

Compensating class imbalance for acoustic chimpanzee detection with convolutional recurrent neural networks

Franz Anders^{a,*}, Ammie K. Kalan^{b,d}, Hjalmar S. Kühl^{c,d}, Mirco Fuchs^a

^aLeipzig University of Applied Sciences, Laboratory for Biosignal Processing, Eilenburger Straße 13, 04317 Leipzig, Germany

^bDepartment of Anthropology, University of Victoria, PO Box 1700 STN CSC, Victoria, BC, V8W 2Y2 Canada

^cGerman Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig, Puschstraße 4, 04103 Leipzig, Germany

^dMax Planck Institute for Evolutionary Anthropology, Deutscher Platz 6, 04103 Leipzig, Germany

Abstract

Automatic detection systems are important in passive acoustic monitoring (PAM) systems, as these record large amounts of audio data which are infeasible for humans to evaluate manually. In this paper we evaluated methods for compensating class imbalance for deep-learning based automatic detection of acoustic chimpanzee calls. The prevalence of chimpanzee calls in natural habitats is very rare, i.e. databases feature a heavy imbalance between background and target calls. Such imbalances can have negative effects on classifier performances. We employed a state-of-the-art detection approach based on convolutional recurrent neural networks (CRNNs). We extended the detection pipeline through various stages for compensating class imbalance. These included (1) spectrogram denoising, (2) alternative loss functions, and (3) resampling. Our key findings are: (1) spectrogram denoising operations significantly improved performance for both target classes, (2) standard binary cross entropy reached the highest performance, and (3) manipulating relative class imbalance through resampling either decreased or maintained performance depending on the target class. Finally, we reached detection performances of 33 % *F1* for drumming and 5 % *F1* for vocalization, which is a > 7 fold increase compared to previously published results. We conclude that supporting the network to learn decoupling noise conditions from foreground classes is of primary importance for increasing performance.

Keywords: CRNN, pan troglodytes, bioacoustics, imbalance, pant-hoot, drumming

1. Introduction

Automatic detection of chimpanzee calls is of primary importance for automatic monitoring of wild chimpanzee populations. There is a multitude of monitoring applications, from assessing chimpanzee home ranges for behavioral studies to early-warnings systems for areas with human-wildlife conflict.[1, 2]

Passive acoustic monitoring (PAM) is one of the most widely employed methods for monitoring wild animals. Here, autonomous recording units (ARUs) are distributed over an area for constant soundscape recording. The main advantages of PAM are: (1) minimal intrusion, as humans are only required for installation and maintenance of devices, (2) sampling over large spatial and temporal scales, and (3) detection of animals in habitats where visual recognition is limited, e.g. dense rain forests. However, PAM also produces large amounts of audio data which quickly become infeasible to curate manually by humans. Consequently, algorithms for automatic detection of target species are of primary importance for PAM settings.[2]

Heinicke et al. [3] investigated an automatic system for detecting calls of various primate species in PAM recordings of a tropical forest. Their algorithm employed a conventional acoustic detection approach based on hand-crafted features and gaussian mixture models. Algorithm performances varied with respect to target species and call type, from 10 % *F1* for Diana monkeys and King colobus monkeys to 4 % for chimpanzee drumming and 0.2 % for chimpanzee vocalizations. To the best of our knowledge, this is the only paper to date which focused on automatic detection of chimpanzee calls in a PAM setting. Dev's Master Thesis [4] also investigated classification of chimpanzee calls, however for classification against the Urbansound8K [5] dataset classes (e.g. car horn or gun shot) instead of detection in natural soundscape recordings.

In recent years, deep-learning based methods have become prevalent and have largely replaced approaches based on hand-crafted features in automatic audio recognition systems. In the annual DCASE-challenges, deep-learning based systems became popular between 2016 and 2017 [6–9]. Research teams working on automatic animal call detection particularly adapted convolutional neural networks (CNNs) with spectrogram inputs: Bergler et al. [10] applied Res-Net [11] variants for detection of orca calls in long-term recordings; Bjorck et al. [12] applied Dense-

*Corresponding author

Email addresses: franz.anders@htwk-leipzig.de (Franz Anders), ammie07@gmail.com (Ammie K. Kalan), hjalmar.kuehl@idiv.de (Hjalmar S. Kühl), mirco.fuchs@htwk-leipzig.de (Mirco Fuchs)

Net CNNs [13] for detecting African forest elephants with PAM; Oikarinen et al. [14] applied siamese CNNs with stereo inputs to the detection of various marmoset monkey calls. Aodha et al. [15] investigated CNNs for bat detection. In an open challenge for bird audio detection in 2017[16], the top placed system "bulbul" likewise applied CNNs to spectrogram inputs.

However, research is still lacking in some areas of automatic primate call detection as well as automatic animal call detection in general.

(1) Consideration of target class rarity in PAM settings. The majority of recordings in long-term PAM recordings will comprise background noise rather than target calls [3, 12, 14]. Consequently, databases feature a heavy imbalance between background and target class. Numerous studies showed that class imbalances can have detrimental effects on automatic system performances, as classifiers are usually biased towards the majority class [17–19]. However, all previously mentioned studies worked with databases with strongly reduced amounts of background samples, biasing the class distribution towards the positive class. This bias was either already present in the respective database, or produced by the authors through discarding fixed percentages of noise samples. Additionally, nearly all papers measured system performances with metrics unsuited for unbalanced settings (accuracy or AUC ROC). These give overly optimistic results in recordings with heavy class imbalance as they are biased towards the majority class [19].

(2) Time-continuous detection. All previously mentioned deep-learning based systems approached detection tasks by classifying broader spectrogram patches of various seconds, e.g. 25-second patches [12]. An evaluated temporal context receives a single class label. This approach decreases the temporal resolution of training annotations as well as test time predictions. In general-purpose detection tasks, convolutional recurrent neural networks (CRNNs) with time-distributed outputs have recently become more prevalent [20]. They are capable of predicting each individual spectrogram time frame while incorporating broader spectrogram contexts. Such networks have not yet been applied to animal call detection tasks.

In this study, we investigated time-continuous detection of chimpanzee calls using CRNNs. The target calls were chimpanzee drumming and vocalizations in long-term PAM audio recordings of an African rain forest (Taï National Park, Côte d'Ivoire). We addressed a particular challenge imposed by the severe imbalance between target classes and background samples in the database, caused by the rarity of chimpanzee calls. The test set, with a total duration of 179 hours, contained merely ≈ 10 minutes of chimpanzee calls. Consequently, we studied various methods for compensating this imbalance. These methods comprised (a) spectrogram denoising for compensating the absolute rarity of target classes, and (b) alternative loss functions and resampling (over & undersampling). The novel contributions of this paper are:

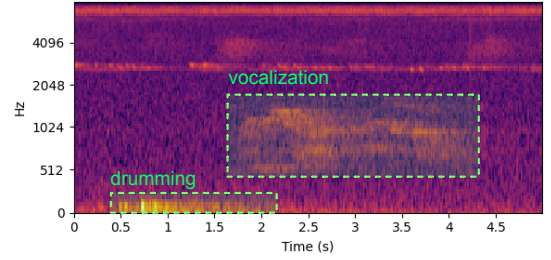


Figure 1: Example spectrogram of target classes.

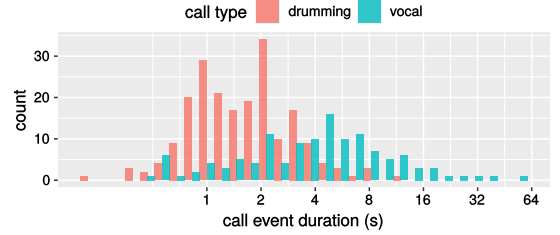


Figure 2: Histogram of call type durations

(1) the investigation of the feasibility of time-continuous detection with CRNNs for chimpanzee detection. To the best of our knowledge, this is the first application of this approach for animal species detection in long-term recordings. (2) a dedicated investigation of various methods for compensating class imbalance, including resampling, alternative loss functions and spectrogram denoising.

2. Materials and Methods

2.1. Dataset

The dataset was originally collected by AKK [1, 2] with the aim of developing an automated approach for detecting primate calls in PAM forest recordings [3]. Heinicke et al. [3] present the evaluation of this automated system.

The recording site was the western section of the Taï National Park, Côte d'Ivoire. The area sampled the territories of two chimpanzee communities. The soundscape of the park featured a wide variety of biogenic sounds, e.g. birds, insects, anthropogenic sounds, rain, wind etc. The recording setup comprised 20 ARUs distributed evenly across an area of $\approx 35 \text{ km}^2$. ARUs recorded in stereo with a sampling rate of 16 kHz and 16 bit depth. The recording period was from November 2011 to May 2012. ARUs recorded daily from 6 am to 6 pm on the full hour for 30 minutes. A total of 12 889 h of audio data was collected.

The original automated approach [3] targeted chimpanzees, *Pan troglodytes* ssp. *verus*, as well as three other primate species. The present study focuses exclusively on chimpanzees. Heinicke et al. defined two chimpanzee call types for detection: (1) **Drumming**, which is produced by chimpanzees when they repeatedly hit buttress roots of trees with their hands and feet. (2) **Vocalizations**, referring primarily to chimpanzee pant-hoots and screams for

long-distance communication. Drumming is characterized by short energy bursts with low frequency. Vocalizations in the data set are characterized by harmonic patterns with an estimated frequency range of 200 - 2000 Hz. Figure 1 shows an example spectrogram excerpt with both call types. For the remainder of this paper, we refer to call types as *classes* and call instances as *events* in accordance with the vocabulary established in general audio detection research [21].

