UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LUÍS AUGUSTO WEBER MERCADO

Detection of Infant Guinea pig Vocalizations using Machine Learning

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Engineering

Advisor: Prof. Dr. Mariana Recamonde Mendoza
Coadvisor: Prof. Dr. Marcelo de Oliveira Dietrich

Porto Alegre
November 2021

“Fui a floresta porque queria viver de verdade.
Eu queria viver profundamente,
sugar toda a essência da vida,
Fazer apodrecer tudo que não era vida
e não, quando eu morrer, descobrir que não vivi.“

— Henry David Thoreau

AGRADECIMENTOS

Gostaria de agradecer primeiramente a minha família – Marlene Elesbão Weber, Rosane Elesbão Prade, Luís Paulo Leopoldo Mercado, Nadja Magali Weber, Luís Augusto Leopoldo Mercado, Luisa Weber Mercado e Marília Weber Mercado – pelo carinho, apoio e inspiração, e principalente, pelo acolhimento nos momentos difíceis que surgiram ao longo desse período. Não pediria outra família se pudesse.

Agradeço principalmente aos meus grandes amigos, irmãos e colegas Matheus Tavares Frigo e Nicholas de Aquino Lau. Muito obrigado por toda essa trajetória no curso e por todo apoio, carinho e transcedentalidade que tivemos um pelo outro: sem vocês eu não estaria aqui. Amo vocês. Agradeço também à Aimée Sousa Calepso por me acompanhar durante grande parte do caminhos até aqui, pelo incentivo, apoio, revisões, parceria e conversas que tivemos/temos até hoje. Agradeço, também, a todes meus amigues que, embora não sejam citados aqui diretamente, fizeram parte dessa trajetória e compartilharam seu carinho comigo. Amo todes vocês.

À minha orientadora Mariana Recamonde Mendoza, por aceitar essa posição, e também por servir de inspiração e guia nesse momento turbulento. Obrigado pelas revisões desse trabalho, pela atenção ao que fiz e pela compreensão das minhas dificuldades. Muito obrigado.

Ao professor Eduardo Zimmer, por ter me indicado para o meu co-orientador, Marcelo de Olveira Dietrich. Sem essa indicação, esse TCC não existiria.

Aos companheiros do Dietrich Lab, pelo apoio e acolhimento, mesmo sem nem ao menos termos nos vistos pessoalmente. Em especial, agradeço ao professor Marcelo de Oliveira Dietrich por me incentivar a explorar habilidades que nem eu mesmo sabia que tinha. Agradeço pelo apoio, compreensão, e por trazer o melhor de mim a tona. Agradeço também ao professor Bruno Zatt por suas contribuições e parceria nesse projeto. Não obstante, agradeço ao Gustavo Madeira Santana pela paciência, mentoria e incentivo desde o início desse projeto. Estaria perdido sem ti.

ABSTRACT

One way to address guinea pigs (*Cavia porcellus*) internal state on a behaviour of interest (e.g., elicit mother retrieval when isolated) is through the quantitative and qualitative analysis of vocalizations emitted by them. Researchers usually annotate these vocalizations for further counting and characterization using visual cues (spectrograms) generated from experiment recordings. This process demands time, effort and attention from these individuals. This work proposes a Machine Learning model for annotating these vocalizations – a fundamental step in both of these analysis types – in raw waveforms with no further human effort by using previously annotated recordings as base knowledge. AudibleSincNet, the proposed model, overcomes the spectrogram-threshold based approach on the task of vocalization detection when trained using a manually annotated dataset with ~5.5 hours and evaluated on a separated test set with 1.5 hours of experiment recordings.

Keywords: Cavia porcellus. vocalizations. audio. raw waveforms. cnn. sinc.

Detecção de vocalizações de filhotes de porquinhos-da-índia usando Aprendizado de Máquina

RESUMO

Uma maneira de avaliar o estado interno de Porquinhos-da-Índia (*Cavias porcellus*) com respeito a um comportamento – como incentivar o resgate da mãe quando isolado – se dá pela análise quantitativa e qualitativa das vocalizações emitidas pelos mesmos. Essas vocalizações são comumente anotadas por pesquisadores para futura contagem e caracterização destas. Para isso, um espectrograma gerado a partir de gravações dos experimentos é interpretado, processo este que demanda tempo, esforço e atenção desses indivíduos. Esse trabalho propõe um modelo de Aprendizado de Máquina para a anotação automática dessas vocalizações – um processo fundamental para ambos os tipos de análise – em formas de onda brutas, usando gravações previamente anotadas, sem a necessidade de esforço destes pesquisadores. O modelo proposto, AudibleSincNet, supera o modelo baseado em análise de espectrograma na tarefa de detectar vocalizações quando treinado em um conjunto de dados com aproximadamente 5.5 horas e avaliado em um conjunto de dados separado para teste com 1.5 horas de gravações de experimentos.

Keywords: Cavia porcellus, vocalizações, audio, ondas puras, cnn, sinc.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AD	Analog-to-Digital
CNNs	Convolutional Neural Networks
DSP	Digital Signal Processing
FN	False Negatives
FP	False Positives
GKDE	Gaussian Kernel Density Estimation
ML	Machine Learning
PSD	Power Spectral Density
SR	Sampling Rate
STFT	Short-time Fourier Transform
TN	True Negatives
TP	True Positives
USV	Ultrasonic Vocalization

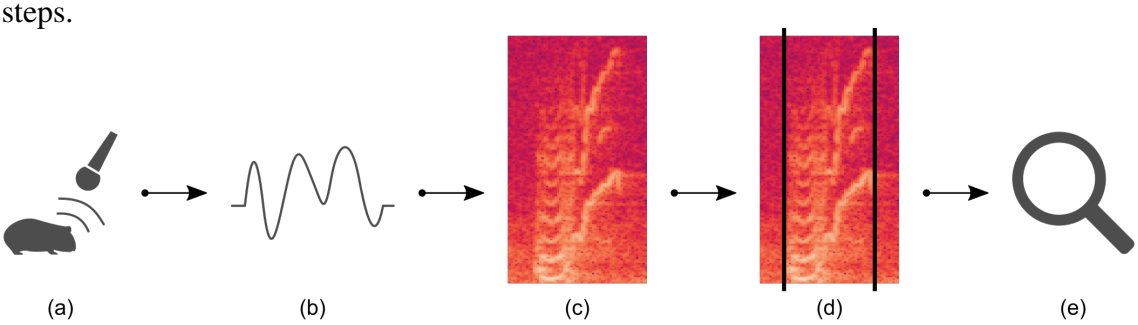
CONTENTS

1 INTRODUCTION

Guinea pigs are rodents that serve as a model to study mother-infant attachment (PORTER; BERRYMAN; FULLERTON, 1974; PORTER; FULLERTON; BERRYMAN, 1973). To draw attention and elicit mother retrieval, guinea pig infants emit vocalizations (HENNESSY, 2014; HENNESSY; RITCHEY, 1987; PETTIJOHN, 1979) when isolated from this caregiver. By showing high vocalization rates, these vocalizations are thought to be associated with distress. Vocal rates return to baseline levels upon reunion with the caregiver. Factors such as environmental temperature and food availability can affect vocalization rate. These vocalization characteristics can reliably quantify these factors consequences on the pup internal state (WEWERS; KAISER; SACHSER, 2003; RITCHEY; HENNESSY, 1987; HENNESSY et al., 1995).

It is necessary to annotate experimental recordings manually to quantify vocal behaviour. Fundamental for the quantitative and qualitative analysis of vocalizations, this process (Figure 1.1) requires excessive time, effort, and attention from researchers who manually annotate these vocals using visual cues on a spectrogram – which must be carefully tuned for optimal annotation. Besides the particular demand from a single researcher, the validation process also demands human resources. Several researchers should annotate these same vocalizations to minimize the dataset's inter-annotator error (i.e., researcher bias). Another should consider these different annotations when producing a final annotated dataset with according annotations.

Figure 1.1: An overview of the process of recording an animal and the analysis of this recording. (a) ilustrates the recording of the animal itself on an isolation paradigm; (b) represents the audio signal – raw waveform – resulting from this recording; (c) the spectrogram generation based on the audio signal and (d) is the annotation based on the generated spectrogram; (e) represents the analysis that is directly dependent of the previous steps.



Source: The Author

Therefore, aiming to propose a solution for this exhaustive and time-consuming step of manual vocalization annotation, this work proposes a Machine Learning model trained and evaluated on several hours of experiment recordings, named AudibleSincNet. AudibleSincNet overcomes the spectrogram-based approach (or baseline), which is state-of-the-art when considering mice Ultrasonic Vocalization (USV) detection.

Following this chapter, background and motivation chapters (Chapters 2 and 3, respectively) describes the main concepts necessary to understand this work, as well state-of-the-art techniques when considering vocalization detection. The 4th Chapter, *Methods*, explains the approach to train and evaluate both models, as well as how they are evaluated and further compared. Results, when trained using a manually annotated dataset with ~5.5 hours and evaluated on a separated test set with 1.5 hours of experiment recordings, are presented in Chapter 5. Discussion of these results and their implications are presented in Chapters 6 and 7, respectively.

2 BACKGROUND

This chapter presents the background concepts and definitions necessary to understand the adopted methods in this work. First, Digital Signal Processing concepts are discussed, including the Fourier Analysis and spectrogram representation. Secondly, the Machine Learning concepts such as Convolutional and Fully Connected Neural Networks. Further information can be found in the references.

2.1 Digital Signal Processing (DSP)

A signal is formally defined as a *function of one or more variables that conveys information on the nature of a physical phenomenon* (HAYKIN; VEEN, 2005). From human communication through speech to representations of daily fluctuations in the prices of stocks and commodities on world markets, signals are present in everyday life. These signals can be multidimensional analogue or digitally coded functions. Analogue signals are continuous, while digital signals are discrete by definition. Nonetheless, a continuous signal can have a discrete representation and vice versa. Functions that rely on a single variable are called one-dimensional, while multidimensional signals depend on two or more variables. Digital signals are most commonly a time-series representation of an analogue signal, a conversion done by an Analog-to-Digital (AD) component. Except for images, most of the relevant signals nowadays are one-dimensional with respect to time – the independent variable and a varying quantity, such as the amplitude in audio signals.

An essential characteristic of this conversion from analogue to digital is the *sampling rate* (SR), which defines how many samples of the original signal (i.e., physical information) are represented in a single second of audio. Hence, choosing a suitable sampling rate can result in a compressed version of the analogue signal at the cost of information integrity. However, the original signal can be theoretically reconstructed from its digital representation without loss only if this sampling is done at a frequency at least twice as high as the maximum frequency of the original signal (NYQUIST, 1928; SHANNON, 1949). Since analogue signals are often not precisely constrained to a specific bandwidth, these signals will not be perfectly reconstructed.

One example of a one-dimensional digital signal is audio, which consists of amplitude (usually in decibels [dB]) over time (in samples) signal. Audio, also referred to as raw waveform, can be synthetically generated by a computer or when recording a sub-

ject or environment using a microphone. This signal can have different formats when represented on a computer (e.g., WAV or MP3), where they differ on compression level, available sampling rates, and channel organisation (ZÖLZER, 2008).

2.1.1 Fourier Analysis

Signals can have alternative representations, where the independent variable defines the domain of that signal. A signal that varies with time is said to be in the *time-domain*. Similarly, a signal is represented with its independent variable set as the frequency is in the *frequency-domain*. Any given signal can be converted to another domain, given a transform function. Choosing the ideal representation for a signal is crucial, as different signal processing techniques can yield better results depending on the chosen domain.

Transforming a time-domain signal to its frequency-domain representation is a fundamental part of signal processing. Fourier analysis (FOURIER, 1878) is the basis for many of the signal processing techniques in use today. For a given signal $x(t)$ in the time-domain, where t is time, its frequency-domain representation $X(\omega)$ for a given frequency in radians per second ω by using the Fourier Transform is defined as:

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt \tag{2.1}$$

where j is the imaginary unity, defined as $j = \sqrt{-1}$. The Inverse Fourier Transform can be used to transform a signal in the frequency-domain back to the time-domain:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) e^{j\omega t} d\omega \tag{2.2}$$

Due to its many applications and widespread use in engineering and many scientific fields, modifications and improvements regarding computational effort and complexity of the Fourier Transform were motivated. One example of optimisation is the Short-time Fourier Transform, or STFT (ALLEN, 1977). The intuition behind the STFT algorithm is to break down a signal into smaller segments of equal length and compute the Fourier Transform individually for each segment. This approach also helps in dealing with non-stationary spectral features of a signal.

