Analysis of Mouse Vocal Communication (AMVOC): a deep, unsupervised method for rapid detection, analysis and classification of ultrasonic vocalisations

Vasiliki Stoumpou\textsuperscript{a}, César D. M. Vargas\textsuperscript{b, c}, Peter F. Schade\textsuperscript{b, c}, J. Lomax Boyd\textsuperscript{d}, Theodoros Giannakopoulos\textsuperscript{e, f} and Erich D. Jarvis\textsuperscript{b, f}

\textsuperscript{a}School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece; \textsuperscript{b}Laboratory of Neurogenetics of Language, The Rockefeller University, New York, NY, USA; \textsuperscript{c}Laboratory of Neural Systems, The Rockefeller University, New York, NY, USA; \textsuperscript{d}Berman Institute of Bioethics, Johns Hopkins University, Baltimore, MD, USA; \textsuperscript{e}Computational Intelligence Lab, Institute of Informatics and Telecommunications, National Center of Scientific Research 'Demokritos', Athens, Greece; \textsuperscript{f}Howard Hughes Medical Institute, Chevy Chase, MD, USA

**ABSTRACT**

Some aspects of the neural mechanisms underlying mouse ultrasonic vocalisations (USVs) are a useful model for the neurobiology of human speech and speech-related disorders. Much of the research on vocalisations and USVs is limited to offline methods and supervised classification of USVs, hindering the discovery of new types of vocalisations and the study of real-time free behaviour. To address these issues, we developed AMVOC (Analysis of Mouse VOcal Communication) as a free, open-source software to detect and analyse USVs. When compared to hand-annotated ground-truth USV data, AMVOC’s detection functionality (both offline and online) has high accuracy and outperforms leading methods in noisy conditions. AMVOC also includes an unsupervised deep learning approach that facilitates discovery and analysis of USV data by training a model and clustering USVs based on latent features extracted by a convolutional autoencoder. The clustering is visualised in a graphical user interface (GUI) which allows users to evaluate clustering performance. These results can be used to explore the vocal repertoire space of individual animals. In this way, AMVOC will facilitate vocal analyses in a broader range of experimental conditions and allow users to develop previously inaccessible experimental designs for the study of mouse vocal behaviour.

**1. Introduction**

Over the past two decades, there has been a growing interest in the usage and signalling of vocalisations in mice, with large efforts going into studying the underlying neurobiological mechanisms for auditory processing (Pomerantz et al. 1983; Liu et al. 2003; Holy and Guo 2005; Neilans et al. 2014; Perrodin et al. 2020) and the production of vocalisations (Arriaga et al. 2012; Chabout et al. 2016; Okobi et al. 2019; Zimmer et al. 2019; Gao...
et al. 2019; Tschida et al. 2019; Michael et al. 2020). The tools available for experiments in mice provide a promising model for studying the neural basis of vocalisations, as well as the effects of genes on the origin and development of vocal neural anatomy (Grimsley et al. 2011; Michael Bowers et al. 2013; Chabout et al. 2016; Tabler et al. 2017).

Mice produce ultrasonic vocalisations (USVs; 30–110 kHz) relative to the human hearing range (2–20 kHz). Both sexes of mice produce USVs from an early natal age through adulthood (Grimsley et al. 2011). Efforts have been made to better understand these USVs in terms of their structure, classifying their multi-syllabic structure, as well as the non-random sequencing of syllables (Holy and Guo 2005; Chabout et al. 2016; Calbick et al. 2017; Castellucci et al. 2018; Hertz et al. 2020). USVs exhibit considerable variation across inbred strains, such that innate USV repertoires can be used as a phenotyping marker for different genotypes (Melotti et al. 2021), and as a behavioural readout for genetic modifications and speech-related mutations (Luisa Scattoni et al. 2009; Chabout et al. 2016; Castellucci et al. 2016). USVs can also vary within individuals based on social environment and affective state. For example, adult male mice modify the temporal organisation and the rate of vocalisations for different social contexts (Chabout et al. 2015). In order to advance our understanding of the neurobiology of vocalisations, it is important to more precisely understand the vocalisations themselves as units of behaviour.

Historically, vocal research was done by hand-annotating spectrograms. Successive technological improvements have led to many new automated methods that allow for unsupervised detection and classification of vocalisations.

These advances in vocal detection include tools developed specifically for analysing the USV repertoire of mice. Early tools used pre-determined classification methods that rely explicitly on the acoustic parameters of the recordings. One such tool is Mouse Song Analyser (MSA) (Holy and Guo 2005; Arriaga et al. 2012; Chabout et al. 2015). MSA first generates a spectrogram from the audio recording, that is subsequently thresholded to remove white noise. Frequencies outside of mouse USV range of 35–125 kHz are discarded. The spectrogram is used to compute frequency, amplitude, spectral purity, and discontinuity across the entire recording. A combination of user-defined thresholds for each of these features is used to detect USVs. Lastly, a duration filter is applied to remove any detected sounds that fall below a predefined value, and all remaining detections are considered USVs. The detected USVs are then classified based on the number of gaps, or ‘pitch jumps’, that are present within the detected USVs (Chabout et al. 2015). USVSEG (Tachibana et al. 2020) is also a MATLAB tool used for vocalisation detection, emphasising noise removal of the spectrogram in the cepstral domain before thresholding to detect whether a segment contains a vocalisation or not. Another tool is Axe (Neunuebel et al. 2015), which seeks to detect vocal signals by keeping time-frequency points of the spectrogram that significantly exceed noise values.

As new methods have arisen for the unsupervised assessment of USVs (Van Segbroeck et al. 2017), some have implemented machine learning and/or neural networks in their processes (Coffey et al. 2019; Fonseca et al. 2021). Machine learning methods have been used to improve detection accuracy, but performance is limited by how well a network is trained. Thus, pretrained networks may not generalise well to other experiments, making comparisons difficult to interpret.
This includes Mouse Ultrasonic Profile ExTraction (MUPET), a MATLAB open-source tool, developed by Van Segbroeck et al. (2017) to detect USV syllables, analyse the vocalisation features and cluster the syllables depending on these features. MUPET first filters the signal to keep high frequencies (25–125 kHz). It then uses spectral subtraction to remove stationary noise, and at last, it computes the power of the spectral energy in the ultrasonic range above a specific threshold. The vocalisations are converted to representations by using negative matrix factorisation (NMF) and gammatone filters. The filtered spectrograms are then used to cluster vocalisations based on spectral shape similarities. The clustering is done by using K-Means and user-defined number of ‘repertoire units’ (clusters). The authors note that MUPET can also be used with many non-rodent species’ vocalisations.

DeepSqueak is a software suite for USV detection and analysis (Coffey et al. 2019). It splits the recording into areas of interest, computes the corresponding spectrograms and passes them to a Faster-RCNN (Recurrent Convolutional Neural Network) object detector, which consists of two networks. The first network is a region detection network, which proposes sections of the spectrogram that could contain actual vocalisations. These sections are then used as inputs to a second network, a convolutional neural network (CNN), which classifies the content of that section as vocalisation or noise. This process has been recently updated to use a You Only Look Once (YOLO) network to improve detection quality. These networks can also be retrained with new vocalisations. DeepSqueak can also be used for clustering the detected syllables, either with a supervised or unsupervised method. Their unsupervised approach gives the user the opportunity to define three weighted input features: shape, frequency and duration of the vocalisation. An important difference between DeepSqueak and MUPET is that the clustering function of MUPET takes syllable amplitude into account, whereas Deep-Squeak does not. This can be considered an advantage of DeepSqueak, given that the amplitude of a vocalisation can depend on the recording setup itself, among other factors.

In another recent approach, Premoli et al. (2021) compared different USV classification methods, using supervised learning with 10 predefined classes, as proposed in (Luisa Scattoni et al. 2008). The team tested the performance of different methods, including Multi-layer Perceptrons, a Convolutional Neural Network (CNN), Support Vector Machines (SVMs) and Random Forests. The authors showed that CNN, which exploits time and frequency information from the whole spectrogram, outperformed the other machine learning methods that considered only a subset of spectrogram-derived numerical features.

VocalMat is a MATLAB tool (Fonseca et al. 2021), which uses image processing techniques and differential geometry analysis on the spectrogram of a recording to detect vocalisation candidates. It can then classify detected USVs into 12 predefined categories (including noise), by using a CNN. As with DeepSqueak, VocalMat’s networks can be retrained as well.

A more recent tool is Deep Song Segmenter (DeepSS), which has been used for annotation of songs of mice, birds and flies (Steinfath et al. 2021). DeepSS learns a representation of sound features directly from raw audio recordings using temporal convolutional networks (TCNs), based on dilated convolutions. It is a comparatively fast, supervised annotation method since the network is trained with manually annotated recordings. It can also be combined with unsupervised approaches to reduce the amount of manual annotation required.
There are, however, limitations to each of these approaches. The supervised classification methods are limited in their ability to analyse datasets for changes or novelty in the USV repertoire across individuals and genotypes. Predefined categories have long been used, but it has remained uncertain whether such constructed categories are ethologically relevant. Another technical hurdle that is not addressed by existing methods is the ability to perform analyses in real time, which would facilitate closed-loop experimental designs such as those used in songbirds (Sober and Brainard 2012; Xiao et al. 2018), which are currently inaccessible for mouse vocal behaviour. To address these pitfalls, we sought to create a tool for mouse USVs that is both computationally efficient and accurate and that can provide a less biased classification of USVs, without being restricted by predefined patterns imposed by humans.

