Deep Learning to Extract Laboratory Mouse Ultrasonic Vocalizations from Scalograms

Adam A. Smith
Department of Mathematics and Computer Science
University of Puget Sound
Tacoma, Washington 98416
Email: aasmith@pugetsound.edu

Drew Kristensen
Department of Mathematics and Computer Science
University of Puget Sound
Tacoma, Washington 98416
Email: dkristensen@pugetsound.edu

***Abstract*—We tested two techniques that can assist in the automatic extraction and analysis of laboratory mouse ultrasonic vocalizations. First, we substituted a Morlet-wavelet-based scalogram in place of the commonly used Fourier-based spectrogram. The frequencies in a scalogram are logarithmically scaled, which likely makes it better model a mouse's pitch perception based on the structure of its inner ear. Works that use the linear spectrogram are more likely to overemphasize frequency change at high frequencies, and so might misclassify calls when partitioning them according to frequency modulation. We showed that laboratory mice do indeed modulate their calls more when making high-frequency calls, which we would expect if they perceive pitch logarithmically. Secondly, we used "deep" convolutional neural networks to automatically identify calls within the scalogram. A convolutional neural network performs much like the animal visual cortex, and has enjoyed recent success in computer-vision and related problems. We compared the convolutional network to a conventional fully-connected neural network and the Song filter used by recent papers, and found that our convolutional network identifies calls with greater precision and recall.**

I. Introduction

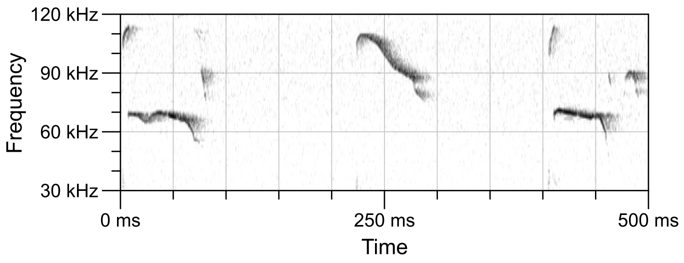
Laboratory mice (*Mus musculus*) are one of the most common animal models used in medical research. This work focuses on the automatic extraction of their ultrasonic vocalizations (USVs) without immediate human supervision. It is commonly thought that a mouse's vocalizations provide a window into the animal's affective state. For scientists who study human mental illness (e.g. schizophrenia, autism spectrum disorder, traumatic brain injury), a better understanding of a mouse's vocalizations may be crucial to understanding mouse models. However, obtaining and using this data has been challenging. Recording mice and extracting their vocalizations is very expensive and labor-intensive, and requires specialized equipment. It can take a human technician hours (if not days) to process mere minutes of recording. This paper makes two primary contributions. First, we replace the traditional Fourier transform with Morlet wavelets, in order to address likely problems related to mouse pitch perception. Second, we use deep-learning convolutional neural networks to automatically extract the vocalizations from a sound file with minimal human oversight.

The characteristics of vocalizations emitted by a mouse are dependent on several factors [1]. USVs vary in the presence of neurological disorders, including conditions that model autism spectrum disorder [2]–[4], schizophrenia [4], and Alzheimer's disease [5]. USVs also differ depending on social condition [6], [7] or the odors of other mice [8], [9]. Mice exploring novel environments emit vocalizations that vary by the environment being explored [10]. Further, exposure to drugs can influence USVs, including drugs of abuse [11], [12]. Of all the qualities of laboratory mouse vocalizations, the number of calls and the frequency modulation within those calls are likely to be among the most important attributes [13], [14].

