# Exploration of Music Object Detection using Detection Transformer (DETR): Baseline Comparisons and Open Issues

**Tianquan Di, Hantang Li, Zhenyu Wang**
Department of Computer Science
University of Toronto
Toronto, ON M5S 2E4
{tianquan.di, hantang.li, zhenyu.wang}@mail.utoronto.ca

## Abstract

In this paper, we will experiment with the performances of the Detection Transformer (DETR) on object detection in sheet music datasets, and compare it against Faster-R-CNN, which has been consistently researched and used as a baseline model in recognizing music objects datasets and general object datasets in the past[3][8]. We will run these two models against three different types of datasets, conduct quantitative and qualitative analysis on these two methods, and conclude about existing strengths and weaknesses that they have. We will then discuss future works and directions of improvement that researchers could approach.

## 1 Introduction

Optical Music Recognition (OMR) has been an existing problem researched over the last five decades, and the primarily goal of OMR is to recognize the music symbols and notations from images and then convert them into digital formats (e.g. MIDI files, XML files, etc.) [1] OMR is a challenging problem due to ambiguous notation, large variations in musical font, high density of information, and musical image's structural complexity [2]. However, as neural networks becoming more applicable, we see a new potential in solving the OMR problem. The pipeline of a deep-learning based OMR system are divided into four steps, according to ref. [3]:

- Preprocessing: Increasing the contrasts of the image, removing unnecessary noises on the image input, scaling the image, removing staff lines.
- **Music Object Detection:** Given the preprocessed image, detect the locations and the labels of the music symbols that are presented in the image.
- Semantic Reconstruction: Reconstruct the semantic of the music based on the obtained the locations and the labels of the music symbols in the image.
- Encoding: Based on the reconstructed music semantic, output the final digital music format such as MIDI, XML, etc. These types of format can be imported into music editing software to conduct further processing and music production.

The second step, i.e. music object detection, is the most challenging step [2][3][4], and it will be the main focus of our experiment.

In March 2020, Carion et al. introduced the Detection Transformer (DETR) [5], which opened another possibility for approching object detection, by using Transformer-based models. Using DETR on music scores dataset is a novel combination that no researchers have ever attempted before, so we believe that it is an interesting experiment for us to explore. Also, due to the unique nature

Preprint. Under review.

of music scores datasets, every objects in a score image will be having interconnected relations [6]. Therefore, we would like to see if DETR is capable to utilize these interconnections between objects to achieve a superior performance. To set up a baseline for comparison, we will also be using Faster Region-based Convolutional Neural Network (Faster R-CNN), which was originally proposed by Ren et al. [7] to compare against DETR. Although the performance of Faster R-CNN has already been bench-marked against some other non-Transformer models in music object detection (such as ref. [3][8]), there have never been an experimental comparison against Transformer-based model such as DETR in the field of OMR. Thus, how would DETR perform in regards to Faster R-CNN is also a novel aspect that we would explore in this experiment.

## 2 Experimental Setup

### 2.1 The Architectures

DETR: As mentioned in Section 1, DETR is a recently introduced transformer-based detector. The main ingredients of DETR are a set-based global loss that forces unique predictions via bipartite matching and a transformer encoder-decoder architecture. [5] We will use ResNet-50 as the feature extraction stage as well for being a controlled variable.

Faster R-CNN: Faster R-CNN is a wildly used two-stage detector, with the first stage generating a sparse set of region proposals classified and further refined in the second stage[7]. In order to make sure the comparison with DETR is fair, we will also use ResNet-50 as the feature extractor.

### 2.2 Datasets

We will be using two distinct datasets, namely DeepScores and MUSCIMA++ v2. The original version of these datasets consist of over 10,000 music scores, but to simplify the training process, we will be using simplified versions of these dataset organized by [6]. We adapted works from literature [3] to normalize both dataset by removing unused images and unify the image name and annotation format. Both datasets contains a large number of music score images, as well as the annotations of the music symbols. However, images in DeepScores only consists of printed scores, whereas images in MUSCIMA++ v2 consists of hand-written scores.

