Improved Optical Music Recognition (OMR)

Justin Greet
Stanford University
jgreet@stanford.edu

Abstract

This project focuses on identifying and ordering the notes and rests in a given measure of sheet music using a novel approach involving a Convolutional Neural Network (CNN). Past efforts in the field of Optical Music Recognition (OMR) have not taken advantage of neural networks. The best architecture we developed involves feeding proposals of regions containing notes or rests to a CNN then removing conflicts in regions identifying the same symbol. We compare our results with a commercial OMR product and achieve similar results.

1. Introduction

Optical Music Recognition (OMR) is the problem of converting a scanned image of sheet music into a symbolic representation like MusicXML [9] or MIDI. There are many obvious practical applications for such a solution, such as playing back the audio for a piece of music the computer has never seen.

The input to our algorithm is an image of a single measure of sheet music. Note that this measure is expected to be monophonic so that there is at most a single note active at a time. We then propose regions potentially containing a single note or rest, pass each region to a CNN to classify it as background or a music symbol, remove duplicate regions identifying the same symbol, then order each symbol by the horizontal location of its region. The output of the algorithm is an ordered set of the pitches and durations of the symbols present in the measure.

We evaluate each predicted note in the output along its duration, pitch, and order in the measure. For each symbol in the labelled input image, we determine how well it is represented in the output along the same dimensions. More precisely, given an input measure X with notes and rests $R = \{r_1, r_2, ..., r_n\}$ and an output measure Y with notes and rests $S = \{s_1, s_2, ..., s_m\}$ we calculate a percentage accuracy by comparing the pitch and detection of each note in R to the note at the same index in S, and comparing the duration of each rest in R to the duration of the rest at the same index in S.

Observe that the order of the notes is implicitly measured. Any note missing from the output, out of place, or erroneously added heavily penalizes the percentage. More qualitatively, we can render the output of the algorithm as sheet music and visually check how it compares to the input measure.

Multiple commercial OMR products exist and their effectiveness has been studied to be imperfect enough to preclude many practical applications [4, 20]. We run the test images through one of them (SmartScore [12]) and use the evaluation criteria described above to serve as a benchmark for the proposed algorithm.

2. Related Work

The topic of OMR is heavily researched. The research can be broadly broken up into three categories: evaluation of existing methods, discovery of novel solutions for specific parts of OMR, and classical approaches to OMR.

Research into evaluation of existing methods includes areas like combining the output of multiple commercial OMR products to achieve a better overall result in general [4, 20] and in more narrowly-defined problems [22, 5]. The research also explores facilities for a more standard method of evaluation [11, 16].

The general problem of digitizing an entire piece of sheet music contains many component parts, and research has been conducted into a number of them. This includes the detection of rhythm [14, 7], measures comprising the sheet music [29], and the instruments for which the piece is intended [13, 27]. In addition, there has been some exploration of analyzing the quality of scanned images [25].

There have also been numerous approaches to the same problem as our algorithm attempts to solve, none of which take advantage of modern deep learning techniques. The most common approach is to employ some kind of traditional pattern matching [18], using flavors such as k-nearest neighbors [6] , hidden Markov models [21], correlations followed by an SVM [23], and a single convolution [24]. While relative success has been achieved, these methods all suffer from the fundamental lack of ability to deal with var-

ied conditions present in real-world images.

