DEEP WATERSHED DETECTOR FOR MUSIC OBJECT RECOGNITION

Lukas Tuggener[1,3] **Ismail Elezi**[1,2]
Jürgen Schmidhuber[3] **Thilo Stadelmann**[1]
[1] ZHAW Datalab, Zurich University of Applied Sciences, Winterthur, Switzerland
[2] Dept. of Environmental Sciences, Informatics and Statistics, Ca'Foscari University of Venice, Italy
[3] Faculty of Informatics, Università della Svizzera Italiana, Lugano, Switzerland
tugg@zhaw.ch, ismail.elezi@unive.it, juergen@idsia.ch, stdm@zhaw.ch

ABSTRACT

Optical Music Recognition (OMR) is an important and challenging area within music information retrieval, the accurate detection of music symbols in digital images is a core functionality of any OMR pipeline. In this paper, we introduce a novel object detection method, based on synthetic energy maps and the watershed transform, called Deep Watershed Detector (DWD). Our method is specifically tailored to deal with high resolution images that contain a large number of very small objects and is therefore able to process full pages of written music. We present state-of-the-art detection results of common music symbols and show DWD's ability to work with synthetic scores equally well as on handwritten music.

1. INTRODUCTION AND PROBLEM STATEMENT

The goal of Optical Music Recognition (OMR) is to transform images of printed or handwritten music scores into machine readable form, thereby understand the semantic meaning of music notation [2]. It is an important and actively researched area within the music information retrieval community. The two main challenges of OMR are: first the accurate detection and classification of music objects in digital images; and second, the reconstruction of valid music in some digital format. This work is focusing solely on the first task.

Recent progress in computer vision [10] thanks to the adaptation of convolutional neural networks (CNNs) [8, 15] provide a solid foundation for the assumption that OMR systems can be drastically improved by using CNNs as well. Initial results of applying deep learning [14, 27] to heavily restricted settings such as staffline removal [26], symbol classification [21] or end-to-end OMR for monophonic scores [5], support such expectations.

In this paper, we introduce a novel general object detection method called Deep Watershed Detector (DWD) motivated by the following two hypotheses: a) deep learning can be used to overcome the classical OMR approach of having hand-crafted pipelines of many preprocessing steps [22] by being able to operate in a fully data-driven fashion; b) deep learning can cope with larger, more complex inputs than simple glyphs, thereby learning to recognize musical symbols in their context. This will disambiguate meanings (e.g., between staccato and augmentation dots) and allow the system to directly detect a complex alphabet.

DWD operates on full pages of music scores in one pass without any preprocessing besides interline normalization, detects handwritten and digitally rendered music symbols without any restriction on the alphabet of symbols to be detected. We further show that it learns meaningful representation of music notation and achieves state-of-the art detection rates on common symbols.

The remaining structure of this paper is as follows: Sec. 2 puts our approach in context with existing methods; in Sec. 3 we derive our original end-to-end model, and give a detailed explanation on how we use the deep watershed transform for the task of object recognition; Sec. 4 reports on experimental results of our system on the *DeepScores* digitally rendered dataset in addition to the *MUSCIMA++* handwritten dataset before we conclude in Sec. 5 with a discussion and give pointers for future research.

2. RELATED WORK

The visual detection and recognition of objects is one of the most central problems in the field of computer vision. With the recent developments of CNNs, many competing CNN-based approaches have been proposed to solve the problem. R-CNNs [9], and in particular their successors [24], are generally considered to be state-of-the-art models in object recognition, and many developed recognition systems are R-CNN based. On the other hand, researchers have also proposed models which are tailored towards computational efficiency instead of detection accuracy. YOLO systems [23] and Single-Shot Detectors [19] while slightly compromising on accuracy, are significantly faster than R-CNN models, and can even achieve super real-time performance.