In this work, we present Analysis of Mouse Vocal Communication (AMVOC), a new open-source tool for real-time mouse USV research. AMVOC’s purpose is twofold: (a) it uses dynamic spectral thresholding to detect the presence of USVs in audio recordings both offline and online; and (b) similar to other machine learning approaches, it analyses and visualises the detected USVs using feature representations extracted from deep convolutional autoencoders, and uses these feature representations for clustering and data exploration. AMVOC’s ability to detect USVs was extensively evaluated using real recordings. AMVOC’s accuracy outperformed most state-of-the-art tools in various acoustic environments, allowing for more flexible experimental set-ups. In addition, while many of the other USV detection tools available are specifically developed for offline analysis of vocalisations, AMVOC is unique in that it can detect and measure USVs in real-time with an accuracy that rivals the accuracy of offline approaches and detection speeds at behaviourally relevant timescales. Lastly, the proposed deep feature extraction technique, using a convolutional autoencoder, produces unsupervised USV classifications that can be used as a basis to discover biologically relevant USV clusters that previously introduced methods may have missed. This provides an unparalleled usage that opens up new avenues to better understand mouse vocal behaviour and its associated neurobiology.

2. Materials and methods

2.1. Laboratory setup and dataset

We used recordings from two sources: a) from (Chabout et al. 2015), recorded at Duke University. The recordings are publicly available on mouseTube (Torquet et al. 2016); and b) mice recorded for this study, recorded at The Rockefeller University.

2.1.1. Animals

The mice in (Chabout et al. 2015) were adult B6D2F1/J mice purchased from Jackson Laboratory. For the mice at Rockefeller, adult C57BL/6J mice were purchased from Jackson Laboratory and maintained in a colony bred and maintained at The Rockefeller University. Before recording the experiments, all mice were group housed (2–5 per cage) and kept on a 12-h light/dark cycle and received ad-libitum food and water. All procedures were approved by the Institutional Animal Care and Use Committee of The Rockefeller University.
2.1.2. Vocal behavioural recordings

For audio recordings, we used procedures similar to those previously described by Chabout et al. (2015, 2016, 2017). Briefly, before being used for recordings, adult male mice (>8 weeks old) were sexually socialised overnight with a sexually mature female mouse. The next morning, the female mouse was removed and returned to her home cage. Prior to the vocal recording experiments, the experimental male was removed from his home cages and acclimated in a clean cage with no bedding (to reduce noise) inside the recording chamber for 10 min. After this acclimation period, either an adult female mouse (different from the one he was sexually socialised with) was introduced into the cage (recordings referred to as FemaleLiveDirected), an anaesthetised adult female (recordings referred to as AnaesthetisedFemaleDirected) or urine from a female mouse was placed in the cage (recordings referred to as FemaleUrineDirected). After the female or urine stimulus was introduced, the recording was started. Recording sessions lasted up to 5 min. For the mouse pup recording, the mouse was removed from its home cage and placed in a weigh boat that was located within the recording chamber. The pup’s age was at postnatal day 7 (P7). For this recording, the microphone was moved closer to the mouse than in adult recordings. The 5-min recording was initiated immediately after placing the mouse in the chamber and closing the door. After the recording period was completed, the animals were returned to their home cages. The recording chamber is a 15” × 24” × 12” beach cooler which acts as an acoustically insulated environment. No lighting was provided inside the cooler, as mice are nocturnal and vocalised more in the dark. A microphone was hung from the top of the chamber at ~15 cm from the bottom of the cage (~10 cm for pups). To challenge the detection capacity of AMVOC, some recordings were made intentionally noisy, increasing the spectral energy in lower frequencies, by using Corn Cob (Lab Supply, Fort Worth, TX) or Cellu-nest™ (Shepherd Speciality Papers, Watertown, TN) bedding on the bottom of the cage. For real-time detection, we used an Ultramik 384K BLE ultrasonic microphones (Dodotronics, Italy), sampling at 384 kHz, connected to a Raspberry Pi 4 Model B (8GB RAM) and monitored vocalisations via live spectrograms using the UltraSoundGate CM16/CMPA recording system (Avisoft Bioacoustics®, Berlin, Germany).

2.1.3. Datasets

We created four vocal datasets (D1, D2, D3 and D4). Dataset D1 was created to evaluate and compare vocalisation detection methods, and contains a combination of recordings from both B6D2F1/J mice and C57BL/6J mice. Specifically, we compiled a ground-truth dataset of 9 audio segments of 5–10 s each, containing 245 syllables in total from 14 different male mice. A manual ground-truth annotation was performed by a domain expert by simply declaring the frames that correspond to actual vocalisations, with a time resolution of 1 ms. The recordings used, along with the annotations and instructions on how to reproduce the results, are openly available and can be found in the AMVOC repository https://github.com/tyiannak/amvoc/tree/master.

Dataset D2 consists of 26 different recordings from 9 different male mice (from both B6D2F1/J and C57BL/6J strains), used as the training set of our convolutional auto-encoder, and explained in Section 3.3.2. These recordings can be found under https://drive.google.com/drive/folders/14l-zJmXcjSR9cucnq8lwUnsSIRPxe5 and https://drive.google.com/drive/folders/1M976oaxiMpefN9dfm5Kd5kHeGliNhbw.
Dataset D3 was created for the experimental evaluation of the clustering configurations (explained in Section 3), containing recordings from B6D2F1/J mice. We used 72 behavioural recordings, explained in 3.1.2, 36 in the category FemaleUrineDirected and 36 in the category FemaleLiveDirected. These recordings come from 12 different male mice. We have randomly selected 20 s from each recording, where the vocalisation rate is at least 2.5 vocalisations/sec. We then concatenated the 20-s interval from each recording to a new file. We generated four files, two from the FemaleUrineDirected category and two from the FemaleLiveDirected category. These recordings can be found under https://drive.google.com/drive/folders/1l7qUw0SVvd1dzNrr35FT7XOxbxhDp7Kqn.

Dataset D4 was created as a testing dataset to better evaluate the performance of AMVOC and other detection methods and minimise any bias that could exist when testing on datasets with which AMVOC or other methods were developed. This dataset consists of six different recordings from C57BL/6J mice: five recordings are from adult male mice recorded in three different social contexts and one from a P7 pup (see Section 3.1.2 above). The recordings used, along with the ground truth annotations, can be found under https://drive.google.com/drive/folders/1621UaXEGaY2HWT35XBo3-qY2zwCWZbBX.

2.2. Detection of mouse vocalisations

The first step of AMVOC’s processing pipeline is to detect mice USVs. Given an input recording, we want to determine the parts of the signal that contain a USV. Below are methods for offline and online detection.

2.2.1. Offline USV detection

To detect mouse USVs, we first compute a spectrogram of the whole recording. This is done by splitting the signal into non-overlapping short-term windows (frames) of duration $w = 2$ ms (time resolution) and calculating the Short-Term Fourier Transform (STFT) for each time frame. Frequency resolution $f_r$ is calculated as:

$$f_r = \frac{1}{w}$$  \hspace{1cm} (1)

This means that in our case, if $w = 2$ ms, $f_r = 0.5$ kHz. In simple terms, the ability of the 2-dimensional (time & frequency) spectrogram representation to discriminate between different frequency coefficients is 0.5 KHz (i.e. frequency resolution). This is less fine-scaled than is typical for human speech analytic methods, but because mice vocalisations are found within the frequency range of 30–110 kHz and each vocalisation can individually span many kHz, this resolution is more than sufficient.

As soon as the spectrogram is extracted, the USVs are detected on a time frame-basis, using two separate criteria, time-based thresholding (TT; criterion 1) and frequency-based thresholding (FT; criterion 2), that take into account the values of the distribution of the signal’s energy at the different frequencies (Figure 1(a)).

Both of these criteria are based on the spectral energies, however they differ in the way the thresholding criteria are calculated and applied. The details of the two criteria are as follows:
- Time-based thresholding (TT), criterion 1: This involves a simple temporal thresholding of the spectral energy values. To do this, for each time frame, we calculate the spectral energy by summing the energy value at each frequency. We do this procedure for the frequency range of interest, which is, as mentioned above, from 30 kHz to 110 kHz. If we denote the spectrogram value at time frame \( i \) and frequency \( j \) as \( E_{ij} \), spectral energy \( S_i \) is calculated as:

\[
S_i = \sum_{j=30kHz}^{110kHz} E_{ij}
\]  

(2)
where the step of \( j \) is equal to 0.5 kHz. We then compute a dynamic sequence of thresholds for the spectral energy. In particular, for each frame \( i \), for which we have extracted the spectral energy \( S_i \), we compute the dynamic threshold:

\[
T_i = \frac{1}{2} \frac{\sum_{j=0}^{N-1} S_j}{N} + \frac{1}{2} \frac{\sum_{j=0}^{K-1} S_{i-j}}{K}
\]

(3)

where \( N \) is the number of time frames, so the first term refers to the mean spectral energy. In the second term, \( K \) is the size of a moving average filter in time frames. Here we use a 2-s-long filter \( (K = \frac{2s}{w} = 100 \text{ time frames}) \), which is convolved with the sequence of spectral energy values. In other words, the dynamic threshold \( T_i \) is defined at each frame \( i \) as the average of the current spectral energy \( (S_i) \) and the moving average of the spectral energies of the last \( K \) frames.

- Frequency-based thresholding (FT), criterion 2: This second criterion is associated with applying a thresholding rule, based on the per-frame distribution of energies on the different frequencies. A simple dynamic threshold at each time frame of the spectrogram (as described above) is not enough because there are also time frames where high spectral energy occurs due to noise.