To construct sets for training and validation of the automated system, the data pool was sampled, partitioned into sets of recordings, and annotated. Table 1 summarizes the datasets used in this study: The *complete test set* corresponds to the original test set constructed by Heinicke et al. [3]. They randomly sampled 358 recordings (of 30 minutes each) from the data pool, balanced across ARUs (one file per ARU per week) and time of day. This procedure ensured that the test set reflected diverse acoustic conditions for varying seasons, daytimes and sampling sites. We additionally constructed a *reduced test set* which comprised all recordings from the complete test set with at least one chimpanzee event. We used the reduced test set for repeated evaluation runs, as the complete test set was computationally expensive due to its size. The *training set* contained 44 additional recordings which were likewise randomly sampled (i.e. training and test set comprised of distinct recordings). Each individual recording in the training and test set was 30 min long. The *validation dataset* contained 25 additional recordings collected during a pilot study at the same location in 2010. Contrary to the other sets, recordings in the validation set had varying lengths with a mean duration of 1.3 min. Two trained experts for primate vocalizations annotated call events in these recordings with precise start- and end times [3].

A central characteristic of the dataset is the rarity of the target classes. The complete test set with approximately one week of recording time merely contained a total of 2.5 min of drumming and 7 min of vocalization events. The relative amount for drumming and vocalizations was 0.02 % and 0.06 % respectively. This imbalance is representative of the real-world prevalence of chimpanzee calls obtained using PAM in natural settings. Even the reduced test set, which biased the class distribution in favor of the target classes, contained 0.16 % and 0.47 % of drumming and vocalization events respectively. The training set contained 0.2 % and 0.35 % of drumming and vocalization events respectively.

We highlight that the imbalance between the number of positive class examples (i.e. target calls) and negative class examples (i.e. background samples) has two effects, according to the taxonomy of Weiss et al. [17]:

- *Absolute rarity*, i.e. low amounts of training examples for the target classes. This causes classifiers to overfit individual examples rather than learning generalized patterns for the target classes, particularly in deep-learning systems [21, 22].

Table 1: **Dataset overview**

		test complete	test reduced	training	validation
ARU record.	# recordings	358	50	44	25
	total duration	179 h	25 h	22 h	0.7 h
	% recordings with at least 1 chimp. call	14 %	100 %	50 %	76 %
drum.	# events	100	100	78	29
	total duration	149 s	149 s	159 s	96 s
	mean duration	1.29 s	96 s	1.76 s	2 s
vocal.	# events	50	50	53	23
	total duration	431 s	431 s	270 s	88 s
	mean duration	5.72 s	5.72 s	4.35 s	3 s

- *Relative imbalance* between background class and target class examples. This usually induces a prediction bias into the classifier to favor the majority class, while the minority class often is of greater interest to the user [17–19, 23]. However, the magnitude of the negative effect depends on the complexity of the classification task at hand. The algorithm might be completely unaffected if classes are linearly separable [19, 24].

2.2. Detection pipeline

Figure 3 gives an overview of the detection pipeline at training and test time. The pipeline consist of a series of *pipeline stages* which progressively process an input audio signal (i.e. ARU recording). At training time, the pipeline outputs a trained CRNN. At test time, the pipeline outputs indications on target class presence for spectrogram time frames.

The general approach and stage arrangement of *feature extraction*, *segmentation*, *CRNN*, *output concatenation* and *output thresholding* originates from Cakir et al.[20]. We additionally added the stages *spectrogram denoising* and *resampling*. These stages, together with the choice of the loss function, are the three components aimed at compensating class imbalance investigated in this study. Although Cakir’s pipeline is aimed at polyphonic detection tasks, we only considered single class detection in this study, i.e. a separate CRNN must be trained for drumming and vocalization. This restriction was imposed to study the effects for both classes individually.

The pipeline stages function as follows:

- (1) *input*: Input are ARU recordings $\mathbf{x}_i \in \mathbb{R}^{l_i}$ of length $l_i \in \mathbb{N}$ as time-domain audio signals, where i is the signal index. Signals are converted to mono and normalized to a peak amplitude of 1 to equalize loudness across signals. Ground truth annotations are tables which list occurrences

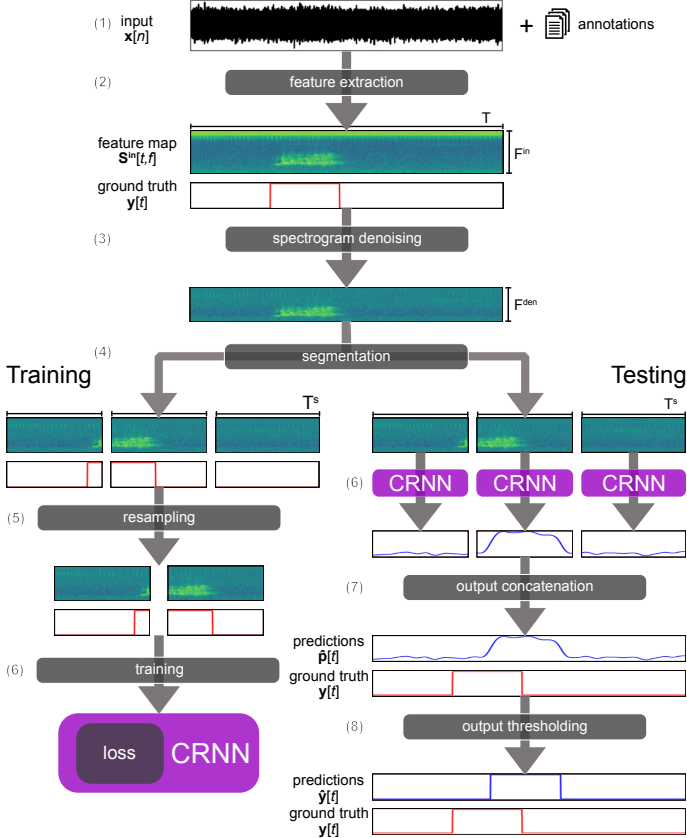


Figure 3: **Overview of the detection pipeline at training and test time.** In this example there is only one input signal \mathbf{x} , so that the index i is omitted.

of target calls with their start- and end times within signals.

(2) *feature extraction*: Signals \mathbf{x}_i are converted into spectrograms $\mathbf{S}_i^{\text{in}} \in \mathbb{R}^{T_i \times F^{\text{in}}}$, where $T_i \in \mathbb{N}$ is the respective amount of spectrogram time frames and $F^{\text{in}} \in \mathbb{N}$ is the amount of frequency bins. We employed mel-scaled spectrograms as it currently is the most prevalent choice in deep-learning based sound recognition systems [7–9, 25]. We adapted the variant of the DCASE baseline systems [26], which is the logarithm of a slaney-stile mel-firfilterbank of the linear magnitude spectrogram. We adapted the frame length of 40 ms and hop-length of 20 ms from Cakir et al. We used $F^{\text{in}} = 80$ mel-bands, i.e. we doubled Cakir’s frequency resolution to reduce the loss of potentially important frequency information, particularly in view of the low signal-to-noise-ratio in this task. The feature extraction process was implemented through the python library `librosa v.0.8`. [27]. Ground truth annotations are likewise converted to binary target vectors $\mathbf{y}_i \in \{0, 1\}^{T_i}$, where $\mathbf{y}_i[t] = 1$ encodes presence and 0 encodes absence of the target class in time frame t (alias ”positive” and ”negative” example).

(3) *spectrogram denoising*: This stage applies a series of denoising functions to spectrograms $f^{\text{den}}(\mathbf{S}_i^{\text{in}}) = \mathbf{S}_i^{\text{den}} \in \mathbb{R}^{T_i \times F^{\text{den}}}$. While the time axis size T_i remains unchanged,

the frequency axis might be reduced to F^{den} . This stage also z-standardizes all spectrograms through global statistics calculated on the training set.

(4) *segmentation*: While all spectrograms $\mathbf{S}_i^{\text{den}}$ contain the same amount of frequency bands, the number of time frames T_i might vary depending on the input duration for signal i . However the CRNN requires inputs with equally sized dimensions to be trained via mini batch training. Therefore spectrograms are segmented across the temporal axis into non-overlapping segments of fixed temporal length T^{seg} . Consequently, $\mathbf{S}_i^{\text{den}}$ is represented as a list of segments $\mathbf{S}_{i,j} \in \mathbb{R}^{T^{\text{seg}} \times F^{\text{den}}}$, where j is the segment index. For training, target vectors are likewise segmented into $\mathbf{y}_{i,j} \in \mathbb{R}^{T^{\text{seg}}}$. In this study, we used a fixed segment length of $T_s \triangleq 10$ s as a balance between computational efficiency and providing sufficient temporal context in spectrogram segments for recognition of target classes. For signals with $T_i \bmod T^{\text{seg}} \neq 0$ (some signals in the validation set), the last segment was partially overlapped with the penultimate to cover T_i completely if the overlap was $< 75\%$, or discarded otherwise.

(5) *resampling*: This stage alters the distribution between positive and negative examples through over and undersampling of segments. This stage is only active for training to not affect class distribution when validating the system. Section 2.6 provides details on the resampling procedure.