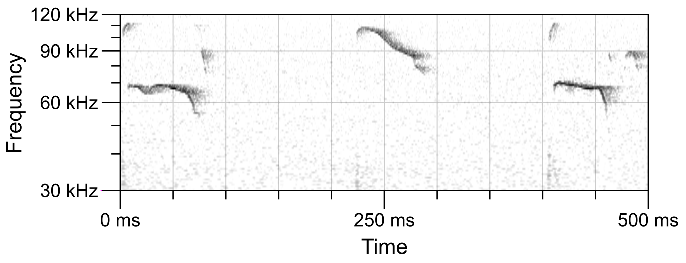
Many software tools already exist in order to facilitate working with bioacoustic data, but most require significant human oversight. Commonly used programs include Praat, Avisoft-SASLab Pro, and MUPET. These all work with *spectrograms*: the result of a windowed Fourier transform of a sound file. A sample spectrogram can be seen in Figure 1a. Praat [15] is an open-source program that has excellent support for spectrogram transforms, but it is geared toward analyzing human speech. Avisoft-SASLab Pro [16] is another standalone program that is commonly used in mouse labs. However its algorithms are proprietary and their precise details are not published. MUPET [17] is a recent open-source library for MATLAB, and it is made to act more autonomously than the other programs. It relies on the Song filter (see below), which may have difficulties identifying the precise frequency of a call.

It is important that any tool used to analyze mouse vocalizations takes into account our understanding of a mouse's pitch perception (as much as we can understand it), to introduce as few artifacts as possible. For example, calls are often partitioned based on their observed frequency modulation (e.g. "flat calls", "upward calls", etc.) [3], [18]. In general, pitch perception in mammals is thought to operate on a roughly logarithmic scale [19], [20].[1] That is, each octave is a doubling of frequency. This is due in part to the logarithmic distribution of lengths of stereocilia (sensing hairs) in the inner ear. While this fact has long been known in humans, only more recently have Müller et al. [20] confirmed it specifically for laboratory

[1]A demonstration of this phenomenon can be found at https://github.com/adam-a-smith/log-linear-scale-mp3s. Here, two sound files contain ten tones, increasing from 200 Hz to 2 kHz. One increases linearly and the other logarithmically. The linear one seems to increase quickly at first and then levels out, whereas the logarithmic one seems to increase by even steps.



(a) Spectrogram



(b) Scalogram

Fig. 1. A spectrogram and a scalogram of the same sequence of FVB mouse calls. In the spectrogram, the middle call appears to be much more modulated than the mice probably perceive, with respect to the other two large calls. The scalogram likely shows a more accurate comparison, due to the physiology of a mouse's inner ear.

mice. Thus, we argue that a call's modulation should be judged relative to the call's base frequency: an increase of 10 kHz in a 40-kHz call should be treated identically to an increase of 20 kHz in an 80-kHz call. Use of a linear frequency scale (such as that used in the spectrograms of the above programs) introduces an inherent bias, since it might cause us to think of the second call as having a greater apparent modulation than the first. This is at odds with what we know about mouse anatomy, and will make high-frequency calls appear to be more modulated than they are likely perceived to be. This could result in many misclassified calls, leading to incorrect conclusions!

II. Background

Here we detail the mathematical tools we use in this paper.

The Song Filter

The Song filter is a tool to identify times containing a bona fide call [21]. It is used by MUPET. Since non-vocal noise most often has a wider frequency range than a vocal call, the filter works by dividing the maximum spectral component at each time by the sum of that time's other spectral components:

$$\text{song}(t) = \frac{\max_{f\in F} D_{t,f}}{\sum_{f\in F} D_{t,f}}. \tag{1}$$

Here t is a time, F is the set of all frequencies in the spectrogram, and $D_{t,f}$ is the spectrogram value at time t and frequency f. If this ratio is above a threshold defined during training, time t is assumed to contain a call. This simple filter works well to identify times with calls, but it was not made to identify the exact frequencies of a call.

Scalograms vs. Spectrograms

Many biologists employing laboratory mice have continued to use the windowed Fourier transform and its resulting spectrograms, even as new tools have been developed. This is due in part to the spectrogram being well suited to the mouse's narrow-band, whistle-like vocalizations. A spectrogram creates an immediately understandable image of a mouse's calls, in which start and stop times, durations, and frequency modulation are all easily apparent. We believe that using a Morlet wavelet transform instead would keep the spectrogram's advantages, while correcting some of its shortcomings. The wavelet's analog to a spectrogram is called a *scalogram*. The two are compared in Figure 1.