In addition, we will also use a preprocessing technique introduced in ref.[9], which proposed a sliding-windows approach that crops a entire music score into small snippets, where each snippet will contain music symbols in one staff group, and the length of each snippet will be less or equal to 250. The DeepScores dataset does not have annotations for the positioning of the staff lines, so we are only able to conduct the cropping to the MUSCIMA++ dataset.

A detailed specification of all the datasets has been summarized in Table 1.

Table 1: A summary of music score datasets that are being used in the experiment

| Name | Typography | # of images | # of objects | Width | Height |
|---|---|---|---|---|---|
| DeepScores | Printed | $\sim 140$ | $\sim 80$ | $\sim 2000$ | $\sim 3000$ |
| MUSCIMA++ v2 (Not Cropped) | Handwritten | $\sim 70$ | 17739 | $\sim 800$ | $\sim 600$ |
| MUSCIMA++ v2 (Cropped) | Handwritten | 4794 | 17739 | $\sim 250$ | $\sim 200$ |

### 2.3 Training and Evaluation

For training and evaluating the performance of DETR, we will conduct several experiments to study the effect of dataset size, dataset content and the effect of applying image cropping.

1. We adapted the DETR training code from Facebook's official github page and we used the Detectron2 framework from Facebook Research for training the Faster-R-CNN. [1] Then we train both models by fine-tuning models that was pretrained on the coco dataset. We use the default hyper-parameter setting for the OMR baseline paper [1] when training Faster R-CNN. We use two images per GPU as the batch size when training DETR because batch size has a minor effect on the test result.

[1] Please see the appendix for more details about codes used for the experiments in this paper.

2. We use the Average Precision (AP) score[2] as defined by the COCO challenge to evaluate the performance of the model. For each training process, we train until the AP score converges and then record the highest value. We keep the models for analyzing their prediction result on test set.

## 3 Experiment Results and Analysis

The performance of DETR and Faster-R-CNN has been summarized in Table 2, which includes the converged AP score as well as variants of the AP scores on the test set. As we can see, both DETR and Faster R-CNN have drastically different performances between cropped datasets and non-cropped datasets.

Table 2: A summary of average precision (AP) produced by each architecture

| Architecture | Dataset | Cropped? | $AP$ | $AP_{50}$ | $AP_{75}$ | $AP_{small}$ | $AP_{medium}$ | $AP_{large}$ |
|---|---|---|---|---|---|---|---|---|
| DETR | DeepScores | No | 0.1 | 0.2 | 0.0 | 0.0 | 0.0 | 0.0 |
| Faster R-CNN | DeepScores | No | **2.7** | **4.9** | **2.7** | **1.2** | **2.5** | **19.7** |
| DETR | MUSCIMA++ v2 | No | 0.0 | 0.1 | 0.0 | 0.0 | 0.1 | 1.0 |
| Faster R-CNN | MUSCIMA++ v2 | No | **2.2** | **4.0** | **2.0** | **0.8** | **2.7** | **7.5** |
| DETR | MUSCIMA++ v2 | Yes | 22.4 | 33.4 | 25.5 | 22.9 | **26.2** | **17.0** |
| Faster R-CNN | MUSCIMA++ v2 | Yes | **25.6** | **34.0** | **29.0** | **27.8** | 23.4 | 13.6 |

Although both architectures performed poorly on non-cropped datasets, it is worth noting that Faster-R-CNN still has an edge over DETR when evaluating against non-cropped datasets. As shown in figure 2, Faster R-CNN architecture can detect most objects across all three datasets. On the other hand, there is a much more significant gap between the performance of DETR on cropped and non-cropped datasets. It obtains 0 in 3 of 6 AP score measurements. It suggests the disappointing fact that DETR is completely dysfunctional on a large, complete music score image.

However, when detecting small, cropped images, DETR architecture reaches the AP score of 22.4 in MUSCIMA++ v2 (Cropped), which is competitive comparing to Faster R-CNN. The mechanism of cropping the MUSCIMA++ dataset is to transform the size of objects (music notations) to be larger while maintaining other features of the objects. Therefore, we can conclude that DETR's capability is significantly increased when the overall sizes of music objects in the image are larger. We also noticed that $AP_{medium}$ and $AP_{large}$ of DETR is marginally better than Faster R-CNN. This indicates DETR has a slightly stronger ability in large object detection comparing to Faster R-CNN.

