Two Convolutional Neural Networks for Bird Detection in Audio Signals

Thomas Grill and Jan Schlüter
Austrian Research Institute for Artificial Intelligence
Freyung 6/6, Wien, Austria
Email: {thomas.grill,jan.schlueter}@ofai.at

***Abstract*—We present and compare two approaches to detect the presence of bird calls in audio recordings using convolutional neural networks on mel spectrograms. In a signal processing challenge using environmental recordings from three very different sources, only two of them available for supervised training, we obtained an Area Under Curve (AUC) measure of 89% on the hidden test set, higher than any other contestant. By comparing multiple variations of our systems, we find that despite very different architectures, both approaches can be tuned to perform equally well. Further improvements will likely require a radically different approach to dealing with the discrepancy between data sources.**

I. Introduction

Detecting the presence of bird calls in audio recordings can serve as a basic step for wildlife and biodiversity monitoring. To help advance the state of the art in automating this task, Stowell et al. [1] organized a *Bird audio detection challenge*.[1] Specifically, participants were asked to build algorithms that predict whether a given 10-second recording contains any type of bird vocalization, regardless of the species. For recent surveys of existing approaches, see [1, Sec. 3] and [2, Sec. 2].

The authors took part in the challenge with two independent submissions (*bulbul* and *sparrow*), both deploying convolutional neural networks applied to spectrograms. In the following, we describe the common denominators as well as individual prerequisites and strengths of the approaches. Section II describes the data used in the challenge, before Section III goes into depths regarding the methods of supervised learning used to tackle the problem. Section IV provides an overview of the results obtained, joined by a conclusion and outlook in Section V.

II. Data

A. Data sources

The *Bird audio detection challenge* provides data from three different sources, as described on its website: First, field recordings from the freefield1010 project [3], a collection of excerpts from field recordings originating from the FreeSound[2] online database, being very diverse in location and environment. Second, ten-second smartphone audio recordings, coming from a bird-sound crowdsourcing research spinout called Warblr[3]. The audio covers a wide distribution of UK locations and environments, and includes weather noise, traffic noise, human speech and even human bird imitations. The third dataset comes from the TREE research project[4], which is deploying unattended remote monitoring equipment in the Chernobyl Exclusion Zone, with its audio covering a range of bird vocalizations, weather, large mammal and insect noise sampled across various environments.

B. Data structure

According to the challenge website, the provided training data comes from freefield1010 (7690 examples) and Warblr (8000 examples), the testing data mostly from Chernobyl and to a smaller extent from Warblr (8620 examples altogether). Each training example comes with a single human annotation if birds are present anywhere in the audio (1), or no birds present at all (0). Most of the files are 10 seconds long, but there are exceptions with a duration of up to 22 seconds or down to only one second. Notably, the freefield1010 dataset contains examples that are predominantly negative (25% bird presence), while Warblr contains mostly positively annotated examples (76% bird presence).

The representation of the data we used for machine learning consists of Mel-scaled log-magnitude spectrograms with 80 bands. In order to obtain a clearer picture of the data structure, we performed clustering on some simple features derived from those spectrograms: per example and per frequency mean, standard deviation, 1-percentile (quasi-minimum excluding outliers) and 99-percentile (quasi-maximum excluding outliers), forming a 320-dimensional vector per audio file. After a PCA (variance coverage 95%, reducing to 11 dimensions), we clustered agglomeratively using Ward linkage[5].

In Figure 1, train and test data sets are clustered separately: eight clusters for the training data and four for the test data. Test clusters 1 and 3 are quite similar, comprising 7206 items (84% of the test data) of a rather low audio quality (high 1-percentile, low standard deviation, indicating noisy sound with low dynamics). For these clusters, matches to the training set can only be found partly in train cluster 7 and, vaguely, in

[1] http://machine-listening.eecs.qmul.ac.uk/bird-audio-detection-challenge, visited 2017-02-20