A common aspect of the above-mentioned methods is that they are specifically developed to work on cases where the images are relatively small, and where images contain a small number of relatively large objects [7, 18]. On the

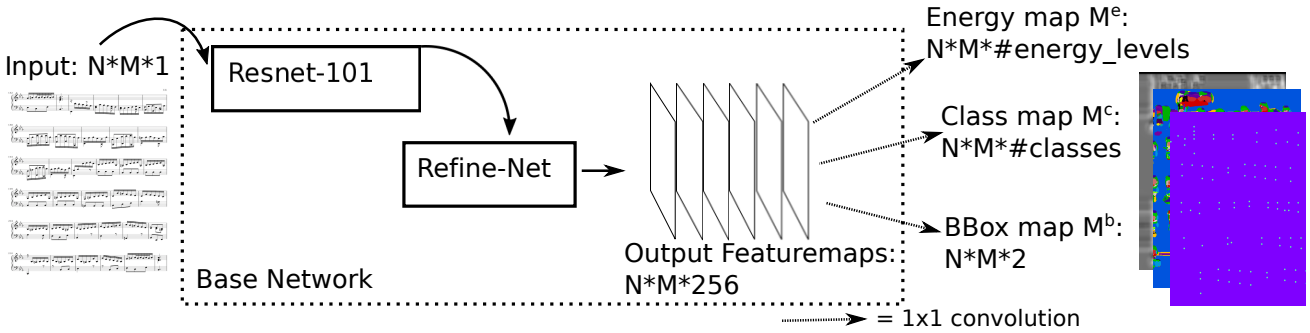


Figure 1. Illustration of the DWD network and its sub-components together with input and outputs. The outputs have been cropped to improve visibility

contrary, musical sheets usually have high-resolution, and contain a very large number of very small objects, making the mentioned methods not suitable for the task.

The watershed transform is a well understood method that has been applied to segmentation for decades [4]. Bai and Urtasun [1] were first to propose combining the strengths of deep learning with the power of this classical method. They proposed to directly learn the energy for the watershed transform such that all dividing ridges are at the same height. As a consequence, the components can be extracted by a cut at a single energy level without leading to over-segmentation. The model has been shown to achieve state of the art performance on object segmentation.

For the most part, OMR detectors have been rule based systems working well only within a hard set of constraints [22]. Typically, they require domain knowledge, and work well only on simple typeset music scores with a known music font, and a relatively small number of classes [25]. When faced with low-quality images, complex or even handwritten scores [3], the performance of these models quickly degrades, to some degree because errors propagate from one step to another [21]. Additionally, it isn't clear what to do when the classes change, and in many cases, this requires building the new model from scratch.

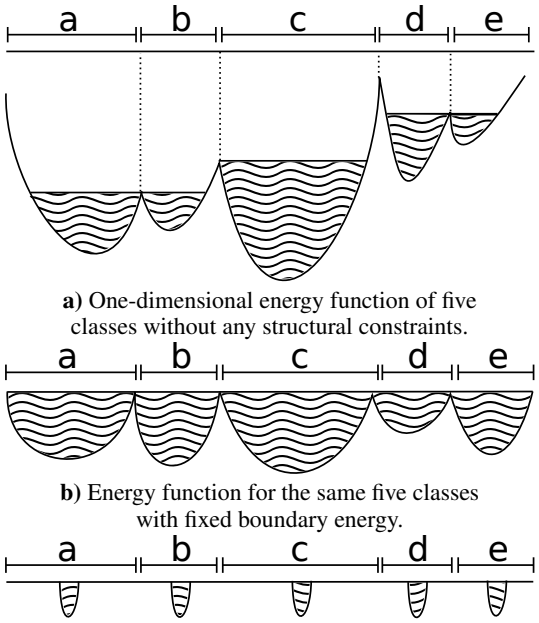
In response to the above mentioned issues some deep learning based, data driven approaches have been developed. Hajic and Pecina [13] proposed an adaptation of Faster R-CNN with a custom region proposal mechanism based on the morphological skeleton to accurately detect noteheads, while Choi et al. [6] were able to detect accidentals in dense piano scores with high accuracy, given previously detected noteheads, that are being used as input-features to the network. A big limitation of both approaches is that the experiments have been done only on a tiny vocabulary of the musical symbols, and therefore their scalability remains an open question.