(6) *CRNN prediction / training*: The CRNN processes spectrogram segments individually to produce corresponding class probability vectors $f^{\text{crnn}}(\mathbf{S}_{i,j}) = \hat{\mathbf{p}}_{i,j} \in [0, 1]^{T_s}$, i.e. $\hat{\mathbf{p}}_{i,j}[t]$ indicates the probability of the target class being present in frame t of segment j for signal i . At training time, predictions are used for iterative optimization of network weights. Section 2.4 provides details on the CRNN configuration. We experimented with various loss functions as described in section 2.5.

(7) *output concatenation*: The stage concatenates the prediction segments to produce one target vector per input spectrogram $[\hat{\mathbf{p}}_{i,0}, \hat{\mathbf{p}}_{i,1}, \dots] = \hat{\mathbf{p}}_i \in \mathbb{R}^{T_i}$.

(8) *output binarization*: This stage binarizes predicted probabilities through thresholding: $\hat{\mathbf{y}}_i = 1$, if $\mathbf{p}_i > C$, else 0. We used the unbiased threshold $C = 0.5$ as in [20].

2.3. Spectrogram denoising

Spectrogram denoising is a common preprocessing step in automatic animal call detection. The aim is to increase the signal-to-noise-ratio, as recordings usually carry high amounts of noise due to the nature of open field settings. In this context, the signal is the target call of interest. Noise refers to *background sounds*, i.e. geophony (environmental sounds such as wind and rain), anthrophony (noise generated by humans, such as traffic) and biophony (sounds of animals not of interest). In the context of this work, we employed spectrogram denoising as a method for combating absolute rarity of target classes. Target event examples are present for only few background noise conditions, possibly causing the classifier to infer false coupling

between noise conditions and event probability. Prior elimination of variability between noise conditions can mitigate this effect. [15, 28, 29]

Among the multitude of available methods, we chose to evaluate: (1) frequency removal and (2) spectral subtraction, as they are among the most prevalent and straightforward methods in automatic animal call detection [15, 28, 29]. Both methods exploit the observation that background noise is fairly consistent over large periods of time, while target classes are comparatively short and seldom.

2.3.1. Frequency removal

Animal target calls usually occupy narrow frequency ranges. Therefore, many systems apply preprocessing filters to remove unneeded frequency ranges [18]. We calculated frequency ranges for target classes through the following proposed method:

The goal is to calculate a *class mask* $\mathbf{r} \in \mathbb{R}^F$, whose values $\mathbf{r}[f]$ indicate the strength of association between mel frequency bins f and the target class. let $T_e^{\text{start}} \in \mathbb{N}$ and $T_e^{\text{end}} \in \mathbb{N}$ be the start and end time of event $e \in \{0, \dots, E-1\}$ indicated as spectrogram time frame indices. $\mathbf{S}_e \in \mathbb{R}^{T \times F}$ and \mathbf{y}_e are the spectrogram and ground truth vectors which contain event e at some points. $t_e \in [T_e^{\text{start}} - T^c, \dots, T_e^{\text{end}} + T^c]$ is the list of time frames for event e , padded by a fixed context size T^c . Then, $\mathbf{S}_e^{\text{context}}$ and $\mathbf{y}_e^{\text{context}}$ are the spectrogram and ground truth vector patches corresponding to t_e , i.e. $\mathbf{S}_e^{\text{context}} = \{\mathbf{S}_e[t, f] \mid t \in t_e\}$. For each event, we obtain an *event mask* $\mathbf{r}_e \in \mathbb{R}^F$ by calculating the Pearson correlation between the mel frequency bin f and the target vector as:

$$\mathbf{r}_e[f] = \frac{\text{cov}(\mathbf{S}_e^{\text{context}}[f], \mathbf{y}_e^{\text{context}}[f])}{\sigma(\mathbf{S}_e^{\text{context}}[f]), \sigma(\mathbf{y}_e^{\text{context}}[f])} \quad (1)$$

The total class mask \mathbf{r} is the average of all event masks:

$$\mathbf{r}[f] = \frac{1}{E} \sum_{e=0}^{E-1} \mathbf{r}_e[f] \quad (2)$$

The intuition behind this calculation method is as follows: Any call event causes an increase in energy of its associated frequency bands relative to the background noise. If an event is surrounded by time invariant noise, this gain is measurable as a positive correlation between a frequency band's energy and the target vector. The class mask value range is $-1 \leq \mathbf{r}[f] \leq +1$, where $+1/-1$ indicates perfect association of a frequency band to the target vector and 0 indicates no association. Positive correlations > 0 are attributed to the actual target class acoustic content, i.e. energies active during target events. Negative correlations < 0 indicate systematic absence of band energies during events, which might be caused by other sounds commonly preceding, succeeding or pausing during target events. Our approach is applicable for calculating frequency ranges of arbitrary sound classes, if the mentioned preconditions are met.

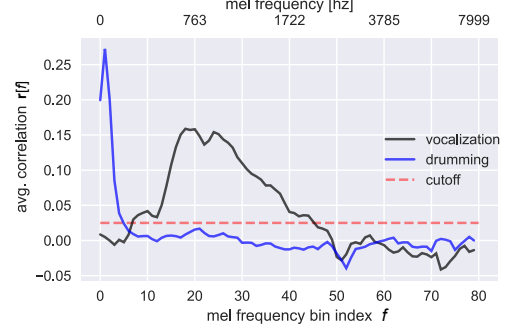


Figure 4: **Class masks.** Graphs indicate the strength of association between mel frequency bands and target classes. $\mathbf{r}[f] = 0$ indicates no association and $\mathbf{r}[f] = +1/-1$ indicates perfect association. The red line indicates the threshold under which we removed frequency bands when applying frequency removal.

Figure 4 shows the class masks for both target classes. These masks were calculated exclusively on events from the training and validation set. The context length was $T^c \triangleq 1$ min which roughly corresponds to the length of the longest call event in our database. When applying frequency removal, we set a correlation threshold C^r to remove frequency bands with $\mathbf{r}[f] < C^r$. We chose $C^r = 0.025$ as a rather low threshold to ensure no significant target class information was lost. We exclusively considered positive correlations as only those were indicative of the actual target class' signal content. Negative correlations, particularly > 2500 Hz, were attributed to variations in the background noise conditions such as periodic insect calls which we aimed to eliminate in this preprocessing step.

This procedure retained the lowest 5 frequency bins for drumming (range 0 – 152 Hz), and 39 frequency bins for vocalization (range 267 – 2097 Hz). Those frequency ranges correspond to estimations of previous studies [3]. Figure 5 visualizes the effect of frequency removal.

2.3.2. Spectral subtraction

Spectral subtraction removes background noise by subtracting a noise profile from signals. If the noise profile is assumed invariant across time/recordings, the profile is commonly estimated by averaging each frequency bin across all time steps inside a background noise region [28]. However, we observed that noise conditions vary strongly between recordings due to time of day, season and ARU location, but are fairly consistent within recordings. Consequently, we estimated local noise profiles for recordings as follows:

A spectrogram \mathbf{S}_i is segmented into non-overlapping segments $\mathbf{S}_{i,j} \in \mathbb{R}^{T^{\text{sub}} \times F}$ of length T^{sub} similar to the segmentation step in the processing pipeline. From each segment we subtract each frequency bin's average value:

$$\mathbf{S}_{i,j}^{\text{denoised}}[t, f] = \mathbf{S}_{i,j}[t, f] - \frac{1}{T^{\text{sub}}} \sum_{t=0}^{T^{\text{sub}}} \mathbf{S}_{i,j}[t, f] \quad (3)$$

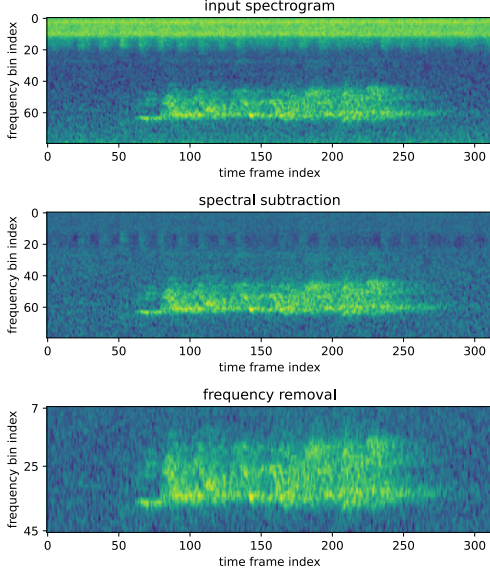


Figure 5: **Visualization of preprocessing operations for an example vocalization event.** Top: Input spectrogram. Middle: Effect of spectral subtraction (see sec. 2.3.2). Bottom: Effect of frequency removal (see sec. 2.3.1)

If T^{sub} is sufficiently large compared to the expected target event duration, the average frequency band energy is dominated by the noise profile rather than the target class profile. Denoised spectrogram segments are then concatenated to reconstruct the input spectrogram progression. This method is similar to the one applied in study [15]. Here we set $T^{\text{sub}} \triangleq 3$ min, so that even the maximum expected call duration contains twice as much noise as the target signal. Figure 5 visualizes the effect of spectral subtraction.