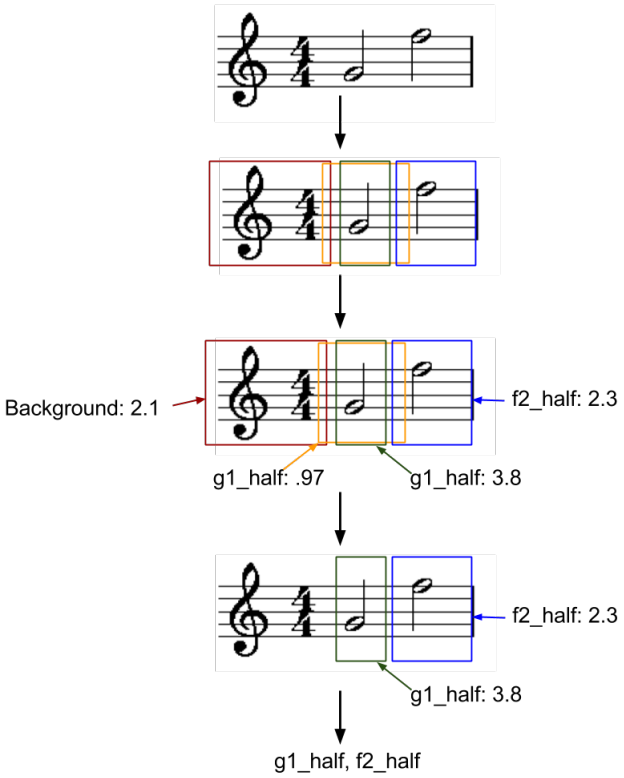
The state of the art is achieved in the commercial space whose methods are opaque.

3. Methods

Our method of symbol detection is loosely based on R-CNN [10] and follows a similar structure. R-CNN is an algorithm for object detection that takes as input an image with any number of objects, and returns both the labels of those objects and predicted bounding boxes surrounding them.

There are three main components to our algorithm. First, we generate proposals for regions of the measure that might contain a single symbol. Next, we run these regions through a CNN which identifies it as either a symbol or background and assigns an associated score. Finally, we remove duplicate regions that identify the same symbol and output an ordered set of notes and rests. See Figure 1 for a sketch of the overall algorithm.

Figure 1: A sketch of the model we employ.



3.1. Region proposals

The input to this stage is a single image of a measure. The goal is to produce a set of symbol-independent regions that capture all of the symbols present in the measure. For our purposes the goal of a region is to capture exactly one symbol in its entirety. A number of algorithms exist for this purpose for the more general problem of object detection such as selective search [28], category-independent object proposals [8], and objectness [2].

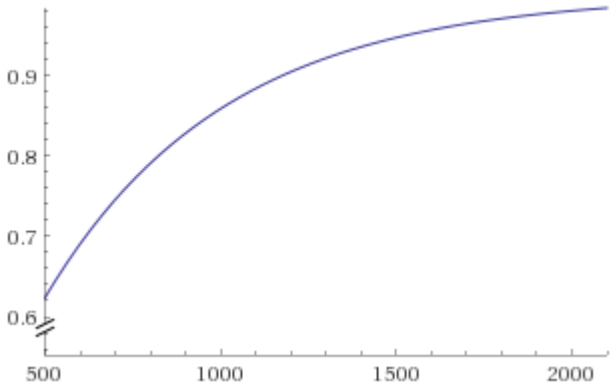
While these algorithms are effective for the general problem of two-dimensional region proposals, our problem is constrained in a way that allows us to make a simplifying assumption. The pitch of a note on a music staff is defined precisely by its vertical position. Therefore for a region proposal to contain all the information needed to identify a note's pitch and duration it needs to capture the entire vertical axis. So our problem is reduced to one dimension. That is, a region proposal on a measure consists solely of two points on the horizontal axis.

So we employ a random approach to region proposals. Assume we're given a measure of width w containing n symbols. Assume for simplicity that each symbol occupies w/n space. If we propose m random regions, the probability that a given symbol is captured by at least one region is (see the appendix for the full proof):

$$1 - \left(1 - \frac{1}{2n^2}\right)^m$$

By inspecting the graph in Figure 2, we see that 1,800 is the lowest round number that gives us at least a 97% chance of capturing a given note in the worst case when $n = 16$. So we propose 1,800 random regions. Note that this is lower than the $2,000$ regions proposed by selective search.

Figure 2: Probability of capturing a note with different number of proposed regions.