To our knowledge, the best results so far has been reported in the work of Pacha and Choi [20] where they explored many models on the *MUSCIMA++* [11] dataset of handwritten music notation. They got the best results with a Faster R-CNN model, achieving an impressive score on the standard mAP metric. A serious limitation of that work is that the system wasn't designed in an end-to-end fashion and needs heavy pre- and post-processing. In particular, they cropped the images in a context-sensitive way, by cutting images first vertically and then horizontally, such that each image contains exactly one staff and has a width-to-height-ratio of no more than $2:1$, with about 15% horizontal overlap to adjacent slices. In practice, this means that all objects significantly exceeding the size of such a cropped region will neither appear in the training nor testing data, as only annotations that have an intersection-over-area of 0.8 or higher between the object and the cropped region are considered part of the ground truth. Furthermore, all the intermediate results must be combined to one concise final prediction, which is a non-trivial task.

3. DEEP WATERSHED DETECTION

In this section we present the Deep Watershed Detector (DWD) as a novel object detection system, built on the idea of the deep watershed transform [1]. The watershed transform [4] is a mathematically well understood method with a simple core idea that can be applied to any topological surface. The algorithm starts filling up the surface from all the local minima, with all the resulting basins corresponding to connected regions. When applied to image gradients, the basins correspond to homogeneous regions of said image (see Fig. 2a). One key drawback of the watershed transform is its tendency to over segment. This issue can be addressed by using the deep watershed transform. It combines the classical method with deep learning by training a deep neural network to create an energy surface based on an input image. This has the advantage that one can design the energy surface to have certain properties. When designed in a way that all segmentation boundaries have energy zero, the watershed transform is reduced to a simple cutoff at a fixed energy level (see Fig. 2b). An objectness energy of this fashion has been used by Bai and Urtasun for instance segmentation [1]. Since we want to do object detection, we further simplify the desired energy surface to having small conical energy peaks of radius n pixels at the center of each object and be zero everywhere else (see Fig. 2c).

More formally, we define our energy surface (or: energy map) M^e as follows:



a) One-dimensional energy function of five classes without any structural constraints.

b) Energy function for the same five classes with fixed boundary energy.

c) Energy function for the same five classes this time with small energy markers at the class centers.

Figure 2. Illustration of the watershed transform applied to different one-dimensional functions.

$$M^e_{(i,j)} = max \begin{cases} \cancel{\text{arg}}\max_{c \in C}[E_{max} \cdot (1 - \frac{\sqrt{(i-c_i)^2+(j-c_j)^2}}{r})] \\ 0 \end{cases} \tag{1}$$

where $M^e_{(i,j)}$ is the value of M^e at position (i,j), C is the set of all object centers and c_i, c_j are the center coordinates of a given center c. E_{max} corresponds to the maximum energy and r is the radius of the center marking.

At first glance this definition might lead to the misinterpretation that object centers that are closer together than r cannot be disambiguated using the watershed transform on M^e. This is not the case since we can cut the energy map at any given energy level between 1 and E_{max}. However, using this method it is not possible to detect multiple bounding boxes that share the exact same center.

3.1 Retrieving Object Centers

After computing an estimate $\hat{M}^e$ of the energy map, we retrieve the coordinates of detected objects by the following steps:

1. Cut the energy map at a certain fixed energy level and then binarize the result.

2. Label the resulting connected components, using the two-pass algorithm [31]. Every component receives a label l in $1...n$, for every component o^l we define O^l_{ind} as the set of all tuples (i,j) for which the pixel with coordinates j and i is part of o^l.

3. The center $\hat{c}^l$ of any component o^l is given by its center of gravity:

$$\hat{c}^l = o^l_{center} = |O^l_{ind}|^{-1} \cdot \sum_{(i,j) \in O^l_{ind}} (i,j) \tag{2}$$

We use these component centers $\hat{c}$ as estimates for the object centers c.

3.2 Object Class and Bounding Box

In order to recover bounding boxes we do not only need the object centers, but also the object classes and bounding box dimensions. To achieve this we output two additional maps M^c and M^b as predictions of our network. M^c is defined as:

$$M^c_{(i,j)} = \begin{cases} \Lambda_{(i,j)}, & \text{if } M^e_{(i,j)} > 0 \\ \Lambda_{background}, & \text{otherwise} \end{cases} \tag{3}$$

where $\Lambda_{backgroud}$ is the class label indicating background and $\Lambda_{(i,j)}$ is the class label associated with the center c that is closest to (i,j). We define our estimate for the class of component o^l by a majority vote of all values $\hat{M}^c_{(i,j)}$ for all $(i,j) \in O^l_{ind}$, where $\hat{M}^c$ is the estimate of M^c. Finally, we define the bounding box map M^b as follows:

$$M^b_{(i,j)} = \begin{cases} (y^l, x^l), & \text{if } M^e_{(i,j)} > 0 \\ (0,0), & \text{otherwise} \end{cases} \tag{4}$$

where y^l and x^l are the width and height of the bounding box for component o^l. Based on this we define our bounding box estimation as the average of all estimations for label l:

