LEARNING TO PINPOINT SINGING VOICE FROM WEAKLY LABELED EXAMPLES

Jan Schlüter
Austrian Research Institute for Artificial Intelligence, Vienna
jan.schlueter@ofai.at

ABSTRACT

Building an instrument detector usually requires temporally accurate ground truth that is expensive to create. However, song-wise information on the presence of instruments is often easily available. In this work, we investigate how well we can train a singing voice detection system merely from song-wise annotations of vocal presence. Using convolutional neural networks, multiple-instance learning and saliency maps, we can not only detect singing voice in a test signal with a temporal accuracy close to the state-of-the-art, but also localize the spectral bins with precision and recall close to a recent source separation method. Our recipe may provide a basis for other sequence labeling tasks, for improving source separation or for inspecting neural networks trained on auditory spectrograms.

1. INTRODUCTION

A fundamental step in automated music understanding is to detect which instruments are present in a music audio recording, and at what time they are active. Traditionally, developing a system detecting and localizing a particular instrument requires a set of music pieces annotated at the same granularity as expected to be output by the system – no matter if the system is constructed by hand or by machine learning algorithms.

Annotating music pieces at high temporal accuracy requires skilled annotators and a lot of time. On the other hand, instrument annotations at a song level are often easily available online, as part of the tags given by users of streaming services, or descriptions or credits by the publisher. Even if not, collecting or cleaning song-wise annotations requires very little effort and low skill compared to curating annotations with sub-second granularity.

As a step towards tapping into such resources, in this work, we explore how to obtain high-granularity vocal detection results from low-granularity annotations. Specifically, we train a Convolutional Neural Network (CNN) on 10,000 30-second song snippets annotated as to whether they contain singing voice anywhere within, and subsequently use it to detect the presence of singing voice with sub-second granularity. As the main contribution of our work, we develop a recipe to improve initial results using multiple-instance learning and saliency maps. Finally, we investigate how well the system can even pinpoint the spectral bins containing singing voice, instead of the time frames only. While we constrain our experiments to singing voice detection as a special case of instrument detection (and possibly the easiest), we do not assume any prior knowledge about the content to be detected, and thus expect the recipe to carry over to other instruments.

The next section will provide a review of related work on learning from weakly-annotated data both outside and within of the music domain, and on singing voice detection. Section 3 explains the methods we combined in this paper, Section 4 describes how we combined them, and Section 5 evaluates the resulting system on four datasets. Finally, Section 6 discusses what we achieved and what is still open, highlights avenues for future research and points out alternative uses for some of our findings.

2. RELATED WORK

The idea of training on weakly-labeled data is far from new, since coarse labels are almost always easier to obtain than fine ones. The general framework for this setting is Multiple-Instance Learning, which we will return to in Section 3.2. As one of the first instances, Keeler et al. [8] train a CNN to recognize and localize two hand-written digits in an input image of about 36×36 pixels, giving only the identities of the two digits as training targets. As a recent work closer to our setting, Hou et al. [6] train a CNN to detect and classify brain tumors in gigapixel resolution tissue images. As such images are too large to be processed as a whole, they propose to train on patches, still using image-level labels only. To account for the fact that not all patches in a tumor image show tumorous tissue, Hou et al. employ an expectation maximization algorithm that iteratively prunes non-discriminative patches from the training set based on the CNN's predictions. As far as we are aware, the only work in music information retrieval aiming to produce fine-grained predictions from coarse training data is that of Mandel and Ellis [14]: They train special variants of SVMs on song, album or artist labels to predict tags on a granularity of 10-second clips. In contrast, we aim for sub-second granularity, and for identifying spectral bins.

Recent approaches for singing voice detection [9,10,18] are all based on machine learning from temporally accurate labels. The current state of the art is our previous work [18], a CNN trained on mel spectrogram excerpts. We will use it both as a starting point and for comparing our results against, and describe it in more detail in Section 3.1.