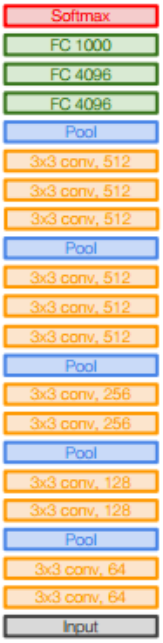


3.2. CNN

Given a region proposal, we'd like to classify it as one of the 60 types of valid symbols or as background. To accomplish this we build a CNN on top of VGGNet [26], a 16-layer network trained on the ImageNet dataset that won

second place in the ImageNet Challenge for classification in 2014. It uses a series of small convolutions of size $3x3$ to reduce the number of trainable parameters while maintaining the same receptive field as a network with larger convolutions. See Figure 3 for an image of the full network architecture.

Figure 3: The network architecture of VGGNet-16.



There are three fully-connected layers at the end of the network. The last one traditionally has a size of 1,000 to assign class scores to each of the 1,000 image types in ImageNet. We replace this last layer will a fully-connected layer of size 61 for each of the 61 potential classes of symbols (60 symbol types plus one for background).

During training we use cross entropy loss, which assigns normalized class probabilities to each of the 61 classes. Because we're dealing with monophonic data each input image has exactly one correct label. More specifically, for a given training image m the loss L_m is:

$$L_m = -\log \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

where f_{y_i} is the class score of the correct label and f_j is the class score for class j. The total loss is the average loss across all input images. We use the Adam optimizer [17], an algorithm for gradient-based optimization which features an adaptive learning rate with momentum. That is, the effective step size changes throughout training and we take steps based on a history of the gradient rather than the instantaneous gradient alone.

We train the last layer on its own for 5 epochs with a batch size of 32 and learning rate of .001, then train the entire network for 7 epochs with a batch size of 32 and a smaller learning rate of .00001.

3.3. Duplicate removal

Once each of the 1,000 regions has a respective label and score, we drop any region classified as background. We also have to be careful to ensure each symbol is captured by at most one region while allowing for the possibility that the same symbol occurs multiple times in a row.

To accomplish this, we drop any region who shares an intersection-over-union of greater than the learned rate of .13 with another region of the same classification and a higher score. In other words, if two regions A and B have an intersection of I, if $I/(A+B-I) >= .13$ and the score of A is less than the score of B, we drop region A. This learned rate is fairly low. That's because we have enough regions so that if we incorrectly reject a region because it intersects with a region of another symbol, there is likely to be another region capturing that original symbol that doesn't intersect with the other.

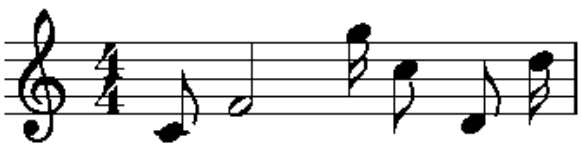
Finally, we order the remaining symbols by the left position of their regions and return that list as output. For a monophonic measure this is enough information to recreate it perfectly.

4. Dataset

Labelled sheet music is hard to come by and very time consuming to create manually. Instead, we generate our own labelled data programatically. We first generate a symbolic representation of a random measure of sheet music with some small actions taken to make the generated data more realistic. This includes things like preventing two consecutive rests and increasing the likelihood of a repeated note. A measure can contain anywhere between 1 and 16 symbols (notes plus rests).

We then use the Lilypond music engraving program [19] to render this representation as an image. Bounding boxes are stored around the locations of each note and rest. See Figure 1 for an example of a generated measure. In total, we generate 22,000 training measures, 8,000 validation measures, and 200 test measures.

Figure 4: Example of a generated measure of sheet music.



While we use the training measures and validation measures in their entirety for experimentation, their primary use is in our CNN. The CNN expects a single image of a note, rest, or background. So we slice each test measure into its component symbols, in addition to random slices for background. There are 13 different pitches our network can accept (plus 1 for rests) and 5 different durations. For each of the 60 kinds of symbols we ensure there are enough measures to generate 175 training samples and 45 validation samples. In addition we have 5,000 background samples for training and 1,000 for validation.