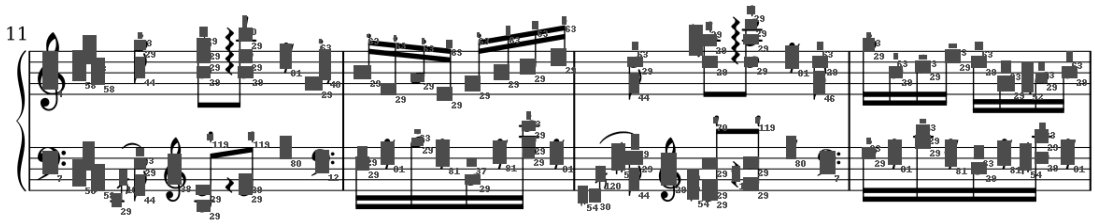
$$(\hat{y}^l, \hat{x}^l) = |O^l_{ind}|^{-1} \cdot \sum_{(i,j) \in O^l_{ind}} \hat{M}^b_{(i,j)} \tag{5}$$

3.3 Network Architecture and Losses

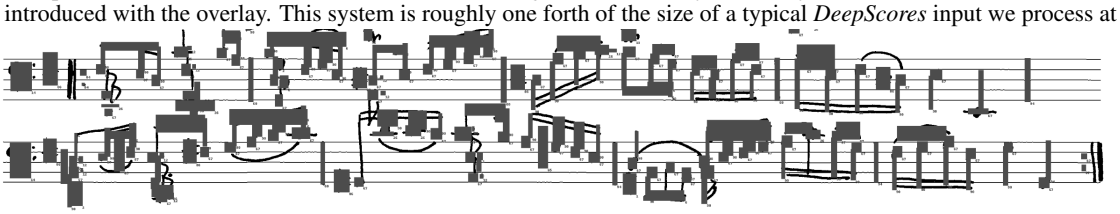
As mentioned above we use a deep neural network to predict the dense output maps M^e, M^c and M^b (see Fig. 1). The base neural network for this prediction can be any fully convolutional network with the same input and output dimensions. We use a ResNet-101 [12] (a special case of a Highway Net [28]) in conjunction with the elaborate RefineNet [17] upsampling architecture. For the estimators defined above it is crucial to have the highest spacial prediction resolution possible. Our network has three output layers, all of which are an 1 by 1 convolution applied to the last feature map of the RefineNet.

3.3.1 Energy prediction

We predict a quantized and one-hot encoded version of M^e, called $M^{e\text{-}oh}$, by applying a 1 by 1 convolution of depth E_{max} to the last feature map of the base network. The loss of the prediction $\hat{M}^{e\text{-}oh}$, $loss^e$, is defined as the cross-entropy between $M^{e\text{-}oh}$ and $\hat{M}^{e\text{-}oh}$.



a) Example result from *DeepScores* with detected bounding boxes as overlays. The tiny numbers are class labels from the dataset introduced with the overlay. This system is roughly one forth of the size of a typical *DeepScores* input we process at once.



b) Example result from *MUSCIMA++* with detected bounding boxes and class labels as overlays. This system is roughly one half of the size of a typical processed *MUSCIMA++* input. The images are random picks amongst inputs with many symbols.

Figure 3. Detection results for *MUSCIMA++* and *DeepScores* examples, drawn on crops from corresponding input images.

3.3.2 Class prediction

We again use the corresponding one-hot encoded version $M^{c\text{-}oh}$ and predict it using an 1 by 1 convolution, with the depth equal to the number of classes, on the last feature map of the base network. The cross-entropy $loss^c$ is calculated between between $M^{c\text{-}oh}$ and $\hat{M}^{c\text{-}oh}$. Since it is not the goal of this prediction to distinguish between foreground and background, all the loss stemming from locations with $M^e = 0$ will get masked out.

