Fast and accurate annotation of acoustic signals with deep neural networks

Elsa Steinfath[1,2], Adrian Palacios[1,2], Julian Rottschäfer[1,2], Deniz Yuezak[1,2], Jan Clemens[1,3,4]

1. European Neuroscience Institute - A Joint Initiative of the University Medical Center Göttingen and the Max-Planck-Society, Göttingen, Germany
2. International Max Planck Research School and Göttingen Graduate School for Neurosciences, Biophysics, and Molecular Biosciences (GGNB) at the University of Göttingen, Göttingen, Germany
3. Bernstein Center for Computational Neuroscience, Göttingen, Germany
4. corresponding author, contact: clemensjan@gmail.com

Abstract

Acoustic signals serve communication within and across species throughout the animal kingdom. Studying the genetics, evolution, and neurobiology of acoustic communication requires annotating acoustic signals: segmenting and identifying individual acoustic elements like syllables or sound pulses. To be useful, annotations need to be accurate, robust to noise, fast.

We introduce DeepSS, a method that annotates acoustic signals across species based on a deep-learning derived hierarchical presentation of sound. We demonstrate the accuracy, robustness, and speed of DeepSS using acoustic signals with diverse characteristics: courtship song from flies, ultrasonic vocalizations of mice, and syllables with complex spectrotemporal structure from birds. DeepSS comes with a graphical user interface for annotating song, training the network, and for generating and proofreading annotations. The method can be trained to annotate signals from new species with little manual annotation and can be combined with unsupervised methods to discover novel signal types. DeepSS annotates song with high throughput and low latency, allowing realtime annotations for closed-loop experimental interventions.

Introduction

Animals produce sounds to foster group cohesion (Chaverri et al., 2012; Haack et al., 1983; Janik and Slater, 1998), to signal the presence of food, friend, or foe (Cäsar et al., 2013; Clay et al., 2012), and to find and evaluate mating partners (Baker et al., 2019; Behr and Helversen, 2004; Holy and Guo, 2005; Sangiamo et al., 2020). Studying acoustic communication not only provides insight into social interactions within and across species; it can also reveal the mechanisms driving complex behaviors: The genetics and evolution of signal production and recognition (Ding et al., 2019, 2016), the genes and circuits driving song learning (Kollmorgen et al., 2020), or the fast and precise sensorimotor transformations involved in vocal interactions (Cator et al., 2009; Coen et al., 2016; Fortune et al., 2011; Okobi et al., 2019). The first step in any study of acoustic communication is song annotation: the segmentation and labelling of individual elements in a recording. Acoustic signals are diverse and range from the repetitive long-distance calling songs of crickets, grasshoppers, and anurans (Gerhardt and Huber, 2002), the dynamic and context-specific courtship songs of vinegar flies or rodents (Clemens et al., 2018; Coen et al., 2014; Neunuebel et al., 2015; Sangiamo et al., 2020), to the complex vocalizations produced by some birds and primates (Lipkind et al., 2013; Weiss et al., 2014).

This diversity in signal structure has spawned a zoo of annotation tools (Arthur et al., 2013; Coffey et al., 2019; Cohen et al., 2020; Goffinet et al., 2019; Koumura and Okanoya, 2016; Tachibana et al., 2020), but existing methods still face challenges: First, assessing vocal repertoires and their relation to behavioral and neural dynamics (Clemens et al., 2018; Coffey et al., 2019; Fortune et al., 2011; Neunuebel et al., 2015; Okobi et al., 2019) requires annotations to be complete and temporally precise even at low signal levels, but annotation can fail when signals are weak (Coffey et al., 2019; Stern et al., 2017). Second, analyses of large datasets and experimental interventions during behavior (Bath et al., 2014; Fortune et al., 2011; Okobi et al., 2019; Stowers et al., 2017; Tschida and Mooney, 2012) need annotations to be fast, but existing methods are often slow. Last, annotation methods should be flexible and adaptable (Clemens et al., 2018; Clemens and Hennig, 2013; Ding et al., 2019, 2016), but existing methods often work only for restricted types of signals or adapting them to new signals requires tedious manual tuning (Clemens et al., 2018).

In brief, an accurate, fast, and flexible framework for annotating song across species is missing. A general framework would not only improve upon existing methods but would also facilitate the study of species for which automated methods do not yet exist. Deep neural networks have emerged as powerful and flexible tools for solving data annotation tasks relevant for neuroscience such as object recognition, pose tracking, or speech recognition (Graves and Jaitly, 2014; Graving et al., 2019; Krizhevsky et al., 2012; Mathis et al., 2018; Pereira et al., 2018). These methods are not only fast and accurate but also easily adapted to novel signals by non-experts since they only require annotated examples for learning. Recently, they have also been used for annotating animal vocalizations (Arthur et al., 2021; Coffey et al., 2019; Cohen et al., 2020; Goffinet et al., 2019; Sainburg et al., 2019).

We here present a new deep-learning based framework for annotating acoustic signals, called *Deep Song Segmenter* (DeepSS). We test the framework on a diverse set of recordings from flies, mice, and birds, and show that DeepSS annotates song in single- and multi-channel recordings with high accuracy. The framework produces annotations with low latency on standard PCs and is therefore ideally suited for closed-loop applications. Small to moderate amounts of manual annotations suffice for adapting the method to a new species and annotation work can be simplified by combining DeepSS with unsupervised methods. We provide DeepSS as an open source software package with a graphical user interface for manually annotating audio, training the network, and inferring and proofreading annotations. Integration into existing frameworks for signal analysis or experimental control is possible using a programmatic interface. The code and documentation for DeepSS is available at https://janclemenslab.org/deepss/.

Results

Architecture and working principle of DeepSS

Acoustic signals are defined by features on multiple timescales—the fast harmonic oscillations of the sound carrier (<10 ms), modulations of amplitude (AM) and frequency (FM) (10–100 ms), and the sequencing of different AM and FM patterns into bouts, syllables, or phrases (10–1000 ms). In most methods, these patterns are typically made explicit using hand-tuned and fixed representations derived from time-resolved Fourier and wavelet transforms (Arthur et al., 2013; Coffey et al., 2019; Cohen et al., 2020; Van Segbroeck et al., 2017). By contrast, DeepSS learns a task-specific representation of sounds features directly from raw audio using *temporal convolutional networks* (TCNs) (Bai et al., 2018; Guirguis et al., 2020; Oord et al., 2016) (Fig. S1A-E). At the core of TCNs are so-called *dilated convolutions* (Yu and Koltun, 2015). In standard convolutions, short templates without gaps slide over the signal and return the similarity with the audio waveform at every time point. In *dilated convolutions*, these templates have gaps, allowing to analyze features on longer timescales without requiring more parameters to specify the template. Stacking dilated convolutions with growing gap sizes results in a hierarchical, multi-scale representation of sound features, which is ideally suited for the hierarchical and harmonic structure of animal vocalizations.

The output of the TCN in DeepSS is a set of confidence scores for each audio sample, corresponding to the probability of each song type (Fig. 1C). Annotation labels are produced by comparing the confidence score to a threshold or by choosing the most probable song type. Brief gaps in the annotations are closed and short spurious detections are removed to smoothen the annotation. For song types that are described as events, like the pulses in fly song (Fig. 1A), the event times are extracted as local maxima that exceed a confidence threshold.

DeepSS accurately annotates the song of flies, mice, and birds

Fly courtship song

We first tested DeepSS on the courtship song of *Drosophila melanogaster*, which consists of two major modes (Fig. 1A): The sine song, which corresponds to sustained oscillations with a species-specific carrier frequency (150 Hz), and two types of pulse song, which consists of trains of short (5-10 ms) pulses with carrier frequencies between 180 and 500 Hz, produced with a species-specific interval (35-45ms in *D. melanogaster*). Males dynamically choose the song modes based on sensory feedback from the female (Calhoun et al., 2019; Clemens et al., 2018; Coen et al., 2016, 2014). Despite the relative simplicity of the individual song elements, an accurate annotation of fly song is challenging because of low signal-to-noise ratio (SNR): The song attenuates rapidly with distance (Bennet-Clark, 1998) and is highly directional (Morley et al., 2018), which can lead to weak signals if the male is far from the microphone (Fig. 1A). Moreover, the interactions between male and female introduce pulsatile noise and complicate the accurate and complete annotation of the pulse song.

We first trained DeepSS to detect the pulse and the sine song recorded using a single microphone (data from (Stern, 2014)) and compared the performance of DeepSS to that of the current state-of-the-art in fly song segmentation, FlySongSegmenter (FSS)(Arthur et al., 2013; Clemens et al., 2018; Coen et al., 2014). DeepSS was trained on recordings from multiple males and then tested on data from a different male of the same species—we thereby assess how well the method generalizes to new individuals. Annotation performance was quantified using *precision*, the fraction of correct detections, and *recall*, the fraction of true song that is detected (Fig. 1E, J, S1F, G). We counted detected pulses within 10 ms of a true pulse as correct detections. 10 ms corresponds to 1/4th of the typical interval between pulses in a train and results are robust to the choice of this value (Fig. S2A). DeepSS detects pulses with a high precision of 97% — only 3% of all detected pulses are false detections — and a high recall of 96% — it misses only 4% of all pulses. This is a substantial improvement in recall over FSS, which has slightly higher precision (99%) but misses 13% of all pulses (87% recall) (Fig. 1D, E). In DeepSS, the balance between precision and recall can be

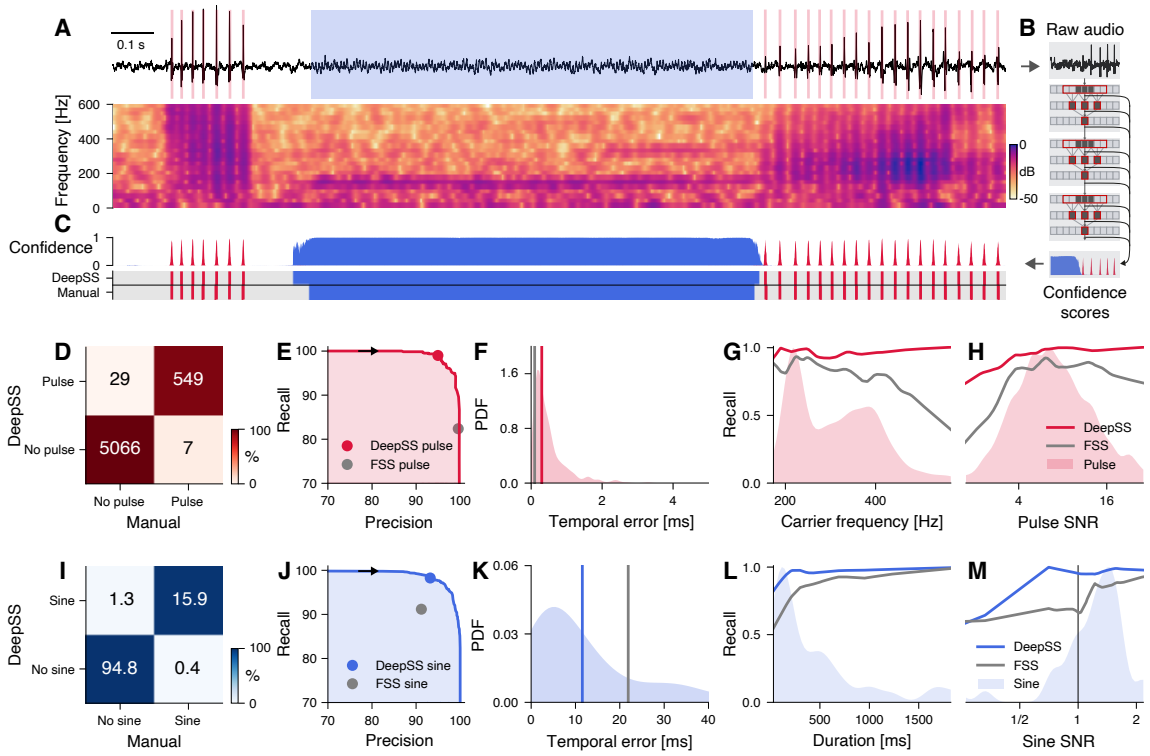

Figure 1: DeepSS performance for fly song.

A Fly song (black, top) with manual annotations of sine (blue) and pulse (red) song. The spectrogram (bottom) shows the signal's frequency content over time (see color bar).

B DeepSS builds a hierarchical presentation of song features relevant for annotation using a deep neural network. The output of the network is a confidence score for each sample and song type.

C Confidence scores (top) for sine (blue) and pulse (red) for the signal in A. The confidence is transformed into annotation labels (bottom) based on a confidence threshold (0.5 for sine, 0.7 for pulse). Ground truth (bottom) from manual annotations shown for comparison.

D Confusion matrix for pulse from the test data set. Color indicates the percentage (see color bar) and text labels indicate the number of pulses for each quadrant. All confusion matrices are normalized such that columns sum to 100%. The concentration of values along the diagonal indicates high annotation performance.