A discrete time-domain signal $x[n]$ can be transformed into its frequency-domain

representation with the STFT using the following equation:

$$X(n, w) = \sum_{-\infty}^{\infty} x[n] \cdot w[n - m] \cdot e^{-j\omega n} \tag{2.3}$$

where $x[n] \cdot w[n-m]$ is segment of the signal $x[n]$ at time n according to some windowing function.

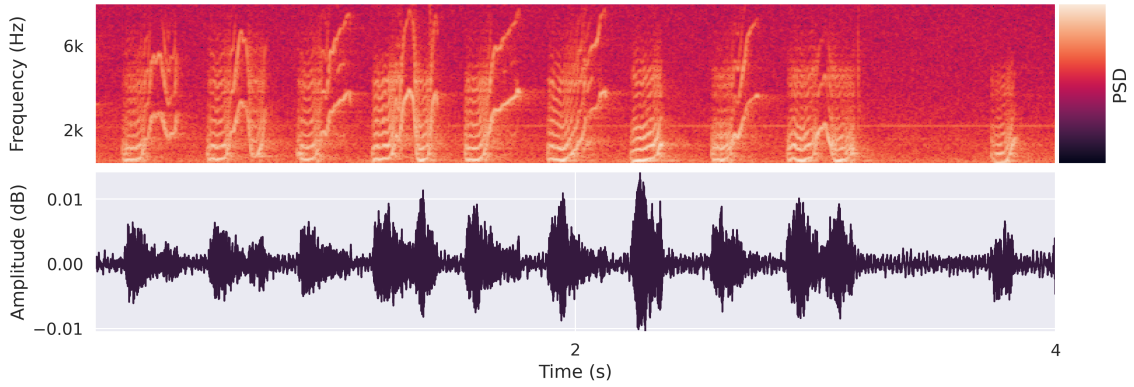
2.1.2 Spectrogram

A convenient way of visualising a signal in the frequency domain is through its spectrogram, a time $\times$ frequency $\times$ magnitude graphical representation. The x-axis represents the signal over time, the y-axis represents the frequency components at any given time, and the pixel colour represents the magnitude of the signal for that frequency component at that point in time, measured using the power spectral density. We can obtain the power spectral density (PSD) of a signal by computing the square of the magnitude of that signal. A log scale is usually applied to the PSD:

$$PSD(X) = \log |X(n, w)|^2 \tag{2.4}$$

Figure 2.1 shows an example of a spectrogram of a sequence of guinea pig vocalizations, followed by its raw audio waveform. Using a spectrogram representation, one can efficiently compute several signal properties that can not be obtained straightforwardly from raw waveforms.

Figure 2.1: Spectrogram of a fragment from a recording of guineapig vocalizations.



Source: The Author

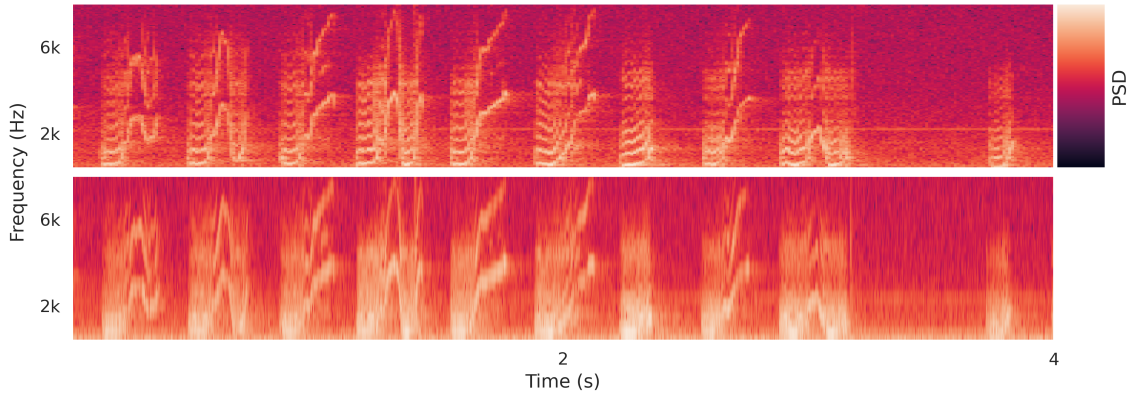
The size of the STFT segments (i.e. window size or NFFT) on the STFT algorithm

determines the frequency resolution. The y-axis on a spectrogram always ranges from 0 up to half of the signal sampling rate. However, the number of different frequency intervals within this range is determined by the NFFT. For example, a window size w of 1024 samples and 44.1kHz as the SR results on frequency bins of $\frac{SR}{w} \approx 43.0664Hz$ in length and a total of 512 frequency bins. Additionally, the frequency resolution can be obtained on the time domain as well, with w_t being the length of the NFFT in seconds:

$$Fr = \frac{1}{w_t} \tag{2.5}$$

Window length determines the time resolution as well. Oversized windows increase the frequency resolution and decrease temporal resolution. Overlap between sequential windows can be introduced to remediate this decrease. Spectrogram generation relies on a set of hyper-parameters, such as the window size, overlap, and window typology. The choice of these hyper-parameters directly impacts the resolution of the time-frequency information, as exemplified in Figure 2.2.

Figure 2.2: Spectrogram visibility is drastically impacted by the chosen hyper-parameters. On the spectrogram at the top, NFFT is 1024 samples and the overlap 512 samples. The bottom spectrogram takes the same signal, but applies a window size is 128 samples and there is no overlap when generating the spectrogram.



Source: The Author

2.2 Machine Learning (ML)

Machine Learning is a sub-field of Artificial Intelligence dedicated to exploring techniques where learning is based on a model experience (RUSSELL; NORVIG, 2019) or the data itself (HAN; PEI; KAMBER, 2011). It does not rely on hand-crafted knowledge programming, but it learns how to represent/extract this knowledge by discovering

patterns on its input. Such learning from experience or established knowledge yield three main learning task types: Supervised Learning, Unsupervised Learning, and Reinforcement Learning. This chapter briefly clarifies common concepts of Machine Learning, focusing mainly on Supervised Learning and Deep Learning methods.

2.2.1 Supervised Learning: Binary classification

Supervised Learning consists of the task to find a model able to generalise based on previously classified data (i.e., previous knowledge). This is done by fitting these model parameters to a training dataset, given an input and output pair. One of the approaches proposed by this work can be expressed as a binary classification problem (i.e., two possible outcomes), which can be generically stated as: *Given a label* $y \in \{0, 1\}$ *and an input* $\boldsymbol{x}$*, learn a mapping function* f *such that* $f(\boldsymbol{x}) = \hat{y}$*, and minimizes the loss* $L(\hat{y}, y)$.

This process of finding a function – or fitting a model – is achieved by minimising a loss function $L(\hat{y}, y)$, which quantifies how far the prediction is from the correct classification. The function $\hat{f}$ (i.e., the model) that can approximate the ideal mapping f is called a hypothesis, and the class of supervised learning techniques defines the space from which this function can be drawn (RUSSELL; NORVIG, 2019).

2.2.2 Deep Learning and Neural Networks

According to Russell e Norvig (2019), *deep learning* is a broad family of techniques for machine learning in which potential models take the form of complex algebraic circuits with tunable connection strengths – a network of interconnected nodes, where each of these connections has a weight. The word *deep* refers to the fact that these networks are typically organised into many layers, which means that computation paths from inputs to outputs have many steps. Deep learning is currently the most widely used approach for applications such as visual object recognition, machine translation, speech recognition, speech synthesis, and image synthesis.

These networks are formed by several nodes organised in several layers, where each of these nodes computes a weighted sum of its input and a bias weight followed by the application of an activation function on the resulting sum. This result is then

passed to subsequent nodes, which repeat the process. Several activation functions are usually nonlinear. The fact that these activation functions are usually nonlinear makes it possible for a sufficiently large network to approximate arbitrary functions (RUSSELL; NORVIG, 2019). The network of interconnected nodes between the first and last layers is called hidden layers, while the first layer is the input layer, and the last layer is the output layer. The optimisation of such networks is done by applying a method called back-propagation (or a variation of it), which propagates the error on the output (computed by a loss function) – the gradient – to every weight associated with every node connection. These weights are then updated according to their contribution to this error on the model output given by an optimiser algorithm rule. Different optimisers are suitable for different network architectures.

A common way to address a binary classification problem using neural networks is to have an architecture with an arbitrary amount of layers and weights and a particular output node with a sigmoid activation function – an activation function that maps its input to an interval between 0 and 1. The output of this model can be interpreted as the probability of the model input being from the positive class. Regarding the first part of this model, several network types arise depending on how these interconnections are made. One example of these models is a feedforward network (or fully connected networks – FC), where the connections between nodes from different layers are only in one direction.

2.2.3 Fully Connected and Convolutional Neural Networks

A fully connected layer (or FC layer) works as a hidden layer from a multi-layer node. It is composed of O sets of weights (for output of size O), called neurons, and each set has the same size as the input vector, so it is possible to do a dot product between them. Since one weight directly corresponds to an input feature, this approach is heavily impacted by input feature locality (RUSSELL; NORVIG, 2019).

In Convolutional Neural Networks (CNNs), the learned filters by convolutional[1] layers have shared weights, which removes the locality dependency with its input. Convolutional layers can receive either matrices or vectors as input, and they can also have multiple channels. For example, on two-dimensional signal processing such as images, one can have an input with dimensions C $\times$ M $\times$ N – C is the number of input channels,

[1]The convolution process referenced here is cross-correlation, even though this type of network use *Convolutional* on its name.

and MxN is the matrix dimension. On one-dimensional signals, the input format consists of C $\times$ L, where L is the length of the input signal. The weights or parameters of this layer used during the convolution are called kernels or filters, and they can have an arbitrary shape K.

The convolution also has a stride parameter S. Defines the spacing between two successive convolutions on the input signal. In order to control the output size, a padding parameter P is also introduced. Padding inserts zero or other values to the input. To increase the depth-of-filed of a filter, one can use dilation D, which inserts spaces internally the kernel weights.

Convolutional layer output L_{out}, given an input length L_{in}, has as many channels as the number of kernels, and it is defined as:

$$L_{out} = \lfloor \frac{L_{in} + 2 \cdot P - D \cdot (K-1) - 1}{S} + 1 \rfloor \tag{2.6}$$

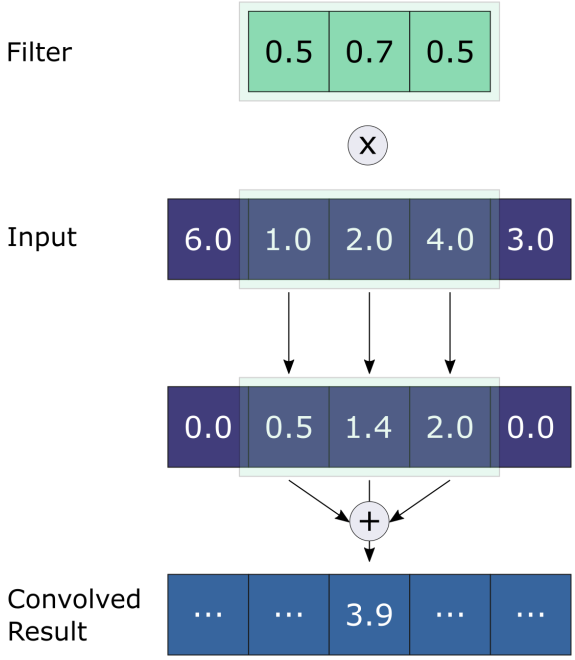
Different from the fully connected layer, the convolutional layer keeps the spatial information of its input in the output: a kernel that learns to respond to a particular pattern in the input will always give the same output, and it will have a position relative to the input. For this reason, when processing digital signals, it is more interesting to keep this kind of layer before the fully connected layers since they extract signal local independent features. Figure 2.3 has an example of the convolution for a one-dimensional signal.

In CNNs, the input must pass through several layers until the final layer can process an output, such as a class probability. Generally, there is a significant difference between the input and output dimension size. The layers have to reduce the input size until it gradually reaches the output, mainly due to model complexity and parameter number. Smaller inputs are also better for the model's efficiency since it will take less time to convolve a smaller input. In such cases, it is helpful to insert max-pooling layers in the network. This kind of layer works similarly to the convolutional layers, having kernel size and stride hyper-parameters. However, instead of multiplying, it returns only the largest value of the input region under the kernel. Another main difference is that it pools values channel-wise, so its output always keeps the number of input channels.

2.2.3.1 Sinc-based convolution

Introduced by Ravanelli e Bengio (2018) into CNNs, this approach to the first convolutional layer aims to reduce the number of learnable parameters and improve the

Figure 2.3: One-dimensional convolution example.



Source: The Author

general interpretability of this type of network.