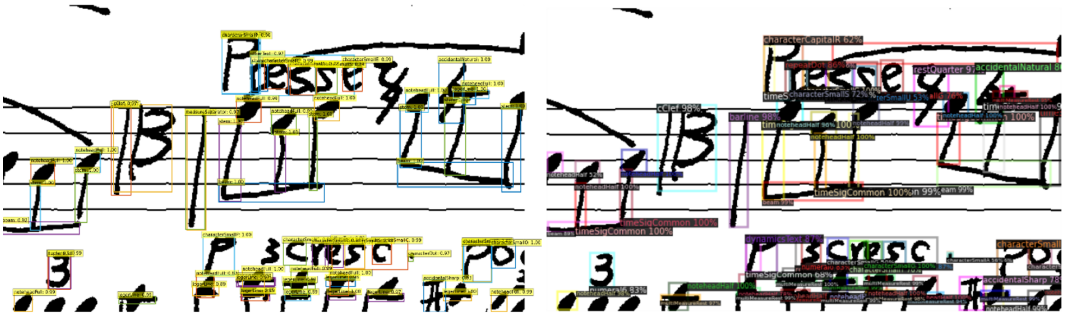

Figure 1: Left: DETR's output on MUSCIMA++ v2 (Cropped) test set. Right: Faster R-CNN's output on MUSCIMA++ v2 (Cropped) test set.

In comparison with Faster R-CNN, DETR attains a high accuracy in detecting items inside a bounding box, but more objects are missed and more bounding boxes are incorrectly stacked. Such as the model draws bounding boxes for both "ss" and each "s" inside. On the other hand, faster R-CNN has more misclassification than DETR. For instance, it misclassified the character "p" at the top to "R".

For better understanding and analyze DETR's performance, we print DETR's decoder and encoder's self-attention heatmap for a specific object to study how the model learns and predicts contents.

[2]The official definition of AP score is defined by COCO at https://cocodataset.org/#detection-eval

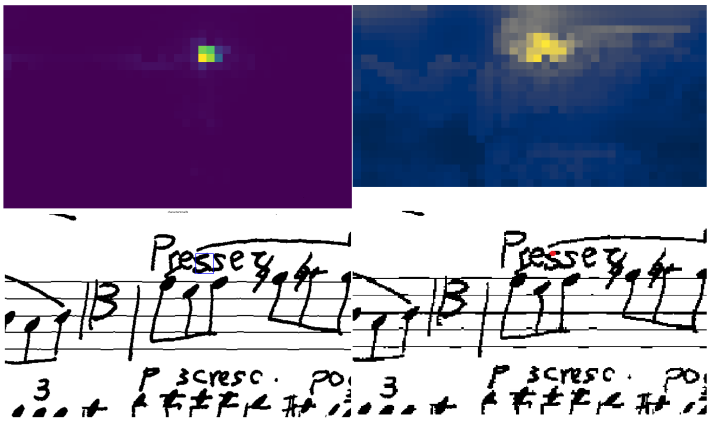

Figure 2: Left: decoder attention weights heatmap, Right: encoder attention weights heatmap

Since DETR compresses the images' height and weight by 32, the heatmap is mosaic shaped. The decoder's attention weight is obtained by tracing the index of the decoder's bounding box. By looking at the decoder's attention weight heatmap, we observed that the model also pays attention to nearby words while deciding the bounding box for "s." Furthermore, that may cause the model to have low confidence while detecting "s." The encoder's attention weight is obtained by selecting a point in the picture and then calculating the corresponding attention weight's index on the layer. We can understand what model learnt by examining the encoder's attention weight. According to what model learnt, object "s" has a relationship with nearby objects. Since theoretically detecting a letter in a word should not pay a similar amount of attention to both the letter itself and surrounding letters, the model might be overfitted as there are limited words that contain the letter "s" in music scores.