The spectral energy value in these high-noise time frames may surpass the threshold, but this does not correspond to any vocalisation (Figures 1(a), 2, 3, and 4). Our goal is to filter out these false-positive vocalisations. It is easy to observe that in the vast majority of cases, noise appears as high-energy values, spread across a large frequency range in each time frame (e.g. Figures 1(a) and 3(b)), compared to the time frame energy distribution in vocalisations, which is concentrated in a small frequency range in each time frame (e.g. Figures 1(a) and 3(a)). Our filtering criterion was to keep only time frames where the peak energy value \( P_i \) is larger than the mean spectral energy \( (M_i) \) of a 60 kHz range around the frequency of the peak energy (truncated if the range goes below 30 kHz or above 110 kHz). If we denote the energy value at time frame \( i \) and frequency \( j \) as \( E_{ij} \), the equations describing the two quantities above, are the following:

\[
P_i = \max_{j=30, \ldots, 110 \text{kHz}} E_{ij}
\]

(4)

\[
M_i = \frac{1}{N_f} \min_{j = \max(p_i - 30 \text{kHz}, 30 \text{kHz})} \sum_{j = \max(p_i + 30 \text{kHz}, 110 \text{kHz})} E_{ij}
\]

(5)

where the step of \( j \) is equal to 0.5 kHz and, as a result, \( N_f = 2 \cdot (\min(p_i + 30 \text{kHz}, 110 \text{kHz}) - \max(p_i - 30 \text{kHz}, 30 \text{kHz})) \), and \( p_i \) is the frequency of the peak energy at time frame \( i \), i.e. \( p_i = \arg \max(E_{ij}) \).

Both criteria TT and FT are applied to each short-term frame \( i \) as follows: the threshold conditions require that the spectral energy is higher than 50% of the dynamic threshold computed in step 1 and that the maximum energy is larger than the mean spectral energy by a factor of 3.5. Let \( V \) be a sequence of frame-level vocalisation decisions, i.e. \( V_i = 1 \) if time frame \( i \) is part of a vocalisation and \( V_i = 0 \) if not (Figure 2(a)). Then, the above rule can be expressed as follows:
Both weights $t$ and $f$ have been selected after experimentation and can be considered as configurable (see Section 2.2.3).

After the twofold thresholding rule has been applied as described above, we apply a smoothing step. Specifically, after the sequence $V$ of 1 s and 0 s occurs (Figure 2(a)), this sequence is smoothed using a box moving average filter with a duration of 20 ms ($L = \frac{20}{W} = 10$ time frames) (Figure 2(b)), so that the neighbourhood of the possible vocalisation is taken into account:

$$V_i = \begin{cases} 1 & \text{if } (S_i > t \cdot T_i) \ AND \ (P_i > f \cdot M_i) \text{, where } t = 0.5 \text{ and } f = 3.5 \\ 0 & \text{else} \end{cases}$$

(6)

Both weights $t$ and $f$ have been selected after experimentation and can be considered as configurable (see Section 2.2.3).

After the twofold thresholding rule has been applied as described above, we apply a smoothing step. Specifically, after the sequence $V$ of 1 s and 0 s occurs (Figure 2(a)), this sequence is smoothed using a box moving average filter with a duration of 20 ms ($L = \frac{20}{W} = 10$ time frames) (Figure 2(b)), so that the neighbourhood of the possible vocalisation is taken into account:

$$F_i = \frac{\sum_{j=0}^{L-1} V_{i-j}}{L}$$

(7)

where $L$ is the size of the filter in time frames. A time frame $i$, for which $F_i > 0$ indicates the detection of a vocalisation. As a final step, if the successive positive frames found in $F$ are separated by <11 ms, they are concatenated to form segments of mice vocalisations (Figure 2(c)). Vocalisations of duration <5 ms are filtered out, since practical evaluation showed that in most cases these are false-positive detections.

2.2.2. Online USV detection

In addition to the offline detection that is standard for most USV analyses, we developed an online version of AMVOC using streaming sound recorded from the computer’s sound card. This is achieved by following the aforementioned analysis steps, though these cannot be applied to the whole signal at once, since, in a real-time setup, the signal is recorded simultaneously with the detection procedure. For this process, we needed the
processing interval to be as small as possible, in order to detect the vocalisations fast. Smaller processing frames come with other issues: 1) processing the signal too often increases the probability of cutting a vocalisation in the border between two successive blocks; 2) the signal statistics that need to be calculated for the detection steps described in the preceding section will become less robust if they are computed on smaller segments. We therefore chose to process the signal in blocks of a fixed duration, which for our case we set to 750 ms. We estimated the number of USVs that could occur per processing window by running 20 recordings from live female and 20 recordings from female urine social contexts by processing the files with the online computations of AMVOC in an offline format. In order to address the first of the issues with small processing frames and to prevent USVs from being separated, we always process the newest 750 ms block with the last 100 ms of the previous block. In this way, we add a minor computational delay (as we repeat the process for 13% of the data), but we manage to eliminate the errors caused by USVs being split between two successive blocks, and therefore lost by the detection method. A 750 ms window with buffer provides a long enough period for high accuracy detection and minimally interrupting the number of individual syllables that would be processed (Figure 3A and B). AMVOC requires about

![Figure 3](image-url)

**Figure 3.** Effect of different weights on F1 event and temporal scores. a and b) Demonstrates the effect of different weights for \( t \) (with constant \( f = 3.5 \)) and \( f \) (with constant \( t = 0.5 \)) on temporal and event F1 scores (blue and orange lines, respectively) during offline detection. c and d) Demonstrates the effect of different weights for \( t \) (with constant \( f = 3.5 \)) and \( f \) (with constant \( t = 0.5 \)) on temporal and event F1 scores (blue and orange lines, respectively) during online detection. Our choice of \( t = 0.5 \) and \( f = 3.5 \) in both detection modes are highlighted in red. These parameters were selected since each of these achieved the highest event F1 score, without compromising much of the temporal F1.
5 ms to process a 750 ms window (Figure S1C). This processing time is not appreciably impacted by the number of USVs per window, and appears constant across tested files (Figure S1D and E).

The main algorithmic difference between the online and the offline detection is the calculation of the dynamic threshold. If we denote \( k \) as the current block, the dynamic threshold is the same for all frames belonging to that block and it is computed as follows:

\[
T_k = 0.3 \frac{\sum_{j=1}^{k} B_j}{k} + 0.7 \cdot B_k
\]

(8)

where \( B_k \) is the mean spectral energy of block \( k \) in the 30–110 kHz frequency range (as described in Section 2.2.1). In simple terms, the block’s threshold is computed as the weighted average of the mean of the spectral energies of the blocks recorded up to that point and the current block’s spectral energy. The weights (0.3, 0.7) were chosen after extensive experimentation (Figure S2). The sequence of thresholds that occurs is then multiplied by the threshold weight percentage \( t \), as in the offline method (Section 2.2.1).

2.2.3. Vocalisation detection configuration

As described above, the two parameters which determine the vocalisation detection procedure are the threshold weight percentage \( t \) and a factor \( f \). Factor \( f \) is a threshold such that the energy of a 60 kHz area \( M_i \) around the frequency of peak energy \( p_i \) must surpass peak energy \( P_i \) by the factor \( f \). These two parameters are adjustable, and the user can change their values according to the expected recording conditions and application requirements. Parameters used in our current study were optimised to include small events while minimising false positives. To select these parameters, we used Dataset D1 (Figure 3). As expected, increasing either of these parameters results in a more strict thresholding, which means that the precision increases, and recall decreases. The
opposite is observed when either of the parameters is reduced. From a more qualitative point of view, increasing the threshold might result in splitting a vocalisation with relatively low peak energy in intermediate time frames. On the other hand, a very low threshold can merge successive vocalisations. Based on our experimental goals, it was more essential to maximise event detection over the accurate detection of the temporal boundaries of detected vocalisations. The values of $t = 0.5$ and $f = 3.5$ provided the best optimisation for maintaining high event F1 scores (Figure 3).

2.2.4. Vocalisation detection evaluation
For evaluation and comparison, we adopted two performance metrics, namely the temporal F1 and the event F1 score:

- For the temporal F1 score, we interpret the vocalisation detection as a classification task of each 1 ms time frame into two categories (vocalisation or no vocalisation). We then calculate the precision and recall rates by comparing the detected vocalisations to the ground-truth vocalisations. Their harmonic mean is the temporal F1 score.
- The event F1 score is the harmonic mean of the two following fractions: (a) the number of events a method detected that are annotated in the ground-truth data by the number of events detected by a method (i.e. precision), and (b) the number of events annotated in ground-truth data that are detected with a method by the number of events of ground-truth annotations (i.e. recall).

2.3. Deep unsupervised learning for mouse vocalisation clustering
We next developed a method for unsupervised clustering of the detected vocalisations, by using a convolutional autoencoder, from which features are derived.

2.3.1. Convolutional autoencoders
2.3.1.1. Autoencoders. Autoencoders are neural networks that are trained to attempt to reproduce their input as their output. An autoencoder consists of three components: encoder, code and decoder. The encoder compresses the input and produces the code, and the decoder then reconstructs the input using only this code. The encoder is described by a function $h = f(x)$, where $x$ is the input and $h$ is the code and the decoder produces a reconstruction $r = g(h)$. The encoder maps the input to a representation that is assumed to contain the most important information (features) of the input. The decoder then constructs the output from this intermediate representation, also referred to as latent-space representation.

By training the autoencoder to perform the input reproduction task, we hope that the latent-space representation $h$ will be an efficient representation of the input and will contain important information about the input training set. One way to obtain useful features from the autoencoder is to constrain $h$ to have a smaller dimension than $x$. An autoencoder whose code dimension is less than the input dimension is called under-complete. Learning an undercomplete representation guides the autoencoder to capture the most salient features of the training data (Goodfellow et al. 2016). The learning process is described simply as minimising a loss function $L(x, g(f(x)))$, where $L$ is a loss.
function penalising $g(f(x))$ for being dissimilar from $x$. Commonly, the encoder and the decoder are feedforward neural networks, whose task is to learn the function $f$ and $g$, respectively. In our case, where the inputs are images, the encoder and the decoder are convolutional neural networks.