E Precision-recall curve for pulse depicts the performance characteristics of DeepSS for different confidence thresholds (from 0 to 1, black arrow points in the direction of increasing threshold). Recall decreases and precision increases with the threshold. The closer the curve to the upper and right border, the better. The red circle corresponds to the performance of DeepSS for a threshold of 0.7. The grey circle depicts the performance of FlySongSegmenter (FSS).

F Probability density function (PDF) of temporal errors for all detected pulses (red shaded area), computed as the distance between each pulse annotated by DeepSS and the nearest manually annotated pulse. Lines depict the median temporal error for DeepSS (red line, 0.3 ms) and FSS (grey line, 0.1 ms).

G, **H** Recall of DeepSS (red line) and FSS (grey line) as a function of the pulse carrier frequency (G) and signal-to-noise ratio (SNR) (H). Red shaded areas show the distributions of carrier frequencies (G) and SNRs (H) for all manual pulses. DeepSS outperforms FSS for all carrier frequencies and SNRs.

I Same as in D but for sine. Color indicates the percentage (see color bar) and text labels indicate seconds of sine for each quadrant.

J Same as in E but for sine. The blue circle depicts the performance for the confidence threshold of 0.5 used in A.

K Distribution of temporal errors for all detected sine on- and offsets. Median temporal error is 12 ms for DeepSS (blue line) and 22 ms for FSS (grey line).

L, **M** Recall for DeepSS (blue line) and FSS (grey line) as a function of sine duration (L) and SNR (M). Blue shaded areas show the distributions of durations and SNRs for all manually annotated sine songs. DeepSS outperforms FSS for all durations and SNRs.

controlled via the confidence threshold, which corresponds to the minimal confidence required for labelling a pulse (Fig. 1C): Lowering this threshold from 0.7 to 0.5 yields a recall of 99% for pulse song with a modest reduction in precision to 95%. The performance gain of DeepSS over FSS for pulse stems from better recall at high frequencies (>400Hz) and low SNR (Fig. 1G, H). To assess DeepSS performance for sine song, we evaluated the sample-wise precision and recall. DeepSS has similar precision to FSS (92% vs 91%) but higher recall (98% vs. 91%) (Fig. 1I, J). Recall is higher in particular for short sine songs (<100ms) and at low SNR (<1.0) (Fig. 1L, M). The performance boost for pulse and sine arises because DeepSS exploits context information, similar to how humans annotate song: For instance, DeepSS discriminates soft song pulses from pulsatile noise based on the pulse shape and also because song pulses occur in regular trains while noise does not (Fig. S2C).

The temporal precision of annotations is crucial, for instance for mapping sensorimotor transformations based on the timing of behavioral or neuronal responses relative to individual song elements (Benichov and Vallentin, 2020; Coen et al., 2014; Long and Fee, 2008; Srivastava et al., 2017). We therefore quantified the temporal error of the annotations produced by DeepSS. For pulse song, the temporal error was taken as the distance of each pulse annotated by DeepSS to the nearest true pulse. The median temporal error for pulse is 0.3 ms which is negligible compared to the average duration of a pulse (5-10ms) or of a pulse interval (~35-45ms) (Deutsch et al., 2019). For sine song, the median temporal error for on- and offsets was 12 ms, which is almost half of that of FSS (22 ms). Sine song can have low SNR (Fig. 1M) and fades in and out, making the precise identification of sine song boundaries difficult even for experienced manual annotators (see Fig. 1A, C).

During naturalistic interactions in larger behavioral chambers, multiple microphones are required to record the song produced by animals at any position in the chamber (Coen et al., 2014; Neunuebel et al., 2015). To demonstrate that DeepSS can process multi-channel audio, we trained DeepSS to annotate recordings from a chamber tiled with 9 microphones (Coen et al., 2014) (Fig. S3, S1B). As is the case for existing methods (Arthur et al., 2013), we achieved maximal performance by training separate networks for the pulse and for the sine song, likely because the different signal properties require different strategies for merging information across channels (Table S1). In multi-channel recordings, DeepSS annotates pulse song with 98% precision and 94% recall, and sine song with 97% precision and 93% recall, and matches the performance of FSS (FSS pulse precision/recall 99/92, sine 95/93) ((Fig. S3D-L). Annotations of multi-channel audio have high temporal precision for pulse (DeepSS 0.3 ms, FSS 0.1 ms) and sine (DeepSS 8 ms, FSS 15 ms) (Fig. S3E, J). Overall, DeepSS performs better or as well as the current state-of-the-art method for annotating single and multi-channel recordings of fly song.

Mouse ultrasonic vocalizations

Mice produce USVs in diverse social contexts ranging from courtship to aggression (Neunuebel et al., 2015; Sangiamo et al., 2020; Warren et al., 2020). We tested DeepSS using audio from an intruder assay, in which an anesthetized female was put into the home cage and the vocalizations produced by a resident female or male were recorded (Ivanenko et al., 2018). The female USVs from this assay typically consist of pure tones with weak harmonics and smooth frequency modulations that are often interrupted by frequency steps (Fig. 3A, B). The male USVs are similar but also contain complex frequency modulations not produced by the females in this assay (Fig. 3C, D). Recording noise from animal movement and interaction as well as the frequency steps often challenge spectral threshold-based annotation methods and tend to produce false positive syllables (Coffey et al., 2019; Tachibana et al., 2020). Moreover, weak signals often lead to missed syllables or imprecisely delimited syllables. We first trained and tested DeepSS on recordings of a female mouse interacting with an anesthetized female intruder (Fig. 3A). DeepSS annotates the female USVs with excellent precision (98%) and recall (99%) (Fig. 3E) and low median temporal error (0.3 ms) (Fig. 3F). DeepSS is robust to noise: Even for weak signals (SNR 1/16) the recall is ~90% (Fig. 3G). These performance values compare favorably to that of methods specialized to annotate USVs (Coffey et al., 2019; Tachibana et al., 2020; Van Segbroeck et al., 2017). USVs of female and male residents have similar characteristics (Fig. 3A, B) and the female-trained DeepSS network also accurately

annotated the male vocalizations (Fig. 3H). Notably, even the male syllables with characteristics not seen in females in the paradigm were detected (Fig. 3D). Overall, DeepSS accurately and robustly annotates mouse USVs and generalizes across sexes.

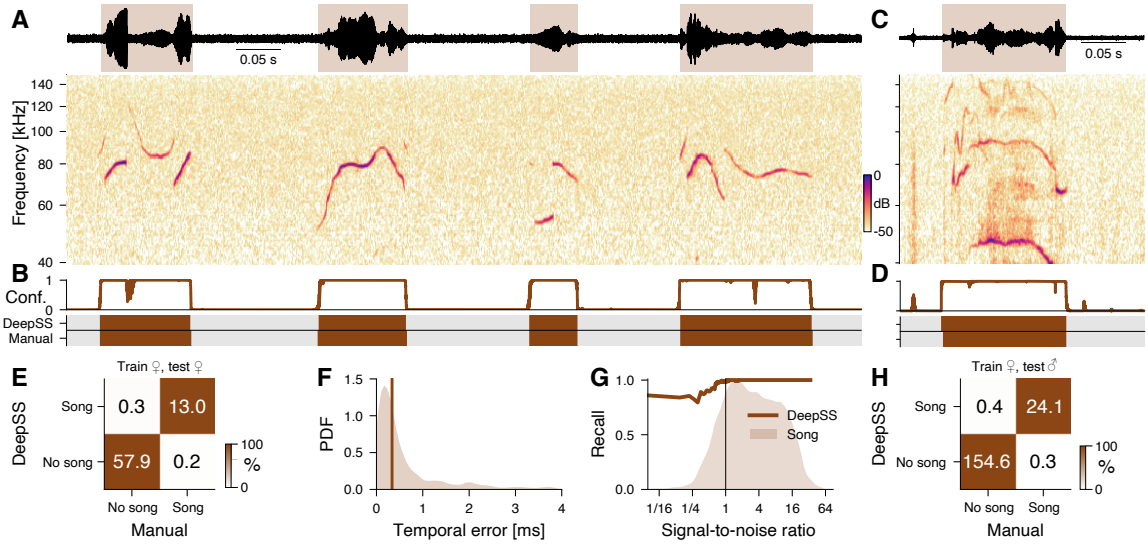

Figure 2: DeepSS performance for mouse ultrasonic vocalizations.
A Waveform (top) and spectrogram (bottom) of USVs produced by a female mouse in response to an anesthetized female intruder. Shaded areas (top) show manual annotations.
B Confidence scores (top) and DeepSS and manual annotations (bottom) for the female USVs in A. Brief gaps in confidence are filled to ensure smooth annotations.
C Example of male USVs with sex-specific characteristics produced in the same assay.
D Confidence scores (top) and DeepSS and manual annotations (bottom) for the male USVs in C from a DeepSS network trained to detect female USVs.
E Confusion matrix from a female-trained network for a test set of female USVs. Color indicates the percentage (see color bar) and text labels the seconds of song in each quadrant.
F Distribution of temporal errors for syllable on- and offsets in female USVs. Brown line indicates the median temporal error (0.3 ms).
G Recall of the female-trained network (brown line) as a function of SNR. Brown shaded area represents the distributions of SNRs for all USV samples. Recall is high even at low SNR.
H Confusion matrix of the female-trained DeepSS for a test set of male USVs (see C, D for examples). Color indicates the percentage (see color bar) and text labels the seconds of song in each quadrant.

Bird song

Bird song is highly diverse and can consist of large, individual-specific repertoires. The spectral complexity and large diversity of the song complicates the annotation of syllable types. Traditionally, syllable types are annotated based on statistics derived from the segmented syllable spectrogram. Recently, good annotation performance has been achieved with unsupervised methods (Goffinet et al., 2019; Sainburg et al., 2019) and specialized deep neural networks (Cohen et al., 2020; Koumura and Okanoya, 2016). We trained DeepSS to annotate the song from four male Bengalese finches (data and annotations from (Nicholson et al., 2017)). The network was then tested on a random subset set of the recordings from all four individuals which contained 37 of the 48 syllable types from the training set (Fig. S4A, B). DeepSS annotates the bird song with high accuracy: Sample-wise precision and recall are both 97% and syllable on- and offsets are detected with sub-millisecond precision (median temporal error 0.3 ms). The types of 98.5% off all syllables are correctly annotated (Fig. 4D), with only 0.3% false positives (noise annotated as a syllable), 0.2% false negatives (syllables annotated as noise), and 1% type confusions (Fig. S4C-E). This results in a low sequence error (corresponding to the minimal number of substitutions, deletions, or insertions required to transform the true sequence of syllables into the inferred one) of 0.012. Thus, DeepSS performs as

well as or better than specialized deep learning-based methods for annotating bird song (Cohen et al., 2020; Koumura and Okanoya, 2016).

In summary, DeepSS accurately and robustly annotates a wide range of signals—from the pulsatile song pulses of flies to the spectrally complex syllables in bird song.

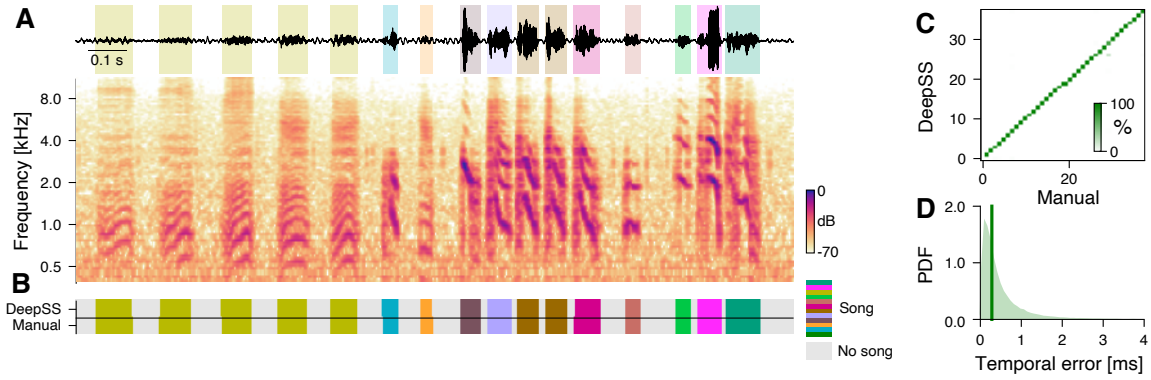

Figure 3: DeepSS performance for bird song.
A Waveform (top) and spectrogram (bottom) of the song of a male Bengalese finch. Shaded areas (top) show manual annotations colored by syllable type.
B DeepSS and manual annotation labels for the different syllable types in the recording in A (see color bar). DeepSS accurately annotates the syllable boundaries and types.
C Confusion matrix for the different syllables in the test set (see color bar). Rows depict the probability with which DeepSS annotated each syllable as any of the 37 types in the test dataset. The type of 98.5% of the syllables were correctly annotated, resulting in the concentration of probability mass along the main diagonal.
D Distribution of temporal errors for the on- and offsets of all detected syllables (green shaded area). Green line indicates the median temporal error (0.28 ms).