2.4. Convolutional Recurrent Neural Network

Figure 6 visualizes the network architecture scheme. The scheme is largely adapted from Cakir et al. [20]. It contains a sequence of network stages which process a spectrogram $\mathbf{S}_{i,j} \in \mathbb{R}^{T^{\text{seg}} \times F^{\text{den}}}$ to produce an output prediction vector $\hat{\mathbf{p}}_{i,j} \in [0, 1]^{T^{\text{seg}}}$. For simplicity, we denote these elements as $\mathbf{S} \in \mathbb{R}^{T \times F}$ and $\hat{\mathbf{p}} \in \mathbb{R}^T$ in this section. A central network property is that the temporal dimension T remains intact throughout the network, i.e. the temporal alignment between spectrogram frames and output time steps is maintained.

2.4.1. Architecture Scheme

The *input stage* appends an empty channel dimension to the spectrogram in preparation for the convolutional stage $\mathbb{R}^{T \times F} \mapsto \mathbb{R}^{T \times F \times (C=1)}$,

The *convolutional stage* consists of alternating convolutional and pooling layers similar to conventional CNN architectures [30]. Convolutional layers extract local features via learned filter kernels. They retain the size of the frequency and time dimension of their respective inputs through padding, but alter the channel dimension

according to the number of filter kernels k , i.e. $\text{conv}^k : \mathbb{R}^{T \times F \times C} \mapsto \mathbb{R}^{T \times F \times C' = k}$. Pooling layers reduce the frequency axis size through one-dimensional pooling according to the pooling stride p , i.e. $\text{pool}^p : \mathbb{R}^{T \times F \times C} \mapsto \mathbb{R}^{T \times (F' = F/p) \times C}$. The stage outputs a volume $\in \mathbb{R}^{T \times F^{\text{conv}} \times C^{\text{conv}}}$, where F^{conv} is the frequency size remaining after several pooling operations and C^{conv} is the number of filter kernels in the last convolutional layer.

The *frequency integration stage* removes the frequency dimension by incorporating frequency into channel information: $\mathbb{R}^{T \times F^{\text{conv}} \times C^{\text{conv}}} \mapsto \mathbb{R}^{T \times C^{\text{freqint}}}$. This is done via a time-distributed function, i.e. a function which is applied equally at each time step t .

The *recurrent stage* contains a recurrent layer which processes the output of the frequency integration stage sequentially at every time step. The output of the recurrent layer at time step t depends on the input of the preceding layer at the same time step t as well as the hidden states of the recurrent layer at steps $0, \dots, t-1$. The volume dimensionality transformation is $\mathbb{R}^{T \times C^{\text{freqint}}} \mapsto \mathbb{R}^{T \times C^{\text{rec}}}$.

Finally, the *output layer* consists of a time-distributed fully-connected layer $\mathbb{R}^{T \times C^{\text{rec}}} \mapsto \mathbb{R}^T$, i.e. a single fully-connected neuron which is applied with the same weights at each time step. We initialized the bias parameter of this neuron to the respective training class distribution $\log_e(\text{pos}/\text{neg})$, where *pos* and *neg* are the amount of positive and negative time steps respectively. This way, networks start training with appropriate output distributions which can prevent instability in the initial training steps, as recommended by Lin et al. [31]

The central difference between Cakir’s scheme and ours is the definition of the frequency integration stage as a generic stage, while Cakir defined a single fixed integration operation for flattening the frequency axis. The reason for this change is that we identified the frequency integration operation as one of the most important architecture hyper parameters in our previous studies [32, 33]. We implemented networks and training with `tensorflow v.2.3.1`.

2.4.2. Network Configuration

The following network parameters were fixed. For the convolutional stage, all convolutional layers used kernel sizes (5, 5) and strides of (1, 1). Convolutional layers were always followed by a batch-normalization layer and ReLU activation. Pooling layers always used max-pooling. The recurrent stage contained a single recurrent layer with gated recurrent units (GRU). Networks were trained with Adam optimizer with standard parameters [34] and early-stopping based on the validation set loss. All of these parameters were chosen in accordance to the recommendations of Cakir et al. [20].

The following network parameters were subjected to a parameter search as part of the experimental setup in this study (see section 2.8). The *channel size* $\in \mathbb{N}$ is the number of filters in each convolutional or recurrent layer, i.e. the number of filters/units is held constant across the

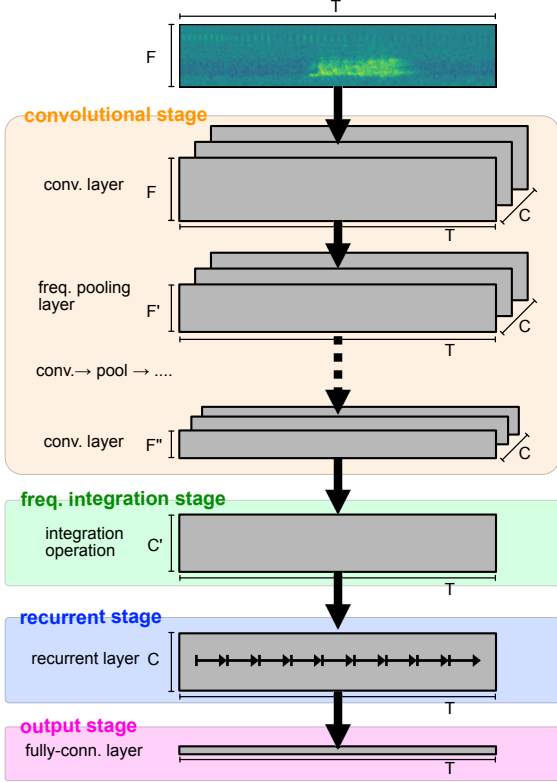


Figure 6: **Convolutional recurrent neural network architecture scheme.**

entire network. The *conv. stage depth* $\in \mathbb{N}$ determines the number of convolutional layers. We chose to search these parameters as they primarily determine the amount of network weights and consequently the capacity to fit the data [22]. The *pooling size* $\in \mathbb{N}$ determines the pooling size and stride for all pooling layers. The *frequency integration operation* $\in \mathbb{R}^{F \times C} \mapsto \mathbb{R}^{C'}$ is the time-distributed operation for integration of the frequency dimension. We searched for these parameters as we found them to be of primary importance for generalization capability in prior studies [32, 33]. Finally, *recurrent bi-directional* $\in \mathbb{B}$ determined whether the recurrent layer was used bi-directionally. If yes, half of the units run forward and half backwards. We searched this parameter as we hypothesized it to be of primary importance for the network’s capability to precisely locate frames of event activity.

Table 2 summarizes the search space. As explained further in section 2.8, we investigated network configuration in combination with the spectrogram denoising stage setting. All search spaces were equal regardless of input class and denoising setting, except for drumming when frequency removal was applied, as this removed 93% of the input frequency axis size.

2.5. Loss

The choice of loss function is among the primary algorithm-level methods for mitigating class imbalance in neural networks [19]. The standard loss for classification problems

Table 2: **Search space for network configuration and spectrogram denoising.** Table header shows input class, the top half shows denoising setting, and the bottom half shows the search space for network configuration parameters. The symbol \leftarrow indicates that a search space corresponds to the respective left cell. Abbreviations: GAP = 1d global average pooling, GMP = 1d global max pooling

input class	vocal		drumming	
freq. removal	no	yes	no	yes
input size $T \times F$	500×80	500×39	500×80	500×5
spectrl. subtr.	{no, yes}	{no, yes}	{no, yes}	{no, yes}
# units / layer	{32, 64, 96}	\leftarrow	\leftarrow	\leftarrow
# conv. layers	{2, 3, 4}	\leftarrow	\leftarrow	{1, 2}
pool size & stride	{2, 3, 4, 5}	\leftarrow	\leftarrow	{2, 3}
freq. integr. op.	{flatten, GAP, GMP}	\leftarrow	\leftarrow	\leftarrow

with output neurons with sigmoid activations is binary cross entropy [20, 22]:

$$BCE(\hat{p}, y) = \begin{cases} -w_1 \log(\hat{p}) & , \text{ if } y = 1 \\ -w_0 \log(1 - \hat{p}) & , \text{ else} \end{cases} \quad (4)$$

where w_1, w_0 are optional weights for the background and the target class. In standard binary cross entropy both classes are weighted equally, i.e. $w_1, w_0 = 1$. Binary cross entropy was the default loss used for network optimization. As part of the experimental setup (see section 2.8), we additionally evaluated a selection of the most prevalent BCE variants [19] aimed at mitigating class imbalance. These were:

- **weighted binary cross entropy** weighs classes according to the relative count of examples: $w_k = |C_k| / (|K| \cdot |C|)$, where $k \in K = \{0, 1\}$ is the class index, C_k is the set of examples for class k , and C is the total set of examples. This calculation method originates from King et al. [35], as implemented with `scikit-learn v.0.23`.
- **focal loss** down-weighs the loss of well-classified examples irregardless of the class trough: $w_0 = (\hat{y})^\gamma$ and $w_1 = (1 - \hat{y})^\gamma$. The hyper parameter γ determines the amount of down-weighting and is set to a default value $\gamma = 2$ in the paper [31]. We used the implementation of `tensorflow-addons 0.11.2`.
- **weighted focal loss** is the combination of focal loss with class weights as previously described.