[2] http://freesound.org, visited 2017-02-20

[3] http://warblr.net, visited 2017-02-20

[4] https://wiki.ceh.ac.uk/display/NRT/NERC+RATE+TREE+Home, visited 2017-02-20

[5] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html, visited 2017-02-20

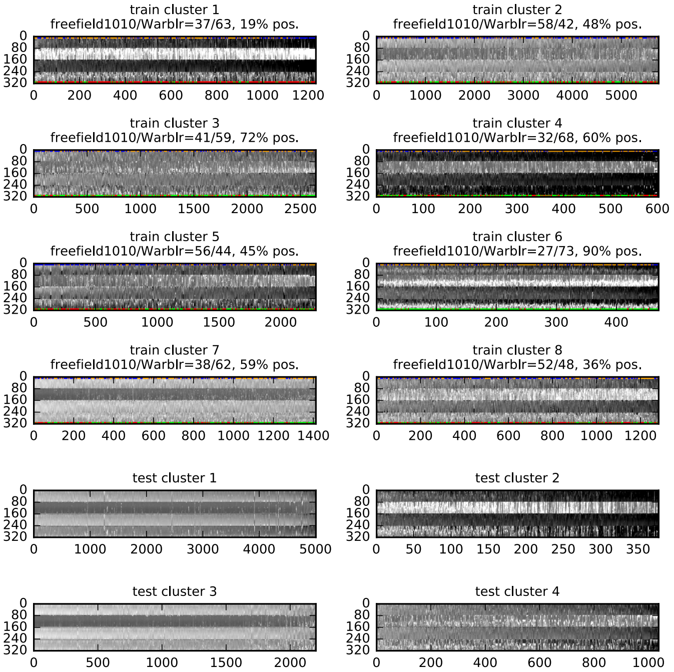


Fig. 1. Clusters in train (top eight) and test data (bottom four). The four discernible bands per subplot (on the y-axis) are 80 components of mean, standard deviation, 1-percentile and 99-percentile, respectively, accumulated over time for each example spectrogram (along the x-axis). On the top of each training data subplot, examples from the freefield1010 dataset are encoded with small blue dots, examples from the Warblr dataset in orange. On the bottom, green/red dots indicate bird presence or absence, respectively.

train cluster 2. Both latter clusters come from mixed sources with quite balanced absence/presence annotations. Test cluster 2 (high dynamics, low noise) with only 377 items can be identified with train clusters 1 and 6, both mostly from the Warblr source (63% and 73%), the first one with mostly negative (80%), the latter with predominantly positive labels (90%). Test cluster 4 (mixed quality, 1037 items) matches parts of train clusters 5 and 8, both of mixed origin and annotation.

All in all, the structure of the data represents a challenging situation for a supervised machine learning approach: Mostly positive examples from one source, mostly negative examples from another source with different characteristics, and test data for which predictions are desired predominantly from yet another source.

III. Method

Our approach to the Bird audio detection challenge deploys feed-forward CNNs trained on Mel-scaled log-magnitude spectrograms. The task poses two main challenges: Firstly, the label of an audio file can be determined by very local events (e.g., short chirps), sometimes less than half a second (see Figure 2a). Secondly, as stated already in Section II, the test data exhibits very different characteristics from training data. We compare two principally different network architectures (see Tables I and II) addressing the former, and attempt to overcome the latter with various training and pre/post processing techniques.

TABLE I
Network architecture of *bulbul* submission

Input	1×1000×80
Conv(3×3)	16×998×78
Pool(3×3)	16×332×26
Conv(3×3)	16×330×24
Pool(3×3)	16×110×8
Conv(3×1)	16×108×8
Pool(3×1)	16×36×8
Conv(3×1)	16×34×8
Pool(3×1)	16×11×8
Dense	256
Dense	32
Dense	1

TABLE II
Network architecture of *sparrow* submission

Input	1×701×80
Conv(3×3)	32×699×78
Conv(3×3)	32×697×76
Pool(3×3)	32×232×25
Conv(3×3)	32×230×23
Conv(3×3)	32×228×21
Conv(3×19)	64×226×3
Pool(3×3)	64×75×1
Conv(9×1)	256×67×1
Conv(1×1)	64×67×1
Conv(1×1)	1×67×1
GlobalMax	1

A. Input features

For each audio file under analysis, we first compute an STFT magnitude spectrogram with a window size of 1024 samples at 22.05 kHz sample rate with 70 frames per second (hop size 315 frames), apply a mel-scaled filter bank of $n = 80$ triangular filters from 50 Hz to 11 kHz (*bulbul*) or 10 kHz (*sparrow*, to leave room for pitch-shifting, see Section III-D) and scale magnitudes logarithmically. The features are normalized per frequency band to zero mean and unit variance. This is implemented using a batch normalization step [4] prior to the first network layer – we found this works as well as manually standardizing the features, but is more convenient. Finally, for the *bulbul* submission, from each spectrogram we subtract its mean over time, as a simple way of removing frequency-dependent (colored) noise.