DeepSS is fast

To efficiently process large corpora of recordings and to be suitable for closed-loop applications, DeepSS needs to infer annotations quickly. We therefore assessed the throughput and latency of DeepSS. Throughput measures the rate at which DeepSS annotates song and high throughput means that large datasets are processed quickly. Across the three species tested here, DeepSS has a throughput of 8-15x realtime on a CPU and of 24-91x on a GPU (Fig. 4A). This means that a 60-minute recording is annotated in less than 5 minutes on a standard desktop PC and in less than 1.5 minutes using a GPU, making the annotation of large datasets feasible (Fig. S5A-F). A measure of speed crucial for closed-loop experiments is latency, which quantifies the time it takes to annotate a chunk of song and determines the delay for experimental feedback. Latencies are short, between 7 and 15 ms (Fig. 4B, Fig. S5G-L) on CPUs and GPUs. One network parameter impacting latency is the chunk size—the duration of audio processed at once—and we find that for fly song, latency can be optimized by reducing chunk size with a minimal impact on accuracy (Fig. S6). The low latency of annotation makes DeepSS well suited for triggering realtime optogenetic or acoustic feedback upon the detection of specific vocalizations (Bath et al., 2014; Stowers et al., 2017).

DeepSS requires little manual annotation

To be practical, DeepSS should achieve high performance with little manual annotation effort. We find that DeepSS can be efficiently trained using an iterative protocol (Pereira et al., 2018) (Fig. 4C, S8): Annotate a small set of recordings and train the network for a few epochs; then generate annotations on a larger set of recordings and correct these annotations. Repeat the predict-correct-train cycle on ever larger datasets until performance is satisfactory. To estimate the amount of

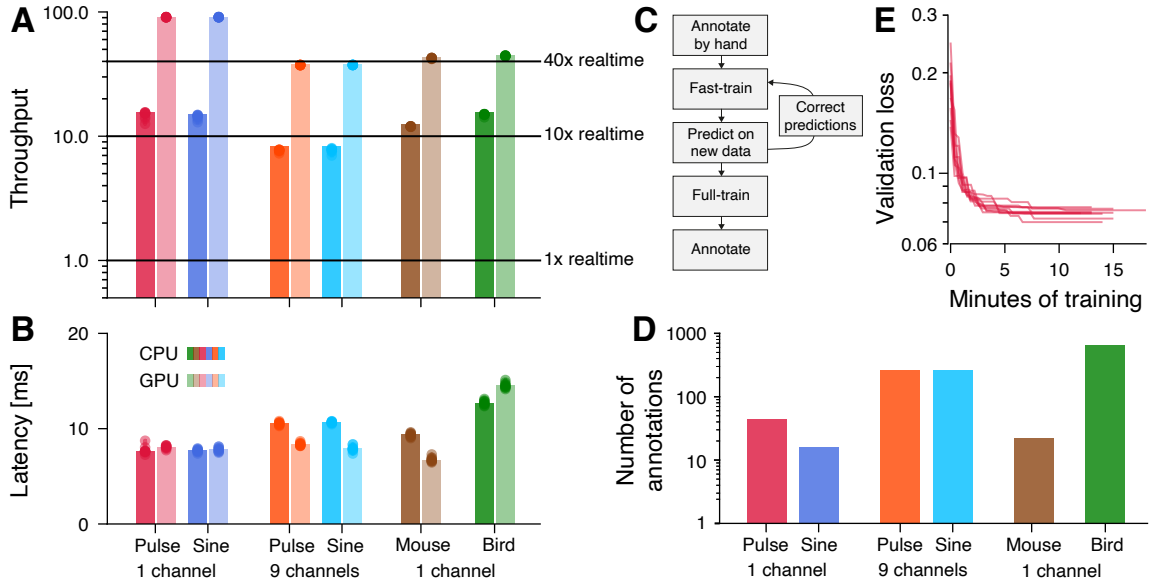

Figure 4: DeepSS annotates song with high throughput and low latency and requires little data.

A, **B** Throughput (A) and latency (B) of DeepSS. Throughput (A) was quantified as the amount of audio data in seconds annotated in one second of computation time. Horizontal lines in A indicate throughputs of 1, 10, and 40. Throughput is around 10x realtime on a CPU (dark shades) and 40x or more on a GPU (light shades). Latency (B) corresponds to the time it takes to annotate a single chunk of audio and is similar on CPU (dark shades) and GPU (light shades). Multi-channel audio from flies was processed using separate networks for pulse and sine. For estimating latency of fly song annotations, we used networks with 25 ms chunks, not the 410 ms chunks used in the original network (see Fig. S6).

C Flow diagram of the iterative protocol for fast training DeepSS.

D Number of manual annotations required to reach 90% of the performance of DeepSS trained on the full data set (see Fig. S7 and Figs. 1, S3, 2, 3). Performance was calculated as the F1 score, the geometric mean of precision and recall. DeepSS requires small to modest amounts of manual annotations.

E Current best validation loss during training for fly pulse song recorded on a single channel for 10 different training runs (red lines, 18 minutes of training data). The network robustly converges to solutions with low loss after fewer than 15 minutes of training (~40 epochs).

manual annotations required to achieve high performance, we evaluated DeepSS trained on subsets of a larger training data set (Fig. S7). We then took the number of manual annotations needed to reach 90% of the performance of DeepSS trained on the full data sets as an upper bound on the data requirements (Fig. 4D). Performance was taken as the F1 score, the geometric mean of precision and recall. For single-channel recordings of fly song, fewer than 50 pulses and 20 sine songs are sufficient to reach 90% of the performance achieved with the full data set. For mouse vocalizations, DeepSS achieves ~90% of its peak performance with fewer than 25 manually annotated syllables. Manually annotating such small amounts of fly and mouse song takes less than 5 minutes. Likewise, for bird song, 1-51 (median 8, mean 17) manual annotations are required per syllable type, with one outlier requiring ~200 syllables (Fig. S7C). Closer inspection reveals that the outlier results from an annotation error and consists of a mixture of three distinct syllable types (Fig. S7D-F). Even with this outlier, only 626 manually annotated syllabes (424 without) are required in total to reach 90% of the test performance of a network trained on >3000 annotated syllables. Data requirements are higher for the multi-channel recordings of fly song (~270 pulses and sine songs), since the network has to learn to combine information across all nine channels, but even in this case, the iterative training protocol reduces the manual annotation work. Overall, DeepSS requires small to moderate amounts of data for reaching high performance. Small training data sets and high throughput (Fig. 4A) translate to short training times (Fig. 4E). The single-channel data sets typically achieve 90% of the performance after less than 10 minutes of training on a GPU. Training on the full data sets typically finishes in less than 5 hours. Thus, DeepSS can be adapted to novel species in short time and with little manual annotation work.

DeepSS can be combined with unsupervised methods

DeepSS is a supervised annotation method: It discriminates syllable types that have been manually assigned different labels during training. By contrast, unsupervised methods can determine in unlabelled data whether syllables fall into distinct types and if so, classify the syllables (Arthur et al., 2021; Clemens et al., 2018; Coffey et al., 2019; Goffinet et al., 2019; Sainburg et al., 2019; Sangiamo et al., 2020; Tabler et al., 2017). While DeepSS does not require large amounts of manual annotations (Fig. 4D), manual labeling of syllable types can be tedious when differences between syllable types are subtle (Clemens et al., 2018) or when repertoires are large (Sangiamo et al., 2020)In these cases, combining DeepSS with unsupervised methods can be further reduce annotation work. To demonstrate the power of this approach, we use common procedures for unsupervised classification, which consist of an initial preprocessing (e.g. into spectograms) and normalization (e.g. of amplitude) of the syllables, followed by dimensionality reduction and clustering (see Methods) (Clemens et al., 2018; Sainburg et al., 2019; Sangiamo et al., 2020). For fly song, DeepSS was trained to discriminate two major song modes, pulse and sine. However, *Drosophila melanogaster* males produce two distinct pulse types, termed P_{slow} and P_{fast} (Clemens et al., 2018), and unsupervised classification robustly discriminates the two pulse types as well as the sine song in the DeepSS annotations (Fig. 5A-C). Mouse USVs do not fall into distinct types (Goffinet et al., 2019; Sainburg et al., 2019; Tabler et al., 2017). In this case, unsupervised clustering produces a low-dimensional representation that allows to group the syllables by the similarity of their spectrograms (Coffey et al., 2019; Sangiamo et al., 2020; Tabler et al., 2017) (Fig. 5D, E). Lastly, for bird song, DeepSS was trained to discriminate dozens of syllable types and the unsupervised method re-discovers these pre-defined labels (Goffinet et al., 2019; Sainburg et al., 2019) (Fig. 5F, G). The unsupervised classification also reveals manual annotation errors: For instance, the bird song syllable that required >200 manual annotations to be annotated correctly by DeepSS (Fig. S7C) is a mixture of three clearly distinct types (Fig. S7D–F). Thus, unsupervised methods simplify annotation work: DeepSS can be trained using annotations that do not discriminate between syllables types and the types can be determined post hoc. If distinct types have been established, DeepSS can be retrained to directly annotate these types using the labels produced by the unsupervised method as training data.

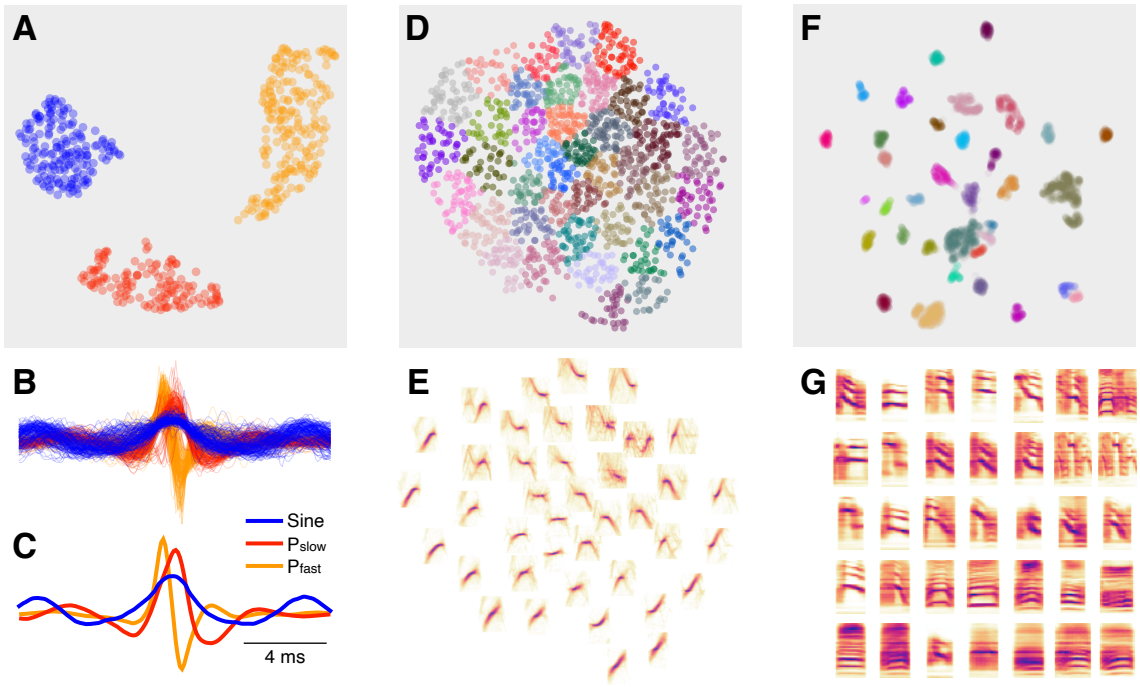

Figure 5: DeepSS can be combined with unsupervised methods for song classification.
A Low-dimensional representation obtained using the UMAP method of all pulse and sine song waveforms annotated by DeepSS in a test data set. Data points correspond to individual waveforms and were clustered into three distinct types (colors) using the density-based method HDBSCAN.
B, **C** All waveforms (B) and cluster centroids (C) in A colored by the cluster assignment. Waveforms cluster into one sine (blue) and two pulse types with symmetrical (red, P_{slow}) and asymmetrical (orange, P_{fast}) shapes.
D Low-dimensional representation of the spectrograms of mouse USVs. Individual syllables (points) form a continuous space without distinct clusters. Song parameters vary continuously within this space, and syllables can be grouped by the similarity of their spectral contours using k-means clustering.
E Mean spectrogram for each cluster in D.
F Low-dimensional representation of the spectrograms of bird song syllables. Syllables separate into distinct types and density-based clustering (colors) produces a classification of syllables that closely matches the manual annotations (homogeneity score 0.96, completeness score 0.89, v-score 0.92).
G Mean spectrogram for each cluster in F.