The first layer of a standard CNN performs a set of time-domain convolutions between the input waveform, and some Finite Impulse Response (FIR) filters (RABINER; SCHAFER, 2010). Each convolution is defined as follows:

$$y[n] = x[n] * h[n] = \sum_{l=0}^{L-1} x[l] \cdot h[n-l] \tag{2.7}$$

where $x[n]$ is a chunk of the 1D signal, $h[n]$ is the filter of length L, and $y[n]$ is the filtered output. In standard CNNs, all the L elements of each filter are learned from data. Sinc-based convolutions are performed with a predefined function g that depends on a few learned parameters. The function g is a band-pass filter on the time domain, and is defined as, which is based on a difference of scaled sinc function:

$$g[n, f_1, f_2] = 2f_2\text{sinc}(2\pi f_2 n) - 2f_1\text{sinc}(2\pi f_1 n) \tag{2.8}$$

where the sinc function is defined as $\text{sinc}(x) = \sin(x)/x$. The two parameters for this function are cutoff frequencies.

Using the function g instead of learning the filters from scratch as standard CNNs reduces the number of learnable weights. For instance, considering a layer composed of F filters of length L, a standard CNN employs $F \cdot L$ parameters against the $2F$ considered

by sinc-based convolutions. Moreover, if the filter length L is doubled, a standard CNN doubles its parameter count, while on this type of sinc-based convolution, the number of learnable parameters remains constant.

2.2.4 Avoiding overfitting

When training a model for further testing or use, two characteristics may arise. Suppose the model is not complex enough to be optimised to learn the pattern within the data (i.e. high loss function value, low value for the evaluation metrics). In that case, the model is *underfitting*. On the other hand, if a given predictor model performs well on the training set (i.e., low loss function value, high value for evaluation metrics), but poorly on the testing set, the model is said to be *overfitting* or unable to generalise for unseen data. One way to solve this is by introducing regularisation techniques, such as Dropout or Batch normalisation.

In neural networks with several layers and unbounded activation functions, the outputs may reach tremendous values, resulting in errors during error calculation. Another problem is the variation of the distribution of each layer's inputs, which can slow down the learning process and require tuning other solver parameters. *Batch normalisation* (IOFFE; SZEGEDY, 2015) is a particular layer created to approach these issues by normalising the output of each layer. It is also shown that batch normalisation helps to reduce the model overfitting. The batch normalisation layer takes the current input batch's mean and variance during the training phase and then normalises its examples. Hence, the output batch has zero mean and unit variance. Then, the output examples are scaled by parameters learned during the training, so the model can regulate how much of the normalisation will be passed to the output. After the training phase, when the network is being tested, the layer uses a pre-estimated mean and variance of the training set batches, making the network outputs deterministic. With a similar purpose of batch normalisation, *layer normalisation* (BA; KIROS; HINTON, 2016) normalises input across features instead of normalising input features across the batch dimension.

When training a deep learning model, one might use Early stopping. *Early stopping* consists of stopping the training on the epoch before the loss function value keeps growing at least p times on a previously defined validation set, where p is called patience on this approach.

The *Dropout layer* (SRIVASTAVA et al., 2014) is another approach to the over-

fitting problem. It is used during training, and it "turns off" randomly a given amount of parameters of a layer so that only part of them will participate in the forward and backward phases of the training. The removed outputs are changed during each iteration, so the layer has different parameters updated by the optimiser. This way, in every iteration, a slightly different model is trained, and the number of these models increases exponentially with the number of parameters. In theory, the chance that each of these models learns the same parameters is meagre, resulting in a low chance of the final model to overfit.

2.2.5 Supervised Learning evaluation metrics

Several metrics are available to identify how a given model performs as on a problem solution or compare different models performances on this same problem. Accuracy, sensitivity/recall, specificity, precision, Matthews Correlation Coefficient (MCC), F_1, and F_β scores are standard metrics to look at when considering a classifier performance. When comparing two models, tests of statistical significance are applied on both models (HAN; PEI; KAMBER, 2011). Metrics associated with the first situation are described in more detail in this section.

A binary classifier prediction can be divided into four categories. True Positives (TP) refers to the number of correctly classified instances with a ground-truth label equal to 1. Analogously, True Negatives (TN) represent the amount of correctly classified instances with a ground-truth label equal to 0. The number of instances with different classifications from the associated ground-truth is called False Negatives (FN) and False Positives (FP). Thus, the amount of ground-truth-equals-1 instances (P) equals the sum of TP and FN. Similarly, the number of "negative" instances (N) is equal to the sum of TN and FP. These definitions are the base of the metrics above – which are formalised as follows:

Accuracy: the percentage of the test set instances that the model correctly classifies.

$$\text{acc} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}} \tag{2.9}$$

Sensitivity/recall and specificity: Sensitivity is also referred to as the true positive (recognition) rate (i.e., the proportion of positive instances that are correctly identified), while specificity is the true negative rate (i.e., the proportion of negative instances

that are correctly identified).

$$\text{sens} = \frac{\text{TP}}{\text{P}} \tag{2.10}$$

$$\text{spec} = \frac{\text{TN}}{\text{N}} \tag{2.11}$$

Precision: a measure of exactness (i.e., the percentage of instances labelled as positive that are such). Thus

$$\text{prec} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.12}$$

Matthews correlation coefficient (MCC): represents how random the prediction of a model is. It is a correlation coefficient between the observed and predicted classifications, returning values ranging from -1 and +1. A coefficient of +1 represents a perfect prediction, 0 is no better than a random prediction, and -1 indicates total disagreement between prediction and the ground-truth.

$$MCC = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP} + \text{FP}) \cdot (\text{TP} + \text{FN}) \cdot (\text{TN} + \text{FP}) \cdot (\text{TN} + \text{FN})}} \tag{2.13}$$

$\mathbf{F_1}$ and $\mathbf{F_\beta}$ scores: a weighted combination of precision and sensitivity/recall. F_1 measure gives both measurements equal importance, while F_β gives different importance to each metric based on the β value. It's important to notice that F_1-score is a specific case of F_β-score. The definition for these metrics is

$$F_1 = \frac{2 \cdot \text{prec} \cdot \text{sens}}{\text{prec} + \text{sens}} \tag{2.14}$$

$$F_\beta = \frac{(1 + \beta^2) \cdot \text{prec} \cdot \text{sens}}{\beta^2 \cdot \text{prec} + \text{sens}} \tag{2.15}$$

These metrics can be computed on a test set (unseen data by the model) to address the model ability of generalisation. The test set can be obtained using different approaches, such as holdout and k-fold cross-validation. In the first method, the given data are randomly partitioned into train and test sets. In the second method, k-cross-validation, the initial data is randomly partitioned into k mutually exclusive subsets (or folds) $D_1, D_2, \ldots, D_k$, each of approximately equal size and ideally with similar class balance. Training and testing are performed k times. In the ith iteration, the partition D_i is reserved as the test set, and the remaining partitions are collectively used to train the model. Unlike holdout, each sample is used the same number of times for training and

once for testing (HAN; PEI; KAMBER, 2011). For classification, the accuracy estimate is the sum of the accuracy on the test set on each interaction, averaged by the number of iterations k.

2.3 Guinea pig as an attachment model

This section presents a brief background of the biological significance, motivation for the experiments and tool development presented in this work.

2.3.1 Mother-infant attachment

In most invertebrates and vertebrates, infants rely on parental care – usually on the mother – for survival. Maternal care is vital for mammals because the mother provides all the resources for infant growth and development. The infant-mother interaction is based on intimacy and synchronicity, and it shapes infants' physiology and behaviour. Mammals are born either immature (altricial) or in a more mature state (precocial). Nevertheless, infant-mother interaction is ubiquitous in mammals. In some mammals, this dyadic bonding occurs only between the biological mother and her infant and requires recognition of both individuals. This unique bonding is defined as *attachment*, first described in (BOWLBY, 1979). The lack of or disruption of this bonding is thought to lead to long-term consequences for the infant, including an increased risk of anxiety, depression, and obesity (WEWERS; KAISER; SACHSER, 2003; RITCHEY; HENNESSY, 1987; HENNESSY et al., 1995). In that sense, guinea pigs (*Cavia porcellus*) are rodents that serve as a model to study mother-infant attachment (PORTER; BERRYMAN; FULLERTON, 1974; PORTER; FULLERTON; BERRYMAN, 1973). For example, the response of infant guinea pigs to the separation of the mother resembles the response of non-human primates in Harlow's (HARLOW, 1959) pioneering studies in separated infant Rhesus Monkey.

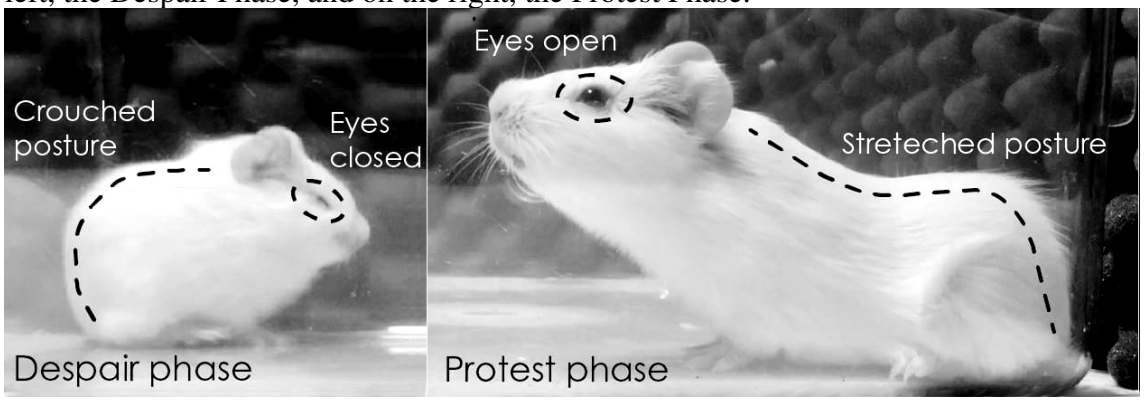
2.3.2 Vocalization characteristics and context

To draw attention and elicit mother retrieval, guinea pig infants (pos-natal) emit vocalization (HENNESSY, 2014; HENNESSY; RITCHEY, 1987; PETTIJOHN, 1979)

when isolated from this caregiver. When in this situation, also called Protesting Phase – active search behaviour (Figure 2.4, right), it is usual that these infants show high vocalization rates. Vocal rates return to baseline levels upon reunion with the caregiver or when at Despair Phase – a state of profound physical inactivity (Figure 2.4, left). Factors such as environmental temperature and food availability can affect the vocalization rate. These effects on the vocalization characteristics can reliably quantify their consequence on the pup internal state.

Varying from 200Hz to 20kHz (BERRYMAN, 1976) – mostly the audible range for humans –, guinea pig vocalizations usually occurs in bouts with a sequence of syllables (Figure 2.1). A *vocalization syllable* is defined as a segment of sound separated at least by 5ms from another. They have a complex inner structure, with some of their types having several harmonic components and frequency modulations.

Figure 2.4: Guinea pig infants display different behavioural phases when isolated. On the left, the Despair Phase, and on the right, the Protest Phase.



Source: Yale Laboratory of Psychology of Behaviour

3 STATE-OF-THE-ART AND MOTIVATION

3.1 Existing tools to detect rodent vocalizations

Several approaches and tools to detect vocalizations of rodents and other species are available in the scientific community. In this section, commonly used tools for rodent vocalization detection are described, even though some of them are general-purpose detectors. A detailed description of these tools is presented below, and Table 3.1 summarizes the main features and characteristics.

A first example is VocalMat (FONSECA et al., 2021). By performing segmentation of vocalizations using computer vision applied on a generated spectrogram for a given audio file, VocalMat performs automated, accurate, and quantitative analysis of mice vocalizations without the need for user inputs. It incorporates particular characteristics of mice USVs in this process, such as the minimum duration for a mice vocalization. These segments of spectrograms are fed into a CNN to distinguish between vocalization and noise. After this differentiation process, segments with vocalizations resulting from this step are then passed for the second module of this tool to classify the vocalization according to its morphology.

A more general approach (although still focusing on rodent USVs) is DeepSqueak (COFFEY; MARX; NEUMAIER, 2019). It is a USV detection and analysis software suite that can automatically perform detection and classification using neural network models for object detection, with architectures varying between different versions. The object (i.e., vocalizations) detection is performed on spectrograms as well.

Although the primary purpose of MUPET (SEGBROECK et al., 2017) is to analyze the relationship between vocalizations and their repertory, the initial step is to detect these vocals. MUPET segments individual vocalizations by measuring the power spectrum in the ultrasonic range and comparing it with a noise threshold previously computed. Therefore, this analysis is made on the frequency domain directly, the same information visually represented by a spectrogram. By pre-computing PSD spectrum properties and using a threshold for classification as well, a recent tool called AMVOC (STOUMPOU et al., 2021) reaches the state of the art results on USV detection.