When slicing the measures we take loose bounds on the notes and rests to add some noise. In addition, we perform data augmentation on the symbol images before passing them to the CNN by performing a random small rotation and randomly adjusting the scale. The images are then resized to 224 x 224 pixels before being passed through the network. See Figure 2 for sample data augmentation before the final 224 x 224 resizing. Note that the test measures undergo similar data augmentation.

(a) Original (b) Rotated (c) Scaled

Figure 5: Data augmentation of note

5. Results and Discussion

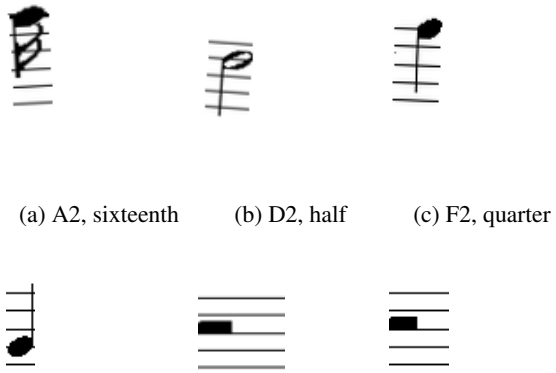
The goal of our algorithm is to understand and be able to re-create the music symbols present in an input image of a measure. To do that we first evaluate individual regions and classify them individually as music symbols or background. In order to understand the results of the algorithm as a whole we first need to evaluate the performance of this classifier.

5.1. CNN Results

As discussed above the CNN we use to classify the music symbols is built largely on top of VGGNet. It takes in an image of either one of the 60 types of notes (12 possible pitches plus one for rests, and 5 possible durations) or background (any image in a measure that doesn't exactly capture a single note) and attempts to classify it.

To test the performance of the CNN we generate 3,000 random measures of labelled music, then slice them into smaller images of only the individual music symbols. We also generate random background images for each measure. We then group the individual images by label and limit the maximum number of test images for any single music symbol to 15 to prevent any single symbol from having an outsize effect on our results. In total we collected 612 individual images for an average of 10.03 images of each symbol.

The CNN is correct for a given image if the symbol predicted by the CNN matches that image's true label. For the 612 input images the CNN predicted the correct label for 606 (99.0%) of them. The incorrect predicted label was the same for all of them: background. The 6 images that were labelled incorrectly can be seen alongside their actual labels in Figure 6. See Figure 7 for the confusion matrix for these notes. The one discernible pattern across the 6 images is the relatively zoomed-in nature for them. On average these images have a width of 78% the width of the average measure across all 612 samples.



(a) A2, sixteenth (b) D2, half (c) F2, quarter

(d) G1, quarter (e) rest, half (f) rest, half

Figure 6: The six images incorrectly labelled as background. Captions are of form of (pitch, duration).

We first train the last fully-connected layer of the network on its own for 5 epochs with a batch size of 32 and learning rate of .001, then train the entire network for 7 epochs with a batch size of 32 and a smaller learning rate of .00001. The Adam optimizer, which is described above, is used. The exact values used for learning rate and batch size do not end up mattering too much. Alterations of these values result in similar performance. This could be for one of two reasons: either the images of music symbols have a lot in common with the ImageNet dataset on which VGGNet was originally trained that the network could suss out, or this is a straightforward classification problem. They both seem likely in ways as we were able to get accuracy of 60.4% with a simple 5-nearest neighbors approach.