Discussion

We present Deep Song Segmenter (DeepSS), an architecture for annotating acoustic signals. We show that DeepSS annotates song in single- and multi-channel recordings from flies (Fig. 1), mice (Fig. 2), and birds (Fig. 3) accurately, robustly, and quickly (Fig. 4A, B).

Using a user-friendly graphical interface, our method can be optimized for new species without requiring expert knowledge and with little manual annotation work (Fig. 4C-E), making automatic annotation and analyses of large corpora of recordings from diverse species accessible. Annotation burden can be further reduced using unsupervised classification of syllable types in particular for species with large or individual-specific repertoires (Fig. 5) (Arthur et al., 2021; Clemens et al., 2018; Coffey et al., 2019; Goffinet et al., 2019; Sainburg et al., 2019; Tabler et al., 2017).

The high inference speed (Fig. S5) allows integration of DeepSS in closed-loop systems in which song is detected and stimulus playback or optogenetic manipulation is triggered with low latency (Bath et al., 2014; Stowers et al., 2017). In combination with realtime pose tracking (Graving et al., 2019; Mathis et al., 2018; Pereira et al., 2018), this provides unique opportunities to tailor optogenetic manipulations to specific behavioral contexts, for instance to dissect the neural circuits underlying acoustic communication in interacting animals (Coen et al., 2014; Fortune et al., 2011; Okobi et al., 2019).

We consider DeepSS a starting point and believe that future extensions of the method can further reduce data requirements. For instance, self-supervised or transfer learning (Devlin et al., 2018; Mathis et al., 2019; Raghu et al., 2019) can produce networks with a general and rich representation of sound features that can be fine-tuned for species-specific tasks using few samples. In addition, DeepSS currently does not work well with recordings in which the signals produced by multiple animals overlap. In the future, DeepSS will be extended with methods for multi-speaker speech recognition to robustly annotate vocalizations from animal groups.

Acknowledgments

We thank Kurt Hammerschmidt (DPZ, Göttingen) for providing mouse data and Sam Sober and David Stern for making publicly available song recordings of birds and flies. We thank Mala Murthy, David Stern, and all members of the Clemens lab for feedback on the manuscript.

This work was supported by the DFG through grants CL 596/1-1 (Emmy Noether) and CL 596/2-1 (SPP2205) to JC and by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Starting Grant agreement No. 851210 NeuSoSen).

Methods

Instructions for installing and using DeepSS can be found at https://janclemenslab.org/deepss. Code for the *dss* module is available at https://github.com/janclemenslab/deepss, code for the unsupervised methods is at https://github.com/janclemenslab/deepss_unsupervised.

Data sources

All data used for testing DeepSS was published previously. Single-channel recordings of *Drosophila melanogaster* (strain OregonR) males courting females were taken from (Stern, 2014) (deposited at https://www.janelia.org/lab/stern-lab/tools-reagents-data). The multi-channel data from *Drosophila melanogaster* (strain NM91) males courting females was recorded in a chamber tiled with nine microphones (Coen et al., 2014) and was previously published in (Clemens et al., 2018). Annotations for fly song were seeded with FlySongSegmenter (Arthur et al., 2013; Coen et al., 2014) and then manually corrected. Recordings of mouse USVs were kindly provided by Kurt Hammerschmidt (DPZ, Göttingen, Germany) and were previously published in (Ivanenko et al., 2018). The USVs were manually labelled using the DeepSS graphical interface. To test DeepSS on bird song data, we used a publicly available, hand-labeled collection of song from four male Bengalese finches (Nicholson et al., 2017).

DeepSS network

DeepSS is implemented in Keras (Chollet and others, 2015) and Tensorflow (Abadi et al., 2016). At its core, DeepSS consists of a stack of temporal convolutional blocks, which transform an input sequence of audio data into an output sequence of labels.

Inputs: DeepSS takes as input raw, single or multi-channel audio. Pre-processing of the audio using a Wavelet or short-time Fourier transform is *not* required. DeepSS processes audio in overlapping chunks (Fig. S1A-D). The chunking accelerates annotations since multiple samples are annotated in a single computational step. Edge effects are avoided by processing overlapping chunks and by discarding a number of samples at the chunk borders. The overlap depends on the number of layers and the duration of filters in the network.

STFT frontend: The trainable STFT frontend is an *optional* step and was implemented using kapre (Choi et al., 2017). Each frequency channel in the output of the frontend is the result of two, one-dimensional strided convolutions which are initialized with the real and the imaginary part of discrete Fourier transform kernels:

$$x(i,f) = \sum_{\tau=0}^{\mathrm{T}-1} x(i*s+\tau)[\cos(2\pi f\tau/\mathrm{T}) - i\sin(2\pi f\tau/\mathrm{T})]$$

$$y(i,f) = \log_{10}(\sqrt{\Re x(i,f)^2 + \Im x(i,f)^2})$$

where f is the frequency, s is the stride, and T is the filter duration. The stride results in downsampling of the input by a factor s.

The kernels are optimized with all other parameters of the network during training. The STFT frontend was used for mouse and bird song, but not for fly song. In the mouse and bird networks, we used 33 STFT filter pairs with a duration $\mathrm{T} = 64$ samples and a stride $s = 16$ samples. The downsampling sped up annotations without reducing accuracy for mouse and bird song. For fly song, the STFT frontend tended to reduce performance and was omitted.

Temporal convolutional blocks: Temporal convolution (TCN) blocks are central to DeepSS and produce a task-optimized hierarchical representation of sound features at high temporal resolution (Bai et al., 2018). Each TCN block consists of a stack of so-called residual blocks (Fig. S1E) (He et al., 2015), which are composed of:

- A *dilated convolutional layer* filters the input with a number of kernels of a given duration: $y_i(t) = \sum_{\tau,\gamma} k_i(\tau,\gamma) * x(t - a\tau, \gamma)$, where $k_i(\tau,\gamma)$ is the ith kernel, $x(t,\gamma)$ is the input on channel γ at time t, $y_i(t)$ the output and a the gap or skip size (Yu and Koltun, 2015). In old-fashioned convolution $a = 1$. Increasing a allows the kernel to span a larger range of inputs with the same number of parameters and without a loss of output resolution. The number of parameters is further reduced for networks processing multi-channel audio, by using *separable* dilated convolutions in the first two TCN blocks (Mamalet and Garcia, 2012). In separable convolutions, the full two-dimensional $k(\tau,\gamma)$ convolution over times and channels is decomposed into two one-dimensional convolutions. First, a temporal convolution, $k^t(\tau, 1)$, is applied to each channel and then several channel convolutions, $k^\gamma(1,\gamma)$, combine information across channels. Instead of $\tau \times \gamma$ parameters, the separable convolution only requires $\tau + N \times \gamma$ parameters. Note that each temporal convolution is applied to each channel, leading to a sharing of filter parameters across channels. This makes explicit the intuition that some operations should be applied to all channels equally. We also tested an alternative implementation, in which individual channels are first processed separately by a single-channel TCN, the output of the TCN blocks for each channel are concatenated, and then fed into a stack of standard TCNs with full two-dimensional convolutions. While this architecture slightly increased performance it was also much slower and we therefore chose the architecture with separable convolutions. Architecture choice ultimately depends on speed and performance requirements of the annotation task.
- A *rectifying linear unit* transmits only the positive inputs from the dilated convolutional layer by setting all negative inputs to 0: $y_i = max(0, y_i)$.
- A *normalization layer* rescales the inputs to have a maximum absolute value close to 1: $y_i/(max(|y_i|) + 10^{-5})$.
- The output of the residual block, $z(t)$, is then routed to two targets: First, it is added to the input: $o(t) = x(t) + z(t)$ and fed into subsequent layers. Second, via a so-called skip connection, the outputs of all residual blocks are linearly combined to produce the network's final output (Oord et al., 2016).

A single TCN block is composed of a stack of five residual blocks. Within a stack, the skip size a doubles — from 1 in the first to $2^5 = 16$ in the final residual block of a stack. This exponential increase in the span of the filter $\tau * a$ allows the TCN block to produce a hierarchical representation of its inputs, from relatively low-level features on short timescales in early stacks to more derived features on longer timescales in late stacks. Finally, a full network consists of a stack of 3 or 4 TCN blocks, which extract ever more derived features (Fig. S1A-D).

Output: The network returns a set confidence scores—one for each song type (and for no song)—for each sample from a linear combination of the output of each residual block in the network, by using a single dense layer and a softmax activation function. Re-using information from all blocks via so-called skip connections ensures that downstream layers can discard information from upstream layers and facilitates the generation of specialized higher-order presentations. If the input recording got downsampled by a STFT frontend, a final upsampling layer restores the confidence scores to the original audio rate by repeating values.

The parameters of all used networks are listed in Table S1.

Training

Networks were trained using the categorical cross-entropy loss and the Adam optimizer (Kingma and Ba, 2014) with a batch size of 32. Prediction targets were generated from fully annotated recordings and one-hot-encoded: Segments were coded as binary vectors, with $y_i(t) = 1$ if a segment of type i occurred at time t, and $y_i(t) = 0$ otherwise. To encode uncertainty in the timing of fly song pulses, the pulses were represented as Gaussian bumps with a standard deviation of 1.6 ms. A "no song" type was set to $y_{\text{no song}}(t) = 1 - \sum_i y_i(t)$. That way, y corresponds to the probability of finding any of the annotated song types or no song. For bird song, short gaps (6.25 ms, 200 samples at

32 kHz) were introduced between adjacent syllables to aid the detection of syllable on- and offsets after inference.

Typically, multiple fully annotated recordings were combined in a data set. 10% of the recordings were held out for testing the model and the remaining recordings were split 90:10 into a training and a validation set. The validation set was randomly taken from the first, the middle or the last 10% of each recording. For bird song, each recording was split 90:10:10 into a training, validation and test set. Training was set to stop after 400 epochs or earlier if the validation loss was not reduced for at least 20 epochs. Training typically stopped within 40-80 epochs depending on the dataset. The test set was only used for evaluating the model performance after training.

Generation of annotations from the network output

The confidence scores produced by the model correspond to the sample-wise probability for each song type. To produce an annotation label for each sample, the confidence scores were further processed to extract event times and syllable segments.

Event times for event-like song types like fly pulse song were determined based on local maxima in the confidence score, by setting a threshold value between 0 and 1 and a minimal distance between subsequent peaks (using `peakutils` (Negri and Vestri, 2017)). For the pulse song of flies, we set a minimal distance of 10 ms and a threshold of 0.7 for single channel data (Fig. 1) and 0.5 for multi-channel data (Fig. S3).

For segment-like song types like fly sine song or the syllables of mouse and bird song, we first transformed the sample-wise probability into a sequence of labels using $o(t) = \text{argmax}_i\ y(i,t)$. The resulting annotation of segments was then smoothed by filling short gaps (flies 20 ms, mice 10 ms, birds 5 ms) and removing very short detections (flies 20 ms, mice 5 ms, birds 30ms). These values were chosen based on the statistics of song found in the annotation data. Syllable on- and offsets were detected as changes from no-song to song and song to no-song, respectively. For bird song, syllable labels were determined based on a majority vote, by calculating the mode of the sample-wise labels for each detected syllable.

Evaluation

DeepSS was evaluated on segments of recordings that were not used during training.

Events: For events—fly pulse song, or the on- and offsets of segments—we matched each true event with its nearest neighbor in the list of true events and counted as true positives only events within a specified distance from a true event. For the pulse song of flies as well as for the onsets and offsets of mouse and bird syllables, this distance was set to 10 ms. Results were robust to the specific choice of the distance threshold (Fig. S2A). For the onsets and offsets of fly sine song, we set this distance to 40 ms, since sine song tends to fade in and out, making the delineation of exact boundaries difficult. False positive events were counted if the distance from a detected event to the nearest true event exceeded the distance threshold or if another detected event was closer to each true event within the distance threshold. If several detected pulses shared the same nearest true pulses, only the nearest of those was taken as a true positive, while the remaining detections were matched with other true pulses within the distance threshold or counted as false positives.

False negatives were counted as all true events without nearby detected events. For pulse, true negative events were estimated as the number of tolerance distances (2x tolerance distance) fitting into the recording, minus the number of pulses. These true negatives for pulse do not influence precision, recall, and F1-scores and are only used for the confusion matrices in Figs. 1D, I and S3C, H. Pulse and sine song were evaluated only up to the time of copulation.

Matching segment labels: For songs with only one syllable type, we compared the predicted and true labels for each sample to compute the confusion matrix (Fig. S1F, G). In the case of multiple syllable types, the mode of the true and predicted labels for the samples of each detected

syllable were compared. A true positive was counted if the mode of the true labels was the same for the samples covered by the detected syllable. Using the true syllables as reference produces near identical results (Fig. S4D, E).