Singing voice detection does not entail identifying the spectral bins. The closest task to this is singing voice extraction, which aims to extract a purely vocal signal from a single-channel audio recording and thus has to estimate its spectral extends. It differs from general blind source separation in that it can leverage prior knowledge about the two signals to be separated – vocals and background music. As an improvement over Nonnegative Matrix Factorization (NMF) [20], which can only encode such prior knowledge in the form of spectral templates, REPET [16] uses the fact that background music is repetitive while vocals are not. Kernel-Additive Modeling [11] generalizes this method and uses a set of assumptions on local regularities of vocals and background music to perform singing voice extraction. We will use it as a mark to compare our results to.

3. INGREDIENTS

Our recipe combines a few methods that we shall introduce up front: a singing voice detector based on CNNs, multiple-instance learning, and saliency maps.

3.1 CNN-based Singing Voice Detection

The base of our system is a CNN trained to predict whether a short spectrogram excerpt contains singing voice at its center. We mostly follow Schlüter et al. [18], and will limit the description here to what is needed for understanding the paper, as well as how we deviated from that approach.

Input signals are converted to 22.05 kHz mono and processed by a Short-Time Fourier Transform with 70 1024-sample frames per second. Phases are discarded, magnitudes are scaled by $\log(1+x)$, the spectrogram is cropped above 8 kHz (keeping 372 bins) and each frequency band is normalized to zero mean and unit variance over the training set. We skip the mel-scaling step of Schlüter et al. to enable more accurate spectral localization of singing voice.

As in [18], the network architecture starts with 64 and 32 3×3 convolutions, 3×3 max-pooling, 128 and 64 3×3 convolutions. At this point, the 372 frequency bands have been reduced to 118. We add 128 3×115 convolutions and 1×4 max-pooling. This way, the network learns spectrotemporal patterns spanning almost the full frequency range, applies them at four different frequency offsets and keeps the maximum activation of those four, effectively introducing some pitch invariance. We finish with three dense layers of 256, 64 and 1 unit, respectively. Except for the final layer, each convolution and dense layer is followed by batch normalization [7] and the leaky rectifier $\max(x/100, x)$ [13]. The final layer uses the sigmoid activation function.

Training is done on excerpts of 115 frames (about 1.6 sec) paired with a binary label for the excerpt's central frame. We follow the training protocol described in [18, Sec. 3.3], augmenting inputs with random pitch-shifting and time-stretching of $\pm30\%$ and frequency filtering of ±10 dB using the code accompanying [18].

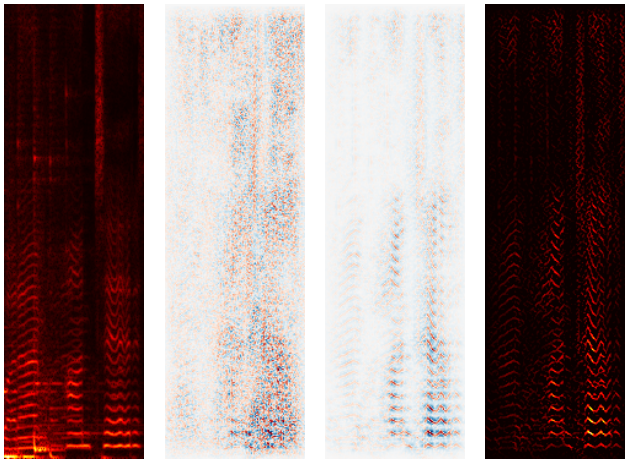
We arrived at this system by selectively modifying our previous work [18] to work well with linear-frequency spectrograms, on an internal dataset with fine-grained annotations. It slightly outperforms [18] on this dataset.

3.2 Multiple-Instance Learning

While we train our network on short spectrogram excerpts, we actually only have a single label per 30-second clip. In the Multiple-Instance Learning (MIL) framework, each explicitly labeled 30-second clip is called a *bag*, and the excerpts we train on are called *instances*. In our setting, a bag is labeled positively if and only if at least one of the instances contained within are positive (referred to as the *standard MI assumption* [4]). This gives an interesting asymmetry: If a 30-second clip is labeled as "no vocals", we can infer that neither of its excerpts contains vocals. If a clip contains vocals, we only know that *some* excerpts will contain vocals, but neither which ones nor how many.