Deep Song Segmenter or DeepSS (STEINFATH et al., 2021) is another recent tool that was tested on different animal species songs such as mice, flies and birds. It is the only tool that looks into raw waveforms instead of spectrogram or other related information on

the context of animal vocalizations. The classification provided by different Temporal CNN models (one for each species) is at the sample level. This classification can address binary classification (vocalization or not) and multi-class (i.e., vocal repertoire).

Regarding specific tools for guinea pig vocalization detection, none was proposed until the date of this proposal. However, due to its methodology, DeepSqueak and especially DeepSS could be used to automate this process, although no studies showing these applications were found until the date of this work. The detection DSP technique based on AMVOC could be applied as well since the spectrogram specificities are hyperparameters. Therefore, the last approach was adapted to guinea pigs and served as the baseline model for this work since it addresses the differences when comparing vocal detection on a spectrogram and in raw waveforms.

Table 3.1: Summary of the existing tools to detect rodent and other species vocalizations.

Model	**Species**	**Guinea pig appl.**	**Input type**	**Technique**
VocalMat (FONSECA et al., 2021)	Mice	No	Spectrogram	Computer vision; Convolutional neural network
DeepSqueak (COFFEY; MARX; NEUMAIER, 2019)	Mice; rat	Yes	Spectrogram	Object detection; Convolutional neural network
MUPET (SEGBROECK et al., 2017)	Mice	No	Spectral information	Power spectrum analysis
AMVOC (STOUMPOU et al., 2021)	Mice	Yes	Spectral information	Power spectrum analysis
DeepSS (STEINFATH et al., 2021)	Mice	Yes	Raw-waveform	Temporal Convolutional Network

Source: The Author

3.2 Spectrograms vs Raw waveforms

Machine Learning and audio signals are widely used on speech-related tasks. Audio data is intrinsically highly dimensional due to its recording settings, such as the sampling rate. Several hand-crafted features can be computed to summarize and simplify this data type so it can be fed to a prediction model, such as the computation of

FBANK and MFCC coefficients (VARIANI et al., 2014; RICHARDSON; REYNOLDS; DEHAK, 2015; SNYDER et al., 2017). Other approaches rely on feeding the model with spectrogram bins directly since they retain more information than hand-crafted features (ZHANG; KOISHIDA; HANSEN, 2018; BHATTACHARYA; ALAM; KENNY, 2017; NAGRANI; CHUNG; ZISSERMAN, 2017). The previous section shows that feeding a deep learning model relying on spectrogram bins or power spectrum information is widely used for vocalization detection. However, the spectrogram generation requires careful tuning of crucial hyper-parameters (Figure 2.2), especially on the vocalization analysis domain (BRUDZYNSKI, 2018).

For this reason, a more recent trend is to learn from raw waveforms directly, thus altogether avoiding any feature extraction step (RAVANELLI; BENGIO, 2018; OORD et al., 2016; PALAZ; COLLOBERT et al., 2015; HOSHEN; WEISS; WILSON, 2015).

3.3 Proposed work

This work proposes a machine learning model, AudibleSincNet, able to detect guinea pig vocalizations directly from raw waveforms. New metrics are also introduced to correlate the biological aspect intrinsic to this work with the ML standard metrics. Performance on vocalization detection is done by comparing the proposed approach with a spectrogram-based approach, or baseline model.

The contribution of this work is not restricted to guinea pig vocalization detection. Using raw waveforms as input for a model can give insights on developing a species-agnostic approach to detect vocalizations since it does not rely on spectrogram generating or its species specificities.

4 METHODS

This chapter describes the methodology of training and evaluation of the two proposed approaches. Appendix A and B have information about tested hyper-parameters and used technologies on the detection models described below.

4.1 Dataset

Guinea pig vocalizations were recorded using a microphone inside a custom-designed experiment cage covered with acoustic foam by the Laboratory of Psychology of Behavior (Yale University) team. Recordings were re-sampled to 44.1kHz, if necessary.

Annotations of vocalizations were manually created, evaluated and filtered by a third party. The dataset has five audio files, containing each ~90 minutes except for one file that was cropped to 60 minutes – resulting in ~7 hours of recording material with 39,561 manually annotated vocalizations.

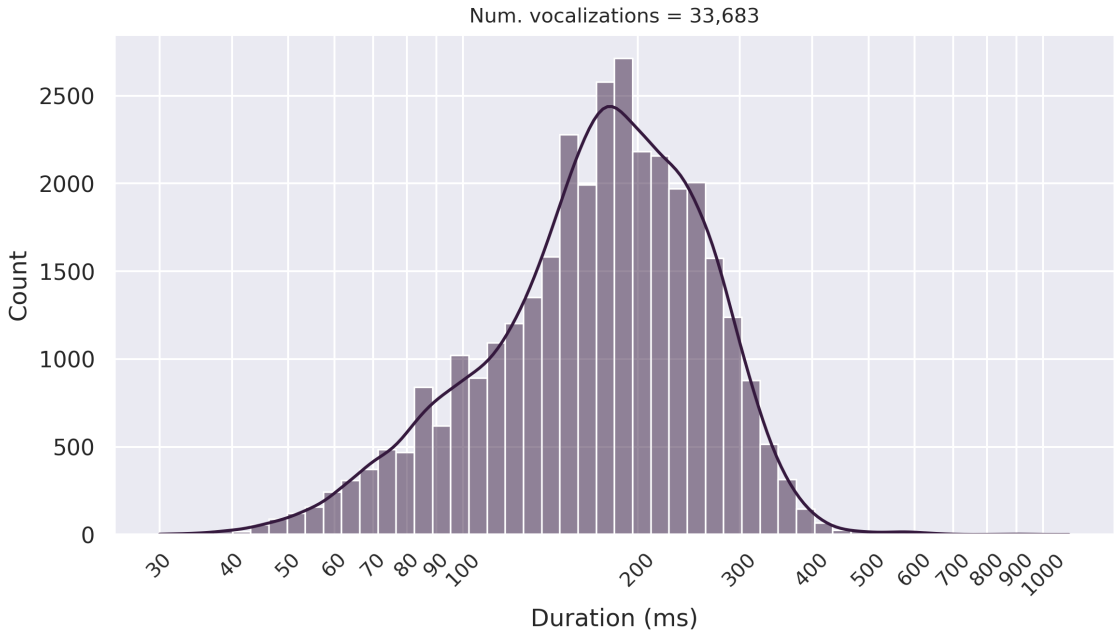
Four out of the five files were assigned as the training set and the remaining as the test set. The test set is used to address the generalization ability for both of the approaches described below. No audio pre-processing technique was used. Vocalization duration distribution for the train set and the test set along with their data Gaussian Kernel Density Estimation (GKDE) – a method for visualizing the distribution of observations, by representing the data using a continuous probability density curve in one or more dimensions[1] – are presented on Figures 4.1 and 4.2, respectively.

4.2 AudibleSincNet

The proposed model, named AudibleSincNet, is based on the SincNet (RAVANELLI; BENGIO, 2018) and is composed of three different blocks. The first applies several convolutions based on sinc functions for different frequencies. The second applies standard convolutions, and the last one represents the fully connected part of the network. Figure 4.3 contains a schematic with a detailed structure of the model. The output logit is then transformed into a probability between 0 and 1 using a sigmoid σ activation function.

[1] Bandwith for the GKDE is defined using a rule of thumb called Scott's Rule (SCOTT, 2015).

Figure 4.1: Distribution of vocalization duration on the train set.



Source: The Author

4.2.1 Model description

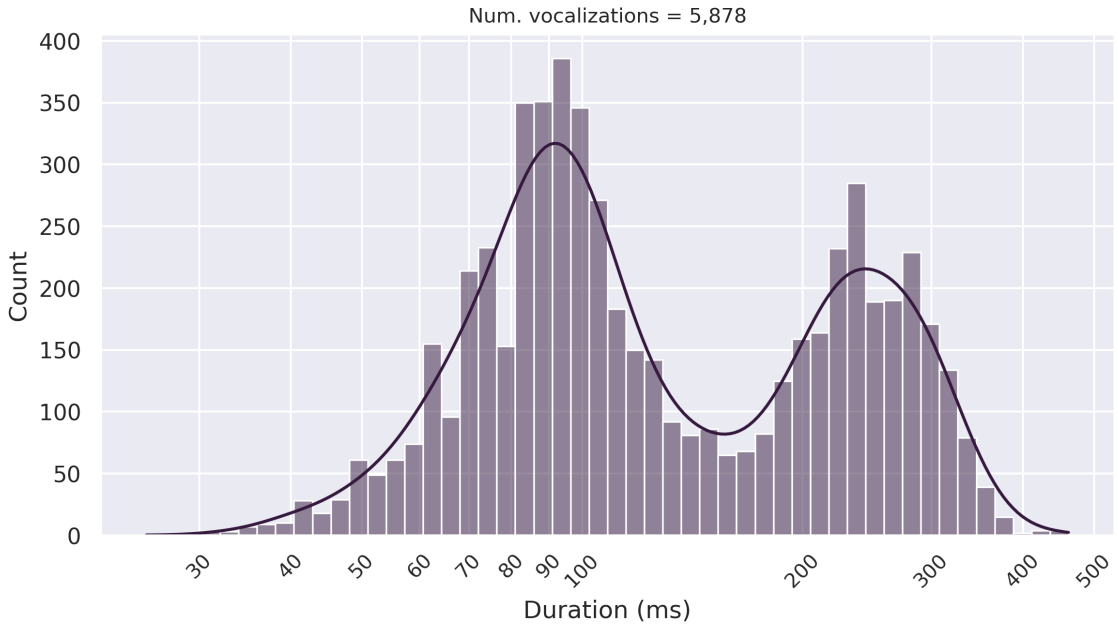
AudibleSincNet uses as input raw waveform windows instead of spectrogram windows or any spectral information. One way to accomplish this is by dividing the recording's raw waveform into windows, where each window is the input for a given model. Although this approach removes the dependency on the audio duration – by providing a fixed input size – it also introduces two hyper-parameters: the window length and, potentially, the overlap length. The window length represents the duration (at sample level) of audio that will serve as an input for the model. The overlap length defines the temporal redundancy between two consecutive windows.

The input size for this network is 441 samples (equivalent to 10ms of audio), which aims to give enough signal for the convolutions to work as the way that they are intended to [2].

For temporal redundancy, each audio file was divided into overlapping windows (25% of overlap or 110 samples). A label was assigned for each window based on its annotation coverage. The window coverage is hereby defined as segment percentage that contains an annotation – or part of one. This process aims to balance the loss of precision

[2]One might use 0-padding to compensate for smaller input sizes or use rules that preserve size when convoluting. However, since the original SincNet uses a more classical CNN architecture (besides the SincConv), these approaches will not be explored in this work.

Figure 4.2: Distribution of vocalization duration on the test set.



Source: The Author

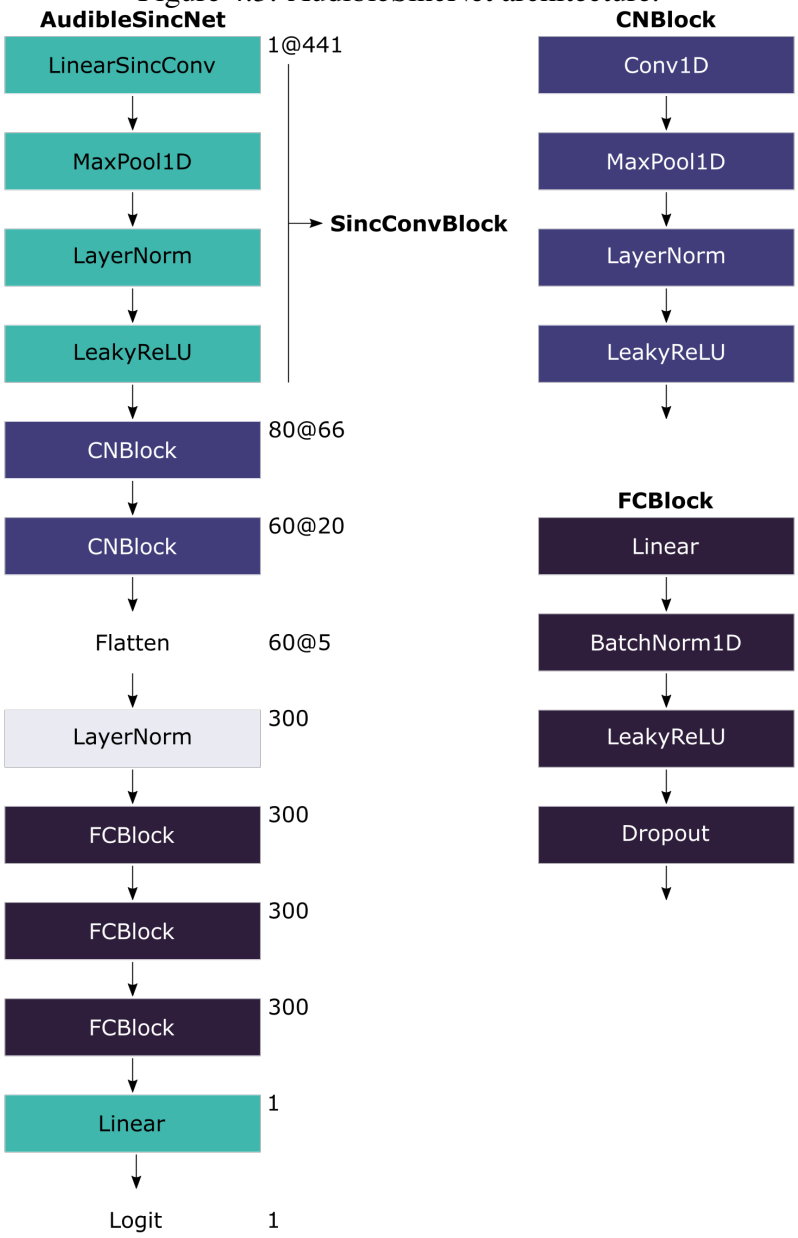
on annotation borders due to the windowing process. The window coverage is inversely proportional to the overlap size: for an overlap of 25%, a minimum coverage of 75% by annotation is desirable, so the label becomes 1 (has vocalization), otherwise 0.