B. Global architecture (Submission bulbul)

This highest-scoring submission to the challenge[6] uses a network with a wide receptive field of 1000 frames (14 s) processed into a single binary output. As shown in Table I, a sequence of four combinations of convolution and pooling condenses the input of 1000×80 into 16 feature maps of 11×8 units. Three dense layers with 256, 32 and 1 unit(s) classify the condensed features. Except for the sigmoid output layer, each convolution and dense layer is followed by the leaky rectifier nonlinearity $\max(x, x/100)$. The total number of trainable network parameters is 373169.

C. Local architecture (Submission sparrow)

A possible disadvantage of the global architecture is that the network has to learn to detect birds at different temporal positions within the receptive field, to predict the correct label even if a file contains just a single chirp. In a separate line of submissions, we attempted to treat bird detection as a local task, with a short receptive field of 103 frames (1.5 s). Since we do not know the label of short excerpts, only for a full recording, this is a multiple-instance learning problem. It follows the *standard MI assumption* [5]: a recording is labeled positively if and only if at least one of its excerpts is positive.

[6] Code repository on https://jobim.ofai.at/gitlab/gr/bird_audio_detection_challenge_2017, visited 2017-02-23

The architecture in Table II reflects this: It uses convolutional and pooling layers to process the spectrogram into a one-dimensional sequence, then takes the global maximum. As in the *bulbul* submission, every convolution is followed by the leaky rectifier except for the final one, which has a sigmoid. The total number of network parameters is 309843.

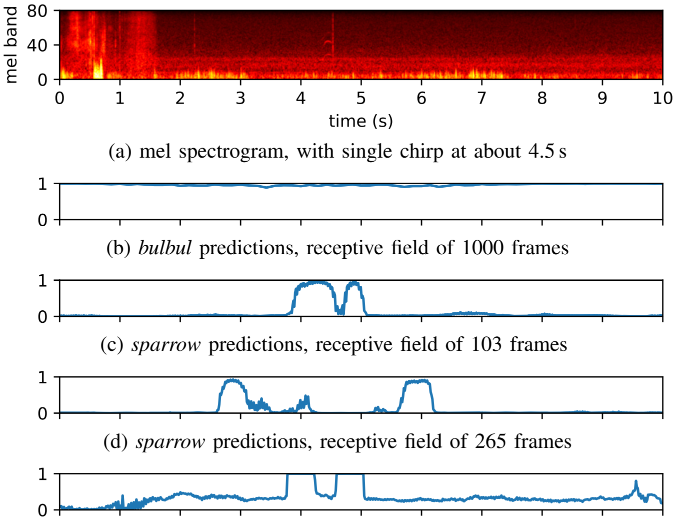
Note that the way the network is designed, it can be applied to any recording of at least 103 frames, producing a temporal sequence of local predictions the maximum is taken over. Each local prediction considers a 103-frame excerpt, with consecutive excerpts overlapping by 94 frames.

D. Training

Training is done by stochastic gradient descent on mini-batches of 64 (*bulbul*) or 32 (*sparrow*) examples, using the ADAM update rule [6] with an initial learning rate of 0.001, reduced by a factor of 10 two times during training. *sparrow* uses a fixed scheme, training for 80,000 updates with learning rate drops after 40,000 and 60,000 updates. *bulbul* uses a variable scheme dropping the learning rate whenever the training error does not improve over three consecutive episodes of 1500 updates, resulting in about the same number of updates. *sparrow* is trained on excerpts of 701 frames, *bulbul* on 1000 frames. Files shorter than required are looped up to the length needed.

Especially with the strongly different test data characteristics, a critical point in training is regularization, to avoid overfitting not only to the specific training examples, but also to the sources they are drawn from. As a general measure, for both architectures, we apply 50% dropout to the inputs of the last three layers. In *sparrow*, we also apply batch normalization to all layers. Specific to the task, we employ different ways of augmenting the training data: In order to achieve temporally translational invariance (the position of a bird vocalization in the spectrogram is irrelevant), the training examples are cyclically shifted in time. To become less sensitive to the exact pitches of bird calls, we employ random pitch shifting: up to ±1 mel band for *bulbul*, by linearly interpolated shifting of the mel spectrograms, and up to ±10% for *sparrow*, by spreading/compressing the mel filterbank. Finally, to generalize to different noise floors, in training the *sparrow* system, the first 8 examples of each mini-batch are mixed with the central frames of the last 8 examples of each mini-batch, with a coefficient between 0 and 0.4 for the noise and a corresponding coefficient between 1 and 0.6 for the signal. This provides a sound floor constant over time, encouraging the network to ignore static background. We also tried mixing full recordings, adapting the label accordingly, but this deteriorated results for both architectures.