### 2.3.1.2. Convolutional neural networks

CNNs are networks that process data with a known grid-like topology, like image data. They are named after the mathematical operation convolution, which they employ. Their fundamental characteristics, which we make use of, are:

- **2D convolution**: The goal of a 2D convolution is to produce the activation map. The convolution operation takes place at every location $(x, y)$ of the image, taking into account a neighbourhood of this location. This neighbourhood is called receptive field and selects a region of pixels of the image, around $(x, y)$. A set of weights, arranged in the same shape as the receptive field and known as a kernel, slides along with the receptive field over the image and, for each location $(x, y)$, the image region defined by the receptive field and the kernel is convolved. The weights of the kernel are the same for all receptive fields of the image since we want this specific kernel to be responsible for the detection of one specific feature at all locations of the image. As a result, the 2D convolution contributes to learning the local characteristics of the image. Each kernel is convolved with the whole image and produces a convolutional activation map. After the convolution, we will have activation maps of depth equal to the number of the filters applied.

- **Activation function**: The output of the convolutional layer is fed to an activation function, whose purpose is to increase the non-linearity of the activation map. There are plenty of activation functions that can be used. One of the most popular is ReLU, which introduces non-linearity by keeping all the positive inputs unchanged and setting all negative inputs to zero. This operation is important, because it enables the model to learn more complex features, since real images are generally non-linear.

- **Pooling**: It is performed at the output of the ReLU activation function and reduces the dimensionality of the activation maps. This procedure is important, because the computational load of the next layers of the network is reduced, since the width and height of the convolutional activation maps is reduced. It also achieves translational invariance, meaning that if a small translation occurs, most of the pooled values will not change. This property is important, especially if we are interested in whether a specific feature is present in the image, rather than where exactly it is. By reducing the size of the activation maps after every convolution, we ensure that the intermediate representation will indeed have lower dimensions compared to the initial image.

### 2.3.1.3. Application of the autoencoder

Autoencoders are powerful tools for dimensionality reduction or feature learning. Applications of undercomplete autoencoders include compression, recommendation systems and outlier detection. Convolutional autoencoders are frequently used in image compression and denoising. They may also be used
in image search applications since the latent representation is often thought to hold semantic representations. Recently, theoretical connections between autoencoders and latent variable models have brought autoencoders to the forefront of generative modelling. More specifically, the variational autoencoder is a generative model which can be trained and used to generate images (Kingma and Welling 2014; Goodfellow et al. 2016).

In our case, the purpose of adopting a convolutional autoencoder is to create more computationally informative feature extractors of the mouse vocalisation spectrograms than is possible with supervised approaches. The inputs of the autoencoder are 2D images (spectrogram parts) that contain a vocalisation, and our goal is to obtain an intermediate representation (code $h$), in order to use it as a feature vector that will describe each image. From this representation, an image will be reconstructed with the decoder, and the output will be compared to the input in each step in order to calculate the loss function. The structure and the training of our autoencoder are described below.

### 2.3.2. Autoencoder architecture and training

To train the autoencoder, we used Dataset D2 (see Section 2.1.3) and we calculated the spectrogram for each of its recordings. This dataset contained 22,409 detected syllables. Each vocalisation was represented by a spectrogram in the detected time interval and defined frequency range. Therefore, these spectrograms vary in width, which corresponds to the respective syllable duration. So, we have to specify the width of the images that we are going to feed to the autoencoder, since the frequency y-axis is the same for all spectrograms: 80 kHz range (from 30 to 110 kHz) at a 0.5 kHz frequency resolution, giving a dimension of 160. Selecting a fixed-sized time dimension for our spectrograms requires taking into consideration a trade-off between losing important temporal information from the larger spectrograms (if we crop them) and reducing the importance of the shape and details of the smaller spectrograms (if we zero-pad them), which are more numerous than large spectrograms (Figure 4).

To decide the final fixed size of the spectrograms for autoencoder input, we plotted a histogram of the initial durations of all the detected syllables in the training set (Figure 4). To ensure uniform sizing, we zero padded small spectrograms and cropped larger spectrograms, keeping the central part of the vocalisation spectrogram (see below for details on image sizes). Based on the histogram, we selected a fixed-sized duration of 64 windows since it is larger than both the mean and the median of the durations (Figure 4). We noted that this window size balanced the information trade-off mentioned above. It is also a power of 2, which is convenient for the pooling operations in the autoencoder. This length of 64 frames corresponds to 64 time frames $\times$ 0.002 sec/time frame $= 0.128$ sec $= 128$ ms. The aforementioned process of cropping or expanding spectrograms to a fixed-sized width of 64 windows leads to spectrograms of a final resolution of 64 time frames $\times$ 160 frequency bins.

The training set consists of 22,409 images, each one with size $64 \times 160$. We fed the images to the encoder (a CNN with three convolutional layers), each followed by a non-linearity step, achieved through a ReLU activation function and a max pooling layer (Figure 5(a)). The first convolutional layer uses 64 filters, with dimensions $3 \times 3$ each. After that, a max pooling layer decreases the spatial dimensions of the activation maps by a factor of 2. This means that the output of the max
pooling layer is a $32 \times 80 \times 64$ representation. The next convolutional layer consists of 32 filters, with a max pooling layer generating an output with dimensions $16 \times 40 \times 32$. The third and final layer includes eight filters, and a max pooling layer, resulting in a convolutional activation map for each image with dimensions $8 \times 20 \times 8$. This flattened intermediate representation is the feature vector that describes the input image (i.e., the code).

Because the task is unsupervised, the second part of the autoencoder (the decoder) is responsible for reconstructing the image we fed to the encoder from the intermediate representation (Figure 5(a)), this output can be compared to the input image to allow training based on the loss calculated between the output and input images. The decoder reverts the steps of the encoder, using 32, 64 and 1 filter in the last layer. In each decoder layer, we use filters of size $2 \times 2$ and a stride of 2, so that after each layer the size of the representation increases by a factor of 2 and the final output of the autoencoder is an image with the same size as the original input.

As mentioned above, after each convolutional layer, a ReLU activation function is used (Equation 9). An exception is the last deconvolution of the decoder, where a sigmoid activation function (Equation 10) is used for the reconstruction of the image and the calculation of the Binary Cross Entropy Loss function, as it outputs a value between zero and one.
\[ R(z) = \max(0, z) \]  

\[ \sigma(z) = \frac{1}{1 + e^{-z}} \]

The loss function is calculated based on the differences between the reconstructed and the original image. This loss function indicates the quality of the autoencoder’s performance and can be improved with increased training epochs. If we denote the output of the decoder at a specific pixel as \( \hat{y} \) and the actual value of the input image as \( y \), then Binary Cross Entropy Loss is calculated as:

\[ L(y, \hat{y}) = -y \cdot \log(\hat{y}) - (1 - y) \cdot \log(1 - \hat{y}) \]

2.3.2.1. Parameter tuning. The basic parameters for configuring our proposed autoencoder procedure are the following:

- **Number of layers of the encoder.** We tested a range of different numbers of layers (2–4 layers). Using two layers appeared problematic since the autoencoder could not clearly reconstruct input images. Four or more layers resulted in too many parameters, which slowed down the training, resulting in a bigger loss, without the final reconstruction being any better than the one obtained from a 3-layer autoencoder (Figure S3(A)).

- **Number of filters per layer.** We used the most filters in the first layers (as is typical for classical neural networks) and reduced the number of filters as we went deeper in the network. The critical choice we had to make was the number of filters in the last encoder layer, the output of which we use as the representation for the specific image in the clustering task. We tested varying numbers of filters (2, 4 and 8 filters). Using eight filters resulted in a smaller loss, as expected, although four filters were enough for the reconstruction of the images (Figure 5(c)), meaning enough features for the reconstruction were extracted with fewer filters. Fewer filters resulted in losing information from the images, while more resulted in too many parameters to be trained. We selected eight filters in our final design in order to ensure that all the details of the various shapes of the vocalisations are properly extracted.

- **Filter size.** We experimented with \( 3 \times 3 \), \( 5 \times 5 \) and \( 3 \times 5 \) kernels. A \( 3 \times 5 \) kernel appeared reasonable because of the non-square shape of the images but gave similar results to \( 3 \times 3 \) kernels, so we selected the latter, since symmetric kernels are more commonly used (Figure S3(B)).

- **Size of max pooling kernels.** We experimented with reducing the activation map size by a factor of 2 in both dimensions or by a factor of 2 in time dimension and by 4 in frequency dimension due to the non-square image. A 2x symmetrical reduction provided better results (Figure S3(C)).

For other hyperparameters, we used the Adam optimiser, with learning rate equal to 0.001 and batch size equal to 32.
Training epochs were determined experimentally. We found that two or three epochs were enough for a good reconstruction of the images, as loss did not decrease much after three epochs (Figure 5(b)). We also did not want to overfit the training data. Thus, we elected to train the model for just two epochs.

An example of the input and output of the autoencoder is shown in Figure 5(c). The input comes from a recording that was not used in the training Dataset D2. The reconstruction is lossy, due to our use of an undercomplete autoencoder.

2.3.3. Feature extraction and clustering
After the model has been trained in the unsupervised manner described above, it is ready to be used in the feature extraction procedure (Figure 7(a)). An audio file is selected and its spectrogram is calculated, and individual USVs are detected. The raw spectrograms of the USVs are fed to the autoencoder in batches of 32, and the intermediate representations are derived. These are the feature vectors. Each flattened feature vector has a dimension of 1,280 ($8 \times 20 \times 8$) after a dimensionality reduction from a dimension of 10,240 in the initial flattened vector since each image started with a shape of $64 \times 160$.