2.6. Resampling

Resampling is the most prevalent data-level technique for mitigating relative class imbalance. Resampling seeks to rebalance the distribution between class examples by undersampling, i.e. discarding examples from the majority class, and oversampling, i.e. duplicating examples from

Table 3: Search space for loss and resampling

stage	component	default	search space
training	loss	BCE	{weighted BCE, FL, weighted FL}
	oversampling dupl. amount	2	{0, 2, 4, 8, 16}
resampling	undersampling disc. percentage	75%	{0%, 50% 75%, 90%, 95%}

the majority class. Both techniques have been successfully applied for deep-learning based systems in imbalanced settings [19]. Among the set of available resampling techniques, we chose to experiment with the most prevalent and straight-forward implementations: random over and undersampling.

We implemented resampling as follows. After the segmentation step, each ARU recording was represented as a list of spectrograms $\mathbf{S}_{i,j}$ and target vectors $\mathbf{y}_{i,j}$ where i is the signal and j is the segment index. We separated each list into disjoint subsets, those containing "negative" segments, i.e. segments with exclusively background time steps $Neg_i = \{j \mid \max(\mathbf{y}_{i,j}) = 0\}$, and those containing "positive" segments, i.e. segments with at least one time step with a target call $Pos_i = \{j \mid \max(\mathbf{y}_{i,j}) = 1\}$.

The strength of undersampling is determined through a parameter $U \in [0, \dots, 1]$ which indicates the percentage of discarded negative examples. The strength of oversampling is determined through a parameter $O \in \mathbb{N}$ which indicates the number of duplications of all positive examples, e.g. $O = 1$ means that each positive example is duplicated once. Undersampling and oversampling was performed for each input signal separately. Using this approach, we ensured that the set of background examples displayed a certain degree of diversity even for higher discarding percentages, as noise had greater variance between than within signals.

Table 3 shows the default settings and search space for the resampling stage. As shown, the default setting applies some amount of over- and undersampling, while not completely balancing distributions. The reasons were (a) decreasing training time for the initial experiments by reducing the data set size, (b) ensuring that the extreme imbalance in the training set does not prevent training convergence [19], and (c) to imitate the default setting commonly used in animal call detection systems, which usually start with already undersampled databases [10, 12, 14].

2.7. Evaluation

We chose the following metrics for evaluation of system performance: (1) average precision (AP), also known as the "area under the precision-recall-curve". This metric is based on the un-binarized prediction probabilities, i.e. comparing $\hat{\mathbf{p}}$ and \mathbf{y} after the concatenation step described in section 2.2 (predictions / ground truth vectors for all

signals were concatenated). (2) $F1$, calculated on the binarized prediction probabilities $\hat{\mathbf{y}}$ and \mathbf{y} . Both metrics are recommended for the evaluation of imbalanced class distributions [12, 36]. All metrics were implemented with `scikit-learn v.0.23`

We chose the *segment based* approach as the evaluation method, i.e. metrics were based on comparing fixed-length time intervals as evaluation instances [37]. We chose the following temporal resolutions:

- Frame-wise resolution: Evaluation segments corresponded directly to spectrogram frame indications $\hat{\mathbf{p}}$ or $\hat{\mathbf{y}}$ and \mathbf{y} . Consequently, the evaluation segment length was 20 ms. This resolution measures the system capacity for precise localization of events. It implicitly weighs target events according to the amount of time frames, i.e. longer events influence metric scores more than short events. This resolution was noted through the subscript AP_{frm} and $F1_{\text{frm}}$.
- 5 second resolution: Ground truth and prediction vectors were down-sampled to 5 s intervals as evaluation instances. The down-sampling was performed through max-pooling in non-overlapping segments of 5 s length. This resolution measures the system capacity for coarse localization of events. It also compensates the effect of event importance being weighted by length, as all events with length < 5 s contribute the same amount of evaluation segments. However, it also over-penalizes short false-positive-peaks. This resolution was noted through the AP_5 and $F1_5$.
- Averaged resolution: The average of both resolutions, e.g. $F1_{\text{avg}} = (F1_{\text{frm}} + F1_5)/2$

We used AP_{avg} as the primary evaluation metric for the following reasons. (1) It is independent of a binarization threshold, which imposes another hyperparameter which might require tuning. (2) End users for this application prefer unbinarized predictions as being more informative and (3) It summarizes the system capacity for precise and coarse event localization. We indicated all metrics in the range $[0, \dots, 1]$, where 0 is the worst and 1 is a perfect score.

2.8. Experimental Setup

The central goal of the experimentation was to evaluate the influence of pipeline stages for mitigating class rarity. Tables 2 and 3 summarize the stage's search spaces and default settings. However, the search space was too large to exhaustively evaluate via a full-factorial design, i.e. evaluating each possible combination of parameters ("grid search"). Therefore, we split the experiment into two rounds to investigate local combinations of stages full-factorially:

Round 1: Optimization of network architecture + spectrogram denoising. For each setting of the denoising stage (no preprocessing / freq. removal / spec.

sub. / freq. removal + spec. sub.) we performed a full architecture grid search according to the search space shown in Table 2. The goal was to identify the optimal combination of denoising setting and network architecture configuration. As this round performed hyper-parameter selection, performance was measured on the validation set. Resampling and loss-stages used their default values as shown in Table 3. The reason for optimizing these components first were: (a) the network architecture is the central and only obligatory part of the pipeline and thus of primary importance for optimization, and (b) we hypothesized interaction effects between denoising and network architecture parametrization, since the denoising setting altered the dimensionality and nature of the input features.

Round 2: Optimization of loss type + resampling. For each loss variant, we investigated each possible combination of over & undersampling according to Table 3. Network architecture and spectrogram denoising were fixed to the optimal configuration found in round 1. We investigated these stages in combination, since we suspected interaction effects, e.g. loss variants specialized in balancing stages might prefer less oversampling. For this round we used the reduced test set for evaluation, as it more accurately mirrors the real class distribution than the validation set.

We performed all experiments separately for each class. To reduce performance variability due to random model initializations and dataset shuffling, we repeated each network training and evaluation 5 times and averaged performance results.

3. Results

3.1. Optimization of network architecture + spectrogram denoising

Figure 7 shows the validation set performances achieved by networks in the architecture grid search, grouped by denoising operations. To statistically assess the influence of the investigated parameters, we constructed conditional inference trees (c-trees), shown in Figure 10. C-trees are regression trees where the splitting criterion is the statistical significance (p-value). At each node, the tree splits instances into two subgroups based on a singular feature, choosing the split with the highest significance, until no significant split can be made. The advantage of c-trees for statistical analysis are that (1) they indicate hyper-parameter importance, as the globally most important hyper-parameters occur higher to the root, and (2) they implicitly account for interaction effects, i.e. hyper-parameter effects only occurring in subgroups. We fit regression trees on the validation performance AP_{avg} with all denoising and architectural hyper-parameters as predictors. The p-value-cutoff was 0.001 with Bonferroni-correction to limit trees to the most essential effects. C-trees were implemented with R-package `party` v 1.4-5 [38].

We highlight the following observations based on these visualizations:

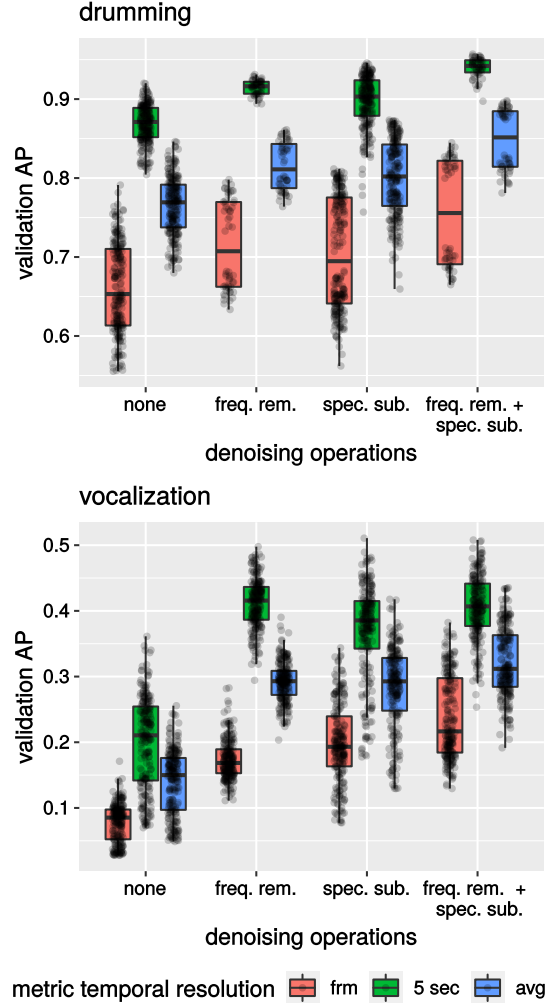


Figure 7: **Performances of network configurations, grouped by denoising operations.** The x-axis indicates spectrogram denoising operations, the y-axis indicates validation performance. Each data point presents the performance of a network configuration produced in the architecture grid search, averaged over 5 runs.

(1) Detection performance was drastically higher for drumming than for vocalization, regardless of network architecture and denoising setting (drumming AP_{avg} range: .55 - .95, vocalization: .03 - .50). Drumming performances reached up to almost perfect scores, while the worst drumming performance was still higher than the best vocalization performance.

(2) AP_5 values were generally higher than AP_{frm} values for both target classes. This means that networks were better at detecting rough localization of target events than frame-precise localization (see Fig. 7).