As another way to better generalize towards the test set, we experimented with pseudo-labeling: After training a first model, we compute predictions for the test examples and add some of them to the training set for a second model – either using the real-valued predictions as soft labels, or using hard labels, limited to the most confidently predicted test examples. This did not improve results for either of our systems.



(a) mel spectrogram, with single chirp at about 4.5 s

(b) *bulbul* predictions, receptive field of 1000 frames

(c) *sparrow* predictions, receptive field of 103 frames

(d) *sparrow* predictions, receptive field of 265 frames

(e) *sparrow* predictions, 103 frames, global mean pooling

Fig. 2. Predictions of different variants on file warblrb10k/92ee2259-6ed8-4120-9511.wav, a recording with a single short chirp. *bulbul* (b) confidently detects the bird call for all cyclic rotations of the input. At test time, only a single prediction is computed. *sparrow* (c–e) detects the call whenever it is near the edge of its receptive field, producing a double peak (see Sect. IV-B). At test time, the maximum over the local predictions is taken. Training with global mean instead of global maximum strongly impairs discrimination (e).

E. Predicting

After training, to obtain a prediction for a recording, we loop it as needed to fill the network's receptive field. For *bulbul*, we then obtain a prediction for non-overlapping 1000-frame excerpts (for most files in this dataset, there only is a single such excerpt) and take their mean. For *sparrow*, we cyclically pad the recording with half a receptive field on either side, and modify the network to internally produce a prediction at every frame instead of every 9th frame (using overlapping pooling and dilation [7], [8]). As in training, the network then takes the global maximum over these local predictions.

To improve results, for both submissions, we average the file-wise predictions of five networks trained on each of five cross-validation splits of the training data. For *sparrow*, we also tried averaging the local predictions instead, but this worked worse in cross-validation on the training set.

IV. Results

The Bird audio detection challenge featured a submission site where contestants could upload their predictions for the test set, at most once every 24 hours. A 'preview score' was then computed giving the AUC (area under ROC curve) for a subset of 1293 files from the test set. Scores for the full test set[7] were published after the contest deadline, deviating from the preview scores by some tenths of a percent for the top submissions. For development, we also computed the AUC using five-fold cross-validation on the training set.

[7] http://c4dm.eecs.qmul.ac.uk/events/badchallenge_results, last visited 2017-03-04

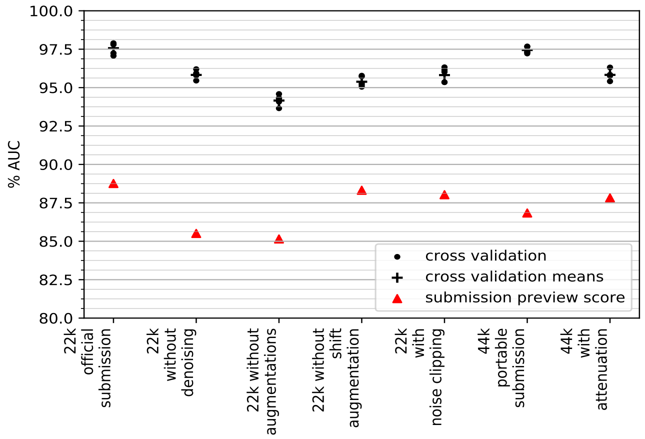


Fig. 3. Results for variants of the *bulbul* architecture.

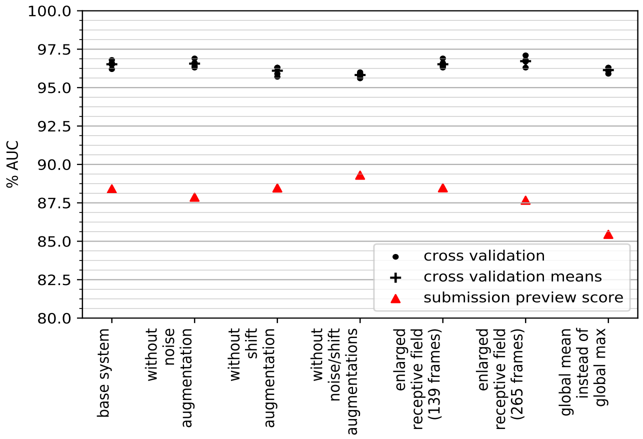


Fig. 4. Results for variants of the *sparrow* architecture.