After the features are extracted, we further reduced the dimensions by excluding features that have the smallest variance and will likely be less impactful for the discrimination of USVs. After extensive qualitative experimentation, assuming we have $N$-dimensional feature vectors, we selected a threshold equal to:

$$v_t = 1.2 \cdot \frac{1}{N} \sum_{j=1}^{N} \sigma_j^2$$  \hspace{1cm} (12)

where $\sigma_j^2$ is the variance of feature $j$. In this case, $v_t$ is the mean of the features’ variances; features with variance less than $v_t$ will be excluded, resulting in feature vectors undergoing a dimensionality reduction of a factor approximately equal to 4.

Next, we normalise the features using a Standard Scaler. Each feature is scaled according to the following equation:

$$X_{\text{transformed}} = \frac{X - X_{\text{mean}}}{X_{\text{std}}}$$  \hspace{1cm} (13)

where $X_{\text{mean}}$ is the mean of the feature values for all samples and $X_{\text{std}}$ is the standard deviation of the feature values for all samples.

In the last preprocessing step, we used PCA to further reduce the dimensionality of the feature vectors. We choose the smallest number of components which maintains 95% of the variance of the features before the PCA, which can be a variable number of components across recordings and/or datasets. Overall, our goal was to both extract many features from the images using the encoder, so that details of the images are taken into account, and simultaneously reduce them as much as possible by ignoring the non-significant features. This final reduced feature representation is then used for clustering. The final visualised cluster is plotted as a two-dimensional tSNE reduction of the data (Figure 7(b)).
Each cluster should consist of vocalisations that share some common features that allow them to belong in the same group. Users can choose the number of clusters, and between the following clustering methods: Agglomerative, Birch, Gaussian Mixture Models, K-Means and Mini-Batch K-Means. Users can choose any combination of these options in the implemented GUI (see Section 3.5).

### 2.4. Baseline feature extraction

To evaluate the quality of AMVOC’s deep feature extraction and clustering, we compared clustering on deep features against clusters derived from hand-picked acoustic parameters (Figure 8(a)). The hand-picked features were measured as follows:

1. We first calculate the spectrogram in the specific time segment that corresponds to the vocalisation and in the defined frequency range (30–110 kHz).

2. We then perform frequency contour detection:
   - Detect the position and the value of the peak energy in each time frame. If we denote the spectrogram value at time frame $i$ and frequency $j$ as $E_{ij}$, the equations describing the two quantities above are the following:
   
   $$ P_i = \arg \max_{j=30\ldots110kHz} E_{ij} $$

   $$ p_i = \max_{j=30\ldots110kHz} E_{ij} $$

   - Use a thresholding condition to keep only the points $i$ where the peak energy is higher than 20% of the highest energy value in the specific time interval:
     
     $$ \text{if } P_i > t \cdot \max_i P_i, \text{ where } t = 0.2, \text{ keep point } (i, p_i) $$

   - Train a regression SVM to map time coordinates to frequency values, using the chosen points $(i, p_i)$ as training data.

   - Predict the frequencies for the same time range. After that, for each vocalisation, a frequency contour $c$ is created, along with a corresponding time vector $v$, which matches every frame $i$ to its actual time of occurrence. This estimated sequence $c$ captures the ‘most dominant’ frequency in each time frame, so we can think of it as a spectral shape sequence of each mice vocalisation.

3. After the frequency contour $c$ is produced, we proceed to the feature extraction step. We selected four different features, all based on the frequency contour:

   - Duration of the vocalisation $d$. If we denote the number of frames of which the vocalisation consists as $N$:
     
     $$ f_1 = d = v[N - 1] - v[0] $$

   - Time position of the minimum frequency (of the predicted frequencies), normalised by the duration of the vocalisation:
\[ f_2 = \frac{v[\arg\min_{i=0..N-1} c_i] - v[0]}{d} \] (18)

- Time position of the maximum frequency (of the predicted frequencies), normalised by the duration of the vocalisation:

\[ f_3 = \frac{v[\arg\max_{i=0..N-1} c_i] - v[0]}{d} \] (19)

- Bandwidth is calculated as the difference between the first and the last predicted frequency value, normalised by the mean frequency of the vocalisation \( m_f \):

\[ f_4 = \frac{c_{N-1} - c_0}{m_f}, \text{ where } m_f = \frac{\sum_{i=0}^{N-1} (c_i)}{N} \] (20)

If we interpret the frequency contour as a 2-dimensional graph, where the x-axis corresponds to time and the y-axis to frequency, the second feature is the x-position of the minimum of the curve, the third feature is the x-position of the maximum of the curve and the fourth feature is the normalised difference between y-positions of the first and last points. For example, the last feature can discriminate contours with different slopes. After the features are extracted, we scale them using a Standard Scaler (see Equation (13)).

### 2.5. Implementation and library description

The functionalities described throughout the paper were implemented in Python 3.8. The code can be found in the repository [https://github.com/tyiannak/amvoc/tree/master/data/vocalizations_evaluation](https://github.com/tyiannak/amvoc/tree/master/data/vocalizations_evaluation). For the implementation of the autoencoder, we used the Pytorch deep learning framework, and clustering algorithm implementations from Scikit-Learn. The visualisation of the data in the offline mode, including the clustering, specific vocalisations, and the evaluation choices are all displayed and accessible in the Dash GUI.

### 3. Results

#### 3.1. Experimental evaluation of the AMVOC detection method

Our objective was to design and implement a robust (in terms of detection performance) but also computationally efficient USV detection method, as our vision was to build a real-time, online pipeline. Because of the online vision of the design, we wanted to ensure that our method could still perform with high accuracy despite the demands of online processing.

We compared our proposed detection methodology with several popular USV detection tools, such as MSA (Arriaga et al. 2012), MUPET (Van Segbroeck et al. 2017), VocalMat (Fonseca et al. 2021), and DeepSqueak (Coffey et al. 2019).

Because of a flaw we noted in MSA detection, we rewrote MSA, from the original MATLAB implementation (MSA1) to a Python implementation (MSA2) with added filtering components to improve USV detection rates. We noted that MSA1 was cutting
off parts of beginning of syllables, underrepresenting the full spectral duration of the USV syllables, as well as misrepresenting the timestamps of the detected USVs. MSA2 first bandpasses the raw audio (30–115 kHz) and then generates a spectrogram. The spectrogram is thresholded according to the signal-to-noise ratio at each frequency.

Due to the vast range of possible parameters that could be tuned in each method we compared against, we used default settings, unless there were other documented improved settings that were used (Chabout et al. 2017). In order to evaluate and compare the aforementioned methods, we used Dataset D1, which consists of 9 audio segments of 5–10 s each, containing 245 annotated syllables in total from 14 male mice (see Section 2.1.3).

To evaluate the range of experimental contexts recordings could be taken from, we split the recordings into two categories: normal and noisy. Normal parts of the recording are the ones where the vocalisation detection is relatively straightforward, because the energy easily surpasses the background energy. Noisy parts contain background noise, such as cage bedding that we purposefully put in to challenge the algorithms or physical interactions between the mice, any of which makes the detection of vocalisations more difficult and ambiguous, even for the human eye observing the raw spectrograms.

To evaluate each method’s ability to detect vocalisations in our datasets, we calculated the temporal and event F1 scores (see Section 2.2.4). Using Dataset D1, derived from Chabout et al. (2015), we found that AMVOC outperforms the other methods with respect to event F1 score, both in clean and noisy segments of the recordings. MSA2 and DeepSqueak performed slightly better than the others with respect to temporal F1 score (Table 1), largely due to the more successful detection in the noisy parts of the recordings.

We also assessed the trade-off between processing speed and detection by determining the processing ratio for AMVOC and each method (Table 2). This metric provides an assessment of how quickly a method is able to process a batch of audio data, which is important for considering the feasibility of real-time processing. AMVOC had an intermediate real-time processing ratio to detect the vocalisations (Table 2). MUPET was the fastest method, whereas VocalMat and DeepSqueak were the slowest (Table 2). The reason for the latter two methods being slower is likely due to their image processing steps used to detect USVs. It is also meaningful that we take into account both of the temporal and event F1 scores since it may be important for particular experimental requirements that a certain method combines accurate and fast detection. We compared the average F1 score with the real-time Processing Ratio for both temporal and event F1 scores (Figure 6(a,b)). AMVOC, DeepSqueak and MSA2 achieved the highest temporal and event F1 score, but MSA2 and

### Table 1. F1 scores of AMVOC and other methods. Scores represent a percent value.

|            | AMVOC |
|------------|-------|
|            | Offline | Online | MSA1 | MSA2 | MUPET | VocalMat | DeepSqueak |
| Temporal   | Normal  | 84     | 85   | 46   | 88    | 85       | 91         | 83         |
| Event      | Noisy   | 67     | 68   | 23   | 71    | 53       | 58         | 76         |
|            | Mean    | 75.5   | 76.5 | 34.5 | 79.5  | 69       | 74.5       | 79.5       |
| Event      | Normal  | 97     | 97   | 66   | 94    | 93       | 90         | 93         |
| Event      | Noisy   | 84     | 83   | 33   | 72    | 57       | 58         | 81         |
| Event      | Mean    | 90.5   | 90   | 49.5 | 83    | 75       | 74         | 87         |

Higher scores indicate a greater similarity to our ground-truth annotations and thus correspond to more accurate detection.
Table 2. Real-time processing ratio of all compared methods. Real-time processing ratio is defined as:
\[
rt = \frac{\text{duration of the recording}}{\text{processing time}}
\]
and is shown for each method. The processing time is calculated as the time
needed to just detect the USVs. The experiments carried out to compute the real-time ratio were
executed for five different recordings, three times for each, and the average time for each method was
calculated. A high real-time processing ratio means that a small processing time is required in order to
detect the vocalisations of a certain signal (e.g. \( rt = 30 \) means that the respective method is 30 times
faster than real-time, meaning it takes 1 min to process 30 min of audio information).

| Methods   | Real-time processing ratio |
|-----------|-----------------------------|
| MUPET     | 32.4                        |
| MSA2      | 29.9                        |
| MSA1      | 28.1                        |
| AMVOC     | 21.2                        |
| DeepSqueak| 8.2                         |
| VocalMat  | 4.3                         |

AMVOC had a considerably better time performance relative to DeepSqueak. While
AMVOC, DeepSqueak and MSA2 were the most accurate overall, AMVOC outperformed
the other methods in segments where noise energy was near that of the USVs (Figure 6(c)).