It is also interesting to note that the network seems a little too eager to assign the background label. This is borne out further when we test the classifier on a clean image of a

		Predicted class					
		A2, sixteenth	D2, half	F2, quarter	G1, quarter	rest, half	background
Actual class	A2, sixteenth	**13**	0	0	0	0	**1**
	D2, half	0	**9**	0	0	0	**1**
	F2, quarter	0	0	**13**		0	**1**
	G1, quarter	0	0	0	**14**	0	**1**
	rest, half	0	0	0	0	**13**	**2**

Figure 7: The confusion matrix for the subset of notes that were classified incorrectly at least once.

note we did not generate. Like the 6 images above, this image also gets the label of background (see Figure 8 for this image). This is a signal that the noise we generate in the images isn't realistic enough in the general sense, and also perhaps that background images played too big a role in training.

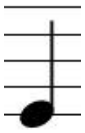


Figure 8: A note that we didn't generate that gets misclassified as background.

5.2. End to end results

To check that our system works we want to verify that the input measure can be re-created from the symbols we detect. That is, the system is correct if for a given image if we predict exactly the right music symbols in the right order. More formally, given an input measure X with notes and rests $R = \{r_1, r_2, ..., r_n\}$ and a measure Y with notes and rests we predict $S = \{s_1, s_2, ..., s_m\}$ we calculate a percentage accuracy by comparing the pitch and detection of each note in R to the note at the same index in S, and comparing the duration of each rest in R to the duration of the rest at the same index in S.

Observe that the order of the notes is implicitly measured. Any note missing from the output, out of place, or erroneously added heavily penalizes the percentage. More qualitatively, we can render the output of the algorithm as sheet music and visually check how it compares to the input measure.

To perform our analysis we generate 200 test measures. 50% of those measures have no additional noise applied, while the remaining 50% undergo the same small transformations as the individual notes we generate: some combination of change of scale and rotation. We present the results with both a normalized view (divide the percentage by the total number of symbols in the input measure) and a non-normalized view.

Overall the results are encouraging. As we can see in Figure 9, 107 of the 200 test measures were predicted with 90-100% accuracy. The remaining 93 measures performed fairly uniformly between 10-89% accuracy. It is important to re-emphasize that an extraneous note predicted early in a sequence throws off the performance of the entire measure. In fact, this exact thing happened in 87.1% of the measures with less than a perfect score.

While the overall performance is encouraging, there is a large discrepancy between the performance of the measure as a whole versus the CNN to predict music symbols. This can largely be attributed to one thing. The notation for an eighth note and a sixteenth note is identical to that of a quarter note with additional "tails" coming off the side. See Figure 10 for an illustration of this. The region proposals did a very effective job of capturing instances of eighth and sixteenth notes. But they also captured regions of these notes with the tails cut off so that it appeared as a quarter note. This explains the additional note in the majority of measures predicted imperfectly.

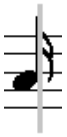


Figure 10: The left side of the line looks like a quarter note, while the entire note is actually a sixteenth note.

The remedy for this can be one of two things: make the region proposal algorithm more sophisticated so that it never captures a portion of an eighth or sixteenth note that looks like a quarter note; or make the duplicate removal logic more sophisticated. Rather than simply calculate intersection over union between regions with the same classification, this can be extended to operate on scenarios where two notes of the same pitch but different durations overlap.