As a consequence from the differences between the train and test data, the scores computed on the test set deviate considerably from our cross-validation scores. The correlation between scores calculated on the train and test domains is low, with a Pearson correlation value of 0.40 (for 19 samples), implying that effects of experimental variations hardly extrapolate from cross-validation scores to the test scores. We will thus always report both the cross-validation and the preview scores.

In the following, we will look at variations of our two submissions, to see how important their different components are, and also investigate some unexpected behaviors.

A. *Submission bulbul*

Figure 3 shows AUC results for the *bulbul* architecture, including both the submission preview scores and cross-validation scores. The leftmost entry shows the architectural variant yielding the highest preview score on the test set (88.76%). Leaving away the denoising preprocessing step considerably degrades performance on both the cross-validation and preview scores (85.51%). As expected, computation without any augmentations (especially the cyclic shifting) also impairs both scores, with the preview at 85.15%. Omitting just the spectral shift augmentation still has a notable impact on the cross-validation score, without much effect on the preview score (88.32%). Many of the audio examples exhibit silence, clicks, etc. at the beginning of the files, obviously from switching on the recording device. A preprocessing step for clipping these noises was introduced, not improving the results though (preview score 88.03%).

It must be noted that details of the audio preprocessing can have a crucial impact on the result. We discovered that the choice of algorithm for resampling the audio signal to 22 kHz can be responsible for a significant degradation of bird detection performance, potentially lowering AUC by some 2%.[8] This causes a considerable portability issue. The reason seems to be the type of low-pass filter employed prior to the resampling. In the context of our problem, a (usually deemed 'bad') shallow filter slope works better than a 'good' steep (brick-wall type) filter. The effects could be shown by artificially imposing a comparable frequency attenuation on the outcome of the latter filter, recovering half of the performance loss. At this time, though, we cannot fully pinpoint why the spectral characteristics are not sufficiently straightened by the batch normalization step.

B. *Submission sparrow*

Figure 4 shows results for the *sparrow* architecture. The leftmost entry denotes the system as described in the previous section, obtaining a preview score of 88.41%. Omitting the noise augmentation lowers the preview score (87.87%) without affecting the cross-validation score on the training data. Conversely, omitting the pitch shift augmentation lowers the cross-validation score without affecting the preview score. Surprisingly, omitting both augmentations lowers the cross-validation score and *raises* the test set preview score to 89.30%. Without access to the test set labels, we are unable to explore the reason.

While the scores confirm the hypothesis that bird calls are local events that can be detected with a small receptive field, a larger receptive field might allow the network to better adapt to the specific recording conditions and noise floor of a file, which vary wildly between recordings and data sources. However, increasing the field from 103 (1.5 s) to 139 frames (2 s) (by extending the 9×1 convolution in Table II to 13×1) does not change the scores compared to the base system, and increasing it further to 265 frames (3.8 s) even reduces the preview score. Looking at the networks' local predictions (before taking the maximum over time), we find something curious: For most bird calls, the predictions contain *two* peaks, half a receptive field before and after the event (see Figure 2). Investigating further, we find that these peaks are merged in the early stages of training, and become separated afterwards. The most likely explanation are mislabeled training examples:[9] When a training example has a negative label, but contains a

[8] The conversion software ffmpeg 2.8.10-0ubuntu0.16.04.1 as used by QMUL in comparison to our avconv version 9.18-6:9.18-0ubuntu0.14.04.1

[9] Manual inspection of errors on the validation set revealed many mislabeled files. For example, the file shown in Figure 2 has a negative label.

bird, the network will be trained to reduce the prediction at its current maximum, possibly leaving two side lobes. Once split, there is no incentive to rejoin the peaks. When changing the train/validation splits or the augmentation, some double peaks are merged, confirming the dependency on training data. Changing the training hyperparameters did not have any effect.