To mitigate a bias of only comparing accuracies for our lab’s recordings, we also
compared AMVOC with the Dataset published with VocalMat (hereafter referred to as
VM1) by (Fonseca et al. 2021). VM1 consists of seven different recordings from seven
mice (5–15 days old, of both sexes) (Fonseca et al. 2021). We did not change AMVOC’s
pre-determined configuration (parameters \( t \) and \( f \)) for this evaluation, in order to examine
how robust the selection of \( t \) and \( f \) is for recordings produced in different conditions.

The recordings of VM1 contained a constant noise artefact at 30 kHz; to remove
unwanted distortions we used a high pass filter with a higher cut-off frequency, between
45 and 110 kHz, instead of 30–110 kHz. Because the ground truth annotations by
(Fonseca et al. 2021) only declare the start time of each vocalisation, we examined start
times of the different methods compared to ground truth start times with 20 ms tolerance,
similar to what (Fonseca et al. 2021) used in their assessment of VocalMat vs. Deep-
Squeak. We also considered a ground truth vocalisation as ‘found’ when the ground truth
start time was between the start and end time of a detected vocalisation (in case some
method merged successive vocalisations). Since we only have ground truth start times, we
only used event and not temporal evaluation. We calculated precision, recall and F1-
score of VM1 detection results as we had done with Dataset D1, for each of the seven
recordings separately, and then their mean.

On the VM1 dataset, VocalMat achieved the best detection results, followed by
AMVOC, in both offline and online modes, and then DeepSqueak and MUPET
(Table 3). VocalMat’s detection method is trained on data similar to test data from
Dataset VM1, so a high detection quality by VocalMat on its own data could be expected.
Meanwhile, AMVOC’s scores were surprising since we did not specifically tune our
parameters for testing on the VM1 dataset, suggesting that AMVOC can generalise well
across many vocal recording conditions.

We further compared our proposed method by using a new testing Dataset (D4) from
male animals in our Rockefeller University mouse colony, consisting of five adult
recordings and one P7 pup recording. For each recording in D4, the first 50 vocalisations
start and end times were manually annotated, meaning that the time intervals of interest
**Figure 6.** Accuracy of AMVOC and other methods. a-b) Event and temporal accuracy of different USV detection methods compared against our ground truth data in different qualities of recordings. c) Examples of the two highest performing methods, AMVOC and DeepSqueak, on clean (top) and noisy (bottom) segments of USV recordings. Contrast modified to facilitate visualisation in the noisy spectrogram.

**Table 3.** Event F1-score of all methods, tested on dataset VM1.

| Methods          | Event F1-score (%) |
|------------------|--------------------|
| AMVOC offline    | 95.6 (± 2.5)       |
| AMVOC online     | 93.6 (± 3.8)       |
| VocalMat         | 98.5 (± 0.6)       |
| MUPET            | 76.0 (± 10.2)      |
| DeepSqueak       | 85.1 (± 3.8)       |

All measurements provided as mean and standard error of the mean (SEM) from n = 7 samples (recordings).
varied across recordings. For example, the 50th USV in one recording could occur at 40 s, while the 50th USV of another recording could occur at 120 s. In order to evaluate fairly across methods, we only kept the timestamps given by each method for which their start time was less than or equal to the 50th annotated USV’s end time:

\[
st_{\text{tool}}[i] \leq et_{\text{gt}}[50]
\]

where \(st_{\text{tool}}[i]\) is the start time of the \(i^{th}\) USV detected by a method and \(et_{\text{gt}}[50]\) is the end time of the 50th ground truth USV. This means that if the 50th ground truth USV is at 40 s, and Tool A’s 36th USV is at 42 s, we will only consider Tool A’s first 35 USVs for analysis.

To test for significant differences in detection accuracy across methods, we performed a Kruskal–Wallis one-way analysis of variance after normalising for recording quality by subtracting the mean performance of all methods for each recording. Similar to dataset D1, AMVOC, in both offline and online modes, achieved higher Event F1-scores compared to the other methods (\(H\) statistic = 12.5, \(p\)-value = 0.01), without any parameter tuning on D4 (Table 4). There was no significant difference in the Temporal F-1 Scores (\(H\) statistic = 2.25, \(p\)-value = 0.68). Notably, VocalMat’s detection had the highest Temporal F1-score, although it had the least accurate Event F1 metric in this dataset.

### 3.2. Experimental evaluation of the AMVOC clustering method

After evaluating the segmentation and detection performance of AMVOC, we wanted to develop a way to assess the repertoire composition of the USVs. To do so, we implemented an autoencoder approach (see Section 2.3) and then clustered the latent features for manual inspection and evaluation (Figure 7). Although others have shown autoencoders can extract relevant information about an individual mouse’s vocalisations (Goffinet et al. 2021), we wanted to ensure that the autoencoder we use could capture features of USVs that may not be obviously calculated or discerned relative to what could be obtained from simple feature descriptions that are often used for comparing USVs of different mice.

#### Table 4. Temporal and Event F1-score on D4 of all methods.

| F1 score | Recording | AMVOC-offline | AMVOC-online | VocalMat | MUPET | DeepSqueak |
|----------|-----------|---------------|---------------|----------|-------|------------|
| **Temporal** | Rec1 | 64.9 | 64.9 | 63.7 | 73 | 63.4 |
| | Rec2 | 67.6 | 67.6 | 85.3 | 61.9 | 79.5 |
| | Rec3 | 89.1 | 90.9 | 90.7 | 94.9 | 89.6 |
| | Rec4 | 69.9 | 69.6 | 83.7 | 68.9 | 61.6 |
| | Rec5 | 72.1 | 69.1 | 83 | 67.4 | 58.6 |
| | Rec6 | 41.9 | 43.8 | 34.5 | 27.8 | 77.3 |
| **Mean (SEM)** | Rec1 | 67.6 (±6.2) | 67.7 (±6.1) | 73.5 (±8.7) | 65.7 (±8.9) | 71.7 (±5.0) |
| | Rec2 | 93.9 | 93.9 | 85.5 | 97 | 81.8 |
| | Rec3 | 99 | 99 | 90.1 | 91.7 | 90.5 |
| | Rec4 | 100 | 100 | 98 | 96.9 | 96.9 |
| | Rec5 | 94.1 | 93.2 | 78.1 | 88.9 | 79.1 |
| | Rec6 | 89.5 | 83.2 | 73.5 | 71.6 | 78.7 |
| **Mean (SEM)** | Rec1 | 92.4 (±3.3) | 91.1 (±3.7) | 80.7 (±5.6) | 86.6 (±4.6) | 85.3 (±2.9) |

Mean of each method reported as mean and standard error of the mean (SEM) from \(n = 6\) samples (recordings).
We compared the deep feature extraction method (see Section 3.3) to the baseline feature extraction method using hand-picked features (e.g. bandwidth, duration, frequency max, and frequency min, Figure 5(a)), by evaluating the derived clustering of the two kinds of features. We used data from four different annotators, two of whom are domain experts, and the other two are not. By using the AMVOC GUI, each annotator evaluated the four recordings from Dataset D3, which were generated by selecting segments of 72 different recordings (2.1.3). The annotators evaluated three different clustering configurations for each recording: Agglomerative clustering with six clusters; Gaussian Mixture clustering with six clusters; and K-Means clustering with six clusters. In order to ensure impartiality and objectivity, the annotators evaluated the clustering derived from both feature extraction methods (named as Method 1 for deep features and Method 2 for hand-picked features), without prior knowledge of which Method refers to the deep features or the hand-picked (simple). Using a scale from 1 to 5, 5 being the best, the evaluation metrics used are the following:

Figure 7. Overview of deep feature extraction procedure. a) Flow diagram of the general procedures used to take image data from USV spectrograms into clusters. b) Cluster example using deep features with K-Means clustering and 6 clusters.
(1) Global annotations: The annotator defines a score to describe how successful the whole clustering is.

(2) Cluster annotations: The annotator defines a score to describe how successful each cluster is.

(3) Point annotations: The annotator selects points from different clusters and declares whether they should be approved or rejected in the specific cluster to which they have been assigned. Approximately 100 points were annotated by each user per configuration, for each method.

For global annotations of each annotator $i$, we calculated the mean score $\mu$ of each clustering configuration $s$ (KMeans-6, GMM-6 and Agg-6), using the scores from the four recordings:

$$\mu_{i,s} = \frac{1}{4} \sum_{j=1}^{4} G_{i,s,j}$$

where counter $j$ refers to the four recordings and $G_{i,s,j}$ is the global score of each configuration $s$, in recording $j$, set by annotator $i$. Then, the mean $m$ and standard deviation $d$ of these mean scores of the four users is calculated (Figure 8(b)).