One approach for training a neural network in this setting is based on the observation that the label of a bag is simply the maximum over its instance labels. If we define the network's prediction for a bag to be the maximum over the predictions for its instances, and the objective function to measure the discrepancy between this bag-wise prediction and the true label, minimizing it by gradient descent directly results in the following algorithm (BP-MIP, [24]): Propagate all instances of a bag through the network, pick the instance that gives the highest output, and update the network weights to minimize its discrepancy with the bag label. Unfortunately, this scheme is very costly: It computes predictions for all instances of a bag, then performs an update for a single instance only. Furthermore, it is easy to overfit: It is enough for the network to produce a strongly positive output for a single chosen instance per bag and negative outputs for all others. For 10,000 30-second clips, it would require merely learning 10,000 short excerpts by heart.

A different approach is to present all instances from negative bags as negative examples (we know they are negative) and all instances from positive bags as positive examples (they *could* be positive), and use a classifier that can underfit the training data, i.e., that may deviate from the training labels for some examples. This naive idea alone can produce good results, but it is also the basis for algorithms iteratively refining this starting point: The mi-SVM algorithm [1] uses the predictions of the initial classifier to re-label some instances from positive bags to become negative, and alternates between re-training and re-labeling until convergence. A variant proposed by Hou et al. [6] is to prune instances from positive bags that are not clearly positive, and also iterate until convergence. We will find that for our task, the idea of improving initial results by relabeling instances is important as well.



(a) (b) (c) (d)

Figure 1: Demonstration of saliency mapping: Network input (a), gradient (b), guided backpropagation (c) and its positive values (d). Best viewed on screen.

3.3 Saliency Mapping

Saliency maps for neural networks have been popularized by Zeiler et al. [23] as a means of inspecting how a trained neural network forms its decisions. One of the most elegant forms computes the saliency map as the gradient of the network's output [1] with respect to its input [19]. For a single data point, this tells how and in which direction each input feature influences the prediction for that data point. In our case, the input is a spectrogram excerpt and the gradient shows for each spectrogram bin how an infinitesimal increase would affect the probability of predicting singing voice (demonstrated in Figure 1a and 1b, respectively). Unfortunately, for a deep neural network, an input feature can influence the output in convoluted ways: Some input may increase the output by decreasing activities in hidden layers that are negatively connected to the output unit.

To get a clearer picture, Springenberg et al. [21] propose *guided backpropagation*: At each layer, only propagate the positive gradient values to the previous layer. This limits the saliency map to showing how input features affect the output by a chain of changes in the same direction. Figure 1c demonstrates this for our example. A positive value (displayed in red) for a bin means increasing this bin will increase the output *by increasing activities in all layers in between*. Likewise, a negative value (displayed in blue) for a bin means that increasing it will decrease the output.

Note that the negative saliencies are not very useful: They form "halos" around the positive saliencies, indicating that the network hinges on the local contrast, and they are much less sharply localized. Assuming the hidden layers show a similar picture, this explains why ignoring negative gradients in guided backpropagation gives a sharper saliency map. To obtain a map of spectrogram bins corresponding to what the network used to detect singing voice, we keep the positive saliencies only (Figure 1d).

[1] Precisely, the pre-activation of the output unit, before applying the nonlinearity, as the sigmoid would dampen gradients at high activations.

4. RECIPE

Having the basic ingredients in place, we will now describe our recipe. We begin by showing how to use a naively trained network for temporal detection and spectral localization, and then highlight an observation about saliency maps that enables higher precision. Finally, we give a three-step training procedure that further improves results.

4.1 Naive Training

The easiest solution to dealing with the problem of incomplete labels – and the starting point of our recipe – is to pretend the labels were complete. We train an initial network by presenting all excerpts from instrumental songs as negative, and all excerpts from vocal songs as positive. This already works quite well: even on the training data, it produces lower output for excerpts of vocal songs that do not contain voice than for those that do.