Since the Mel scale is a perceptual scale of pitches judged by human listeners to be equal in distance from one another (STEVENS; VOLKMANN; NEWMAN, 1937), SincConv was initialized with linear scaled filters instead of the proposed initialization on Ravanelli e Bengio (2018). Linear initialization is more suitable for guinea pig vocalizations analysis since it does not rely on human perception.

4.2.2 Training phase and testing

AudibleSincNet was trained using the RMSprop optimization algorithm with smoothing constant $\alpha = 0.95$, $\epsilon = 10^{-7}$ (for numerical stability) (RAVANELLI; BENGIO, 2018) and a learning rate γ of 0.001. Binary Cross Entropy with Logits loss function L (Equation 4.1) is used to compute how far is the prediction from the ground-truth. In order to compensate the class imbalance (~2.3 more interval windows than vocalization windows), a positive-weight w_p is introduced in the loss function with a corresponding value to the

Figure 4.3: AudibleSincNet architecture.



Source: The Author

class proportion:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{\sum^{n} w_p \cdot y_n \cdot \log\left(\sigma(\hat{y}_n)\right) + (1 - y_n) \cdot \log\left(1 - \sigma(\hat{y}_n)\right)}{N} \tag{4.1}$$

where the first term in the sum refers to the positive case prediction and the last term to negative prediction; the sigmoid activation function σ is defined as $\sigma(x) = (1 + e^{-x})^{-1}$.

Hyper-parameter optimization, such as the learning rate, was performed by using k-fold cross-validation with $k = 10$. The selection of the best hyper-parameter was based

on the resulting mean Accuracy, F1-measure and MCC over ten iterations.

The model was then trained from scratch using the best learning rate value and early stopping with patience $p = 30$ to prevent overfitting, with 15% of the training set used for validation. Loss computation on the validation set was performed after every training epoch, and the early stopping process ran after every validation epoch. The trained model was then applied to the test set and evaluated conforming to annotation detection metrics.

4.3 Baseline model

This model has the advantage that it does not rely on annotated vocalizations. It is an unsupervised approach that was recently applied to mice USVs (STOUMPOU et al., 2021). Using spectral energy as the primary metric, this model relies on two thresholds: time-based t and frequency-based f.

4.3.1 Time-based thresholding

The time-based thresholding t involves simple temporal thresholding of the spectral energy values (STOUMPOU et al., 2021). The spectral energy S consists of the sum of every frequency energy value for each time step (or spectrogram window) based on the spectrogram. More formally, the spectral energy at the time frame i, S_i, can be defined as:

$$S_i = \sum_j E_{ji} \tag{4.2}$$

where E_{ji} is the energy associated with a frequency in i. The step for i is equal to the frequency range step. This computation can be used to filter time frames with energy above a given threshold. Associated with this threshold, the dynamic threshold T_i is defined at each time frame as the average of the current spectral energy (S_i) and the moving average of the spectral energies of the last K windows:

$$T_i = \frac{1}{2}\frac{\sum_{j=0}^{N-1} S_j}{N} + \frac{1}{2}\frac{\sum_{j=0}^{K-1} S_{i-j}}{K} \tag{4.3}$$

where N is the number of time frames and K is the size of the moving average.

4.3.2 Frequency-based thresholding

The frequency-based thresholding f is defined as applying a threshold to the energy distribution across frequencies on each time frame. One way to approach this (STOUMPOU et al., 2021) is to keep only time frames where the peak energy value P_i is larger than the mean spectral energy M_i of a range F around the frequency of with the peak energy. Both quantities can be formally defined as:

$$P_i = \max_j E_{ji} \tag{4.4}$$

$$M_i = \frac{1}{N_f} \sum_{j\,=\,p_i-\frac{F}{2}}^{p_i+\frac{F}{2}} E_{ji} \tag{4.5}$$

where N_f is the number of frequency bins in the sum range and $p_i = \mathrm{argmax}_j\, E_{ji}$.

4.3.3 Model description

For each of the windows that composes the spectrogram (i.e., time frame), the spectral energy S (Equation 4.2), dynamic temporal threshold T (Equation 4.3), mean spectral energy M (Equation 4.5), and the peak energy P (Equation 4.4) were computed. To assign a label for each window y_i (1 if it is part of a vocalization or 0 otherwise) the following rule was applied:

$$y_i = \begin{cases} 1, & \text{if } (S_i > t \cdot T_i) \text{ and } (P_i > f \cdot M_i) \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

The spectrogram for each file was generated with a window size of $NFFT = 220$ (or ~4.9887ms) and no overlap. Overlap is not used due to its impact on the mean spectral energy M. The window size was defined mainly based on the frequency resolution considering the trade-off between the time and frequency resolutionS. A $NFFT$ of 220 samples generates a spectrogram with 110 bins, where each bin varies ~200Hz from the previous one. Only frequency bins within the range of interest [400, 16000]Hz were considered for computing the spectral energy and any other associated metric when detecting guinea pig vocalizations. The dynamic temporal threshold was computed using $K = 2$ (STOUMPOU et al., 2021). By considering a range that encompasses most of the vocal-

ization harmonic components, the mean spectral energy was computed using $F = 4kHz$.

4.3.4 Training phase and testing

Different combinations of the temporal-based threshold t and the frequency-based threshold f were tried during the training phase. The best model was evaluated based on the detected annotation metrics (defined at the end of this chapter).

4.4 Post-processing

The post-processing step is required to transform the binary signal (i.e., one label for each window) into annotations. Given labels for each window, a smoothing filter is applied on the sequence of labels, so potential missing windows within a vocalization are filtered out. The smooth filter length was optimized by testing different values (more details on Appendix A) for each model.

After the smoothing filter, annotations are generated by considering the beginning and end of each peak (i.e., values different than 0). Computed annotations with a duration smaller than 20ms (minimum length of a vocalization according to Berryman (1976)) were discarded. Consecutive annotations separated by less than 5ms were merged.

4.5 Evaluation

In this section, metrics specific to this proposal's domain are described and their importance for the biological aspect of this work.

4.5.1 Window classification-based metrics

Although the standard metrics to evaluate a prediction model are good enough from the Machine Learning view, they do not encompass the biological aspect, which is fundamental for this work's problem. Therefore, some metrics can be used to give the model predictions a biological sense, such as accuracy per vocalization/interval and pair accuracy.

The first one, *window accuracy*, can be described as the percentage of correctly predicted windows within a vocalization or interval, named TP:

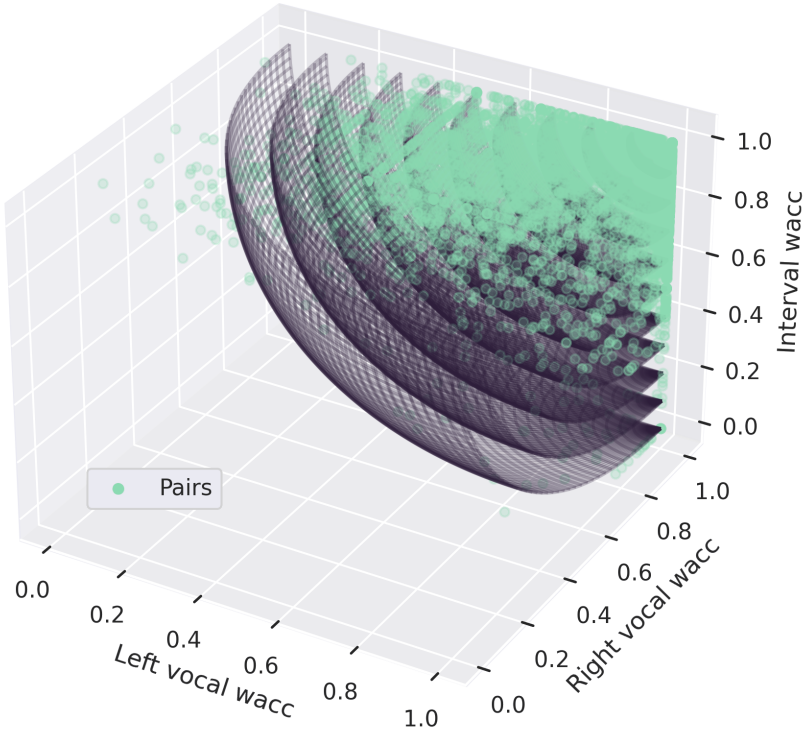
$$\text{wacc}(\mathbf{s}) = \frac{TP}{\text{len}(\mathbf{s})} \tag{4.7}$$

where $len(\mathbf{s})$ is the number of windows associated with a specific vocalization or interval.

Unlike the accuracy per vocalization/interval, the pair distance takes into account a pair of consecutive vocalizations and the interval in between. Since the mean computation result can be affected by outliers, the euclidean distance to the perfect case scenario is computed for each pair. The perfect-case scenario is defined as both vocalizations and the interval between them having a window accuracy of 1. With geometrical inspiration (Figure 4.4), and varying from 0 to 1 (with 1 indicating the perfect prediction), this metric can be defined as

$$\text{pacc}(\mathbf{v_1}, \mathbf{i}, \mathbf{v_2}) = 1 - \frac{\text{euclidean\_distance}([\text{wacc}(\mathbf{v_1}),\ \text{wacc}(\mathbf{i}),\ \text{wacc}(\mathbf{v_1})],\ \text{best\_case})}{\sqrt{3}} \tag{4.8}$$

Figure 4.4: Geometrical inspiration for the Pair Accuracy (pacc), a metric that correlates the window accuracy from pairs of vocalizations and the interval between them.



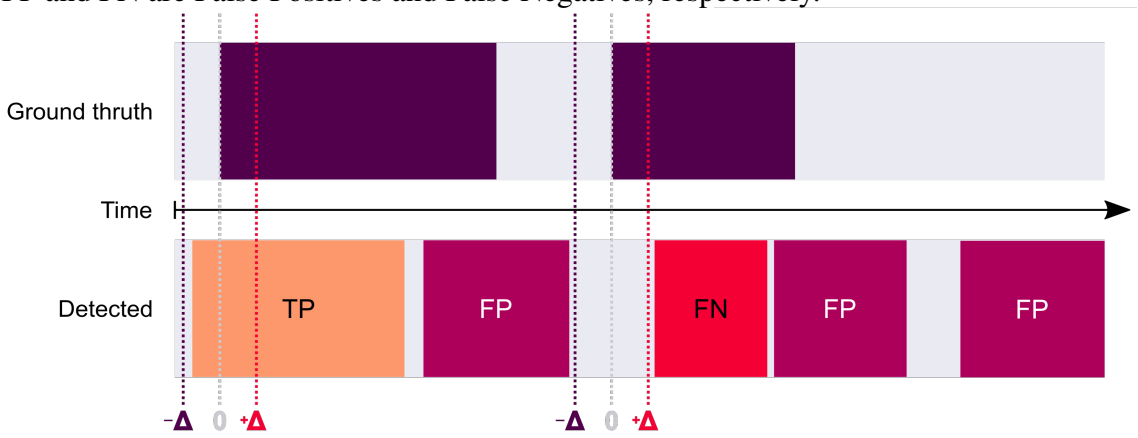
Source: The Author

The computed distance (Figure 4.4, green spheres) from the best-case scenario can be clustered into delimited regions at certain distances from the optimal prediction for further analysis of the distribution of the pairs in this space. An example of this distribution is present on Figure 5.4.