By analyzing the structure of both algorithms, we can have a better understanding of the test result. Firstly, for Faster R-CNN, the region proposal network generates anchors with diverse sizes and locations to cover possible small object candidates. In the uncropped MUSCIMA++ dataset, a sample image has 800 by 600 resolution, approximately 17000 anchors are generated (with default hyperparameter setting).[10] Secondly, it was found that Faster R-CNN has a strong scale-invariant property.[11] Generated proposals are in different sizes; Rol Pooling layers transform them into fixed-length vectors. For DETR, the model struggles to perform well when the target is small and crowded. Compare to Faster R-CNN's anchors, DETR learns N positional encodings as the decoder's input and draws N bounding boxes based on a combination of non-scale-invariant loss and a scale-invariant loss. Note N equals to 100 for our model. As most uncropped music scores contain more than 100 objects to detect on a page, we have a post hoc hypothesis that the more number of objects in an image exceeds N, the worse the model performs. And our experiment result does verify our hypothesis. Note that music scores in the DeepScores dataset contain more objects than music scores in MUSCIMA++ V2.

## 4 Conclusion and Future Works

In conclusion, we have taken the initiative to experiment with a Transformer-based model on music object detection and conducted a formal comparison against Faster R-CNN. We discovered that DETR struggled to conduct proper prediction on complete music score images but produced a slightly superior result (comparing to Faster RCNN) on cropped music scores where the target sizes are relatively increased. While it is obvious that failing to detect the targets on a full-page music score makes DETR far from being a perfect music object detector, it is still worth noting that DETR still has considerable potential if its advantage in medium-large targets can be transferred to small targets in the future. For instance, recently, more Transformer-based approaches aim to address the issue of DETR on small targets and claim to have superior performance than DETR. Hence, we believe that there are opportunities for researchers to evaluate these recent models against music scores datasets in the future. For instance, Zhu et al. presented a variant of DETR, namely "Deformable DETR", which claims to be superior to DETR, and it fixes the issue of detecting small objects[12], which is prominent in our experiment. Overall, It would be interesting for future OMR researchers to explore the possibility of using more advanced Transformer architectures other than DETR.

# Appendix

### Source codes for reproducing the experiment

We used Google Colab (Pro) to facilitate the experiment and store the codes used during the experiment. The hyper links at each row of table 3 corresponds to one colab notebook with the specific architecture and dataset configuration.

Table 3: Source codes

| Architecture | Dataset | Cropped? | Hyper Link |
|---|---|---|---|
| DETR | DeepScores | No | **here** |
| DETR | MUSCIMA++ v2 | Yes | **here** |
| DETR | MUSCIMA++ v2 | No | **here** |
| Faster R-CNN | DeepScores | No | **here** |
| Faster R-CNN | MUSCIMA++ v2 | Yes | **here** |
| Faster R-CNN | MUSCIMA++ v2 | No | **here** |

All the experiments are conducted on Google Colab, and you can click into any of these notebook to reproduce all the experiments. **If you cannot click the hyper link, we also have a github repository that contains a readme file, which stores links to all the colab notebooks we used during this experiment: https://github.com/AndyTQ/csc413-omr-detr-colab**

Additional Note:

1. Inside all the colab notebooks we also used a data preprocessor to preprocess the image, which is adapted and modified from github.com/apacha/MusicObjectDetector-TF and github.com/apacha/MusicObjectDetection/. The original author of this data preprocessor is Alexander Pacha.
2. You can find our modified data preprocessor repo at the following address: https://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@github.com/AndyTQ/csc413-omr
3. For notebooks whose architecture is DETR, these notebooks are adapted from a fork of Facebook's official DETR colab notebook.
   - The original Facebook DETR colab notebook is at: https://colab.research.google.com/github/facebookresearch/detr/blob/colab/notebooks/detr_demo.ipynb
   - The fork that we adapted for fine-tuning the DETR is at: https://colab.research.google.com/github/woctezuma/finetune-detr/blob/master/finetune_detr.ipynb.
4. For notebooks whose architecture is Faster R-CNN, we adapted the Detectron2 framework from Facebook Research. The Detectron2 framework allow us to quickly set up popular object detection architectures (including Faster R-CNN) and speed up the research. We adapted a Colab version of the Detectron2 framework, which is available at http://tiny.cc/ipawtz.