3.3.3 Bounding box prediction

M^b is predicted in its initial form using an 1 by 1 convolution of depth 2 on the last feature map of the base network. The bounding box loss $loss^b$ is the mean-squared difference between M^b and $\hat{M}^b$. For $loss^b$, the components stemming from background locations will be masked out analogous to $loss^c$.

3.3.4 Combined prediction

We want to jointly train in all tasks, therefore we define a total loss $loss^{tot}$ as:

$$loss^{tot} = w_1 * \frac{loss^e}{v^e} + w_2 * \frac{loss^c}{v^c} + w_3 * \frac{loss^b}{v^b} \quad (6)$$

where the $v^{\cdot}$ are running means of the corresponding losses and the scalars $w_{\cdot}$ are hyper-parameters of the DWD network. We purposefully use very short extraction heads of one convolutional layer; by doing so we force the base network to do all three tasks simultaneously. We expect this leads to the base network learning a meaningful representation of music notation, from which it can extract the solutions of the three above defined tasks.

4. EXPERIMENTS AND RESULTS

4.1 Used Datasets

For our experiments we use two datasets: *DeepScores* [30] and *MUSCIMA++* [11].

DeepScores is currently the largest publicly available dataset of musical sheets with ground truth for various machine learning tasks, consisting of high-quality pages of written music, rendered at 400 dots per inch. The dataset has $300,000$ full pages as images, containing tens of millions of objects, separated in 123 classes. We randomly split the set into training and testing, using $200k$ images for training and $50k$ images each for testing and validation. The dataset being so large allows efficient training of large convolutional neural networks, in addition to being suitable for transfer learning [33].

MUSCIMA++ is a dataset of handwritten music notation for musical symbol detection. It contains $91,255$ symbols spread unto 140 pages, consisting of both notation primitives and higher-level notation objects, such as key signatures or time signatures. It features 105 object classes. There are $23,352$ notes in the dataset, of which $21,356$ have a full notehead, $1,648$ have an empty notehead, and 348 are grace notes. We randomly split the dataset into training, validation and testing, with the training set consisting of 110 pages, while validation and testing each consist of 15 pages.

4.2 Network Training and Experimental Setup

We pre-train our network in two stages in order to achieve reasonable results. First we train the ResNet on music symbol classification using the *DeepScores* classification dataset [30]. Then, we train the ResNet and RefineNet jointly on semantic segmentation data also available from *DeepScores*. After this pre-training stage we are able to use the network on the tasks defined above in Sec. 3.3.

Since music notation is composed of hierarchically organized sub-symbols, there does not exist a canonical way to define a set of atomic symbols to be detected (e.g., individual numbers in time signatures vs. complete time signatures). We address this issue using a fully data driven approach and detecting the unaltered labels as they are provided by the two datasets.

Class	AP@$\frac{1}{2}$	Class	AP@$\frac{1}{4}$
rest16th	0.8773	tuplet6	0.9252
noteheadBlack	0.8619	keySharp	0.9240
keySharp	0.8185	rest16th	0.9233
tuplet6	0.8028	noteheadBlack	0.9200
restQuarter	0.7942	accidentalSharp	0.8897
rest8th	0.7803	rest32nd	0.8658
noteheadHalf	0.7474	noteheadHalf	0.8593
flag8thUp	0.7325	rest8th	0.8544
flag8thDown	0.6634	restQuarter	0.8462
accidentalSharp	0.6626	accidentalNatural	0.8417
accidentalNatural	0.6559	flag8thUp	0.8279
tuplet3	0.6298	keyFlat	0.8134
noteheadWhole	0.6265	flag8thDown	0.7917
dynamicMF	0.5563	tuplet3	0.7601
rest32nd	0.5420	noteheadWhole	0.7523
flag16thUp	0.5320	fClef	0.7184
restWhole	0.5180	restWhole	0.7183
timeSig8	0.5180	dynamicPiano	0.7069
accidentalFlat	0.4949	accidentalFlat	0.6759
keyFlat	0.4685	flag16thUp	0.6621

Table 1. AP with overlap 0.5 and overlap 0.25 for the twenty best detected classes of the *DeepScores* dataset.