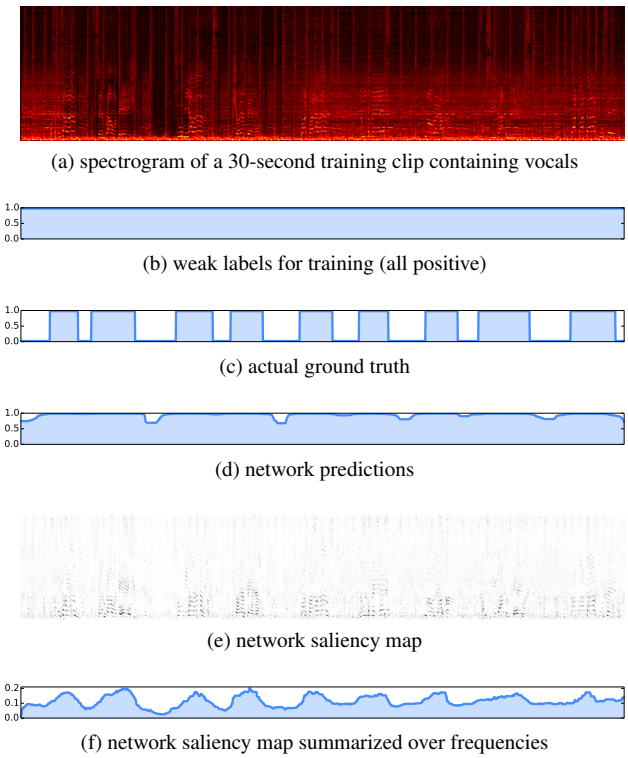
To obtain a temporal detection curve for a test song, we pass overlapping spectrogram excerpts of 115 frames through the network (with a hop size of 1 frame), recording each prediction. This way, for each spectrogram frame, we obtain a probability of singing voice being present in the surrounding ± 57 frames. As in [18], we post-process this curve by a sliding median filter of 56 frames (800 ms).

For spectral localization, we also pass overlapping spectrogram excerpts through the network, each time computing the 115×372-pixels saliency map for the excerpt. To combine these into a single map for a test song, we concatenate the central frames of the saliency maps. This gives a much sharper picture than an overlap-add of the excerpt maps. When used for vocal extraction, we apply two post-processing steps: As the saliency maps are very sparse, we apply Gaussian blurring with a standard dev. of 1 bin. And as the saliency values are very low, we scale them to a range comparable to the spectrogram and take the element-wise minimum of the scaled map and spectrogram.

4.2 Overshoot Correction

When examining the predictions of the initial, naively trained network, we observe that it regularly overshoots at the boundaries of segments of singing voice. Figure 2 illustrates the problem for a training example: Predictions only decline far away from vocal parts, and short pauses are glossed over. This is an artifact from training on weak labels: The network predicts singing voice whenever its 1.6-second input excerpt contains vocals, even if only at the edge, because such excerpts have never been presented as negative examples during training.

The saliency map provides additional information, though. By computing the saliency of the central frame of each excerpt (Figure 2e), we can check whether it was important for that excerpt's prediction. Summing up the saliency map over frequencies (Figure 2f) gives an alternative prediction curve that can be used to improve the precision of temporal detection. In the next section, we will use this observation to improve the network.



(a) spectrogram of a 30-second training clip containing vocals

(b) weak labels for training (all positive)

(c) actual ground truth

(d) network predictions

(e) network saliency map

(f) network saliency map summarized over frequencies

Figure 2: Network predictions (d) overshoot vocal segments (c) because input windows only partially containing vocals were always presented as positive examples (b). Summarizing the saliency map (e) over frequencies (f) allows to correct such overshoots.

4.3 Self-Improvement

A basic idea we described in Section 3.2 is that the naively trained network could be used to relabel the positive training instances, which necessarily contain false positives (compare Figure 2b to 2c). The intuition is that correcting even just a few of those and re-training should improve results.

We tried several variants of this idea: Relabeling by thresholding the initial network's predictions (using a low threshold to only relabel very certainly false positives), weighting positive examples by the initial network's predictions (so confidently positive examples would affect training more than others), or removing positive examples the initial network is not confident about. However, the only effect was that the bias of the re-trained network was lower, and iterating such a scheme lead to a network predicting "no" all the time. In hindsight, this is not surprising: The only positive instances we relabel this way are those the network got correct with naive training already, so it will not learn anything new when re-training.