### Author Contributions

- Tianquan Di: proposed the direction of the topic (Optical Music Recognition); wrote project proposal; decided which dataset to use for the experiment; gathered necessary open source data loader to preprocess the data and adapted them specifically for this project; gathered necessary open source code for DETR and Faster R-CNN architectures and adapted them in order to run the experiment; conducted the actual training of the models and recorded the data; debug the issues encountered during the training; wrote section 1, 2.2, and 4 in the final report; helped with editing and improving section 3 in the final report.

- Zhengyu Wang: gathered neccessary open source code for feature visualization (i.e. heatmap) for DETR; conducted the actual training of the models and recorded the data; debug the issues encountered during the training; wrote section 3 in the final report; explore and explain the result in architectural perspective; helped with improving other sections in the final report.
- Hantang Li: analyzed the mechanism of both encoder and decoder heatmaps found by Zhengyu, then use those heatmaps for analyzing DETR's output and structure; helped with DETR model training and experiments; tested DETR and Faster R-CNN's output on each dataset's test set; wrote section 2.1, 2.3, 3 (analyze output and heatmaps) in the final report.

**Abbreviations**

We used the following abbreviations throughout this paper:
OMR: Optical Music Recognition
AP: Average Precision
DETR: Detection Transformer
R-CNN: Region Based Convolutional Neural Networks
COCO: Common Objects in Context
MIDI: Musical Instrument Digital Interface
XML: Extensible Markup Language

## References


[1] Shatri, Elona & Fazekas, György. (2020) "Optical Music Recognition: State of the Art and Major Challenges." *arXiv preprint*. arXiv:2006.07885, .

[2] Calvo-Zaragoza, J., Jr., J. H., Pacha, A. (2020). Understanding optical music recognition. *ACM Computing Surveys*, 53(4), 1-35. doi:10.1145/3397499

[3] Pacha, A., Hajič, J., Calvo-Zaragoza, J. (2018). A baseline for general music object detection with deep learning. *Applied Sciences*, 8(9), 1488. doi:10.3390/app8091488

[4] Pacha, A., Choi, K., Couasnon, B., Ricquebourg, Y., Zanibbi, R., Eidenberger, H. (2018). "Handwritten music object detection: Open issues and baseline results". *13th IAPR International Workshop on Document Analysis Systems (DAS)*. doi:10.1109/das.2018.51

[5] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S. (2020). "End-to-end object detection with transformers". *arXiv preprint* arXiv:2005.12872

[6] J. j. Hajic and P. Pecina. (2017) "In search of a dataset for handwritten ˇoptical music recognition: Introducing MUSCIMA++" *arXiv preprint* arXiv:1703.04824

[7] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. (2015) "Faster R-CNN: Towards real-time object detection with region proposal networks." *Neural Information Processing Systems (NIPS)*

[8] Tuggener, Lukas & Satyawan, Yvan & Pacha, Alexander & Schmidhuber, Jurgen & Stadelmann, Thilo. (2021). "The DeepScoresV2 Dataset and Benchmark for Music Object Detection." *International Conference on Pattern Recognition*

[9] Lukas T., Ismail E., Jürgen S., Marcello P. & Thilo Stadelmann (2018). "DeepScores – A Dataset for Segmentation, Detection and Classification of Tiny Objects." *IEEE International Conference on Pattern Recognition*

[10] Bharat.S & Larry S. (2018). "An Analysis of Scale Invariance in Object Detection – SNIP" *Conference on Computer Vision and Pattern Recognition*.

[11] Wang, Y., Liu, Z., Deng, W. (2019). Anchor generation optimization and region of interest assignment for vehicle detection. *Sensors*, 19(5), 1089. doi:10.3390/s19051089

[12] Zhou, X., Su, W., Lu L., Li B., Wang X., Dai J. (2021). "Deformable DETR: Deformable Transformers for End-to-End Object Detection". *International Conference on Learning Representations (ICLR)*