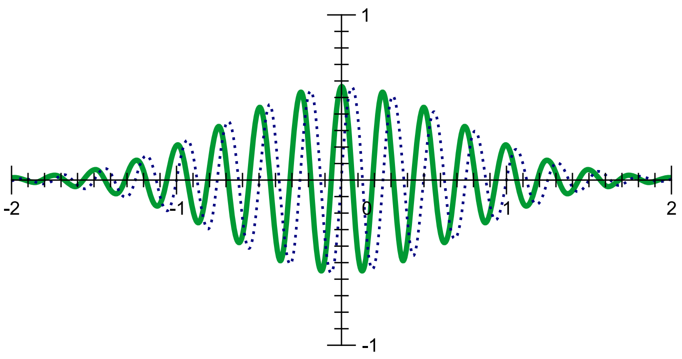


Fig. 2. A complex Morlet wavelet. The solid line shows the real component, while the dotted line is the imaginary component. Using a complex-valued wavelet maintains high convolution values when the signal and the wavelet have similar frequencies but are out of phase.

A complex Morlet wavelet (shown in Figure 2) is determined by the formula:

$$\Psi(t, T, F_c) = \frac{1}{\sqrt{\pi T}} \cdot e^{-\frac{t^2}{T}} \cdot e^{2\pi i F_c t}. \tag{2}$$

This is a Gaussian envelope function multiplied by a complex exponential. T is the time-decay parameter that determines the width of the Gaussian, which affects the relative importance of the side oscillations. F_c is the central wavelength being investigated. The value t is the time, e is Euler's constant ($\sim$2.71828) and i is $\sqrt{-1}$. By working in complex space, it is able to detect waveforms even when they are out of phase with the wavelet. The real and imaginary components of the wavelet are defined as follows:

$$\Psi_{\text{R}}(t, T, F_c) = \frac{1}{\sqrt{\pi T}} \cdot e^{-\frac{t^2}{T}} \cdot \cos(2\pi i F_c t), \tag{3}$$

$$\Psi_{\text{I}}(t, T, F_c) = \frac{i}{\sqrt{\pi T}} \cdot e^{-\frac{t^2}{T}} \cdot \sin(2\pi i F_c t). \tag{4}$$

These consist of a cosine and sine respectively with wavelength F_c, multiplied by the same Gaussian envelope function as above.

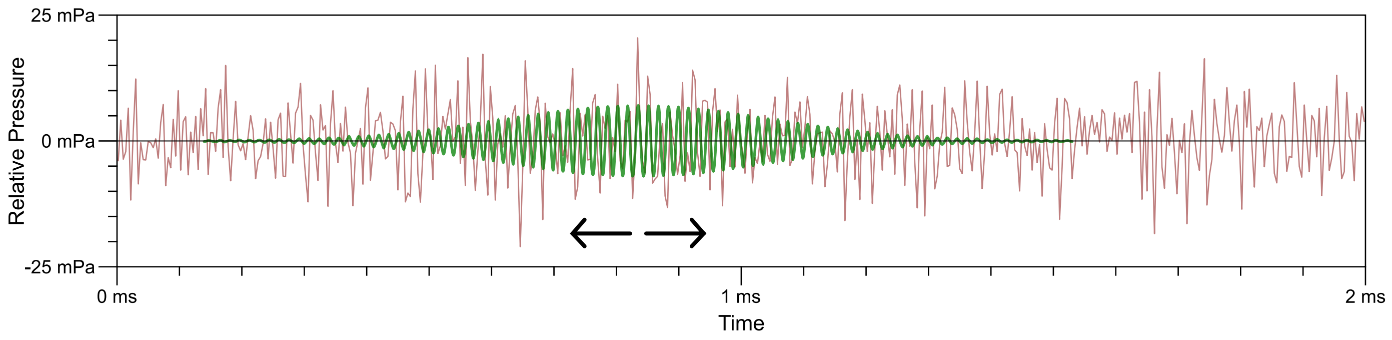


Fig. 3. A Morlet wavelet transform of high-quality (250 kHz) audio data. Because of the wavelet's symmetry, the convolution operation is essentially a cross-correlation. For clarity, the imaginary component of the wavelet is not shown.