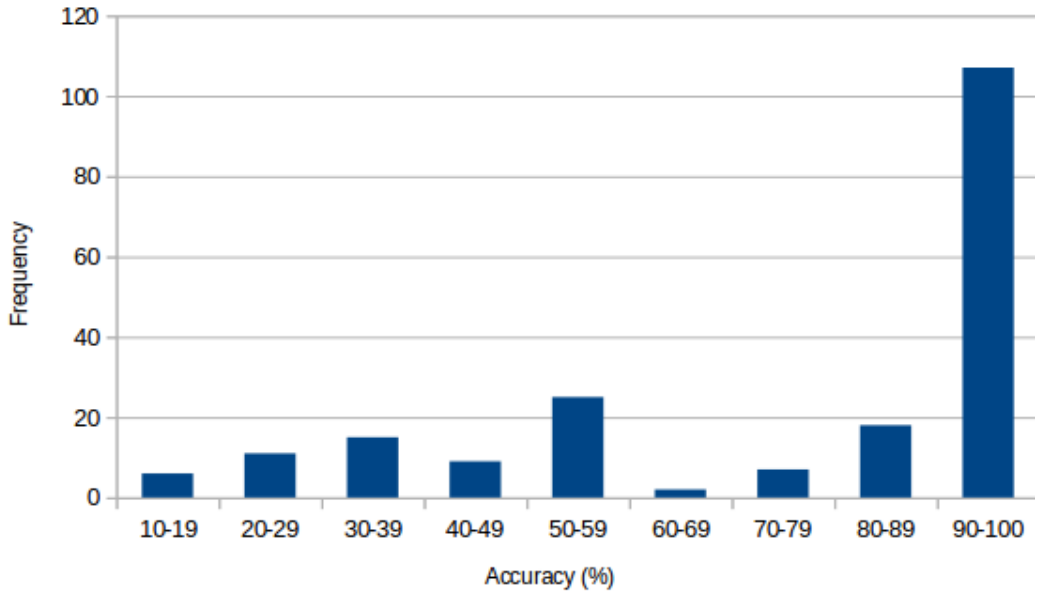


Figure 9: The success of the algorithm in predicting the music symbols contained in 200 test images.

We ran a handful of the test measures through a commercial OMR program called SmartScore, which is the presumed state-of-the-art. SmartScore performed with 100% accuracy on each of the 10 test images we ran through it. While the analysis is far from complete, it seems like the state-of-the-art performs well in situations with a small amount of noise.

6. Conclusion

The early results for our algorithm are encouraging. This is a proof of concept for an OMR procedure that does the following: propose random regions for an input image of a monophonic music measure, then classify those regions and remove duplicates. It works best with around 1,800 region proposals followed by the removal of regions with the same classification and an intersection over union of .13. It struggles in scenarios where the true symbol for an eighth note or sixteenth note gets misinterpreted as a quarter note.

Going forward, we'd like to improve the algorithm to work with near-perfect accuracy on the relatively clean data we're currently generating. This would entail improving both the region proposal algorithm and the method we use to remove duplicate regions.

There is also a big open question in the area of data generation. Can we produce renderings of sheet music realistic enough so that we can correctly label real-life photographs of music? This is an exciting area for exploration. With more time and resources we would explore using multiple different music engraving systems or developing our own for the purpose of diversity of input data. In addition, we would explore the types of noise that can be added to a rendering to make it more realistic. This can deal with things like varying lighting conditions or the natural curve of a page in a book.

7. Appendix

7.1. Proof of region proposal probability

Assume we're given a measure of containing n notes where each is assumed to occupy a uniform width and m region proposals. Each note occupies $1/n$ percentage of the measure. We split the space each note occupies in two, so each half takes up $1/2n$ percent of the measure. A region with two vertical walls successfully captures a note if both halves of a note are occupied by one wall.

The likelihood that the first wall of a region occupies a given note is $1/n$. The likelihood that the second wall occupies the other half of that note is $1/2n$. So the likelihood of a given wall capturing a given note is

$$\left(\frac{1}{n}\right)\left(\frac{1}{2n}\right) = \frac{1}{2n^2}$$

So the likelihood of a given note *not* being captured by a given region is

$$1 - \frac{1}{2n^2}$$

The likelihood of that given note not being captured by *any* of the m regions is

$$\left(1 - \frac{1}{2n^2}\right)^m$$

Therefore the likelihood that a given note is captured by at least one of the m regions is

$$1 - \left(1 - \frac{1}{2n^2}\right)^m$$

7.2. Implementation notes

The algorithm was implemented in Python. The CNN model was written using PyTorch [1] and the training code was based on code written by Justin Johnson [15]. Lastly, I used Abjad [3] to help render the training data.

References

[1] Pytorch. https://github.com/pytorch/pytorch, 2017.

[2] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11):2189–2202, Nov. 2012.

[3] T. Baca and J. W. Oberholtzer. Abjad formalized score control. https://github.com/Abjad/abjad, 2017.

