowledgements

The authors would like to thank Chenjuan Guo for providing valuable supervision and feedback throughout the project.

References

[1] Jorge Calvo-Zaragoza, Jan Hajič Jr., and Alexander Pacha. "Understanding Optical Music Recognition". In: *ACM Comput. Surv.* 53.4 (July 2020). ISSN: 0360-0300. DOI: 10.1145/3397499. URL: https://doi.org/10.1145/3397499.

[2] María Alfaro-Contreras, Jorge Calvo-Zaragoza, and José M. Iñesta. "Approaching End-to-End Optical Music Recognition for Homophonic Scores". In: *Pattern Recognition and Image Analysis*. Ed. by Aythami Morales et al. Cham: Springer International Publishing, 2019, pp. 147–158.

[3] Antonio Ríos-Vila, Jorge Calvo-Zaragoza, and José M. Iñesta. "Exploring the two-dimensional nature of music notation for score recognition with end-to-end approaches". In: *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2020, pp. 193–198. DOI: 10.1109/ICFHR2020.2020.00044.

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

[5] Ana Rebelo et al. "Optical music recognition: state-of-the-art and open issues". English. In: *International Journal of Multimedia Information Retrieval* 1.3 (2012), pp. 173–190. DOI: 10.1007/s13735-012-0004-6.

[6] Elona Shatri and György Fazekas. *Optical Music Recognition: State of the Art and Major Challenges*. 2020. arXiv: 2006.07885 [cs.CV].

[7] *MIDI*. URL: https://www.midi.org/ (visited on 06/06/2021).

[8] *MusicXML*. URL: https://www.musicxml.com/ (visited on 06/06/2021).

[9] *Music Encoding Initiative*. 2021. URL: https://music-encoding.org/ (visited on 06/06/2021).

[10] Sebastian Ruder. *An Overview of Multi-Task Learning in Deep Neural Networks*. 2017. arXiv: 1706.05098 [cs.LG].

[11] Li Liu et al. "Deep learning for generic object detection: A survey". In: *International journal of computer vision* 128.2 (2020), pp. 261–318.

[12] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[13] Kaiming He et al. "Mask R-CNN". In: *IEEE Transactions on Pattern Analysis and*

Machine Intelligence 42.2 (2020), pp. 386–397. DOI: 10.1109/TPAMI.2018.2844175.

[14] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[15] Wei Liu et al. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[16] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 213–229. ISBN: 978-3-030-58452-8.

[17] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[18] Alexander Pacha et al. "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.

[19] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *arXiv preprint arXiv:1506.01497* (2015).

[20] Lukas Tuggener et al. "Deep watershed detector for music object recognition". In: *arXiv preprint arXiv:1805.10548* (2018).

[21] Jan Hajic Jr et al. "Towards Full-Pipeline Handwritten OMR with Musical Symbol Detection by U-Nets." In: *ISMIR*. 2018, pp. 225–232.

[22] Olaf Ronneberger, Philipp Fischer, and Thomas 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.

[23] David Bainbridge. *Extensible optical music recognition*. PhD Thesis. 1997.

[24] David Bainbridge and Tim Bell. "The Challenge of Optical Music Recognition". In: *Computers and the Humanities* 35.2 (2001), pp. 95–121. ISSN: 00104817. URL: http://www.jstor.org/stable/30204846.

[25] Dorothea Blostein and Henry S. Baird. "A Critical Survey of Music Image Analysis". In: *Structured Document Image Analysis*. Ed. by Henry S. Baird, Horst Bunke, and Kazuhiko Yamamoto. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 405–434. ISBN: 978-3-642-77281-8. DOI: 10.1007/978-3-642-77281-8_19. URL: https://doi.org/10.1007/978-3-642-77281-8_19.

[26] K.T. Reed and J.R. Parker. "Automatic computer recognition of printed music". In: *Proceedings of 13th International Conference on Pattern Recognition*. Vol. 3. 1996, 803–807 vol.3. DOI: 10.1109/ICPR.1996.547279.

[27] M. Szwoch. "Guido: A Musical Score Recognition System". In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Vol. 2. 2007, pp. 809–813. DOI: 10.1109/ICDAR.2007.4377027.

[28] Bertrand Coüasnon et al. "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] Michael Droettboom, Ichiro Fujinaga, and Karl MacMillan. "Optical Music Interpretation". In: *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*. Berlin, Heidelberg: Springer-Verlag, 2002, 378–386. ISBN: 3540440119.