(3) Both spectrogram denoising operations increased performance AP_{avg} on average (i.e. over all network configurations constructed in the grid search). Using no denoising operation yielded significantly ($p < 0.0001$) lower performance than using either or both operations in combination. Using both operations simultaneously yielded a significantly higher ($p < 0.0001$) performance than using ei-

Table 4: **Network architectures with highest validation performance per denoising setting.** The underlined models were selected for subsequent experiments regarding loss functions and resampling. "Complete test performance" means that the complete test set was used for testing (and not the reduced test set, see Table 1). Bold numbers highlight the highest performance reached for each class.

class	denoising setting			architectural hyperparameters				val performance			complete test performance		
	freq. rem.	spec. subtr.	conv. stage depth	pooling size	channel size	freq. int.	bi-direct.	AP_{frm}	$AP_{5\text{sec}}$	AP_{avg}	AP_{frm}	$AP_{5\text{sec}}$	AP_{avg}
drum.	False	False	4	2	96	GAP	True	0.791	0.901	0.846	0.025	0.059	0.042
	True	False	2	2	32	GMP	True	0.794	0.928	0.861	0.084	0.214	0.149
	False	True	4	2	96	GAP	True	0.808	0.939	0.873	0.044	0.079	0.061
	<u>True</u>	<u>True</u>	<u>2</u>	<u>2</u>	<u>96</u>	<u>GAP</u>	<u>True</u>	0.839	0.957	0.898	0.162	0.294	0.228
voc.	False	False	3	3	64	GAP	True	0.171	0.34	0.256	0.008	0.013	0.01
	True	False	2	3	96	GAP	True	0.283	0.498	0.39	0.015	0.02	0.018
	False	True	3	3	32	GMP	True	0.325	0.511	0.418	0.011	0.014	0.013
	<u>True</u>	<u>True</u>	<u>2</u>	<u>4</u>	<u>96</u>	<u>GAP</u>	<u>True</u>	0.382	0.488	0.435	0.016	0.02	0.018

ther or none. The performance difference between using either operation individually was not significant ($p > 0.01$).

(4) Both spectrogram denoising operations increased performance regarding the maximum AP_{avg} reached by a model constructed in the architecture grid search. The order was no preprocessing \mapsto freq. rem. \mapsto spec. sub \mapsto freq. rem. + spec. sub for both classes.

(5) The usage of a bi-directional recurrent layer was the most important network architecture property, i.e. it increased performance for both classes with most denoising operations. For vocalization, they only increased AP_{frm} and not AP_5 , i.e. they only increased the capability for precise localization of target events while rough allocation remained the same. The two clusters in Fig. 7 for drumming with spec. sub. + freq. rem. are explained through bi-directional layers.

(6) The influence of network architecture choices decreased when applying integration operations. This is evidenced by the fact that both c-trees (Fig. 10) have the largest depths in the paths without denoising operations. The network features with the most influence were the depth and the choice of frequency integration operation. However, which depth and integration operation optimized performance was dependent upon the denoising setting. On average, performance increased with depth and global average pooling for frequency integration.

Table 4 shows architectures with the highest validation performance AP_{avg} per denoising setting and their test set performances. We highlight the following observations:

(1) Test performances dropped drastically compared to the validation set. This is largely due to the test set containing far more negative examples than the validation set (see Table 1).

(2) While frequency removal and spectral subtraction performed similarly on the validation set, frequency removal outperformed spectral subtraction on the test set. Combining both operations yielded the highest test performance for drumming, and performed on par with using frequency removal alone for vocalization.

(3) For drumming, the importance of denoising functions increased on the test set. The performance difference between using no and both operations was .05 on the validation set, but .18 on the test set. For vocalization, denoising importance decreased on the test set. The performance difference between using none and both was .28 on the validation set, but only .08 on the test set.

For the experiments on loss function + resampling in section 3.2 we used the setting with the highest validation performance AP_{avg} for each class, i.e. the underlined models in Table 4.

3.2. Optimization of Loss Variant + Resampling

Figure 8 shows the results of the experiments on the loss variant and resampling, measured on the reduced test set as AP_{avg} . Additionally, we calculated the total ratio of positive to negative segments resulting from the over / undersampling settings (positive segment = segment with at least one positive frame, see sec. 2.6). Figure 9 shows the corresponding c-tree analysis analogous to the one of section 3.1 (target: reduced test set AP_{avg} , predictors: loss variant and resampling parameters + pos/neg ratio).

Loss and resampling had greater influence on drumming than vocalization, i.e. performance range was .2 - .5 for drumming and .05 - .09 AP_{avg} for vocalization. For drumming:

(1) Both unweighted loss functions reached higher performances than weighted functions. The global effect of weighted vs unweighted functions was significant ($p < 0.001$). The global difference between unweighted cross entropy and focal loss was not significant. However, standard cross entropy had significantly higher performance than focal loss ($p < 0.01$) when undersampling ≤ 0.5 .

(2) Performance decreased globally with increased undersampling.

(3) For unweighted loss functions there was a significant global association for increased performance with a decreased ratio of positive/negative segments ($p < 0.001$). This also means that performance decreased with increased oversampling.

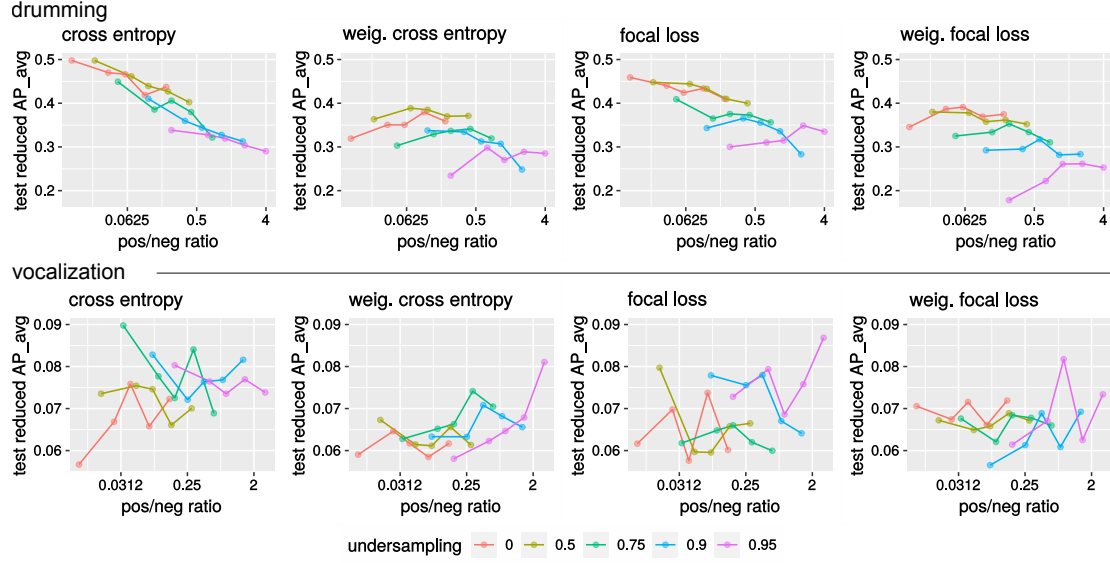


Figure 8: **Influence of resampling grouped by loss functions** Upper row: drumming, lower row: vocalization. The x-axes indicate the the ratio of positive to negative segments. Oversampling is shown implicitly, where 5 data points per undersampling setting correspond to oversampling duplication amounts $\{0, 2, 4, 8, 16\}$.

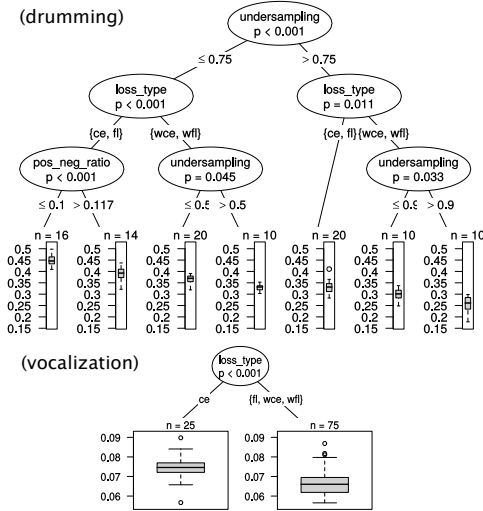


Figure 9: **Conditional inference trees for loss type and resampling**

(4) The highest performance $AP_{avg} = .49$ was reached with binary cross entropy using the “raw” training database without any resampling.

For vocalization, the only significant influence was standard binary cross entropy performing better than the other loss functions, although the influence was still low in absolute terms. Resampling had no systematic influence. The highest value $AP_{avg} = .09$ was reached with no oversampling and undersampling of 0.75.

Table 5 shows the final performance evaluation on the complete test set for the best hyper-parameter settings found in this study. The denoising settings and network architecture setting correspond to the settings underlined in Table 4. The loss type and resampling correspond to

the settings with the highest performance found on the reduced test set (see figure 8). AP_{avg} values dropped by approximately 60 % from the reduced to the complete test set. This is due to the sevenfold increase of negative examples compared to the complete test set, increasing the amount of possible false positive predictions.