![Figure 8](image_url)

**Figure 8.** Global and cluster annotation evaluations. a) Examples of two vocalisations with the autoencoder image generated from deep feature extraction (left column) and the peak energy contour used for hand-picked feature extraction (right column). b) Mean and standard deviation (error bars) of the scores assigned in each configuration for all files evaluated by the annotators for the global clustering scores. c) Mean score and S.D. each cluster in each configuration. d) Mean percentage of approved vocalisations for point annotation evaluation. $p$-values in b and c determined by Student’s $t$-test.
\[ m_s = \frac{1}{N_a} \sum_{i=1}^{i=N_a} \mu_{i,s} \]

\[ d_s = \sqrt{\frac{1}{N_a} \sum_{i=1}^{i=N_a} (\mu_{i,s} - m_s)^2} \]

where \( N_a \) is the number of annotators (in our case, 4).

Based on the global annotation results, deep feature extraction yielded significantly better clustering results (37% higher on average) than simple feature extraction, in all three clustering configurations tested (Figure 8(b); Student’s t-test; K-Means, \( t = 5.0, p = 1.7 \cdot 10^{-4} \); GMM, \( t = 5.1, p = 1.3 \cdot 10^{-4} \), Agg, \( t = 3.1, p = 7.2 \cdot 10^{-3} \)).

The clustering configuration does not seem to affect the performance of the clustering very much, as far as the mean values were concerned. For the cluster annotation scores, for each configuration and annotator, we calculated the mean scores \( \mu' \) of all the six clusters in the four recordings:

\[ \mu'_{i,s} = \frac{1}{4 \cdot c_i} \sum_{j=1}^{j=4} \sum_{k=1}^{k=c_i} C_{i,s,j,k} \]

where counter \( j \) refers to the four recordings, \( k \) to the number of clusters of each configuration \( c_i \) (in our case, six for all configurations) and \( C_{i,s,j,k} \) is the cluster-specific score of cluster \( k \) of configuration \( s \), in recording \( j \), set by annotator \( i \).

These mean scores were then used to calculate the mean and standard deviation values (\( m' \) and \( d' \) respectively) for each configuration:

\[ m'_s = \frac{1}{N_a} \sum_{i=1}^{i=N_a} \mu'_{i,s} \]

\[ d'_s = \sqrt{\frac{1}{N_a} \sum_{i=1}^{i=N_a} (\mu'_{i,s} - m'_s)^2} \]

where \( N_a \) is the number of annotators (in our case, 4). We found that the cluster-level annotation results scores were consistent (30% higher on average) with the global annotation results (Figure 8(c)). As with global scores, all cluster annotation comparisons showed significant differences between deep and simple feature annotations (Student’s t-test; K-Means, \( t = 6.7, p = 1.3 \cdot 10^{-9} \); GMM, \( t = 6.5, p = 3.4 \cdot 10^{-9} \), Agg, \( t = 5.3, p = 8.3 \cdot 10^{-7} \)).

We then assessed the percentage of vocalisations each user approved for each cluster based on their unsupervised assignment for both deep and simple methods. If we denote the approved vocalisations of user \( i \) as \( a_i \), and the total vocalisation annotations they have made as \( t_i \) then the mean percentage of approved vocalisations is calculated as:

\[ M = \frac{1}{N_a} \sum_{i=1}^{N_a} \frac{a_i}{t_i} \]
Users more frequently approved vocalisations clustered by the deep feature extraction method (Figure 8(d)), although this was not significantly different between deep and simple (Student’s $t$-test; $t = 1.73$, $p = 0.18$). This indicates that between both feature types, within-cluster points are consistent, even if deep features had better overall scores across all clusters.

Overall, the deep feature extraction method outperforms the simple method in all terms. This suggests that the encoder has indeed retrieved useful information of each image, resulting in feature vectors that enable a better clustering of the vocalisations based on visually discernible features. Further, this indicates that there are much more complex similarities and differences between vocalisations carried by the representations extracted from the encoder than what is available from hand-picked features.

4. Discussion

We have presented Analysis of Mouse Vocal Communication (AMVOC), an open-source tool for detecting and analysing mice ultrasonic vocalisations. Our evaluation of AMVOC on real datasets demonstrates that it is as accurate and in some cases more accurate than state-of-the-art tools, while being fast enough to function in real-time conditions. AMVOC also includes a novel unsupervised approach for representing and grouping the USVs’ content using convolutional autoencoders. The deep feature extraction method outperforms simple hand-picked features in clustering USVs that are more similar to one another, leading to possibly meaningful and homogeneous clusters. This methodology sets a new context for automatic content characterisation of mouse vocalisations, according to which clusters of similar vocalisations can be automatically ‘learnt’ through deep neural network architectures. While others have used autoencoders to analyse mouse USVs (Goffinet et al. 2021), these methods were not designed to allow users to explore the deep feature clustering and evaluate the results themselves. In addition to the functionalities and usage modes mentioned, the Dash app in which AMVOC runs provides the user with the ability to save the current clustering configuration, along with other user-friendly functionalities. For example, the user can save the feature vectors and the corresponding clusters of vocalisations, and then use them as ground truth data to train a classifier. By using the automatically extracted vocalisation clusters as classes to train a supervised model, users can then load the model in real-time and online behavioural context to classify discovered syllables immediately after the vocalisations are detected. This workflow provides a previously inaccessible way to evaluate mouse USV behaviour, and opens the door for real-time, closed-loop behavioural assessments. A future direction of ours is the application of semisupervised methodologies in USV clustering. The goal would be to integrate human interventions in the classifier used for detection (either offline or online). This could include alternative clusters with reassigned labels for misclustered USVs or comparing pairs of vocalisations, which can be used as pairwise constraints to retrain the encoder part of the model. These retrained models and new classifications could provide new opportunities for approaches such as operant behaviour or new types of experiments in mouse vocal research that are inaccessible with current methodologies. Should future studies find that different strains of mice with different types of vocalisations are not classifiable with AMVOC, then perhaps artificial intelligence approaches could be more adaptable. While much of the work discussed here and by others focuses on descriptions and characterisations of single USVs, applying these techniques to sequences of USVs
(Chabout et al. 2016) could yield a more detailed understanding of rodent vocal behaviour. AMVOC at a sequence-level could provide ways to examine the temporal relationship between syllables across bouts of vocal behaviour. It has been known since male mouse courtship USVs were first identified as ‘songs’ (Holy and Guo 2005) that across the timescale of a recording session mice will use similar sequences of vocalisations. Initial approaches to analysing sequences (Chabout et al. 2015, 2016) have focused on the pairwise probability of transitions in syllables (i.e. how one syllable type follows or precedes another). We propose that future analyses would be best approached at varying timescales and syllable sequence dimensions. By doing so, sequences of vocalisations could be processed and studied, so they can be connected and correlated with various behaviours in mice with a broader appreciation of vocal behaviour.

AMVOC is made possible by improvements in and increased access to high-capacity computing resources. Such resources, especially those that are user friendly and scalable, have led to an explosion of tools to measure as many features of animal behaviour as possible (Robert Datta et al. 2019; von Ziegler et al. 2021; Hausmann et al. 2021). These resources have become invaluable as neuroscientists push to understand behaviours in more naturalistic contexts. Such unrestrained behaviours can allow for more flexible experimentation and a deeper understanding of ethologically relevant brain function or clinically relevant behavioural changes (Jones et al. 2020). One way that this has been approached in most behavioural experiments is to use markerless animal tracking or pose estimation of single or multi-animal contexts (Kabra et al. 2013; Mathis et al. 2018; Gal et al. 2020; Marshall et al. 2021; Hsu and Yttri 2021). As with mouse USV analyses, many of these methods remain offline, with some notable exceptions providing low-latency feedback for behavioural experimentation (Kane et al. 2020). Following in the footsteps of increased data capture and processes that facilitate real-time event detection in animal behaviour, AMVOC sets a new standard for vocally driven analysis of mice behaviour.

**Acknowledgements**

We thank E. Waidmann, R. Agravat, M. Davenport, and other members of the Jarvis lab for their thoughtful conversations on this project. We thank J. Chabout for providing open access to his past mouse recordings.

**Disclosure statement**

No potential conflict of interest was reported by the author(s).

**Funding**

This project was funded by the Howard Hughes Medical Institute and Rockefeller University start-up funds to EDJ, a Kavli Neural Systems Institute Pilot Grant to CDMV. JLB is supported by the Kavli Foundation. CDMV is a Howard Hughes Medical Institute Gilliam Fellow; Kavli Neural Systems Institute [NA]; The Rockefeller University [NA].
References

Arriaga G, Zhou EP, Jarvis ED. 2012. Of mice, birds, and men: The mouse ultrasonic song system has some features similar to humans and song-learning birds. PLoS One. 7(10):e46610. doi:10.1371/journal.pone.0046610.

Calbick D, Castellucci GA, McCormick DA. 2017. Differential temporal structures in mouse ultrasonic vocalizations relay sex and age information about the producer. J Acoust Soc Am. 141(5):3946. doi:10.1121/1.4988955.

Castellucci GA, Calbick D, McCormick D. 2018. The temporal organization of mouse ultrasonic vocalizations. PLoS One. 13(10):e0199929. doi:10.1371/journal.pone.0199929.

Castellucci GA, McGinley MJ, McCormick DA. mar, 2016. Knockout of foxp2 disrupts vocal development in mice. Sci Rep. 6(1). doi:10.1038/srep23305.

Chabot J, Jones-Macopson J, Jarvis ED. 2017. Eliciting and analyzing male mouse ultrasonic vocalization (USV) songs. J Vis Exp. JoVE, (123), 2017 (123). doi:10.3791/54138.

Chabot J, Sarkar A, Dunson DB, Jarvis ED. 2015. Male mouse song syntax depends on social contexts and influences female preferences. Front Behav Neurosci. 9:76. doi:10.3389/fnbeh.2015.00076.

Chabot J, Sarkar A, Patel SR, Radden T, Dunson DB, Fisher SE, Jarvis ED. 2016. A Foxp2 mutation implicated in human speech deficits alters sequencing of ultrasonic vocalizations in adult male mice. Front Behav Neurosci. 10:197. doi:10.3389/fnbeh.2016.00197.