Class	AP@$\frac{1}{2}$	Class	AP@$\frac{1}{4}$
half-rest	0.8981	whole-rest	0.9762
flat	0.8752	ledger-line	0.9163
natural	0.8531	half-rest	0.8981
whole-rest	0.8226	flat	0.8752
notehead-full	0.8044	natural	0.8711
sharp	0.8033	stem	0.8377
notehead-empty	0.7475	staccato-dot	0.8302
stem	0.7426	notehead-full	0.8298
quarter-rest	0.6699	sharp	0.8121
8th-rest	0.6432	tenuto	0.7903
f-clef	0.6395	notehead-empty	0.7475
numeral-4	0.6391	duration-dot	0.7285
letter-c	0.6313	numeral-4	0.7158
letter-c	0.6313	8th-flag	0.7055
8th-flag	0.6051	quarter-rest	0.6849
slur	0.5699	letter-c	0.6643
beam	0.5188	letter-c	0.6643
time-signature	0.4940	8th-rest	0.6432
staccato-dot	0.4793	beam	0.6412
letter-o	0.4793	f-clef	0.6395

Table 2. AP with overlap 0.5 and overlap 0.25 for the twenty best detected classes from *MUSCIMA++*.

We rescale every input image to the desired interline value (number of pixels in between two staff lines). We use 10 pixels for *DeepScores* and 20 pixels for *MUSCIMA++*. Other than that we apply no preprocessing. We do not define a subset of target objects for our experiments, but attempt to detect all classes for which there is ground truth available. We always feed single images to the network, i.e. we only use batch size = 1. During training we crop the full page input (and the ground truth) to patches of 960 by 960 pixels using random coordinates. This serves two purposes: it saves GPU memory and performs efficient data augmentation. This way the network never sees the exact same input twice, even if we train for many epochs. For all of the results described below we train individually on $loss^e$, $loss^c$ and $loss^b$ and then refine the training using $loss^{tot}$. It turns out that the prediction of M^e is the most fragile, therefore we retrain on $loss^e$ again after training on the individual losses in the order defined above, before moving on to $loss^{tot}$. All the training is done using the RMSProp optimizer [29] with a learning rate of 0.001 and a decay rate of 0.995.

Since our design is invariant to how many objects are present on the input (as long as their centers do not overlap) and we want to obtain bounding boxes for full pages at once, we feed whole pages to the network at inference time. The maximum input size is only bounded by the memory of the GPU. For typical pieces of sheet music this is not an issue, but pieces that use very small interline values (e.g. pieces written for conductors) result in very large inputs due to the interline normalization. At about 10.5 million pixels even a Tesla P40 with 24 gigabytes runs out of memory.

4.3 Results and Discussion

Tab. 1 shows the average precision (AP) for the twenty best detected classes with an overlap of the detected bounding box and ground truth of 50% and 25%, respectively. We observe that in both cases there are common symbol classes that get detected very well, but there is also a steep fall off. The detection rate outside the top twenty continues to drop and is almost zero for most of the rare classes. We further observe that there is a significant performance gain for the lower overlap threshold, indicating that the bounding-box regression is not very accurate.

Fig. 3 shows an example detection for qualitative analysis. It confirms the conclusions drawn above. The rarest symbol present, an arpeggio, is not detected at all, while the bounding boxes are sometimes inaccurate, especially for large objects (note that stems, bar-lines and beams are not part of the *DeepScores* alphabet and hence don't constitute missed detections). On the other hand, staccato dots are detected very well. This is surprising since they are typically hard to detect due to their small size and the context-dependent interpretation of the symbol shape (compare the dots in dotted notes or F-clefs). We attribute this to the opportunity of detecting objects in context, enabled by training on larger parts of full raw pages of sheet music in contrast to the classical processing of tiny, pre-processed image patches or glyphs.

The results for the experiments on *MUSCIMA++* in Tab. 2 and Fig. 3b show a very similar outcome. This is intriguing because it suggests that the difficulty in detecting digitally rendered and handwritten scores might be smaller than anticipated. We attribute this to the fully data-driven approach enabled by deep learning instead of hand-crafted rules for handling individual symbols. It is worth noting that ledger-lines are detected with very high performance (see AP@$\frac{1}{4}$). This explains the relatively poor detection of note-heads on *MUSCIMA++*, since they tend to overlap.