[30] Kia Ng. *Music Manuscript Tracing*. DOI: 10.1007/3-540-45868-9n_29.

[31] Christopher Raphael and Jingya Wang. "New Approaches to Optical Music Recognition." In: *ISMIR*. 2011, pp. 305–310.

[32] Lorenzo J Tardón et al. "Optical music recognition for scores written in white mensural notation". In: *EURASIP Journal on Image and Video Processing* 2009 (2009), pp. 1–23.

[33] Donald Byrd and Jakob Grue Simonsen. "Towards a standard testbed for optical music recognition: Definitions, metrics, and page images". In: *Journal of New Music Research* 44.3 (2015), pp. 169–195.

[34] Ichiro Fujinaga. "Optical music recognition using projections". PhD thesis. McGill University Montreal, Canada, 1988.

[35] Władysław Homenda. "Automatic recognition of printed music and its conversion into playable music data". In: *Control and Cybernetics* 25 (1996), pp. 353–368.

[36] Alexander Pacha, Jorge Calvo-Zaragoza, and jr. Jan Hajič. "Learning Notation Graph Construction for Full- Pipeline Optical Music Recognition". In: *Proceedings of the 20th International Society for Music Information Retrieval Conference* (Delft, The Netherlands). Delft, The Netherlands: ISMIR, Nov. 2019, pp. 75–82. DOI: 10.5281/zenodo.3527744. URL: https://doi.org/10.5281/zenodo.3527744.

[37] Xiaojun Chang et al. "Scene Graphs: A Survey of Generations and Applications". In: *arXiv preprint arXiv:2104.01111* (2021).

[38] Jianwei Yang et al.
"Graph r-cnn for scene graph generation".
In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 670–685.

[39] Yikang Li et al.
"Factorizable net: an efficient subgraph-based framework for scene graph generation".
In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 335–351.

[40] Apoorva Dornadula et al. "Visual relationships as functions: Enabling few-shot scene graph prediction".
In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[41] Mengshi Qi et al. "Attentive relational networks for mapping images to scene graphs".
In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3957–3966.

[42] Moshiko Raboh et al. "Differentiable scene graphs".
In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 1488–1497.

[43] Kaiming He et al.
"Deep Residual Learning for Image Recognition".
In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385.
URL: http://arxiv.org/abs/1512.03385.

[44] Ashish Vaswani et al. *Attention Is All You Need*. 2017.
arXiv: 1706.03762 [cs.CL].

[45] Niki Parmar et al. *Image Transformer*. 2018.
arXiv: 1802.05751 [cs.CV].

[46] *Coco Explorer*. 2020.
URL: https://cocodataset.org/#explore (visited on 06/05/2021).

[47] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013.
arXiv: 1301.3781 [cs.CL].

[48] H. W. Kuhn.
"The Hungarian method for the assignment problem".
In: *Naval Research Logistics Quarterly* 2.1-2 (1955), pp. 83–97.
DOI: https://doi.org/10.1002/nav.3800020109.
eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800020109. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109.

[49] Hamid Rezatofighi et al.
"Generalized Intersection over Union". In: (2019).

[50] Jan Hajič jr. and Pavel Pecina.
In Search of a Dataset for Handwritten Optical Music Recognition: Introducing MUSCIMA++. 2017.
arXiv: 1703.04824 [cs.CV].

[51] Alicia Fornés et al. "CVC-MUSCIMA: a ground truth of handwritten music score images for writer identification and staff removal".
In: *International Journal on Document Analysis and Recognition (IJDAR)* 15.3 (2012), pp. 243–251.

[52] Sovit Ranjan Rath.
Evaluation Metrics for Object Detection. 2020.
URL: https://debuggercafe.com/evaluation-metrics-for-object-detection/ (visited on 05/21/2021).

[53] Doug Steen. *Precision-Recall Curves*. 2020.
URL: https://medium.com/@douglaspsteen/precision-recall-curves-d32e5b290248 (visited on 05/25/2021).

[54] *Detection Evaluation*. 2020.
URL: https://cocodataset.org/#detection-eval (visited on 05/22/2021).

[55] Vineeth S. Subramanyam.
IOU (Intersection over Union). 2017. URL: https://medium.com/analytics-vidhya/iou-intersection-over-union-705a39e7acef (visited on 05/25/2021).