The wavelet transform convolves wavelets of different frequencies and offsets with the original signal, as shown in Figure 3. Due to the Morlet wavelet's symmetry, its output is identical to cross-correlation. The result has both real and imaginary components, but by taking the magnitude of the result (the square root of the real portion squared plus the imaginary portion squared), we obtain the scalogram that is qualitatively similar to the well-known spectrogram.

We believe that the wavelet transform offers these three advantages for analyzing mouse vocalizations:

- **A wavelet transform results in a scalogram in which the frequencies are logarithmically scaled.** This almost certainly models a mouse's pitch perception better than the linearly scaled frequencies of the Fourier transform, for the reasons detailed above. A scalogram will not have the inherent linear bias of a spectrogram, and should better represent our best guess of what a mouse actually hears. As a useful side effect, overtone "calls" in a scalogram appear to have the same shape as their base calls, and the first one will be shifted up by exactly one octave. When using spectrograms, overtones can appear distorted and might be less recognizable.
- **Using wavelets allows us to keep the balance between time precision and frequency precision similar across frequencies.** The Heisenberg uncertainty principle states that there is an inherent tradeoff between precision in time and precision in frequency when analyzing a signal [22]. In a wavelet transform, the wavelet is scaled so that its width shrinks as the frequency increases. However, in a windowed Fourier transform the window size must remain constant at every frequency. This means that more wavelengths are allowed into the window at higher frequencies, resulting in a different balance of time and frequency precisions. This one-size-fits-all approach can result in higher or lower frequencies appearing more "out of focus" in a spectrogram. Fix one, and the other blurs. However a wavelet transform adjusts its window size automatically depending on the frequency being analyzed. A wavelength half the length of another results in a window half the length. This keeps the time/frequency tradeoff more consistent between frequencies.
- **Finally, wavelets are ideal because they are intuitive, and the resulting scalograms look very much like spectrograms.** Although the mathematics behind efficient wavelet transformation are very complicated [22], the basic idea behind the transform is quite simple: it is effectively a cross-correlation with a complex sinusoid multiplied by a Gaussian envelope function. Since the output looks much like a spectrogram, mouse biologists can keep their sense of intuition as they figure out the importance behind USVs.

Convolutional Neural Networks

Convolutional neural networks (CNNs) are a subclass of neural networks that excel at finding patterns within visual data, part of the "deep learning" revolution that has yielded many recent advances [23]. CNNs' first big success was in reading handwritten numbers—classifying a scanned character as one of the ten digits [24], [25]. They have since found use in many problems, both visual and non-visual. These include classifying images [26], analyzing video [27], discovering new drugs [28], analyzing natural language [29], and directing self-driving cars [30]. Perhaps most famously, Google's AlphaGo used CNNs in order to find meaningful patterns of pieces on the go board, allowing it to outplay some of the most skilled human players in the world [31].

Traditional neural networks are a mainstay of machine-learning research, that were inspired by natural brains. They consist of several layers of individual neurons, each of which calculates an activation function (e.g. a sigmoid) from the weighted sum of several inputs, and outputs the results to the neurons in the next layer. The final output can be trained to perform basic tasks like classification, using techniques such as backpropagation [32] that set individual parameters and weights for each neuron based on labeled training data. Until recently, it was thought that only two layers were enough to adequately model any function. Although this is true in principle [33], recent advances have showed that more layers of more specialized types (i.e. a "deep" network) can learn specific functions more quickly and accurately [23].