Fig. 4 shows an estimate for a class map with its corresponding input overlayed. Each color corresponds to one class. This figure proofs that the network is learning a sensible representation of music notation: even though it is

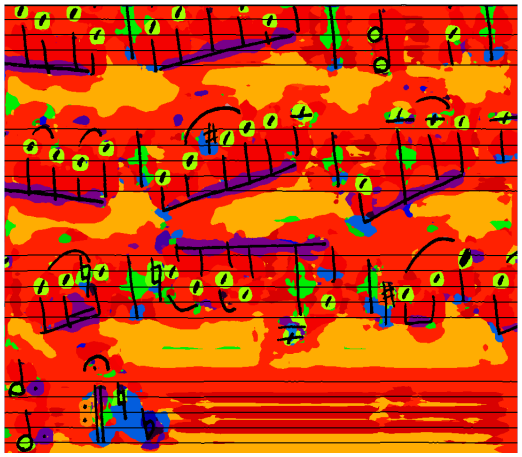


Figure 4. Estimate of a class map $\hat{M}^c$ for every input pixel with the corresponding *MUSCIMA++* input overlayed.

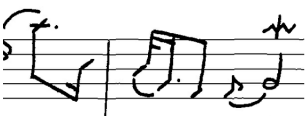


Figure 5. Typical input snippet used by Pacha et al. [20]

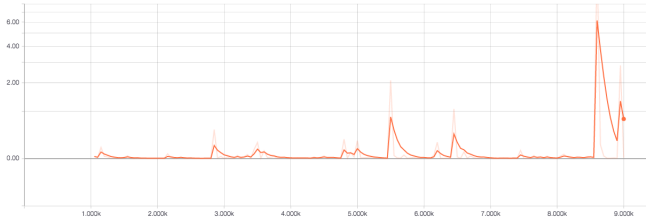


Figure 6. Evolution of $loss^b$ (on the ordinate) of a sufficiently trained network, when training for another 8000 iterations (on the abscissa).

only trained to mark the centers of each object with the correct colors, it learns a primitive segmentation mask. This is best illustrated by the (purple) segmentation of the beams.

5. CONCLUSIONS AND FUTURE WORK

We have presented a novel method for object detection that is specifically tailored to detect many tiny objects on large inputs. We have shown that it is able to detect common symbols of music notation with high precision, both in digitally rendered music as well as in handwritten music, without a drop in performance when moving to the "more complicated" handwritten input. This suggests that deep learning based approaches are able to deal with handwritten sheets just as well as with digitally rendered ones, additionally to their benefit of recognizing objects in their context and with minimal preprocessing as compared to classical OMR pipelines. Pacha et al. [20] show that higher detection rates, especially for uncommon symbols, are possible when using R-CNN on small snippets (cp. Fig. 5). Despite their higher scores, it is unclear how recognition performance is affected when results of overlapping and potentially disagreeing snippets are aggregated to full page results. A big advantage of our end-to-end system is the complete avoidance of error propagation in longer recognition pipeline of independent components like classifiers, aggregators etc [16]. Moreover, our full-page end-to-end approach has the advantages of speed (compared to a sliding window patch classifier), change of domain (we use the same architecture for both the digital and handwritten datasets) and is easily integrated into complete OMR frameworks.

Arguably the biggest problem we faced is that symbol classes in the dataset are heavily unbalanced. In the *DeepScores* dataset in particular, the class *notehead* contains more than half of all the symbols in the entire dataset, while the top 10 classes contain more than 85% of the symbols. Considering that we did not do any class-balancing whatsoever, this imbalance had its effect in training. We observe that in cases where the symbol is common, we get a very high average precision, but it quickly drops when symbols become less common. Furthermore, it is interesting to observe that the neural network actually forgets about the existence of these rarer symbols: Fig. 6 depicts the evolution of $loss^b$ of a network that is already trained and gets further trained for another $8,000$ iterations. When faced with an image containing rare symbols, the initial loss is larger than the loss on more common images. But to our surprise, later during the training process, the loss actually increases when the net encounters rare symbols again, giving the impression that the network is actually treating these symbols as outliers and ignoring them.