The baseline performance corresponds to the performance achieved by Heinicke et al. [3]. Their $F1$ values were computed on event-based metrics with varying event lengths based on the segmentation algorithm in their study, i.e. they are not directly comparable to our segment-based metrics. When comparing the baseline values to $F1_{avg}$ -values, our performances were an improvement, increasing baseline performance. For drumming the increase was 30% $F1$, which is a 7-fold increase. For vocalization, the increase was 5% $F1$, which is a 25-fold increase.

4. Discussion

The $F1$ scores of 33 % for drumming and 5 % for vocalization might seem rather low in absolute terms. However, one has to take into account that even for humans the problem is exceptionally difficult. In addition to the rarity of the target calls, they are also very faint and subtle. This issue requires extensive training by human listeners to label calls reliably. Hence, although our results leave room for further improvement, they represent an improvement compared to previous methods.

The detection performance for chimpanzee drumming was drastically higher than for vocalization, with and without denoising/resampling. The same effect occurred in Heinicke’s [3] method, although absolute values were lower. We hypothesize that the following factors contribute to the

Table 5: **Final performane evaluation**

class	loss	resampling		pos/neg ratio train set		complete test set performance						
		under-samp.	over-samp.	frm	seg	AP_{frm}	AP_5	AP_{avg}	$F1_{\text{frm}}$	$F1_5$	$F1_{\text{avg}}$	base-line $F1$ [3]
drum.	cross entr.	0	0	0.002	0.012	0.308	0.385	0.347	34 %	32.6 %	33.3 %	4.6 %
voc.	cross entr.	0.75	0	0.011	0.034	0.018	0.025	0.021	5 %	5.3 %	5.1 %	0.2 %

difficulty of detecting chimpanzee vocalizations: (1) Vocalizations are more complex with greater intra-class variability than drumming, as they encompass multiple call types with respect to pant hoots and screams. In comparison, drumming has a more "fixed" and stereotypical pattern. (2) The frequency bands for chimpanzee vocalization are also occupied by calls of other primate species which are acoustically similar. However, drumming is the only animal call occupying such low frequency bands in our data set. In our experience, human listeners also have greater difficulty in identifying chimpanzee vocalizations, particularly because they confuse them for other animal calls. Thus we reason that the vocalization class may have needed more training examples to be learned effectively by the network, given the greater difficulty of the task.

Both denoising operations, spectral subtraction and frequency removal, increased performance significantly, particularly for drumming's test set performance. This finding is in accordance with other studies on animal call detection which applied similar operations with success [15, 28, 29]. We were particularly surprised by the magnitude of increase in performance by frequency removal. Theoretically, networks should learn to ignore irrelevant frequency bands by themselves. We give two possible explanations for this: (1) Positive class examples were only present for few recordings. Possibly, this led the network to infer a false association between noise conditions and target call occurrence. Removing uncorrelated frequencies reduced the features which could be used for such false associations. (2) Possibly, the test set contained background noise conditions which were drastically different from the ones in the training set so that they occupy regions far away from the learned manifold and cause faulty forward-passes in the network, similar to adversarial examples [39]. We highlight that frequency removal carries the additional advantage of decreasing computation time.

We found that taking relative class balance did not increase performance. Drumming reached the highest performances using standard binary cross entropy loss without any data set resampling, i.e. using the raw, heavily imbalanced training data. Vocalization also performed best with vanilla cross entropy, but was insensitive to resampling. We draw the conclusion that performance-wise, combating relative class imbalance is unnecessary or even harmful. For drumming, undersampling decreased the per-

formance regardless of the loss function, i.e. displaying diversity of the background class is important even if examples might seem redundant for humans. Still, under-sampling can reduce training time with minimal loss in performance if used only slightly. Drumming reached essentially the same performance using 0 % and 50 % of the training data and began dropping when only using 25 %. This finding is in contrast to other studies [17–19, 31], which usually report positive effects for balancing methods. We give the following possible explanations for this discrepancy: (1) Studies reporting positive effects of resampling commonly worked with imbalanced training sets, but balanced test sets [40, 41]. Consequently, the positive effect of resampling could be attributed to approximating class distributions between training and test set, and not to compensating the imbalance within the training set. (2) Studies reporting positive effects of class weights in classification settings usually performed multi-class single-label classification with softmax activation as in multinomial logistic regression [35, 42, 43]. However, we used binary cross entropy for single-class prediction, which might be inherently more robust to class imbalances. (3) When Lin et al. reported focal loss to outperform binary cross entropy, they performed multi-label detection [31] for 91 classes with one network for the COCO dataset. As the influence of background class multiplies across all positive classes in multi-label detection, loss balancing might become beneficial in such multi-label settings.

In summary, our results show that supporting the network to learn decoupling target class characteristics from background class characteristics is of primary importance for increasing performance. Spectrogram denoising explicitly supports this decoupling by discarding information from signals which are assumed to only be associated with background noise based on prior knowledge. Including more examples from the background class (no under-sampling) implicitly supports this decoupling by displaying a greater amount of background noise variability to the network.

5. Conclusion and Future Work

In this paper we investigated the automatic detection of chimpanzee drumming and vocalizations in long-term forest recordings of PAM. The detection approach was

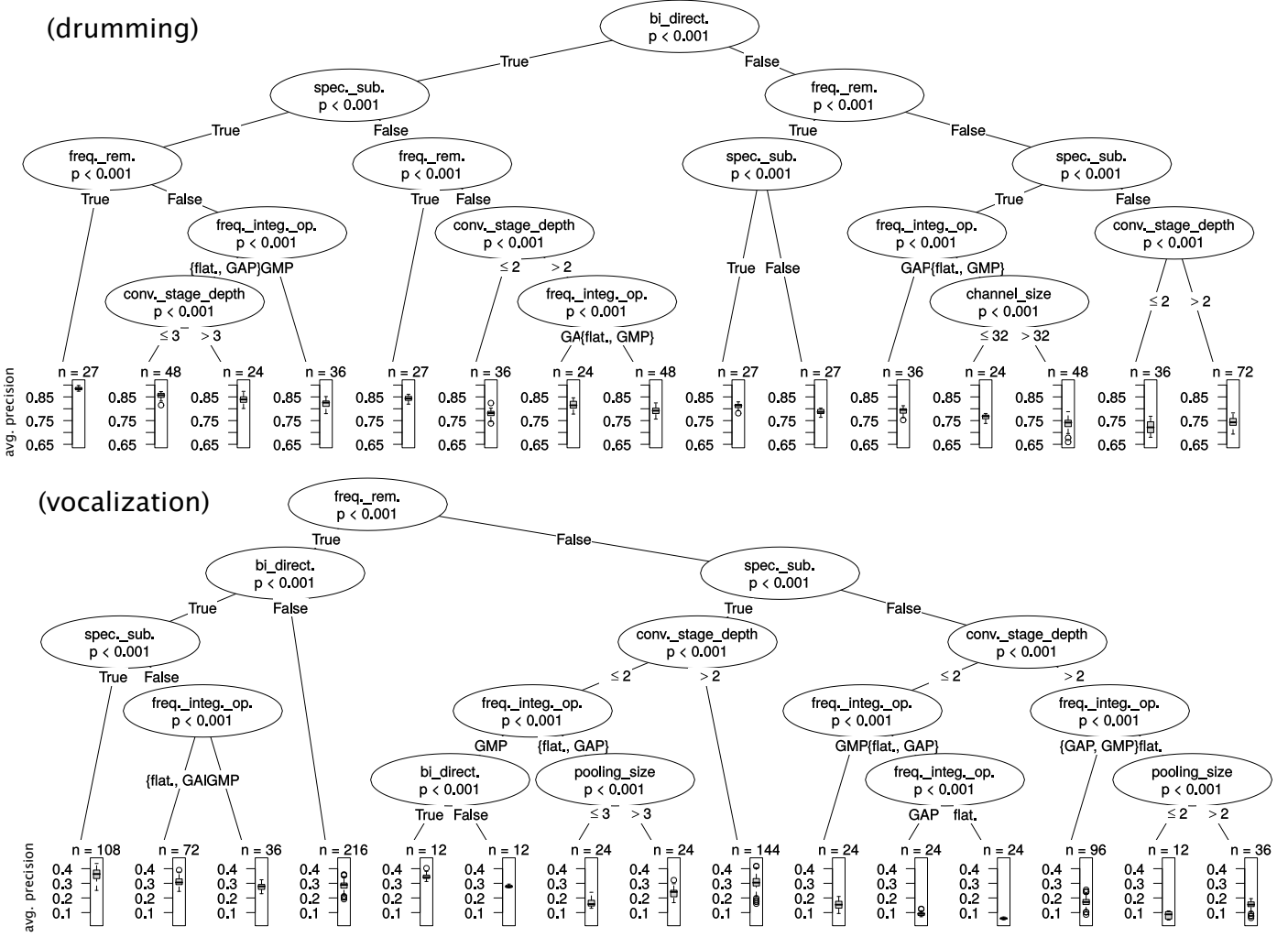


Figure 10: **Conditional inference trees for optimization of network architecture + denoising** Nodes show hyper-parameters with the p-value resulting from splitting the feature into groups indicated by the branches. Leafs show the performance distributions.

based on convolutional recurrent neural networks with spectrogram inputs. The particular challenge of this task was the severe class imbalance and rarity of target calls. We applied various extensions to the pipeline for compensating this imbalance: We evaluated two spectrogram denoising operations frequency removal (i.e. removing frequency bands outside the target call’s range) and spectral subtraction. Both operations significantly improved the performance. For mitigating relative class imbalance, we evaluated various loss functions (weighted / unweighted cross entropy / focal loss) as well as random over/undersampling of segments. The best performing loss was unweighted binary cross entropy. For drumming, any resampling decreased performance, i.e. training on heavily imbalanced training data reached the highest performance. For vocalization, resampling had no significant effect. From this we conclude that a primary factor for increasing performance in animal call detection in PAM settings is aiding the network to learn decoupling background noise conditions from target call characteristics. Final performance results were

an improvement on the previous baseline performance.