Performance scores: From the false negative (FN), false positive (FP), and true positive (TP) counts we extracted several scores: Precision (P)—the fraction of true positive out of all detections TP/(FP+TP)—and recall (R)—the fraction of true positives out of all positives TP/(TP+FN). The F1 score combines precision and recall via their geometric mean: $2 \times P \times R/(P+R)$. Since the birdsong used contained dozens of different syllable types, we used as a summary measure of performance the accuracy—the fraction of correctly labelled syllables: (TP+TN)/(TP+TN+FP+FN). For comparison with other studies, we also provide the error rate for bird song, which is based on the Levenshtein edit distance and corresponds to the minimal number of inserts, deletions, and substitutions required to transform the sequence of true syllable labels into the sequence of predicted syllable labels normalized by the length of the true sequence (Cohen et al., 2020; Koumura and Okanoya, 2016).

Temporal precision: The temporal precision for events (pulses, syllable onsets and offsets) was calculated as the median absolute distance between all matched events.

Bird song annotation errors: The network for annotating bird song was trained on all data. We removed from the test data one syllable type with only a single instance in the test set (which was correctly classified), because the performance could not be assessed reliably based on a single instance. We also excluded as annotation error a syllable type that contained syllables of more than 6 distinct types.

Estimation of signal-to-noise ratio from audio recordings

To assess the robustness of annotation performance to noise, we assessed the recall of DeepSS for epochs with different signal-to-noise ratios (SNRs). Because of fundamental differences in the nature of the signals, SNR values were computed with different methods and are therefore not directly comparable across species.

Pulse: Pulse waveforms were 20 ms long and centered on the peak of the pulse energy. The root-mean square (RMS) amplitudes of the waveform margins (first and last 5 ms) and center (7.5-12.5 ms) were taken as noise and signal, respectively. RMS is defined as $\sqrt{\sum_i x(i)^2}$. For multi-channel recordings, the pulse waveform from the channel with the highest center RMS was chosen to calculate the SNR.

Sine: Signal was given by the RMS amplitudes of the recording during sine song. Noise is the RMS amplitude in the 200 ms before and after each sine song, with a 10 ms buffer. For instance, if a sine song ended at 1000 ms, the recording between 1010 and 1210 ms was taken as noise. From the 200 ms noise, we excluded samples that were labelled as sine or pulse and included intervals between pulses. For multi-channel recordings, the SNR was averaged across channels.

Mouse: We assumed an additive noise model: $\sigma^2_{total} = \sigma^2_{signal} + \sigma^2_{noise}$. σ^2 is the squared signal averaged over a 1ms window. Since noise variance changed little relative to the signal variance in our recordings, we can assume constant noise over time to calculate the signal strength: $\sigma^2_{signal} = \sigma^2_{total} - \sum_t \sigma^2_{noise}$). The sample-wise SNR is then given by $SNR(t) = \sigma_{signal}(t)^2 / \sum_t \sigma^2_{noise}$.

Speed benchmarks

Inference speed was assessed through throughput and latency. Throughput is the number of samples annotated per second and latency is the time it takes to annotate a single chunk. Throughput and latency depend on the chunk duration—the duration of a recording snippet processed by the network at once—and on the batch size—the number of chunks processed during one call. Larger batch sizes maximize throughput by more effectively exploiting parallel computation in modern CPUs and GPUs but this comes at the cost of higher latency, since results are available only after

all chunks in a batch have been processed. Using small batch sizes and short chunks therefore reduces latency, since results are available earlier, but this comes at the cost of reduced throughput because of overhead from data transfer or under-utilized parallel compute resources. To assess throughput and latency, run times of `model.predict` were assessed for batch sizes ranging from 1 to 1024 (log spaced) with 10 repetitions for each batch size after an initial warmup run (Fig. S5A-L). Results shown in the main text are from a batch size corresponding to ~1s of recording for throughput (Fig. 4A) and a batch size of 1 for latency (Figs. 4B, see also Fig. S5). For fly song, latency was optimized by reducing the chunk size to 25.6ms (Fig. S6).

Benchmarks were run on Windows 10, Tensorflow 2.1, with the network either running on a CPU (Intel i7-7770, 3.6GHz) or on a GPU (GTX1050 Ti 4GB RAM).

Data economy

For estimating the number of manual annotations required, we trained the networks using different fractions (0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0) of the full training and validation sets. Performance of all networks trained on different subsets was evaluated on the full test set. The number of manual annotations in each subset was estimated after training. The number of annotations required to exceed 90% of the F1 score of a model trained on the full data sets was calculated based on a lowess fit (Cleveland, 1979) to the data points (Fig. 4A, B).

Unsupervised classification

Sets of signals from flies, mice and birds were clustered using unsupervised methods described previously Sangiamo et al. (2020). First, signals were pre-processed: For fly song, pulse and sine waveforms of duration 15 ms were extracted from the recording, aligned to their peak energy, normalized to unit norm, and adjusted for sign (see (Clemens et al., 2018) for details). For mouse and bird song, we adapted the procedures described in (Sainburg et al., 2019): Noise was reduced in the bird song recordings using the `noisereduce` package (https://github.com/timsainb/noisereduce). For mouse vocalizations, noise reduction tended to blur the spectral contours and was omitted. Then, syllable spectrograms were extracted from mel spectrograms of the recordings. The noise floor of the spectrogram at each frequency was estimated as the median spectrogram over time and each spectral band was then divided by the frequency-specific noise floor value. Finally, the spectrogram values were $\log_2$-transformed and thresholded at 0 for mice and 2 for birds after visual inspection of the spectrograms to further remove background noise. To reduce differences in the duration of different syllables, all syllables were first log resized in time (scaling_factor=8) and then padded with zeros to the duration of the longest syllable in the data set. Lastly, the frequency axis of the spectrograms for mouse syllables were aligned to the peak frequency, to make clustering robust to jitter in the frequency of the thin spectral contours (Sangiamo et al., 2020). The peak frequency of each syllable was calculated from its time-averaged spectrogram, and only the 40 spectrogram frequencies around the peak frequency were retained.

The dimensionality of the pre-processed waveforms (fly) or spectrograms (mouse, bird) was then reduced to two using the UMAP method (McInnes et al., 2018) (mindist=0.5, results are robust to the choice of this parameter). Finally, signals were grouped into types using unsupervised clustering. For fly and bird song, the UMAP distribution revealed distinct groups of syllables and we used a density-based method to cluster the syllables ((Campello et al., 2013), min_samples=10, min_cluster_size=20). For mouse USVs, no clusters were visible in the UMAP distribution and density-based clustering failed to identify distinct groups of syllables. Syllables were therefore split into 40 groups using k-means clustering.

Open source software used

- avgn [https://github.com/timsainb/avgn_paper]
- hdbscan (McInnes et al., 2017)

- ipython (Perez and Granger, 2007)
- jupyter (Kluyver et al., 2016)
- kapre (Choi et al., 2017)
- keras (Chollet and others, 2015)
- keras-tcn [https://github.com/philipperemy/keras-tcn]
- librosa (McFee et al., 2015)
- matplotlib (Hunter, 2007)
- noisereduce [https://github.com/timsainb/noisereduce]
- numpy (Harris et al., 2020)
- pandas (McKinney, 2010)
- peakutils (Negri and Vestri, 2017)
- scikit-learn (Pedregosa et al., 2011)
- scipy (Virtanen et al., 2020)
- seaborn (Waskom et al., 2017)
- snakemake (Köster and Rahmann, 2012)
- tensorflow (Abadi et al., 2016)
- UMAP (McInnes et al., 2018)
- zarr (Miles et al., 2020)
- xarray (Hoyer and Hamman, 2017)

Supplemental table and figures

Table S1A: **Network parameters**

Species	Trained	Sample rate [kHz]	Chunk duration (samples)	Audio channels	STFT down-sampling	Separable conv	TCN stacks	Kernel size	Kernels
D. mel.	pulse & sine	10	4096	1	-	-	3	32	32
D. mel. multi channel	pulse	10	2048	9	-	TCN stacks 1+2	4	32	32
D. mel. multi channel	sine	10	2048	9	-	TCN stacks 1+2	4	32	32
D. mel. multi channel	pulse & sine	10	2048	9	-	-	3	32	32
mice	female song	300	8192	1	4x	-	2	16	32
Beng. finches	song of 4 males	32	1024	1	4x	-	4	32	64

Table S1B: **Network performance**

Species	Trained	Classes	Threshold	Precision %	Recall %	Temporal error [ms]
D. mel.	pulse & sine	3	0.7	97/92 (p/s)	96/98 (p/s)	0.3/12 (p/s)
D. mel. multi channel	pulse	2	0.5	98	94	0.3
D. mel. multi channel	sine	2	0.5	97	93	8
D. mel. multi channel	pulse & sine	3	0.5	97/98 (p/s)	94/88 (p/s)	0.3/6 (p/s)
mice	female	2	0.5	98	99	0.3
Beng. finches	4 males	49	0.5	97	97	0.3

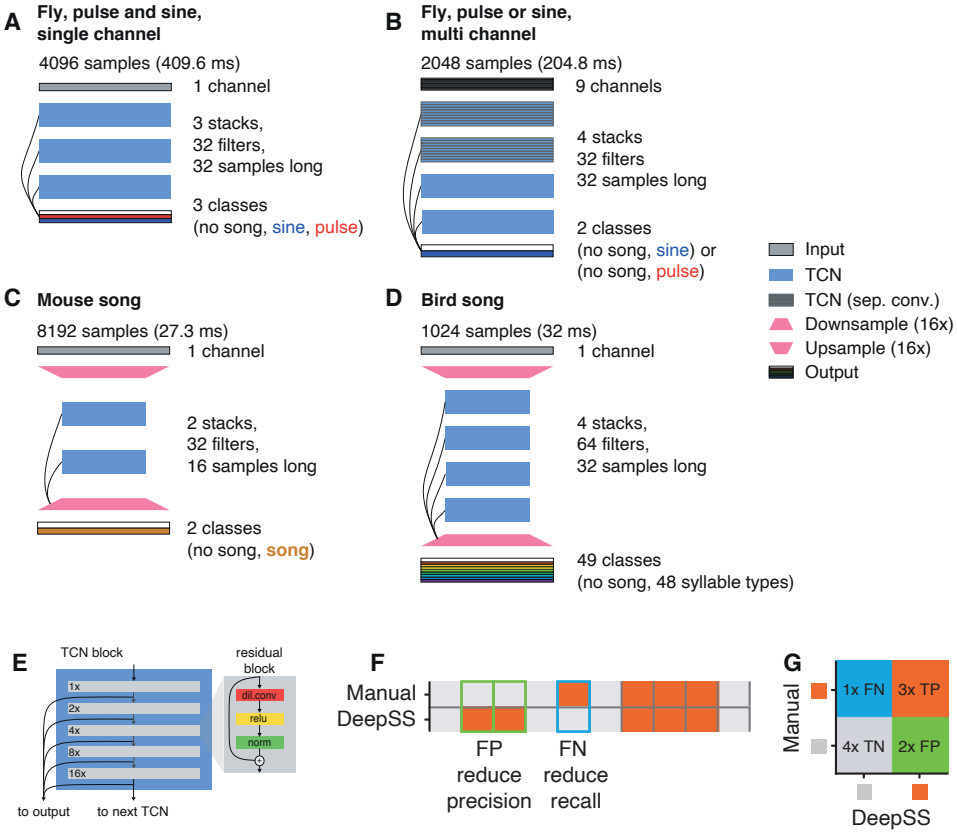

Figure S1: DeepSS architectures and evaluation.
A-D Network architectures for annotating fly song from single (A) and multi-channel (B) recordings, mouse USVs (C), and bird song (D). See legend to the left. Each TCN stack consists of stacks of residual blocks shown in E. See Table S1 for a all network parameters.
E A TCN block (left) consists of a stack of 5 residual blocks (right). Residual blocks process the input with a sequence of dilated convolution, rectification (ReLU) and normalization. The output of this sequence of steps is then added to the input. In successive residual blocks, the dilation rate of the convolution filters doubles from 1x in the first to 16x in the last layer (see numbers to the left of each block). The output of the last residual block is passed as an input to the next TCN block in the network. In addition, the outputs of all residual blocks in a network are linearly combined to predict the song.
F Annotation performance is evaluated by comparing manual annotations (top) with labels produced by DeepSS (bottom). Grey indicates no song, orange song. True negatives (TN) and true positives (TP) are samples for which DeepSS matches the manual labels. False negatives (FNs) are samples for which the song was missed (blue frame) and reduce recall (TP/(FN+TP)). False positives (FP) correspond to samples that were falsely predicted as containing song (green frames) and reduce precision (TP/(TP+FP)).
G Precision and recall are calculated from a confusion matrix which tabulates TP (orange), TN (grey), FP (green), FN (blue). In the example, precision is 3/(3+2) and recall is 3/(1+3).