We found a single scheme that does not deteriorate results: Training a second network to output the temporally smoothed predictions of the initial network for positive instances, and the actual labels for negative instances. It does not result in better predictions either, but in much clearer saliency maps with less noise in non-vocal sections. Using the technique of Section 4.2, we find that for such a second network, the summarized saliency maps alone actually provide a better temporal detection curve than the network output, as it does not suffer from overshoot.

Iterating the latter scheme by re-training on the second network's predictions does not help. However, we can train a third network to output the temporally smoothed summarized saliency maps of the second network for positive instances, again keeping negative instances at their true labels. To bring them to a suitable range for training (i.e., between 0 and 1), we scale them by 10 and apply the $\tanh$ function. Finally, this third network gives predictions that do not need any overshoot correction. It can be seen as finding a more efficient way of computing the summarized saliency map of the second network.

Put together, our recipe consists of: (1) Training a first network (subsequently called CNN-α) on the weak instance labels, (2) training a second network (CNN-β) on the predictions of CNN-α, (3) training a third network (CNN-γ) on the summarized saliency maps of CNN-β.

5. EXPERIMENTS

To test our approach, we train networks on a dataset of 10,000 weakly-annotated 30-second snippets, then evaluate them on two public datasets for temporal detection and two public datasets for spectral localization.

5.1 Datasets

5.1.1 Training and Development

We collected 10,000 30-second song snippets of 10,000 artists. Using a custom web interface, 5 annotators sorted 2,000 snippets each into vocal and non-vocal ones, where "vocal" was defined to be "any use of human vocal chords". Annotators were allowed to skip examples they were very unsure about or that only contained voice-like sound effects, leaving 9751 annotated clips, 6772 of which contain vocals. To check inter-annotator agreement, we had 100 clips be labeled by 6 annotators. Of those, annotators agreed on 97 clips, the remaining 3 were skipped by at least one annotator and disagreed on by others.

We annotate the multiply-annotated clips with sub-second granularity to be used for testing, and keep the remaining clips for training. We further annotate 100 of the training clips finely to train a network equivalent to the one of Schlüter et al. [18] (dataset *In-House A* in that work).

The finely annotated positive clips feature vocals for 70% of the running time (between 13% and 99% per clip). Extrapolating, we thus expect CNN-α to be presented with 30% false positives among its positive training instances.

5.1.2 Testing

For testing temporal detection, we use two public datasets finely annotated with singing voice presence:

– *RWC*, collected by Goto et al. [5] and annotated by Mauch et al. [15], contains 100 pop songs.

– *Jamendo*, curated by Ramona et al. [17], contains 93 songs. It comes with a predefined train/test split, but we use all songs for testing.

		internal		RWC		Jamendo	
		AU-ROC	max acc.	AU-ROC	max acc.	AU-ROC	max acc.
CNN-α	pred.	.911	.888	.879	.856	.913	.865
	sal.	.955	.896	.912	.843	.930	.849
CNN-β	pred.	.922	.888	.890	.861	.923	.875
	sal.	.970	.916	.936	.883	.955	.894
CNN-γ	pred.	.970	.915	.939	.887	.960	.901
	sal.	.965	.914	.931	.884	.950	.898
CNN-fine	pred.	.979	.930	.947	.882	.951	.880
	sal.	.969	.909	.937	.883	.948	.885

Table 1: Temporal detection results for the three steps of self-improvement on weak labels (Section 4.3) as well as a network trained on fine labels, for three datasets.[2]

For spectral localization, we use two public datasets that come with separated vocal tracks:

– *ccMixter*, collected by Liutkus et al. [11], consists of 50 recordings from `ccmixter.org`.

– *MedleyDB*, compiled by Bittner et al. [2], contains 174 songs. We only use the 52 songs that feature vocals (singer or rapper) and do not have bleed between tracks. Downmixes are provided, we obtain the corresponding vocal tracks by mixing all vocal stems per song.