With an algorithm performance of about 30 % for the detection of chimpanzee drumming, this approach may become suitable for continuous field monitoring. As previously demonstrated, species occurrence [1] and movement patterns [2] can be modeled using PAM data. Such applications may be enhanced using the detection process proposed in this study.

We see the greatest chances for further improvement by investigating the following areas:

(1) Applying methods for increasing diversity in the existing positive examples. The most straight-forward method is data augmentation, i.e. applying perturbations to the positive calls to diversify their patterns. Particularly mix-up could prove successful to further teach decoupling of noise and positive examples [44], i.e. randomly overlapping positive segments with noise samples.

(2) Increasing the amount of positives examples. This could be done by applying the algorithm to the yet unlabeled data and collecting additional true positives. Alter-

natively, one could generate synthetic examples through generative adversarial networks (GANs).

(3) Methods for informed undersampling such as SMOTE. Multiple studies show that informed undersampling can outperform random undersampling [17–19]. Even if performance could not be further improved through undersampling, it may be possible to reduce training time by carefully selecting a representative subset of negative examples.

Acknowledgment

This work was funded by the research grant for doctoral researchers at Leipzig University of applied sciences, grant number 3100451136, as well as the European Union as part of the ESF-Program, grant number K-7531.20/434-11; SAB-Nr. 100316843. We thank the Ministère de la Recherche Scientifique, the Ministère de l'Environnement et des Eaux et Forêts and the Office Ivoirien des Parcs et Reserves of Côte d'Ivoire for permissions to work in the country and Taï National Park.

References

- [1] A. K. Kalan, R. Mundry, O. J. Wagner, S. Heinicke, C. Boesch, H. S. Köhl, Towards the automated detection and occupancy estimation of primates using passive acoustic monitoring, *Ecological Indicators* 54 (2015) 217–226.
- [2] A. K. Kalan, A. K. Piel, R. Mundry, R. M. Wittig, C. Boesch, H. S. Köhl, Passive acoustic monitoring reveals group ranging and territory use: a case study of wild chimpanzees (*pan troglodytes*), *Frontiers in zoology* 13 (1) (2016) 1–11.
- [3] S. Heinicke, A. K. Kalan, O. J. Wagner, R. Mundry, H. Lukashevich, H. S. Köhl, Assessing the performance of a semi-automated acoustic monitoring system for primates, *Methods in Ecology and Evolution* 6 (7) (2015) 753–763.
- [4] C. Dev, Automatic detection of chimpanzee vocalizations using convolutional neural networks.
- [5] J. Salamon, C. Jacoby, J. P. Bello, A dataset and taxonomy for urban sound research, in: *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 1041–1044.
- [6] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, M. D. Plumbley, Detection and classification of acoustic scenes and events: Outcome of the dcase 2016 challenge, *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26 (2) (2017) 379–393.
- [7] A. Mesaros, A. Diment, B. Elizalde, T. Heittola, E. Vincent, B. Raj, T. Virtanen, Sound event detection in the dcase 2017 challenge, *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27 (6) (2019) 992–1006.
- [8] A. Mesaros, T. Heittola, T. Virtanen, Acoustic scene classification: an overview of dcase 2017 challenge entries, in: *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, IEEE, 2018, pp. 411–415.
- [9] A. Politis, A. Mesaros, S. Adavanne, T. Heittola, T. Virtanen, Overview and evaluation of sound event localization and detection in dcase 2019, *arXiv preprint arXiv:2009.02792*.
- [10] C. Bergler, H. Schröter, R. X. Cheng, V. Barth, M. Weber, E. Nöth, H. Hofer, A. Maier, Orca-spot: An automatic killer whale sound detection toolkit using deep learning, *Scientific reports* 9 (1) (2019) 1–17.
- [11] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [12] J. Björck, B. H. Rappazzo, D. Chen, R. Bernstein, P. H. Wrege, C. P. Gomes, Automatic detection and compression for passive acoustic monitoring of the african forest elephant, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, 2019, pp. 476–484.
- [13] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.
- [14] T. Oikarinen, K. Srinivasan, O. Meisner, J. B. Hyman, S. Parmar, A. Fanucci-Kiss, R. Desimone, R. Landman, G. Feng, Deep convolutional network for animal sound classification and source attribution using dual audio recordings, *The Journal of the Acoustical Society of America* 145 (2) (2019) 654–662.
- [15] O. Mac Aodha, R. Gibb, K. E. Barlow, E. Browning, M. Firman, R. Freeman, B. Harder, L. Kinsey, G. R. Mead, S. E. Newson, et al., Bat detective—deep learning tools for bat acoustic signal detection, *PLoS computational biology* 14 (3) (2018) e1005995.
- [16] D. Stowell, M. D. Wood, H. Pamula, Y. Stylianou, H. Glotin, Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge, *Methods in Ecology and Evolution*.
- [17] G. M. Weiss, Mining with rarity: a unifying framework, *ACM Sigkdd Explorations Newsletter* 6 (1) (2004) 7–19.
- [18] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, G. Bing, Learning from class-imbalanced data: Review of methods and applications, *Expert Systems with Applications* 73 (2017) 220–239.
- [19] J. M. Johnson, T. M. Khoshgoftaar, Survey on deep learning with class imbalance, *Journal of Big Data* 6 (1) (2019) 27.
- [20] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, T. Virtanen, Convolutional recurrent neural networks for polyphonic sound event detection, *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25 (6) (2017) 1291–1303.
- [21] T. Virtanen, M. D. Plumbley, D. Ellis, Computational analysis of sound scenes and events, Springer, 2018.
- [22] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016, <http://www.deeplearningbook.org>.
- [23] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, N. Seliya, A survey on addressing high-class imbalance in big data, *Journal of Big Data* 5 (1) (2018) 42.
- [24] N. Japkowicz, The class imbalance problem: Significance and strategies, in: *Proc. of the Int'l Conf. on Artificial Intelligence*, Vol. 56, Citeseer, 2000.
- [25] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, et al., Cnn architectures for large-scale audio classification, in: *Acoustics, Speech and Signal Processing (ICASSP)*, 2017 IEEE International Conference on, IEEE, 2017, pp. 131–135.
- [26] A. Mesaros, T. Heittola, T. Virtanen, A multi-device dataset for urban acoustic scene classification, *arXiv preprint arXiv:1807.09840*.
- [27] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, O. Nieto, *librosa: Audio and music signal analysis in python*, in: *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.
- [28] J. Xie, J. G. Colonna, J. Zhang, Bioacoustic signal denoising: a review, *Artificial Intelligence Review* (2020) 1–23.
- [29] I. Himawan, M. Towsey, B. Law, P. Roe, Deep learning techniques for koala activity detection., in: *INTERSPEECH*, 2018, pp. 2107–2111.
- [30] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [31] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.
- [32] F. Anders, M. Hlawitschka, M. Fuchs, Automatic classification of infant vocalization sequences with convolutional neural networks, *Speech Communication*.

- [33] F. Anders, M. Hlawitschka, M. Fuchs, Comparison of artificial neural network types for infant vocalization classification, *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2020) 54–67.
- [34] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980*.
- [35] G. King, L. Zeng, Logistic regression in rare events data, *Political analysis* 9 (2) (2001) 137–163.
- [36] J. Davis, M. Goadrich, The relationship between precision-recall and roc curves, in: *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 233–240.
- [37] A. Mesaros, T. Heittola, T. Virtanen, Metrics for polyphonic sound event detection, *Applied Sciences* 6 (6) (2016) 162.
- [38] T. Hothorn, K. Hornik, A. Zeileis, ctree: Conditional inference trees, *The comprehensive R archive network* 8.
- [39] L. Smith, Y. Gal, Understanding measures of uncertainty for adversarial example detection, *arXiv preprint arXiv:1803.08533*.
- [40] M. Buda, A. Maki, M. A. Mazurowski, A systematic study of the class imbalance problem in convolutional neural networks, *Neural Networks* 106 (2018) 249–259.
- [41] P. Hensman, D. Masko, The impact of imbalanced training data for convolutional neural networks, *Degree Project in Computer Science*, KTH Royal Institute of Technology.
- [42] H. Wang, Z. Cui, Y. Chen, M. Avidan, A. B. Abdallah, A. Kronzer, Predicting hospital readmission via cost-sensitive deep learning, *IEEE/ACM transactions on computational biology and bioinformatics* 15 (6) (2018) 1968–1978.
- [43] R. Anand, K. G. Mehrotra, C. K. Mohan, S. Ranka, An improved algorithm for neural network classification of imbalanced training sets, *IEEE Transactions on Neural Networks* 4 (6) (1993) 962–969.
- [44] H. Zhang, M. Cisse, Y. N. Dauphin, D. Lopez-Paz, mixup: Beyond empirical risk minimization, *arXiv preprint arXiv:1710.09412*.