4.5.2 Number of detected vocalizations

It is necessary to have a way to measure the number of correctly detected vocalizations. One way to accomplish this (FONSECA et al., 2021) is to consider a threshold or tolerance Δ for the beginning of the detected annotation to match the manual annotation or ground-truth. Therefore, vocalizations automatically detected with a start time matching the manual annotation (with Δ tolerance) are considered correctly detected, or True Positives (TP). Manually detected vocalizations with no correspondent annotation given by the detector are considered False Negatives (FN). Finally, detected vocalizations without correspondence in the manual annotation are considered False Positives (FP). Detection preciseness can be taken into account by choosing different values for tolerance. A visualization for this metric is available on Figure 4.5.

Figure 4.5: Detection metric illustration based on a tolerance Δ. TP are True Positives; FP and FN are False Positives and False Negatives, respectively.



Source: The Author

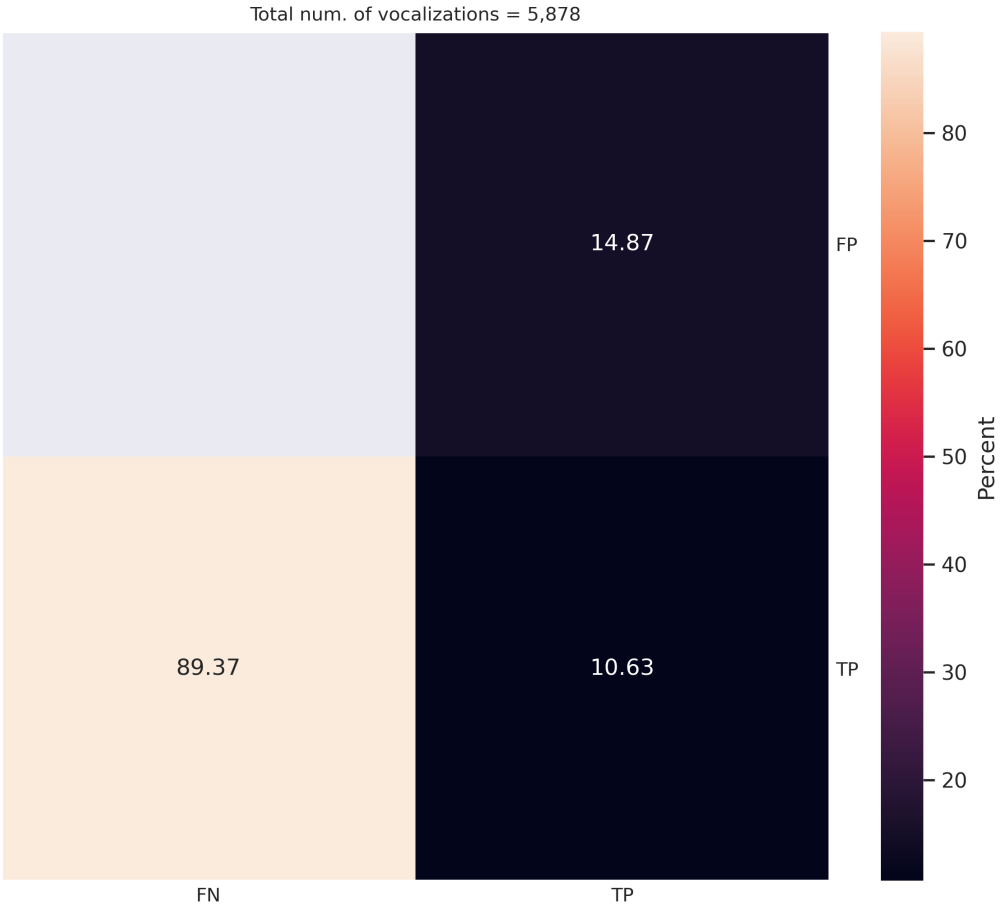
5 RESULTS

This section presents the results for both approaches described above according to the metrics that can be applied to each one. All the metrics were applied to the test set. Results regarding detection metrics are normalized by the number of vocalizations on the ground-truth (manual annotations).

5.1 Baseline model

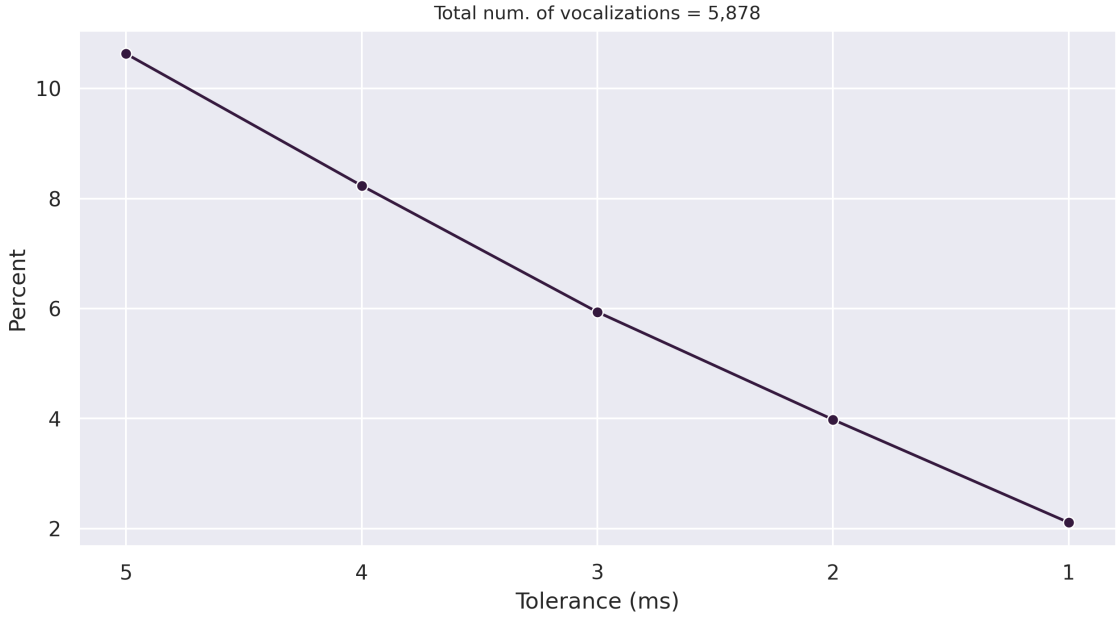
The baseline model detected only 10.63% of the ground-truth vocalizations, while having several false positives (14.87%). These results for 5ms of tolerance are summarized in Figure 5.1. In Figure 5.2, the detection performance for varying tolerance (from 5ms to 1ms) is presented. The presented results are based on the post-processing with smoothing filter of 10ms and model hyper-parameters set $t = 0.7$ and $f = 20$.

Figure 5.1: Baseline results according to the detection metric on the test set ($\Delta = 5$ms).



Source: The Author

Figure 5.2: Detection performance of the baseline model on the test set when considering different values for the tolerance Δ (x-axis).



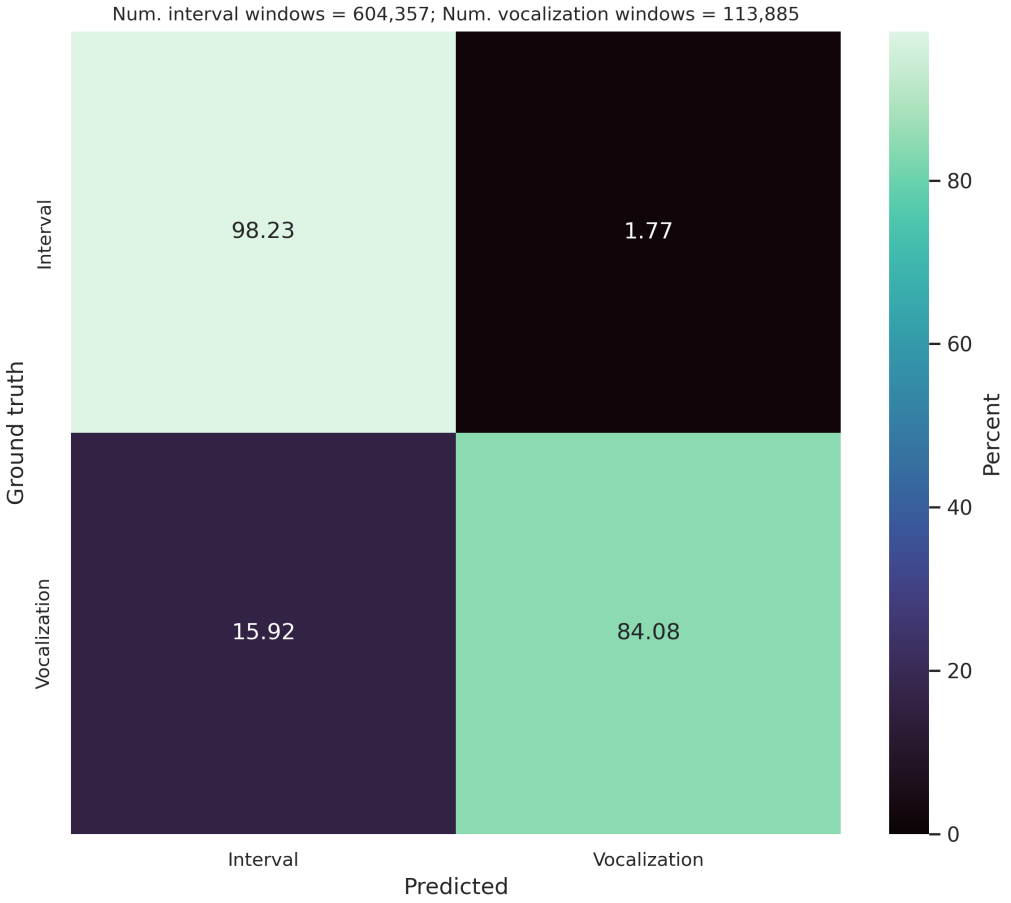
Source: The Author

5.2 AudibleSincNet

AudibleSincNet can be evaluated using common Machine Learning metrics (e.g., Accuracy and F_1-measure) and, most important on the biological aspect, the detection metric. Regarding the first, AudibleSincNet reached on the test set an Accuracy of 0.959, $F_1 = 0.869$ and $MCC = 0.846$. The confusion matrix is shown in Figure 5.3, and is normalized by the total number of windows on each class (interval or vocalization). When considering the classified windows on its context, this model achieved a mean pair accuracy of 0.78 (95% CI $\pm$ 0.003), as shown in Figure 5.4.

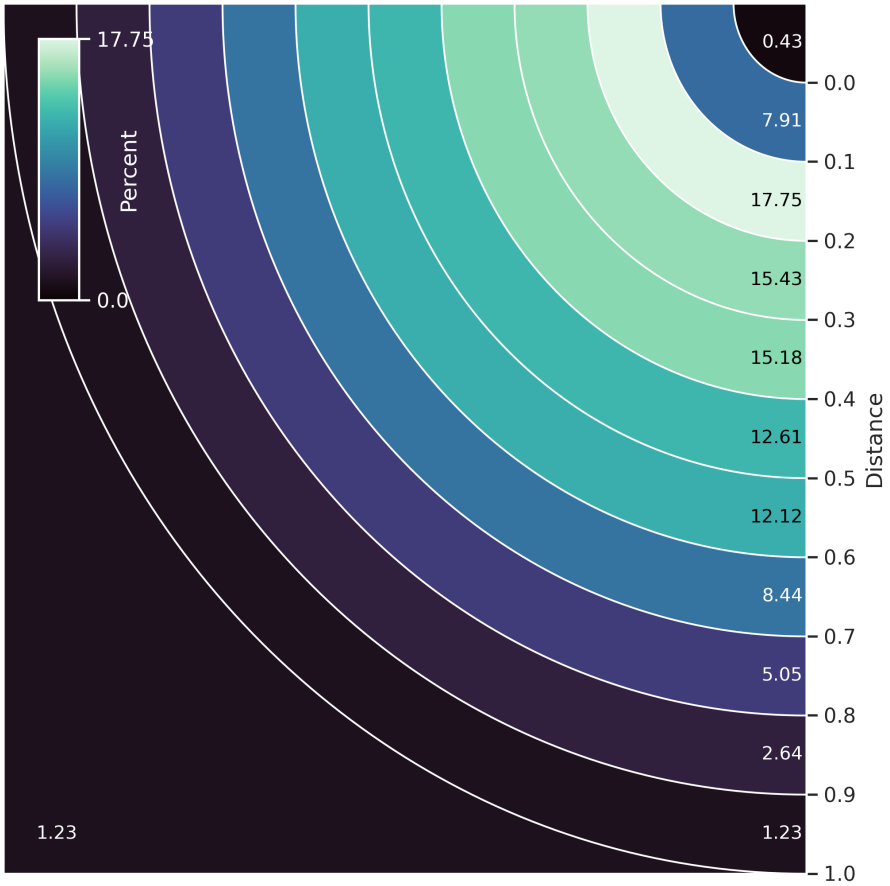
Using a smoothing filter of 30ms on the post-processing step, this model detected 28.72% of the ground-truth vocalizations (TP), while having 13.01% of FP when considering the tolerance $\Delta = 5$ms (Figure 5.5). Similarly for the baseline model, the results when varying this tolerance are presented on Figure 5.6.