Future work will thus concentrate on dealing with the catastrophic imbalance in the data to successfully train DWD to detect all classes. We believe that the solution lies in a combination of data augmentation and improved training regimes (i.e. sample pages containing rare objects more often, synthesizing mock pages filled with rare objects etc.).

Additionally, we plan to investigate the ability of our method beyond OMR on natural images. Initially we will approach canonical datasets like *PASCAL VOC* [7] and *MS-COCO* [18] that have been at the front-line of object recognition tasks. However, images in those datasets are not exactly natural, and for the most part they are simplistic (small images, containing a few large objects). Recently, researchers have been investigating the ability of state-of-the-art recognition systems on more challenging natural datasets, like DOTA [32], and unsurprisingly, the results leave much to be desired. The DOTA dataset shares a lot of similarities with musical datasets, with images being high resolution and containing hundreds of small objects, making it a suitable benchmark for our DWD method to recognize tiny objects.

6. REFERENCES

[1] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In *CVPR*, 2017.

[2] D. Bainbridge and T. Bell. The challenge of optical music recognition. *Computers and the Humanities*, 2001.

[3] A. Baro, P. Riba, and A. Fornés. Towards the recognition of compound music notes in handwritten music scores. In *ICFHR*, 2016.

[4] S. Beucher. The watershed transformation applied to image segmentation. *SCANNING MICROSCOPY-SUPPLEMENT-*, 1992.

[5] J. Calvo-Zaragoza, J. J. Valero-Mas, and A. Pertusa. End-to-end optical music recognition using neural networks. In *ISMIR*, 2017.

[6] K.-Y. Choi, B. Coüasnon, Y. Ricquebourg, and R. Zanibbi. Bootstrapping samples of accidentals in dense piano scores for cnn-based detection. In *GREC@ICDAR*, 2017.

[7] M. Everingham, L. J. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 2010.

[8] K. Fukushima and S. Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 1982.

[9] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[10] R. B. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. Detectron. `https://github.com/facebookresearch/detectron`, 2018.

[11] J. Hajic and P. Pecina. The MUSCIMA++ dataset for handwritten optical music recognition. In *ICDAR*, 2017.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. 2016.

[13] J. H. Jr. and P. Pecina. Detecting noteheads in handwritten scores with convnets and bounding box regression. *CoRR*, abs/1708.01806, 2017.

[14] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 2015.

[15] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1989.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[17] G. Lin, A. Milan, C. Shen, and I. D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017.

[18] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg. SSD: single shot multibox detector. In *ECCV*, 2016.

[20] A. Pacha, K.-Y. Choi, B. Coüasnon, Y. Ricquebourg, and R. Zanibbi. Handwritten music object detection: Open issues and baseline results. In *International Workshop on Document Analysis Systems*, 2018.

[21] A. Pacha and H. Eidenberger. Towards self-learning optical music recognition. In *ICMLA*, 2017.

[22] A. Rebelo, G. Capela, and J. S. Cardoso. Optical recognition of music symbols - A comparative study. *IJDAR*, 2010.

[23] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In *CVPR*, 2017.

[24] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[25] F. Rossant and I. Bloch. Robust and adaptive OMR system including fuzzy modeling, fusion of musical rules, and possible error detection. *EURASIP*, 2007.

[26] A. J. Gallego Sánchez and J. Calvo-Zaragoza. Staff-line removal with selectional auto-encoders. *Expert Syst. Appl.*, 2017.

[27] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 2015.

[28] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. 2015.

[29] T. Tieleman and G. E. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning 4.2*, 2012.

[30] Lukas Tuggener, Ismail Elezi, Jurgen Schmidhuber, Marcello Pelillo, and Thilo Stadelmann. Deepscores - a dataset for segmentation, detection and classification of tiny objects. *ICPR*, 2018.

[31] K. Wu, , E. Otoo, and K. Suzuki. Optimizing two-pass connected-component labeling algorithms. *Pattern Anal. Appl.*, 2009.

[32] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, M. Datcu, M. Pelillo, and L. Zhang. Dota: A large-scale dataset for object detection in aerial images. In *CVPR*, 2018.

[33] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.