5.2 Temporal Detection Results

Singing voice detection is usually evaluated via the classification error at a particular threshold [9, 10, 15, 18]. However, different tasks may require different thresholds tuned towards higher precision or recall. Furthermore, tuning the threshold towards some goal requires a finely-annotated validation set, which we assume is not available in our setting. We therefore opt to assess the quality of the detection curves, rather than hard classifications. Specifically, we compute two measures: The Area Under the Receiver Operating Characteristic Curve (AUROC), and the classification accuracy at the optimal threshold. The former gives the probability that a randomly drawn positive example gets a higher prediction than a randomly drawn negative example, and the latter gives a lower bound on the error.

Table 1 shows the results. We compare four CNNs: CNN-α to CNN-γ from the three-step self-improvement scheme of Section 4.3, and CNN-fine, a network of the same architecture trained on 100 finely-annotated clips, as a reimplementation of the state-of-the-art by Schlüter et al. [18]. For each network, we assess the quality of its predictions and of its summarized saliency map, on the held-out part of our internal dataset as well as the two public datasets. [2]

We can see that for CNN-α, summarized saliencies are better than its direct predictions in terms of AUROC, but worse in terms of optimal classification error. CNN-β, which is trained on the predictions of CNN-α, performs strictly better than CNN-α and gains a lot when using summarized saliencies instead of predictions. CNN-γ is trained to predict the saliencies of CNN-β and matches or even outperforms those. Its saliency maps do not provide any benefit. Figure 3 provides a qualitative comparison of the predictions for the three networks.

Comparing results to CNN-fine, we see that it performs comparable to CNN-γ: It is strictly better on the internal test set (which is taken from the same source as the training data), but not on the public test sets, indicating that CNN-γ profits from its larger training set despite the weak labels. The summarized saliencies of CNN-fine do not provide any improvement, which was to be expected: There is no overshoot they could correct as in Section 4.2.

[2] Note that we did not train on the public datasets and used the full datasets for evaluation, so results are not directly comparable to literature.

	ccMixter			MedleyDB		
	prec.	rec.	F_1	prec.	rec.	F_1
baseline	.324	.947	.473	.247	.955	.361
KAML [12]	.597	.681	.627	.416	.739	.484
CNN-α	.575	.651	.603	.467	.637	.497
CNN-β	.565	.763	.643	.522	.618	.529
CNN- fine	.552	.795	.646	.494	.669	.528

Table 2: Spectral localization results for the baseline of just predicting the spectrogram of the mix, a voice/music separation method, and saliency maps of three networks.

5.3 Spectral Localization Results

Spectral localization of singing voice, i.e., identifying the spectrogram bins containing vocals, is not an established task. The closest to this is singing voice extraction, which is evaluated with standard source separation measures (Source to Distortion Ration, Source to Interference Ratio). However, developing our method into a full-fledged source separator is out of scope for this work. We therefore resort to comparing the saliency map produced by the network for a mixed signal to the spectrogram of the known pure-vocal signal. Specifically, we compute a generalization of precision, recall and F_1-measure towards real-valued (instead of binary) targets: For a predicted saliency map P_{ij} and pure-vocal spectrogram T_{ij}, we define the total amount of true positives as $t = \sum_{i,j} \min(P_{ij}, T_{ij})$. We then obtain precision as $p = t / \sum_{i,j} P_{ij}$, recall as $r = t / \sum_{i,j} T_{ij}$, and F_1-measure as $f = 2pr/(p + r)$.

Results are shown in Table 2. We first compute a simple baseline: Using the spectrogram of the mix as our vocal predictions. In theory, this should give 100% recall, but since the songs are mastered, the mix spectrogram is sometimes lower than the spectrogram of the vocal track, leaving a gap. We then obtain results for KAML [12], a refined implementation of KAM [11] described in Section 2. As the vocal prediction P_{ij}, we compute the spectrogram of the vocal signal it extracts, clipped to 8 kHz to be comparable to our network's saliency maps. Finally, we evaluate the saliency maps of CNN-α, CNN-β and CNN-fine, post-processing them as described in Section 4.1. We omit CNN-γ as it is tailored for temporal detection and will not produce better saliencies than CNN-β.