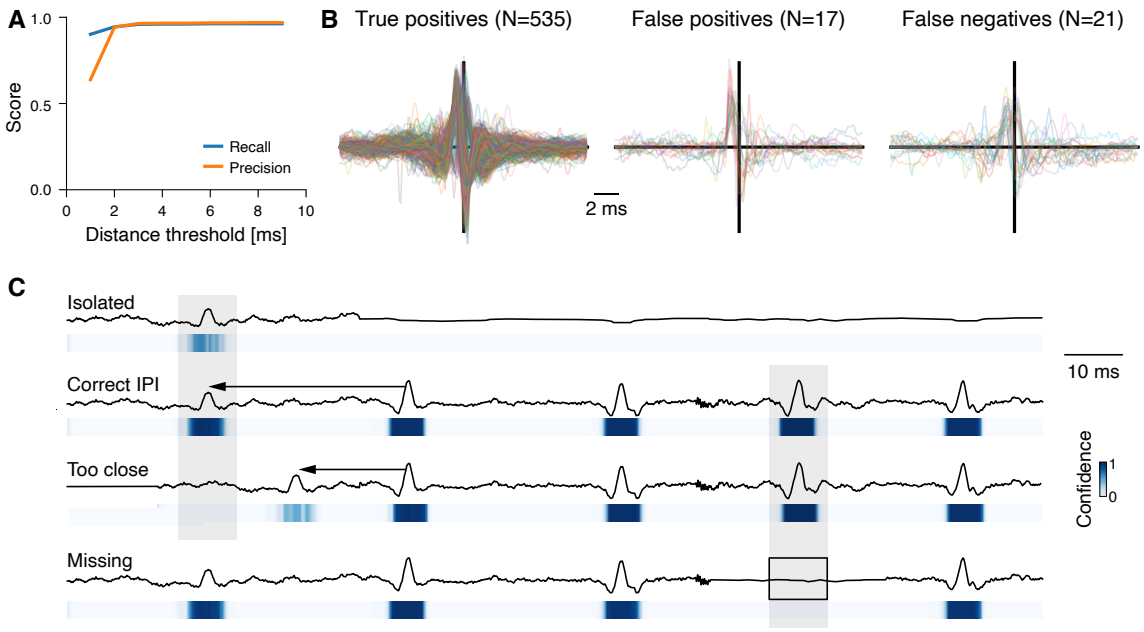

Figure S2: Performance characteristics and the role of context for annotating fly pulse song.
A Recall (blue) and precision (orange) for fly pulse song for different distance thresholds. The distance threshold determines the maximal distance to a true pulse for a detected pulse to be a true positive.
B Waveforms of true positive (left), false positive (middle), and false negatives (right) pulses in fly song. Pulses were aligned to the peak, adjusted for sign, and their amplitude was normalized to have unit norm (see Clemens et al. (2018)).
C Waveforms (top) and confidence scores (bottom, see color bar) for pulses in different contexts. DeepSS exploits context effects to boost the detection of weak signals. An isolated ("Isolated," first row) weak pulse-like waveforms is detected with low confidence, since similar waveforms often arise from noise. Manual annotators exploit context information – the fact that pulses often occur in trains at an interval of ~40ms – to annotate weak signals. DeepSS does the same – the same waveform is detected with much higher confidence due to the presence of a nearby pulse train ("Correct IPI," 2nd row). If the pulse is too close to another pulse ("Too close," 3rd row), it is likely noise and DeepSS detects it with lower confidence. Context effects do not affect strong signals. For instance, a missing pulse within a pulse train ("Missing," last row) does not reduce detection confidence of nearby pulses.

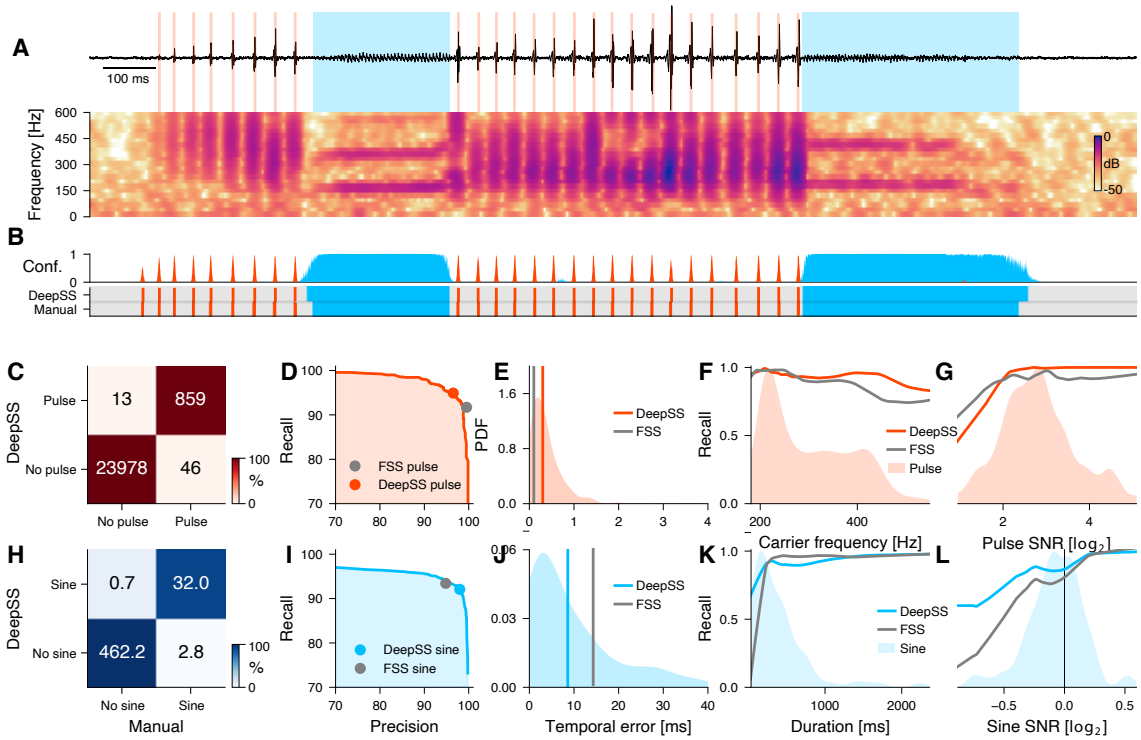

Figure S3: DeepSS performance for multi-channel recordings of fly courtship song.
A Fly song (black) with manual annotation indicating sine (blue) and pulse (red). Traces (top) and spectrogram (bottom, see color bar) show data from the loudest of the nine audio channels.
B Confidence scores (top) for sine (blue) and pulse (red). The confidence is transformed into annotation labels (bottom) based on a confidence threshold (0.5 for sine and pulse). Ground truth (bottom) from manual annotations shown for comparison. DeepSS annotations were generated using separate networks for pulse or for sine.
C Confusion matrix for pulse from a test data set. Color indicates the percentage (see color bar) and text labels indicate number of pulses for each quadrant. All confusion matrices are normalized such that columns sum to 100%. The concentration of values along the diagonal indicates high annotation performance.
D Precision-recall curve for pulse depicts the performance characteristics of DeepSS for different confidence threshold (from 0 to 1). Recall decreases and precision increases with the threshold. The closer the curve to the upper and right border, the better. The red circle corresponds to the performance of DeepSS for a threshold of 0.5. The grey circle depicts the performance of FlySongSegmenter (FSS).
E Probability density function (PDF) of temporal errors for all detected pulses (red shaded area), computed as the distance between each pulse annotated by DeepSS and its nearest manual pulse. Lines depict median temporal error for DeepSS (red line, 0.3 ms) and FSS (grey line, 0.1 ms).
F, **G** Recall of DeepSS (red line) and FSS (grey line) as a function of the pulse carrier frequency (F) and signal-to-noise ratio (SNR) (G). Red areas show the distributions of carrier frequencies (F) and SNRs (G) for all pulses.
H Same as in C but for sine. Color indicates the percentage (see color bar) and text labels indicate seconds of sine for each quadrant.
I Same as in D but for sine. The blue circle depicts the performance for the confidence threshold of 0.5 used in A.
J Distribution of temporal errors for all detected sine on- and offsets. Median temporal error is 9 ms for DeepSS (blue line) and 14 ms for FSS (grey line).
K, **L** Recall for DeepSS (blue line) and FSS (grey line) as a function of sine duration (K) and SNR (L). Blue shaded areas show the distributions of durations (K) and SNRs (L) for all sine songs. DeepSS outperforms FSS for sine songs with short durations and SNRs <1.0.

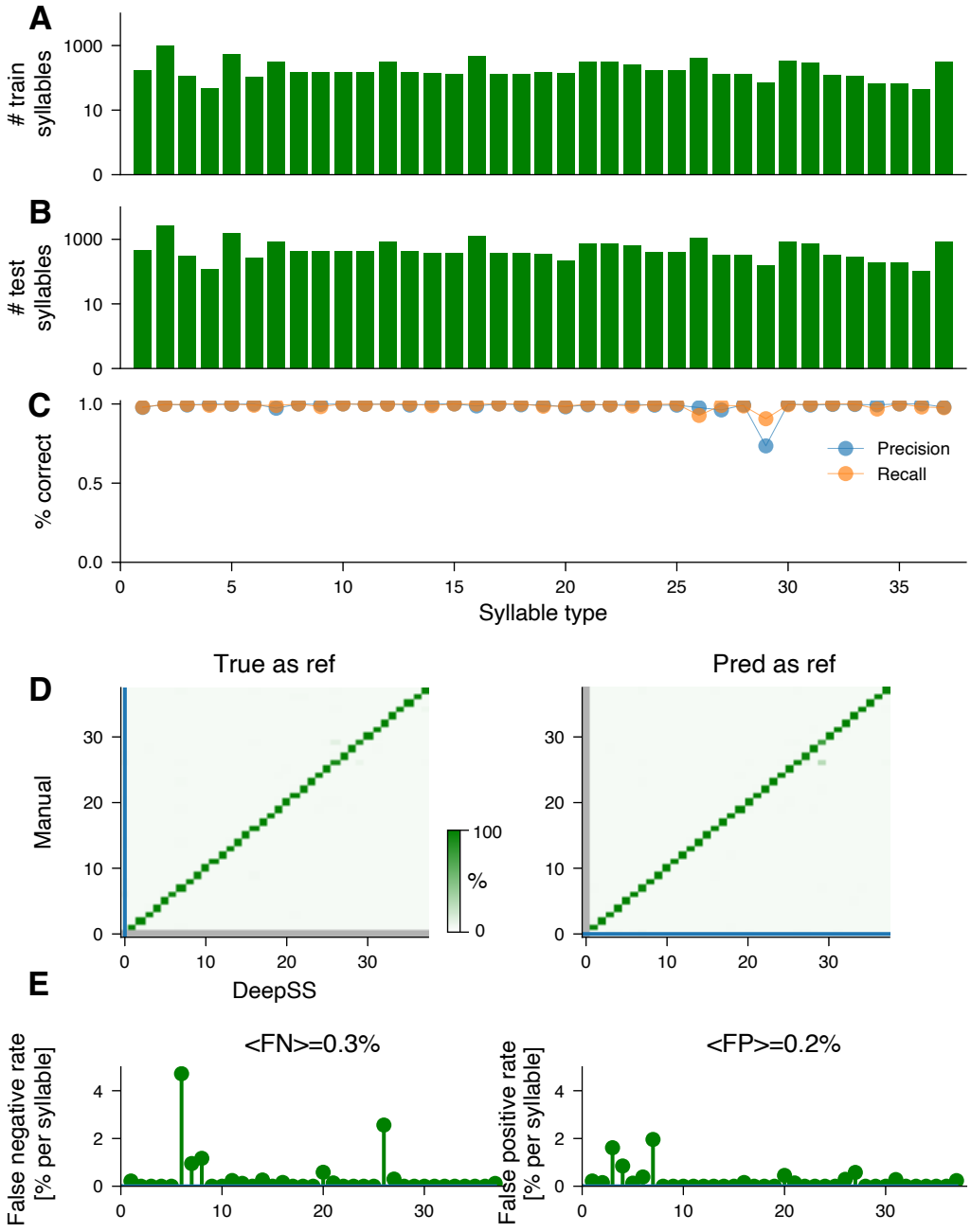

Figure S4: Performance characteristics for bird song.
A, **B** Number of syllables (log scaled) in the train (A) and the test (B) sets for each syllable type present in the test set.
C Precision (blue) and recall (orange) for each syllable type, computed from the confusion matrix in Fig. 3C).
D Confusion matrices when using true (left) and predicted (right) syllables as a reference for determining syllable type. The reference determines the syllable syllable bounds and the syllable label is then given by the most frequent label found in the samples within the syllable bounds. When using the true syllables for reference, there can be no false positives (gray line), but false negatives (blue line, values plotted in E, left). When using the true syllables as reference, there are no true negatives (horizontal gray block at y=0), since all reference syllables are true syllables. By contrast, with the predicted labels as reference, there are no false negatives (vertical gray block at x=0), since all reference syllables are (true or false) positives.
E False negative (left) and false positives (right) rates for each syllable, estimated from the confusion matrices in D. The average false negative and positive rates are 0.3% and 0.2%, respectively.