Finally, we investigated whether taking the maximum over local predictions is the correct approach. During training, it means the network is only updated for the maximal prediction per recording,[10] increasing it for positive examples and decreasing it for negative examples. For a file of a single bird call, this seems optimal. For a file full of bird chatter or devoid of birds, this possibly wastes information. For comparison, we thus modified the base system to take the *mean* over local predictions instead. This updates the network for all local predictions during training. As shown in Figure 2e, this leads to larger predictions on ambient noise, weakening discrimination between birds and background. Consequently, it reduces scores both on the validation and test set (85.45%). As a compromise between max and mean pooling, we can add a sliding average in front of the global maximum, or train on shorter excerpts (so the maximum is taken over a partial recording only). This keeps the validation score high, but also severely reduces the preview score.[11]

C. Comparison

Looking at the architectures again (Tables I/II), both networks mainly use max-pooling over time to reduce a long sequence of input features (the mel spectrogram) into a single prediction: *bulbul* interleaves pooling with feature processing, *sparrow* defers most pooling to the end. Both variants seem to be equally effective on the test set, with *bulbul* performing slightly better on the development set. Investigating validation files the networks classify differently, we find many difficult and mislabeled examples, but no systematic difference between the classifiers. A possible positive aspect of late pooling is that *sparrow* can localize calls in time, but the given datasets lack annotations to assess this quantitatively. Combining the best results of both systems by taking the mean of their predictions for each file, we obtain a preview score of 89.68%.

V. Conclusion

We have presented two deep learning based approaches for detecting bird calls in audio recordings. Despite using different network architectures, they perform very similarly. Moreover, they perform on par with other top submissions to the QMUL bird audio detection challenge (AUC 88.7% for our *bulbul* system, and 88.5%, 88.2%, 88.1%, 88.1% for the next four contestants), all of which use neural networks on spectrograms. This could indicate a glass ceiling: without fundamental changes to the training procedure, no further improvement may be possible.

A promising way forward is to take into account the specific acoustic characteristics of the test data. Our clustering reveals a possible grouping of examples into different sources that we could tap into. Training the network to become invariant to the source characteristics, such as by unsupervised domain adaptation [9] or specialized data augmentation, may reduce the gap between performance on the development and test set. Respective preliminary experiments have shown that this is not easily successful, though.

In any case, the first step should be to investigate whether there is room for improvement at all. To establish an estimate for an upper bound, a subset of both training and test files should be labeled by multiple annotators (see [10]). Given the amount of mislabeled examples we found in the training set, we suspect that we have already reached the limit for this part of the data.

Acknowledgment

The authors would like to thank the Vienna Science and Technology Fund (WWTF project MA14-018), the Austrian Federal Ministry for Transport, Innovation and Technology and the Austrian Science Fund (FWF project TRP 307-N23), and NVIDIA corporation. Furthermore, we thank the authors and co-developers of Theano [11] and Lasagne [12] the experiments were implemented in.

References

[1] D. Stowell, M. Wood, Y. Stylianou, and H. Glotin, "Bird detection in audio: a survey and a challenge," in *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on.* IEEE, 2016, pp. 1–6.

[2] D. Stowell and M. D. Plumbley, "Birdsong and C4DM: A survey of UK birdsong and machine recognition for music researchers," Centre for Digital Music, Queen Mary University of London, Tech. Rep. C4DM-TR-09-12, Aug 2010.

[3] ——, "An open dataset for research on audio field recording archives: freefield1010," *CoRR*, vol. abs/1309.5275, 2013.

[4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, Jul. 2015, pp. 448–456.

[5] J. Foulds and E. Frank, "A review of multi-instance learning assumptions," *Knowledge Engineering Review*, vol. 25, no. 1, pp. 1–25, 2010.

[6] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, San Diego, 2015.

[7] A. Giusti, D. C. Ciresan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," *CoRR*, vol. abs/1302.1700, 2013.

[8] T. Sercu and V. Goel, "Dense prediction on sequences with time-dilated convolutions for speech recognition," *CoRR*, vol. abs/1611.09288, 2016.

[9] Y. Ganin and V. S. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, 2015.

[10] A. Flexer and T. Grill, "The problem of limited inter-rater agreement in modelling music similarity," *Journal of New Music Research*, vol. 45, no. 3, pp. 239–251, 2016, pMID: 28190932.

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

[12] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri *et al.*, "Lasagne: First release." Aug 2015. [Online]. Available: http://dx.doi.org/10.5281/zenodo.27878

[10] Since the output only depends on the maximal prediction, the gradient of the output with respect to any non-maximal prediction is zero.

[11] This is what the official *sparrow* submission to the competition did.