We find that all methods are comparably far from the baseline, with the CNNs obtaining higher F_1-measure than KAML. As in temporal detection, CNN-β has an edge over CNN-α, and is close to CNN-fine. While these results look promising, it should be noted that the saliency maps will not necessarily be better for source separation: A lot of the improvement in F_1-score hinges on the fact that the saliency maps are often perfectly silent in passages that do not contain vocals, while KAML still extracts parts of background instruments. To obtain high recall, for passages that do contain vocals, the post-processed saliency maps include more instrumental interference than KAML.

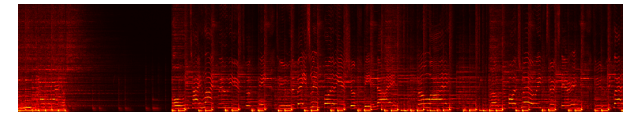
6. DISCUSSION

We have explored how to train CNNs for singing voice detection on coarsely annotated training data and still obtain temporally accurate predictions, closely matching performance of a network trained on finely annotated data. Furthermore, we have investigated a method for localizing the spectral bins that contain singing voice, without requiring according ground truth for training. We expect the recipe to carry over from human voice to musical instruments, if good contrasting examples are available – training on weakly-annotated data can only learn to distinguish instruments that occur independently from one another in different music pieces.

While our results are promising, there are a few shortcomings that provide opportunities for further research. For one, our networks produce good prediction curves, but when used for binary classification, we still need to choose a suitable threshold (e.g., optimizing accuracy, or a precision/recall tradeoff). We did not find good heuristics for selecting such a threshold solely based on weakly labeled examples. Secondly, comparing training on 10,000 weakly-labeled clips against 100 finely-labeled clips is clearly arbitrary. The main purpose in this work was to show that training from weakly-labeled clips can give high temporal accuracy *at all*, which is useful if such weak labels are easy to obtain. For future work, it would be interesting to compare the two methods on more even grounds. Specifically, we could investigate if weak labeling, fine labeling or a combination of both provides the best value for a given budget of annotator time.

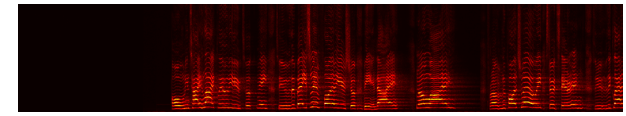
The spectral localization results could be a starting point for instrument-specific source separation. But as for temporal detection, this route would first have to be compared on even grounds to learning from finely-annotated or source-separated training data, and its viability depends on how easily these types of data are obtainable. And in contrast to temporal detection, it requires further work to be turned into a source separation method.

Finally, the kind of saliency maps explored in this work could be used for other purposes: For example, it can be used to visualize and auralize precisely which content in a spectrogram was responsible for a particular false positive given by a network, and thus give a hint on how to enrich the training data to improve results.



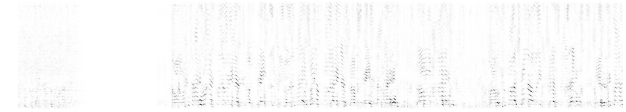
(a) spectrogram of a 30-second test clip containing vocals

(b) corresponding ground truth

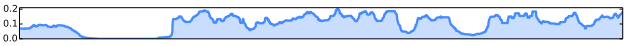


(c) spectrogram of corresponding vocal track

(d) predictions of CNN-α (trained on weak labels)

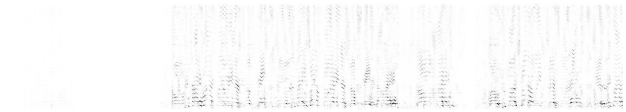


(e) saliency map of CNN-α

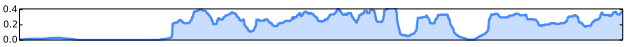


(f) summarized saliency map of CNN-α

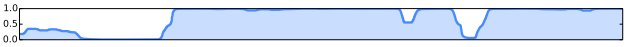
(g) predictions of CNN-β (trained on predictions of CNN-α)



(h) saliency map of CNN-β



(i) summarized saliency map of CNN-β



(j) predictions of CNN-γ (trained on tanh squashed sal. of CNN-β)

Figure 3: Qualitative demonstration of the self-improvement recipe in Section 4.3 for a single test clip (0:24 to 0:54 of "Vermont" by "The Districts", part of the MedleyDB dataset [2]). Visit `http://ofai.at/~jan.schlueter/pubs/2016_ismir/` for an interactive version.