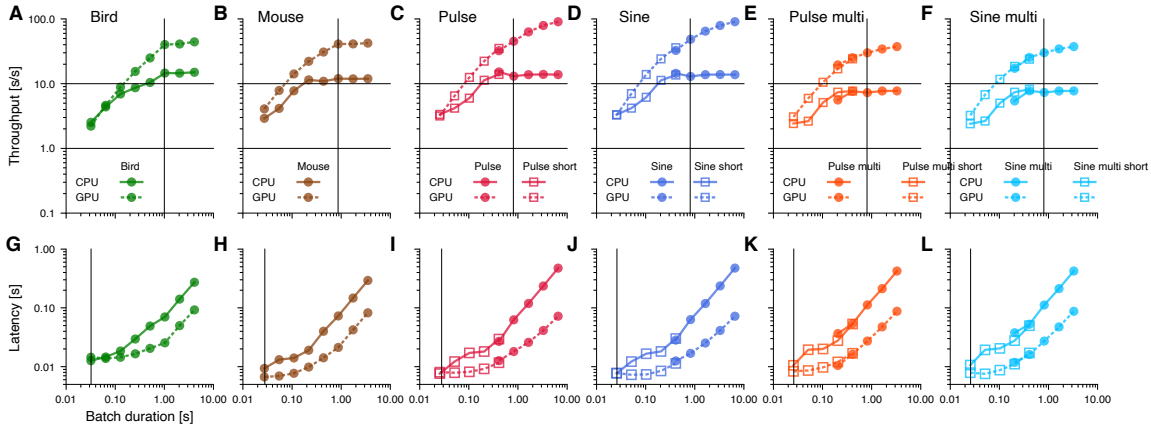

Figure S5: Throughput and latency of inference
Median throughput (A-F) and latency (G-L) across 10 inference runs for different song types. Batch duration is the product of chunk duration and batch size. Error bars () are smaller than the marker size (see Fig. 4A/B). Solid and dashed lines correspond to values when running DeepSS on a CPU and GPU, respectively. Squares and circles in the plots for fly song correspond to networks with short and long chunk durations, respectively (see Fig. S6). In A-F, horizontal lines mark 1x and 10x realtime and vertical lines mark the batch duration used in Fig. 4A. In G-L, the vertical line marks the batch duration used in Fig. 4B. Small differences in these values between species arise from differences in the sample rate and the complexity of the network (number of filters and TCN layers, size of the filters) (see Table S1).

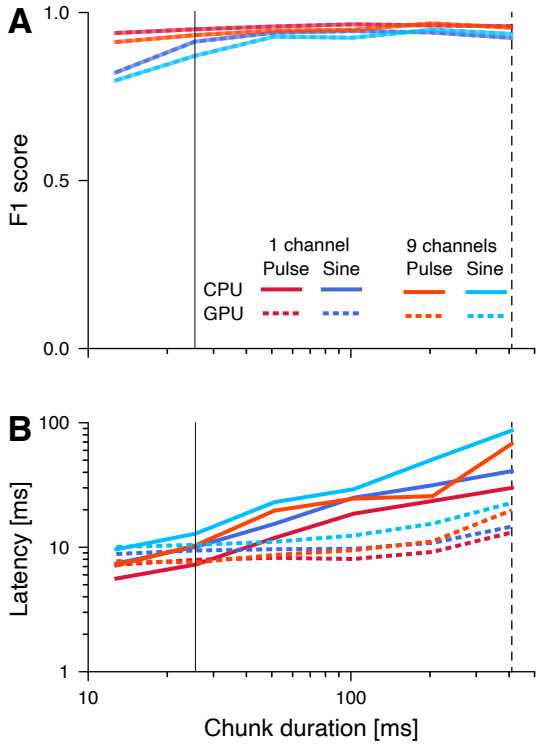

Figure S6: Reducing the chunk duration reduces latency and comes with minimal performance penalties.
A F1 scores of networks with different chunk durations. Reduction of performance due to loss of context information with shorter chunks is minimal.
B Latency of networks with different chunk durations.
Dashed vertical lines indicate the chunk durations of the networks in Figs. 1, 4B and S5. Solid vertical lines indicated the "short" chunk durations used in Fig. 4B and S5.

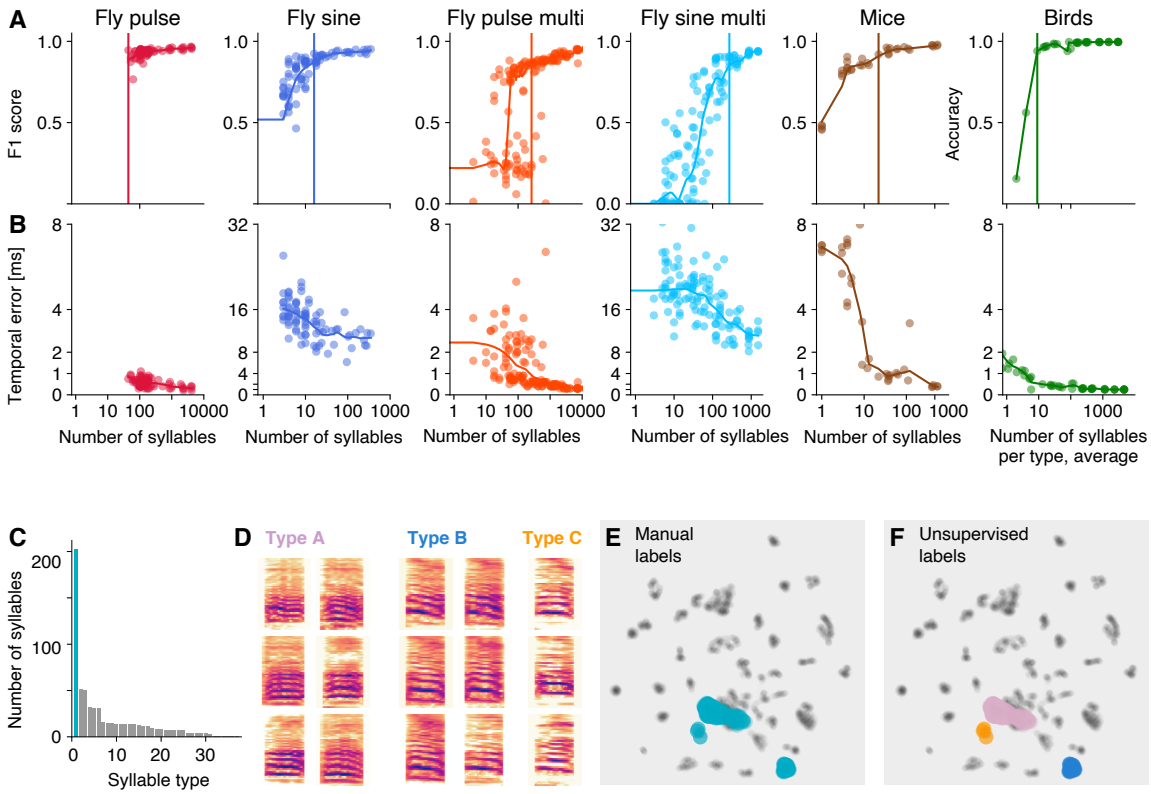

Figure S7: DeepSS requires little data for training.
A Performance as a function of the number of manual annotations in the training set. Individual dots correspond to individual fits using different subsets of the data. For flies and mice, dots indicate the number of annotations for each type and performance is given by the F1 score – the geometric mean of precision and recall. For birds (green, rightmost panel), dots correspond to the median number of syllables per type in the training set and performance is given by the median accuracy over syllables per fit (see C for per-type statistics). Colored curves in A and B were obtained by locally weighted scatter plot smoothing (lowess) of the individual data points. Vertical lines correspond to the number of syllables required to surpass 90% of the maximal performance.
B Temporal error given as the median temporal distance to syllable on- and offsets or to pulses.
C Number of annotations required per syllable type for song birds (range 2-64, median 8, mean 17). One outlier type requires ~200 annotations and consists of a mixture of different syllable types (cyan, see D-F).
D Spectrograms of different types of syllable exemplars from the outlier type in C grouped based on the unsupervised clustering in F.
E, **F** UMAP embedding of the bird syllables with the outlier type from C labelled according to the manual annotations (E, cyan) and the unsupervised clustering (F). The unsupervised clustering reveals that the outlier type splits into two distinct clusters of syllables (pink and blue) and three mislabelled syllables (orange).

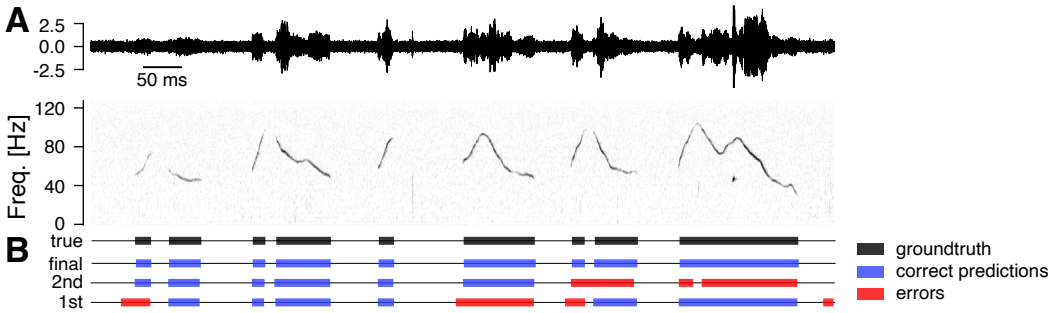

Figure S8: Example of fast training mouse USVs.
DeepSS can be trained using an iterative procedure, in which the network is network trained after annotating few syllables. This initial network is then used to create a larger dataset of annotations, by predicting and manually correcting annotation labels for a longer stretch of audio. This procedure is repeated to create ever larger training datasets until the performance is satisfactory. Shown is an example from mouse song.
A Example of the test recording for mouse song (top - waveform, bottom - spectrogram)
B Manual labels (true, black) and correct (blue) and erroneous (red) annotations from DeepSS for three different stages of training. Even for the first round of fast training, the majority of onsets is detected with low temporal error, requiring only few corrections. Number of syllables, seconds of song and F1-score for the different iterations (1st/2nd/final): 72/248/2706 syllables, 6/26/433 seconds of song, F1 score 96.6/97.9/98.9.

References

Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M, Levenberg J, Monga R, Moore S, Murray DG, Steiner B, Tucker P, Vasudevan V, Warden P, Wicke M, Yu Y, Zheng X. 2016. TensorFlow: A system for large-scale machine learningOSDI'16. USA: USENIX Association. pp. 265–283.

Arthur BJ, Ding Yun, Sosale M, Khalif F, Kim E, Waddell P, Turaga SC, L SD. 2021. SongExplorer: A deep learning workflow for discovery and segmentation of animal acoustic communication signals.

Arthur BJ, Sunayama-Morita T, Coen P, Murthy M, Stern DL. 2013. Multi-channel acoustic recording and automated analysis of Drosophila courtship songs. *BMC Biology* **11**:11.

Bai S, Kolter JZ, Koltun V. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.

Baker CA, Clemens J, Murthy M. 2019. Acoustic Pattern Recognition and Courtship Songs: Insights from Insects. *doiorg* **42**:annurev-neuro-080317-061839.

Bath DE, Stowers JR, Hörmann D, Poehlmann A, Dickson BJ, Straw AD. 2014. FlyMAD: rapid thermogenetic control of neuronal activity in freely walking Drosophila. *Nature methods*.

Behr O, Helversen O von. 2004. Bat serenadescomplex courtship songs of the sac-winged bat (Saccopteryx bilineata). *Behavioral Ecology and Sociobiology* **56**:106–115.

Benichov JI, Vallentin D. 2020. Inhibition within a premotor circuit controls the timing of vocal turn-taking in zebra finches. *Nature communications* **11**:1–10.

Bennet-Clark HC. 1998. Size and scale effects as constraints in insect sound communication. *Philosophical Transactions of the Royal Society B: Biological Sciences* **353**:407–419.

Calhoun AJ, Pillow JW, Murthy M. 2019. Unsupervised identification of the internal states that shape natural behavior. *Nature neuroscience* **16**:1–10.

Campello RJGB, Moulavi D, Sander J. 2013. Density-Based Clustering Based on Hierarchical Density EstimatesAdvances in Knowledge Discovery and Data Mining. Berlin, Heidelberg: Springer, Berlin, Heidelberg. pp. 160–172.

Cator LJ, Arthur BJ, Harrington LC, Hoy R. 2009. Harmonic Convergence in the Love Songs of the Dengue Vector Mosquito. *Science* **323**:1166541+–1166541.

Cäsar C, Zuberbühler K, Young RJ, Byrne RW. 2013. Titi monkey call sequences vary with predator location and type. *Biology Letters* **9**:20130535.

Chaverri G, Gillam EH, Kunz TH. 2012. A call-and-response system facilitates group cohesion among disc-winged bats. *Behavioral Ecology* **24**:481–487.