Figure 5.3: Window classification performance on the test set for AudibleSincNet.



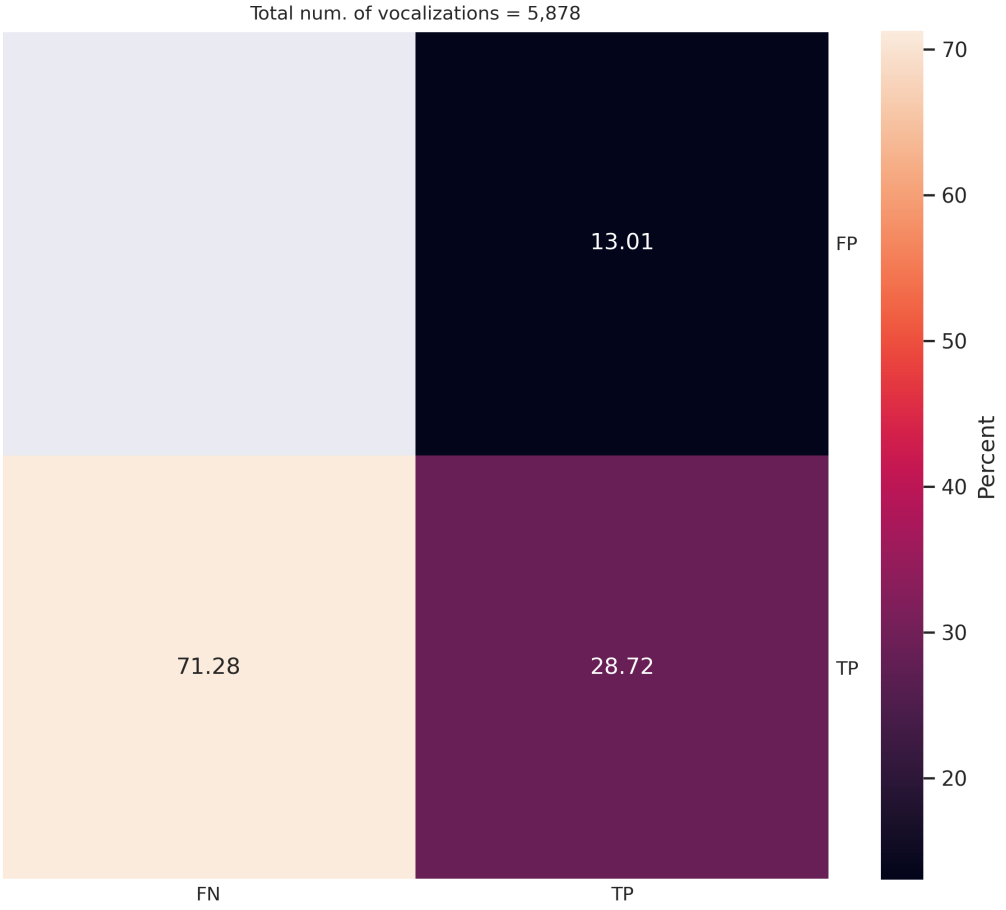
Source: The Author

Figure 5.4: Overall distribution of the pairs when considering AudibleSincNet classified windows on their context.



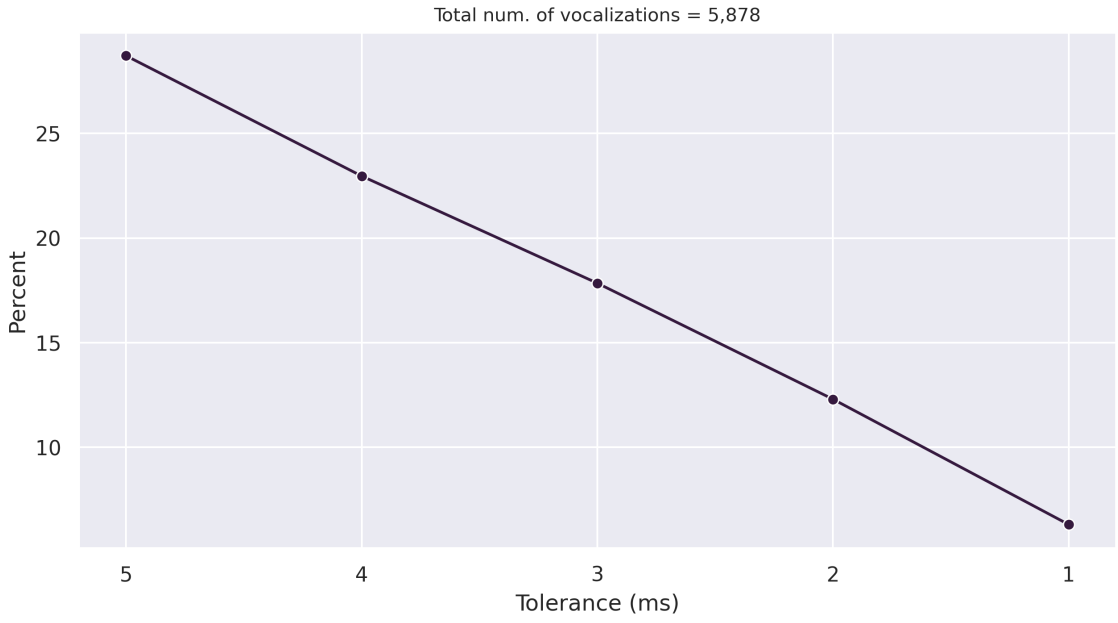
Source: The Author

Figure 5.5: AudibleSincNet results according to the detection metric on the test set ($\Delta =$ 5ms).



Source: The Author

Figure 5.6: Detection performance of AudibleSincNet on the test set when considering different values for the tolerance Δ (x-axis).



Source: The Author

6 DISCUSSION

This section aims to provide insights and analyse the different results for both proposed approaches. More specifically, it aims to answer two questions:

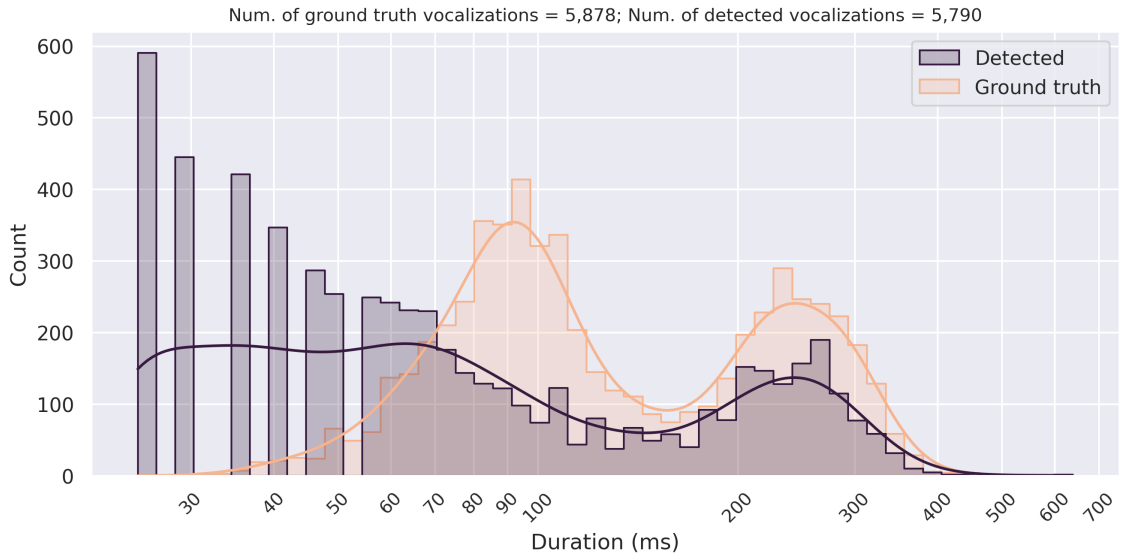
1. *How does AudibleSincNet detect vocalizations differently than the baseline model?*
2. *Why does AudibleSincNet outperform the baseline approach?*

6.1 Insights on the detected annotations

By analysing the distribution of both model detected vocalizations (Figures 6.1 and 6.2), one can infer that the baseline model detected annotations are in general smaller than expected. An intuition of how different the detected vocalizations by the spectrogram-based approach is expressed on the GKDE plot in Figure 6.1.

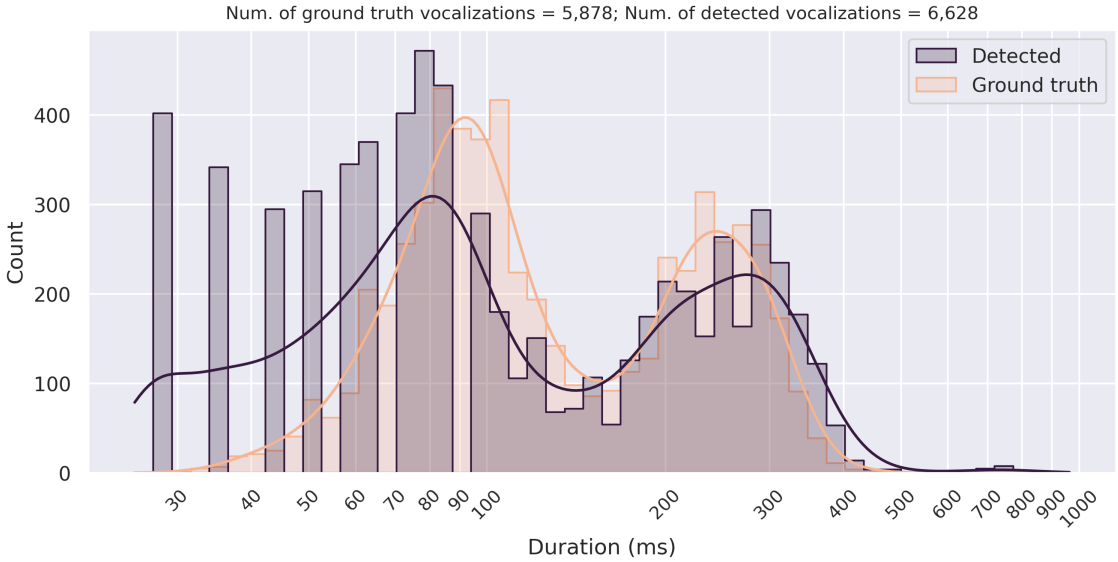
On the other hand, the distribution of the detected vocalizations by AudibleSincNet is closer to the ground-truth, which can be also be visualised by the GKDE plot on Figure 6.2. The proximity of the two distributions justifies the performance of AudibleSincNet over the baseline model. However, some of the detected vocalizations are above 500ms, which imply that several detected vocalizations are being merged together, which generates several false negatives (and consequently a poor performance when considering the number of detected vocalizations, as shown in Figure 5.5).

Figure 6.1: Distribution of automated vocalization detection by the baseline model.



Source: The Author

Figure 6.2: Distribution of automated vocalization detection by AudibleSincNet.