Coffey KR, Marx RG, Neumair JF. 2019. Deepsqueak: A deep learning-based system for detection and analysis of ultrasonic vocalizations. Neuropsychopharmacology. 44(5):859–868. doi:10.1038/s41386-018-0303-6.

Fonseca AH, Santana GM, Bosque Ortiz GM, Bampi S, Dietrich MO. 2021 mar. Analysis of ultrasonic vocalizations from mice using computer vision and machine learning. eLife. 10:e59161. ISSN 2050-084X. doi:10.7554/eLife.59161.

Gal A, Saragosti J, Kronauer DJ. 2020. Antrax, a software package for high-throughput video tracking of color-tagged insects. Elife. 9:e58145. doi:10.7554/eLife.58145.

Gao S-C, Wei Y-C, Wang S-R, Xiao-Hong X. 2019. Medial preoptic area modulates courtship ultrasonic vocalization in adult male mice. Neurosci Bull. 35(4):697–708. doi:10.1007/s12264-019-00365-w.

Goffinet J, Brudner S, Mooney R, Pearson J. 2021. Low-dimensional learned feature spaces quantify individual and group differences in vocal repertoires. eLife. 10:e67855. https://doi.org/10.7554/eLife.67855.

Goodfellow I, Bengio Y, Courville A. 2016. Deep learning. MIT Press. http://www.deeplearningbook.org.

Grimsley JM, Monaghan JJ, Wenstrup JJ. 2011. Development of social vocalizations in mice. PLoS One. 6(3):e17460. doi:10.1371/journal.pone.0017460.

Haumann SB, Marin Vargas A, Mathis A, Mathis MW. 2021. Measuring and modeling the motor system with machine learning. arXiv preprint arXiv:2103.11775.

Hertz S, Weiner B, Perets N, London M. 2020. Temporal structure of mouse courtship vocalizations facilitates syllable labeling. Commun Biol. 3(1):1–13. doi:10.1038/s42003-020-1053-7.

Holy TE, Guo Z. 2005. Ultrasonic songs of male mice. PLoS Biol. 3(12):e386. doi:10.1371/journal.pbio.0030386.
Hsu AI, Yttri EA. 2021. An open source unsupervised algorithm for identification and fast prediction of behaviors. Nat Commun. 12:5188. https://doi.org/10.1038/s41467-021-25420-x .
Jones JM, Foster W, Twomey CR, Burdge J, Ahmed OM, Pereira TD, Wojick JA, Corder G, Plotkin JB, Abdus-Saboor I. 2020. A machine-vision approach for automated pain measurement at millisecond timescales. Elife. 9:e57258. doi:10.7554/eLife.57258.
Kabra M, Robie AA, Rivera-Alba M, Branson S, Branson K. 2013. Jaaba: Interactive machine learning for automatic annotation of animal behavior. Nat Methods. 10(1):64. doi:10.1038/nmeth.2281.
Kane GA, Lopes G, Saunders JL, Mathis A, Mathis MW. 2020. Real-time, low-latency closed-loop feedback using markerless posture tracking. Elife. 9:e61909. doi:10.7554/eLife.61909.
Kingma DP, Welling M. 2014. Auto-encoding variational bayes.
Liu RC, Miller KD, Merzenich MM, Schreiner CE. 2003. Acoustic variability and distinguishability among mouse ultrasound vocalizations. J Acoust Soc Am. 114(6):3412–3422. doi:10.1121/1.1623787.
Luisa Scattoni M, Crawley J, Ricceri L. 2009. Ultrasonic vocalizations: a tool for behavioural phenotyping of mouse models of neurodevelopmental disorders. Neurosci Biobehav Rev. 33 (4):508–515. doi:10.1016/j.neubiorev.2008.08.003.
Luisa Scattoni M, Gandhy SU, Ricceri L, Crawley JN. 2008. Unusual repertoire of vocalizations in the bbr t+t/f/ mouse model of autism. PLoS One. 3(8). doi:10.1371/journal.pone.0003067.
Marshall JD, Aldarondo DE, Dunn TW, Wang WL, Berman GJ, Ölveczky BP. 2021. Continuous whole-body 3D kinematic recordings across the rodent behavioral repertoire. Neuron. 109 (3):420–437. doi:10.7554/eNeuro.2020.11.016.
Mathis A, Mamidanna P, Cury KM, Abe T, Murthy VN, Weygandt Mathis M, Bethge M. 2018. Deeplabcut: Markerless pose estimation of user-defined body parts with deep learning. Nat Neurosci. 21(9):1281–1289. doi:10.1038/s41593-018-0209-y.
Melotti L, Siestrup S, Peng M, Vitali V, Dowling D, Sachser N, Kaiser S, Helene Richter S. 2021. Individuality, as well as genotype, affects characteristics and temporal consistency of courtship songs in male mice. bioRxiv.
Michael Bowers J, Miguel Perez-Pouchoulen NSE, McCarthy MM. 2013. Foxp2 mediates sex differences in ultrasonic vocalization by rat pups and directs order of maternal retrieval. J Neurosci. 33(8):3276–3283. doi:10.1523/JNEUROSCI.0425-12.2013.
Michael V, Goffinet J, Pearson J, Wang F, Tschida K, Mooney R. 2020. Circuit and synaptic organization of forebrain-to-midbrain pathways that promote and suppress vocalization. Elife. 9:e63493. doi:10.7554/eLife.63493.
Neilans EG, Hollo Ath DP, Radziwon KE, Portfors CV, Dent ML. 2014. Discrimination of ultrasonic vocalizations by cba/caj mice (mus musculus) is related to spectrotemporal dissimilarity of vocalizations. PLoS One. 9(1):e85405. doi:10.1371/journal.pone.0085405.
Neunuebel JP, Taylor AL, Arthur BJ, Roian Egnor SE. 2015 may. Female mice ultrasonically interact with males during courtship displays. eLife. 4. doi:10.7554/eLife.06203.
Okobi DE, Banerjee A, Matheson AM, Phelps SM, Long MA. 2019. Motor cortical control of vocal interaction in neotropical singing mice. Science. 363(6430):983–988. doi:10.1126/science.aau9480.
Perrodin C, Verzat C, Bendor D. 2020. Courtship behaviour reveals temporal regularity is a critical social cue in mouse communication. BioRxiv.
Pomerantz SM, Nunez AA, Jay Bean N. 1983. Female behavior is affected by male ultrasonic vocalizations in house mice. Physiol Behav. 31(1):91–96. doi:10.1016/0031-9384(83)90104-4.
Premoli M, Baggi D, Bianchetti M, Gnutti A, Bondaschi M, Drea Mastinu A, Migliorati P, Signoroni A, Leonardi R, Memo M, et al. 2021 Jan. Automatic classification of mice vocalizations using machine learning techniques and convolutional neural networks. PLoS One. 16(1):e0244636. doi:10.1371/journal.pone.0244636.
Robert Datta S, Anderson DJ, Branson K, Perona P, Leifer A. 2019. Computational neuroethology: A call to action. Neuron. 104(1):11–24. doi:10.1016/j.neuron.2019.09.038.
Sober SJ, Brainard MS. 2012. Vocal learning is constrained by the statistics of sensorimotor experience. Proc Nat Acad Sci. 109(51):21099–21103. doi:10.1073/pnas.1213622109.
Steinfath E, Palacios A, Rottschäfer J, Yuezak D, Clemens J. mar, 2021. Fast and accurate annotation of acoustic signals with deep neural networks. eLife. 10. doi:10.7554/eLife.68837.
Tabler JM, Rigney MM, Berman GJ, Gopalakrishnan S, Heude E, Adel Al-Lami H, Yannakoudakis BZ, Fitch RD, Carter C, Vokes S, et al. 2017. Cilia-mediated hedgehog signaling controls form and function in the mammalian larynx. Elife. 6:e19153. doi:10.7554/eLife.19153.
Tachibana RO, Kanno K, Okabe S, Kobayasi K, Okanoya K. 2020 Feb. USVSEG: A robust method for segmentation of ultrasonic vocalizations in rodents. PLoS One. 15(2):e0228907. doi:10.1371/journal.pone.0228907.
Torquet N, De Chaumont F, Faure P, Bourgeron T, Ey E. 2016. Mousetube – A database to collaboratively unravel mouse ultrasonic communication. F1000research. 5. doi:10.12688/f1000research.9439.1.
Tschida K, Michael V, Takatoh J, Han B-X, Zhao S, Sakurai K, Mooney R, Wang F. 2019. A specialized neural circuit gates social vocalizations in the mouse. Neuron. 103(3):459–472. doi:10.1016/j.neuron.2019.05.025.
Van Segbroeck M, Knoll AT, Levitt P, Narayanan S. 2017. Mupet – Mouse ultrasonic profile extraction: A signal processing tool for rapid and unsupervised analysis of ultrasonic vocalizations. Neuron. 94(3):465–485. doi:10.1016/j.neuron.2017.04.005.
von Ziegler L, Sturm O, Bohacek J. 2021. Big behavior: Challenges and opportunities in a new era of deep behavior profiling. Neuropsychopharmacology. 46(1):33–44. doi:10.1038/s41386-020-0751-7.
Xiao L, Chattree G, Garcia Oscos F, Cao M, Wanat MJ, Roberts TF. 2018. A basal ganglia circuit sufficient to guide birdsong learning. Neuron. 98(1):208–221. doi:10.1016/j.neuron.2018.02.020.
Zimmer MR, Fonseca AHO, Iyilikci O, Dai Pra RD, Dietrich MO. 2019. Functional ontogeny of hypothalamic agrp neurons in neonatal mouse behaviors. Cell. 178(1):44–59. doi:10.1016/j.cell.2019.04.026.