Choi K, Joo D, Kim J. 2017. Kapre: On-GPU Audio Preprocessing Layers for a Quick Implementation of Deep Neural Network Models with Keras.

Chollet F, others. 2015. Keras.

Clay Z, Smith CL, Blumstein DT. 2012. Food-associated vocalizations in mammals and birds: what do these calls really mean? *Animal Behaviour* **83**:323–330.

Clemens J, Coen P, Roemschied FA, Pereira TD, Mazumder D, Aldarondo DE, Pacheco DA, Murthy M. 2018. Discovery of a New Song Mode in Drosophila Reveals Hidden Structure in the Sensory and Neural Drivers of Behavior. *Current biology* **28**:2400–2412.e6.

Clemens J, Hennig RM. 2013. Computational principles underlying the recognition of acoustic signals in insects. *Journal of Computational Neuroscience* **35**:75–85.

Cleveland WS. 1979. Robust Locally Weighted Regression and Smoothing Scatterplots. *Journal of the American Statistical Association*.

Coen P, Clemens J, Weinstein AJ, Pacheco DA, Deng Y, Murthy M. 2014. Dynamic sensory cues shape song structure in Drosophila. *Nature* **507**:233–237.

Coen P, Xie M, Clemens J, Murthy M. 2016. Sensorimotor Transformations Underlying Variability in Song Intensity during Drosophila Courtship. *Neuron* **89**:629–644.

Coffey KR, Marx RG, Neumaier JF. 2019. DeepSqueak: a deep learning-based system for detection and analysis of ultrasonic vocalizations. *Neuropsychopharmacology* **231**:1–10.

Cohen Y, Shen J, Semu D, Leman DP, Liberti WA, Perkins LN, Liberti DC, Kotton DN, Gardner TJ. 2020. Hidden neural states underlie canary song syntax. *Nature* **582**:539–544.

Deutsch D, Clemens J, Thiberge SY, Guan G, Murthy M. 2019. Shared Song Detector Neurons in Drosophila Male and Female Brains Drive Sex-Specific Behaviors. *Current biology* **29**:3200–3215.e5.

Devlin J, Chang M-W, Lee K, Toutanova K. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Ding Y, Berrocal A, Morita T, Longden KD, Stern DL. 2016. Natural courtship song variation caused by an intronic retroelement in an ion channel gene. *Nature*.

Ding Y, Lillvis JL, Cande J, Berman GJ, Arthur BJ, Long X, Xu M, Dickson BJ, Stern DL. 2019. Neural Evolution of Context-Dependent Fly Song. *Current biology* **0**.

Fortune ES, Rodríguez C, Li D, Ball GF, Coleman MJ. 2011. Neural mechanisms for the coordination of duet singing in wrens. *Science* **334**:666–670.

Gerhardt CH, Huber F. 2002. Acoustic Communication in Insects and Anurans. University Of Chicago Press.

Goffinet J, Mooney R, Pearson J. 2019. Inferring low-dimensional latent descriptions of animal vocalizations. *bioRxiv* **41**:811661.

Graves A, Jaitly N. 2014. Towards End-To-End Speech Recognition with Recurrent Neural NetworksInternational Conference on Machine Learning. PMLR. pp. 1764–1772.

Graving JM, Chae D, Naik H, Li L, Koger B, Costelloe BR, Couzin ID. 2019. DeepPoseKit, a software toolkit for fast and robust animal pose estimation using deep learning. *eLife* **8**:18.

Guirguis K, Schorn C, Guntoro A, Abdulatif S, Yang B. 2020. SELD-TCN: Sound Event Localization & Detection via Temporal Convolutional Networks.

Haack B, Markl H, Ehret G. 1983. Sound communication between parents and offspring. In: Willott JF, editor. The Auditory Psychobiology of the Mouse. Springfield (Illinois): C. C. Thomas. pp. 57–97.

Harris CR, Millman KJ, Walt SJ van der, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, Kerkwijk MH van, Brett M, Haldane A, Río JF del, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE. 2020. Array programming with NumPy. *Nature*.

He K, Zhang X, Ren S, Sun J. 2015. Deep Residual Learning for Image Recognition. *arXivorg*.

Holy TE, Guo Z. 2005. Ultrasonic Songs of Male Mice. *PLoS Biology* **3**:e386.

Hoyer S, Hamman J. 2017. Xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software* **5**. doi:10.5334/jors.148

Hunter JD. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* **9**:90–95. doi:10.1109/MCSE.2007.55

Ivanenko A, Watkins P, Gerven MAJ van, Hammerschmidt K, Englitz B. 2018. Classification of mouse ultrasonic vocalizations using deep learning. *bioRxiv* **4**:358143.

Janik VM, Slater PJB. 1998. Context-specific use suggests that bottlenose dolphin signature whistles are cohesion calls. *Animal Behaviour* **56**:829–838.

Kingma DP, Ba J. 2014. Adam: A method for stochastic optimization. *arxivorg*.

Kluyver T, Ragan-Kelley B, Pérez F, Granger B, Bussonnier M, Frederic J, Kelley K, Hamrick J, Grout J, Corlay S, Ivanov P, Avila D, Abdalla S, Willing C, team J development. 2016. Jupyter notebooks - a publishing format for reproducible computational workflows In: Loizides F, Scmidt B, editors. Positioning and Power in Academic Publishing: Players, Agents and Agendas. Netherlands: IOS Press. pp. 87–90.

Kollmorgen S, Hahnloser RHR, Mante V. 2020. Nearest neighbours reveal fast and slow components of motor learning. *Nature* **382**:1–5.

Koumura T, Okanoya K. 2016. Automatic Recognition of Element Classes and Boundaries in the Birdsong with Variable Sequences. *PLoS ONE* **11**:e0159188.

Köster J, Rahmann S. 2012. Snakemake–a scalable bioinformatics workflow engine. *Bioinformatics*.

Krizhevsky A, Sutskever I, Hinton GE. 2012. ImageNet Classification with Deep Convolutional Neural Networks 1097–1105.

Lipkind D, Marcus GF, Bemis DK, Sasahara K, Jacoby N, Takahasi M, Suzuki K, Fehér O, Ravbar P, Okanoya K, Tchernichovski O. 2013. Stepwise acquisition of vocal combinatorial capacity in songbirds and human infants. *Nature* **498**:104–108.

Long Ma, Fee MS. 2008. Using temperature to analyse temporal dynamics in the songbird motor pathway. *Nature* **456**:189–194.

Mamalet F, Garcia C. 2012. Simplifying ConvNets for Fast LearningArtificial Neural Networks and Machine Learning ICANN 2012. Berlin, Heidelberg: Springer, Berlin, Heidelberg. pp. 58–65.

Mathis A, Mamidanna P, Cury KM, Abe T, Murthy VN, Mathis MW, Bethge M. 2018. DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nature neuroscience* **20**:1.

Mathis A, Yüksekgönül M, Rogers B, Bethge M, Mathis MW. 2019. Pretraining boosts out-of-domain robustness for pose estimation.

McFee B, Raffel C, Liang D, Ellis DP, McVicar M, Battenberg E, Nieto O. 2015. Librosa: Audio and music signal analysis in pythonProceedings of the 14th Python in Science Conference.

McInnes L, Healy J, Astels S. 2017. Hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software* **2**. doi:10.21105/joss.00205

McInnes L, Healy J, Melville J. 2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.

McKinney W. 2010. Data Structures for Statistical Computing in PythonPython in Science Conference.

Miles A, Kirkham J, Durant M, Bourbeau J, Onalan T, Hamman J, Patel Z, shikharsg, Rocklin M, dussin raphael, Schut V, Andrade ES de, Abernathey R, Noyes C, sbalmer, bot pyup.io, Tran T, Saalfeld S, Swaney J, Moore J, Jevnik J, Kelleher J, Funke J, Sakkis G, Barnes C, Banihirwe A. 2020. Zarr-developers/zarr-python: v2.4.0. Zenodo. doi:10.5281/zenodo.3773450

Morley EL, Jonsson T, Robert D. 2018. Auditory sensitivity, spatial dynamics, and amplitude of courtship song in Drosophila melanogaster. *The Journal of the Acoustical Society of America* **144**:734–739.

Negri LH, Vestri C. 2017. Lucashn/peakutils: v1.1.0. Zenodo. doi:10.5281/zenodo.887917

Neunuebel JP, Taylor AL, Arthur BJ, Egnor SR. 2015. Female mice ultrasonically interact with males during courtship displays. *eLife* **4**:e06203.

Nicholson D, Queen JE, J. Sober S. 2017. Bengalese finch song repository. doi:10.6084/m9.figshare.4805749.v5

Okobi DE, Banerjee A, Matheson AMM, Phelps SM, Long Ma. 2019. Motor cortical control of vocal interaction in neotropical singing mice. *Science* **363**:983–988.

Oord A van den, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A, Kavukcuoglu K. 2016. WaveNet: A Generative Model for Raw Audio.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay É. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*.

Pereira TD, Aldarondo DE, Willmore L, Kislin M, Wang SS-H, Murthy M, Shaevitz JW. 2018. Fast animal pose estimation using deep neural networks. *Nature methods* **16**:1–125.

Perez F, Granger BE. 2007. IPython: A System for Interactive Scientific Computing. *Computing in Science & Engineering*.

Raghu M, Zhang C, Kleinberg J, Bengio S. 2019. Transfusion: Understanding Transfer Learning for Medical Imaging.

Sainburg T, Thielk M, Gentner TQ. 2019. Latent space visualization, characterization, and generation of diverse vocal communication signals. *bioRxiv* **10**:870311.

Sangiamo DT, Warren MR, Neunuebel JP. 2020. Ultrasonic signals associated with different types of social behavior of mice. *Nature neuroscience* **231**:1–12.

Srivastava KH, Holmes CM, Vellema M, Pack AR, Elemans CPH, Nemenman I, Sober SJ. 2017. Motor control by precisely timed spike patterns. *Proceedings of the National Academy of Sciences* **114**:1171–1176.

Stern DL. 2014. Reported Drosophila courtship song rhythms are artifacts of data analysis. *BMC Biology* **12**:38.

Stern DL, Clemens J, Coen P, Calhoun AJ, Hogenesch JB, Arthur BJ, Murthy M. 2017. Experimental and statistical reevaluation provides no evidence for Drosophila courtship song rhythms. *Proceedings of the National Academy of Sciences* **114**:9978–9983.

Stowers JR, Hofbauer M, Bastien R, Griessner J, Higgins P, Farooqui S, Fischer RM, Nowikovsky K, Haubensak W, Couzin ID, Tessmar-Raible K, Straw AD. 2017. Virtual reality for freely moving animals. *Nature methods* **14**:995–1002.

Tabler JM, Rigney MM, Berman GJ, Gopalakrishnan S, Heude E, Al-lami HA, Yannakoudakis BZ, Fitch RD, Carter C, Vokes S, Liu KJ, Tajbakhsh S, Egnor SR, Wallingford JB. 2017. Cilia-mediated Hedgehog signaling controls form and function in the mammalian larynx. *eLife* **6**:320.

Tachibana RO, Kanno K, Okabe S, Kobayasi KI, Okanoya K. 2020. USVSEG: A robust method for segmentation of ultrasonic vocalizations in rodents. *PLoS ONE* **15**:e0228907.

Tschida K, Mooney R. 2012. The role of auditory feedback in vocal learning and maintenance. *Current Opinion in Neurobiology* **22**:320–327.

Van Segbroeck M, Knoll AT, Levitt P, Narayanan S. 2017. MUPETMouse Ultrasonic Profile ExTraction: A Signal Processing Tool for Rapid and Unsupervised Analysis of Ultrasonic Vocalizations. *Neuron* **94**:465–485.e5.

Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov

N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P, SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* **17**:261–272. doi:10.1038/s41592-019-0686-2

Warren MR, Clein RS, Spurrier MS, Roth ED, Neunuebel JP. 2020. Ultrashort-range, high-frequency communication by female mice shapes social interactions. *Scientific reports* **10**:1–14.

Waskom M, Botvinnik O, O'Kane D, Hobson P, Lukauskas S, Gemperline DC, Augspurger T, Halchenko Y, Cole JB, Warmenhoven J, Ruiter J de, Pye C, Hoyer S, Vanderplas J, Villalba S, Kunter G, Quintero E, Bachant P, Martin M, Meyer K, Miles A, Ram Y, Yarkoni T, Williams ML, Evans C, Fitzgerald C, Brian, Fonnesbeck C, Lee A, Qalieh A. 2017. Mwaskom/seaborn: v0.8.1 (september 2017). Zenodo. doi:10.5281/zenodo.883859

Weiss M, Hultsch H, Adam I, Scharff C, Kipper S. 2014. The use of network analysis to study complex animal communication systems: a study on nightingale song. *Proceedings of the Royal Society B: Biological Sciences* **281**:20140460.

Yu F, Koltun V. 2015. Multi-scale context aggregation by dilated convolutions. *arxivorg*.