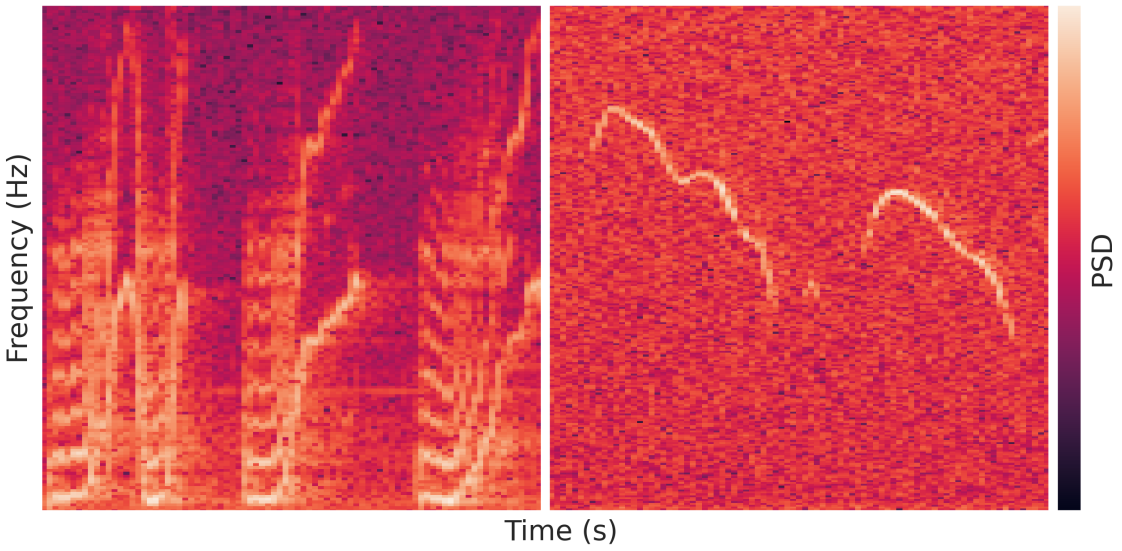
Source: The Author

6.2 AudibleSincNet vs Baseline applicability

The applicability of both of these methods differs since the spectrogram-based model is an adaptation of an unsupervised technique to detect mice USVs. Hence, it is fair to consider the differences between guinea pig and mice vocalizations to understand why AudibleSincNet outperforms the baseline approach.

Guinea pigs vocalise on the audible range. Because audible noise is more prevalent in daily life, vocalizations are harder to differentiate from the background noise, even with proper acoustic isolation (e.g., noise is produced by the animal when it hit the cage wall). On the other hand, mice USVs occur on the ultrasonic range, with few samples of noises like lamps, for example. Additionally, mice vocalizations are in general composed by single frequency modulations or a sequence of them, while guinea pig vocalizations have several harmonics with frequency modulations. Figure 6.3 shows an example of this morphological difference.

Figure 6.3: Difference between guinea pig vocalizations (on the left) and mice USVs (right).



Source: The Author

7 CONCLUSION & FUTURE WORK

This work proposes a Machine Learning solution for the detection of vocalizations of guinea pigs, using as a baseline a spectrogram-based approach that presents state-of-the-art results when considering the detection of mice USVs (STOUMPOU et al., 2021).

As presented, even when a trained AudibleSincNet performs well on the test set regarding single-window classification, this result is not transferred when evaluating the detection of vocalizations itself. Metrics such as pair accuracy are also introduced to evaluate the individual classifications based on their context. However, the intuition provided by this metric still does not map to the results given by the detection metric. Therefore, the detection of guinea pig vocalizations problem remains challenging in the detection task itself and strategies and metrics to evaluate potential models and approaches.

AudibleSincNet, the main approach proposed by this work, relies on the classification of single windows and does not consider their context. Temporal redundancy between two consecutive windows due to overlap is introduced. However, this does not seem to be enough. Methods that consider context, such as recurrent or transformers models, could introduce this feature and improve the results concerning the biological aspect.

Nonetheless, the development of this model with such generalist principles – from the biological point of view (i.e. not relying on spectrogram species specificities) – could also provide insights on the development of a species-agnostic vocalization detection approach. Therefore, future comparison with DeepSS (STEINFATH et al., 2021) is desirable since they have similar objectives.

REFERENCES

ALLEN, J. Short term spectral analysis, synthesis, and modification by discrete fourier transform. **IEEE Transactions on Acoustics, Speech, and Signal Processing**, IEEE, v. 25, n. 3, p. 235–238, 1977.

BA, J. L.; KIROS, J. R.; HINTON, G. E. Layer normalization. **arXiv preprint arXiv:1607.06450**, 2016.

BERRYMAN, J. C. Guinea-pig vocalizations: Their structure, causation and function. **Zeitschrift für Tierpsychologie**, Wiley Online Library, v. 41, n. 1, p. 80–106, 1976.

BHATTACHARYA, G.; ALAM, M. J.; KENNY, P. Deep speaker embeddings for short-duration speaker verification. In: **Interspeech**. [S.l.: s.n.], 2017. p. 1517–1521.

BOWLBY, J. The bowlby-ainsworth attachment theory. **Behavioral and Brain Sciences**, Cambridge University Press, v. 2, n. 4, p. 637–638, 1979.

BRUDZYNSKI, S. **Handbook of Ultrasonic Vocalization: A Window into the Emotional Brain**. Elsevier Science, 2018. (ISSN). ISBN 9780128097731. Disponível em: <https://books.google.com.br/books?id=KnFZDwAAQBAJ>.

COFFEY, K. R.; MARX, R. G.; NEUMAIER, J. F. Deepsqueak: a deep learning-based system for detection and analysis of ultrasonic vocalizations. **Neuropsychopharmacology**, Nature Publishing Group, v. 44, n. 5, p. 859–868, 2019.

FONSECA, A. H. et al. Analysis of ultrasonic vocalizations from mice using computer vision and machine learning. **eLife**, eLife Sciences Publications, Ltd, v. 10, p. e59161, mar 2021. ISSN 2050-084X. Disponível em: <https://doi.org/10.7554/eLife.59161>.

FOURIER, J. B. J. B. **The analytical theory of heat**. [S.l.]: The University Press, 1878.

HAN, J.; PEI, J.; KAMBER, M. **Data Mining: Concepts and Techniques**. Elsevier Science, 2011. (The Morgan Kaufmann Series in Data Management Systems). ISBN 9780123814807. Disponível em: <https://books.google.com.br/books?id=pQws07tdpjoC>.

HARLOW, H. F. Love in infant monkeys. **Scientific American**, JSTOR, v. 200, n. 6, p. 68–75, 1959.

HAYKIN, S. S.; VEEN, B. V. **Signals and systems**. 2.. ed. [S.l.]: Wiley, 2005. 1–802 p. ISBN 9780471707899.

HENNESSY, M. B. Filial attachment and its disruption: insights from the guinea pig. **Developmental psychobiology**, Wiley Online Library, v. 56, n. 8, p. 1747–1754, 2014.

HENNESSY, M. B. et al. Effects of peripherally administered corticotropin-releasing factor (crf) and a crf antagonist: Does peripheral crf activity mediate behavior of guinea pig pups during isolation? **Behavioral Neuroscience**, American Psychological Association, v. 109, n. 6, p. 1137, 1995.

HENNESSY, M. B.; RITCHEY, R. L. Hormonal and behavioral attachment responses in infant guinea pigs. **Developmental Psychobiology: The Journal of the International Society for Developmental Psychobiology**, Wiley Online Library, v. 20, n. 6, p. 613–625, 1987.

HOSHEN, Y.; WEISS, R. J.; WILSON, K. W. Speech acoustic modeling from raw multichannel waveforms. In: IEEE. **2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.], 2015. p. 4624–4628.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: PMLR. **International conference on machine learning**. [S.l.], 2015. p. 448–456.

NAGRANI, A.; CHUNG, J. S.; ZISSERMAN, A. Voxceleb: a large-scale speaker identification dataset. **arXiv preprint arXiv:1706.08612**, 2017.

NYQUIST, H. Certain Topics in Telegraph Transmission Theory. **Transactions of the American Institute of Electrical Engineers**, IEEE, v. 47, n. 2, p. 617–644, 1928. ISSN 00963860.

OORD, A. v. d. et al. Wavenet: A generative model for raw audio. **arXiv preprint arXiv:1609.03499**, 2016.

PALAZ, D.; COLLOBERT, R. et al. **Analysis of cnn-based speech recognition system using raw speech as input**. [S.l.], 2015.

PETTIJOHN, T. F. Social attachment of the infant guinea pig to its parents in a two-choice situation. **Animal Learning & Behavior**, Springer, v. 7, n. 2, p. 263–266, 1979.

PORTER, R. H.; BERRYMAN, J. C.; FULLERTON, C. On the nature of mother-infant interactions in the guinea-pig (cavia porcellus). **Behaviour**, Brill, v. 48, n. 1-4, p. 145–156, 1974.

PORTER, R. H.; FULLERTON, C.; BERRYMAN, J. C. Exploration and attachment behaviour in infant guinea pigs. **Behaviour**, Brill, v. 45, n. 3-4, p. 312–322, 1973.

RABINER, L.; SCHAFER, R. **Theory and applications of digital speech processing**. [S.l.]: Prentice Hall Press, 2010.

RAVANELLI, M.; BENGIO, Y. Speaker recognition from raw waveform with sincnet. In: IEEE. **2018 IEEE Spoken Language Technology Workshop (SLT)**. [S.l.], 2018. p. 1021–1028.

RICHARDSON, F.; REYNOLDS, D.; DEHAK, N. A unified deep neural network for speaker and language recognition. **arXiv preprint arXiv:1504.00923**, 2015.

RITCHEY, R. L.; HENNESSY, M. B. Cortisol and behavioral responses to separation in mother and infant guinea pigs. **Behavioral and neural biology**, Elsevier, v. 48, n. 1, p. 1–12, 1987.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. Pearson, 2019. (Pearson series in artificial intelligence). ISBN 9780134610993. Disponível em: <https://books.google.com.br/books?id=koFptAEACAAJ>.

SCOTT, D. W. **Multivariate density estimation: theory, practice, and visualization**. [S.l.]: John Wiley & Sons, 2015.

SEGBROECK, M. V. et al. Mupet—mouse ultrasonic profile extraction: a signal processing tool for rapid and unsupervised analysis of ultrasonic vocalizations. **Neuron**, Elsevier, v. 94, n. 3, p. 465–485, 2017.

SHANNON, C. E. Communication in the Presence of Noise. **Proceedings of the IRE**, IEEE, v. 37, n. 1, p. 10–21, 1949. ISSN 00968390.

SNYDER, D. et al. Deep neural network embeddings for text-independent speaker verification. In: **Interspeech**. [S.l.: s.n.], 2017. p. 999–1003.

SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. **The journal of machine learning research**, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014.

STEINFATH, E. et al. Fast and accurate annotation of acoustic signals with deep neural networks. **bioRxiv**, Cold Spring Harbor Laboratory, p. 2021.03.26.436927, mar 2021. Disponível em: <https://www.biorxiv.org/content/10.1101/2021.03.26.436927v1https://www.biorxiv.org/content/10.1101/2021.03.26.436927v1.abstracthttps://doi.org/10.1101/2021.03.26.436927>.

STEVENS, S. S.; VOLKMANN, J.; NEWMAN, E. B. A scale for the measurement of the psychological magnitude pitch. **The journal of the acoustical society of america**, Acoustical Society of America, v. 8, n. 3, p. 185–190, 1937.

STOUMPOU, V. et al. Analysis of Mouse Vocal Communication (AMVOC): A deep, unsupervised method for rapid detection, analysis, and classification of ultrasonic vocalizations. **bioRxiv**, Cold Spring Harbor Laboratory, p. 2021.08.13.456283, aug 2021. Disponível em: <https://www.biorxiv.org/content/10.1101/2021.08.13.456283v1https://www.biorxiv.org/content/10.1101/2021.08.13.456283v1.abstract>.

VARIANI, E. et al. Deep neural networks for small footprint text-dependent speaker verification. In: IEEE. **2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)**. [S.l.], 2014. p. 4052–4056.

WEWERS, D.; KAISER, S.; SACHSER, N. Maternal separation in guinea-pigs: A study in behavioural endocrinology. **Ethology**, Wiley Online Library, v. 109, n. 5, p. 443–453, 2003.

ZHANG, C.; KOISHIDA, K.; HANSEN, J. H. Text-independent speaker verification based on triplet convolutional neural network embeddings. **IEEE/ACM Transactions on Audio, Speech, and Language Processing**, IEEE, v. 26, n. 9, p. 1633–1644, 2018.

ZÖLZER, U. **Digital Audio Signal Processing**. 2.. ed. [S.l.]: Wiley, 2008. 1–320 p. ISBN 9780470997857.

APPENDIX A — TESTED HYPER-PARAMETERS

A.1 AudibleSincNet

- *Learning rate*: [0.01, 0.001, 0.0001].

A.2 Baseline

- *Temporal-based threshold* $\mathbf{t}$: [0.1, 0.35, 0.7];
- *Frequency-based threshold* $\mathbf{f}$: [2.0, 3.5, 5.0].

A.3 Post-processing

- *Smoothing filter size*: [5, 10, 20, 30] ms.

APPENDIX B — TECHNOLOGIES USED

B.1 AudibleSincNet

- Python
  - Matplotlib
  - numpy
  - PyTorch
  - PyTorch-Lightning
  - Pandas
  - scipy.signal
  - Seaborn

B.2 Baseline model

- Python
  - Matplotlib
  - numpy
  - scipy.signal
  - Seaborn

B.3 Post processing

- Python
  - numpy