[4] E. P. Bugge, K. L. Juncher, B. S. Mathiasen, and J. G. Simonsen. Using sequence alignment and voting to improve optical music recognition from multiple recognizers. In *ISMIR*, pages 405–410. University of Miami, 2011.

[5] J. A. Burgoyne, L. Pugin, G. Eustace, and I. Fujinaga. A comparative survey of image binarisation algorithms for optical recognition on degraded musical sources. In *ISMIR*, pages 509–512. Austrian Computer Society, 2007.

[6] G. S. Choudhury, M. Droetboom, T. DiLauro, I. Fujinaga, and B. Harrington. Optical music recognition system within a large-scale digitization project. In *ISMIR*, 2000.

[7] M. Church and M. S. Cuthbert. Improving rhythmic transcriptions via probability models applied post-omr. In *ISMIR*, pages 643–648, 2014.

[8] I. Endres and D. Hoiem. Category independent object proposals. In *Proceedings of the 11th European Conference on Computer Vision: Part V*, ECCV'10, pages 575–588, Berlin, Heidelberg, 2010. Springer-Verlag.

[9] J. Ganseman, P. Scheunders, and W. D'haes. Using xquery on musicxml databases for musicological analysis. In *Proceedings of the 9th International Conference on Music Information Retrieval*, pages 433–438, Philadelphia, USA, September 14-18 2008. `http://ismir2008.ismir.net/papers/ISMIR2008_217.pdf`.

[10] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[11] A. Hankinson, L. Pugin, and I. Fujinaga. An interchange format for optical music recognition applications. In *ISMIR*, pages 51–56. International Society for Music Information Retrieval, 2010.

[12] M. Inc. http://www.musitek.com.

[13] Y. Jiang and C. Raphael. Instrument identification in optical music recognition. In *ISMIR*, pages 612–617, 2015.

[14] R. Jin and C. Raphael. Interpreting rhythm in optical music recognition. In *ISMIR*, pages 151–156. FEUP Edições, 2012.

[15] J. Johnson. Pytorch finetune. https://gist.github.com/jcjohnson/6e41e8512c17eae5da50aebef3378a4c, 2017.

[16] J. H. Jr., J. Novotný, P. Pecina, and J. Pokorný. Further steps towards a standard testbed for optical music recognition. In *ISMIR*, pages 157–163, 2016.

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[18] J. R. McPherson. Introducing feedback into an optical music recogniition system. In *ISMIR*, 2002.

[19] H. Nienhuys and J. Nieuwenhuizen. Gnu lilypond. http://www.lilypond.org.

[20] V. Padilla, A. McLean, A. Marsden, and K. Ng. Improving optical music recognition by combining outputs from multiple sources. In *ISMIR*, pages 517–523, 2015.

[21] L. Pugin. Optical music recognitoin of early typographic prints using hidden markov models. In *ISMIR*, pages 53–56, 2006.

[22] L. Pugin and T. Crawford. Evaluating OMR on the early music online collection. In *ISMIR*, pages 439–444, 2013.

[23] C. Ramirez and J. Ohya. Symbol classification approach for OMR of square notation manuscripts. In *ISMIR*, pages 549–554. International Society for Music Information Retrieval, 2010.

[24] C. Raphael and J. Wang. New approaches to optical music recognition. In *ISMIR*, pages 305–310. University of Miami, 2011.

[25] D. Ringwalt and R. B. Dannenberg. Image quality estimation for multi-score OMR. In *ISMIR*, pages 17–23, 2015.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[27] V. Thomas, C. Wagner, and M. Clausen. OCR based post processing of OMR for the recovery of transposing instruments in complex orchestral scores. In *ISMIR*, pages 411–416. University of Miami, 2011.

[28] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[29] G. Vigliensoni, G. Burlet, and I. Fujinaga. Optical measure recognition in common music notation. In *ISMIR*, pages 125–130, 2013.