7. ACKNOWLEDGMENTS

The author would like to thank the anonymous reviewers for their valuable comments and suggestions. This research is funded by the Federal Ministry for Transport, Innovation & Technology (BMVIT) and the Austrian Science Fund (FWF): TRP 307-N23 and Z159. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of a Tesla K40 GPU used for this research. Finally, we thank the authors and co-developers of Theano [22] and Lasagne [3] the experiments build on.

8. REFERENCES

[1] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15*, pages 577–584. 2003.

[2] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *Proc. of the 15th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, Taipei, Taiwan, Oct 2014.

[3] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, et al. Lasagne: First release., Aug 2015.

[4] J. Foulds and E. Frank. A review of multi-instance learning assumptions. *Knowledge Engineering Review*, 25(1):1–25, 2010.

[5] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Popular, classical, and jazz music databases. In *Proc. of the 3rd Int. Conf. on Music Information Retrieval (ISMIR)*, pages 287–288, Oct 2002.

[6] L. Hou, D. Samaras, T. M. Kurç, Y. Gao, J. E. Davis, and J. H. Saltz. Efficient multiple instance convolutional neural networks for gigapixel resolution image classification. *CoRR*, abs/1504.07947v3, 2015.

[7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of the 32nd Int. Conf. on Machine Learning (ICML)*, 2015.

[8] J. D. Keeler, D. E. Rumelhart, and W. K. Leow. Integrated segmentation and recognition of hand-printed numerals. In *Advances in Neural Information Processing Systems 3*, pages 557–563. 1991.

[9] S. Leglaive, R. Hennequin, and R. Badeau. Singing voice detection with deep recurrent neural networks. In *Proc. of the 2015 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, Australia, Apr 2015.

[10] B. Lehner, G. Widmer, and S. Böck. A low-latency, real-time-capable singing voice detection method with LSTM recurrent neural networks. In *Proc. of the 23th European Signal Processing Conf. (EUSIPCO)*, Nice, France, 2015.

[11] A. Liutkus, D. Fitzgerald, Z. Rafii, B. Pardo, and L. Daudet. Kernel additive models for source separation. *IEEE Transactions on Signal Processing*, 62(16):4298–4310, Aug 2014.

[12] A. Liutkus, D. Fitzgerald, and Z. Rafii. Scalable audio separation with light kernel additive modelling. In *Proc. of the 2015 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, Australia, Apr 2015.

[13] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. of the 30th Int. Conf. on Machine Learning (ICML)*, 2013.

[14] M. I. Mandel and D. P. W. Ellis. Multiple-instance learning for music information retrieval. In *Proc. of the 9th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, Philadelphia, USA, 2008.

[15] M. Mauch, H. Fujihara, K. Yoshii, and M. Goto. Timbre and melody features for the recognition of vocal activity and instrumental solos in polyphonic music. In *Proc. of the 12th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2011.

[16] Z. Rafii and B. Pardo. REpeating Pattern Extraction Technique (REPET): A simple method for music/voice separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(1):73–84, Jan 2013.

[17] M. Ramona, G. Richard, and B. David. Vocal detection in music with support vector machines. In *Proc. of the 2008 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1885–1888, 2008.

[18] J. Schlüter and T. Grill. Exploring data augmentation for improved singing voice detection with neural networks. In *Proc of the 16th Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, Malaga, Spain, 2015.

[19] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop of the 2nd Int. Conf. on Learning Representations (ICLR)*, Banff, Canada, 2014.

[20] P. Smaragdis and J. C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, pages 177–180, Oct 2003.

[21] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *Workshop of the 3rd Int. Conf. on Learning Representations (ICLR)*, San Diego, USA, 2015.

[22] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[23] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proc. of the 13th European Conf. on Computer Vision (ECCV)*, pages 818–833, Zürich, Switzerland, 2014.

[24] Z.-H. Zhou and M.-L. Zhang. Neural networks for multi-instance learning. Technical report, Nanjing University, China, Aug 2002.