In a CNN, some of the neuron layers mimic cells found in the animal visual cortex. In the visual cortex, cells focus on limited, overlapping regions of the animal's field of view,

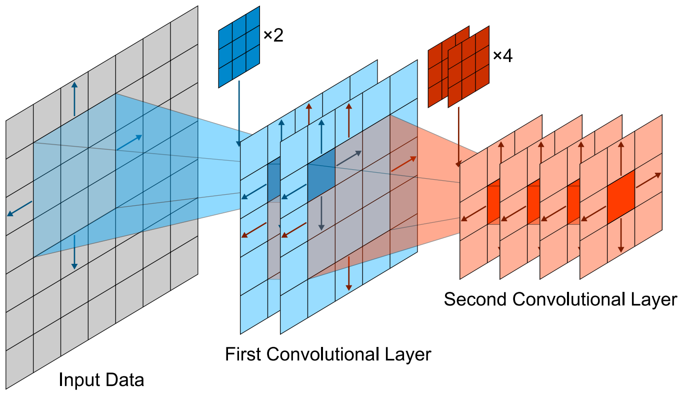


Fig. 4. Two convolutional layers of a convolutional neural network. Here a $7 \times 7 \times 1$ input is convoluted (an inner product) with two $3 \times 3 \times 1$ kernels, making the first layer $5 \times 5 \times 2$. The values are then sent to a second convolutional layer with four $3 \times 3 \times 2$ kernels, so the resulting layer is $3 \times 3 \times 4$. The 36 values of the second layer may be sent to another convolutional layer, or perhaps flattened and sent to a traditional fully connected layer.

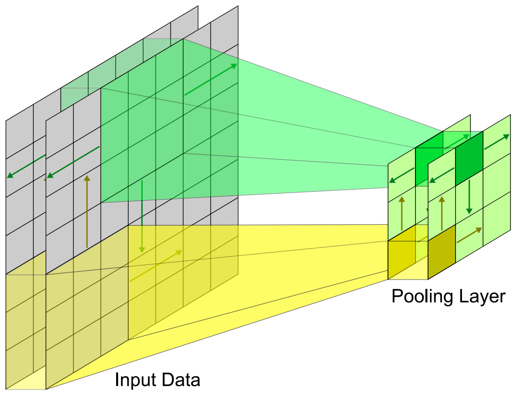


Fig. 5. A pooling layer in a neural network, used to downsample the data. Here the pooling layer has the same depth as its input. Each value in the $3 \times 3 \times 2$ pooling layer is the maximum value of the corresponding 3×3 square in the $7 \times 7 \times 2$ input layer. The stride of 2×2 means that adjacent values in the pooling layer are derived from 3×3 squares that are offset by 2 in each direction.

called *receptive fields*, and activate if some primitive shape is apparent therein (e.g. a horizontal line, a particular curve, etc.) [34]. The outputs of this first line of neurons are transmitted to another group of neurons which can then detect higher-level patterns, and those outputs can be transmitted to still another group, so that the whole visual cortex is able to quickly identify probable objects in the animal's environment.

A CNN works in a similar way, as illustrated in Figure 4. Here, a *kernel* is analogous to a group of cells detecting the same pattern in different regions of the input. A kernel is a three-dimensional tensor of weight values that is keyed to some particular primitive pattern. (The third dimension can be used for multiple inputs, such as the hue, saturation, and value of a color image.) By sliding the kernel against the input and calculating the inner (dot) product at each point, we can find the locations of the pattern within the input. Multiple kernels at each level are used to detect multiple patterns. Thus, the three-dimensional output of a convolutional level can be defined as:

$$o_{x,y,z} = b_z + \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} \sum_{k=0}^{D-1} (p_{x+i,y+j,k} \cdot w_{i,j,k,z}). \tag{5}$$

In this formula $o_{x,y,z}$ is the output of a layer at position (x, y, z), while $p_{x+i,y+j,k}$ is the input from the previous layer or the original data at position $(x+i, y+j, k)$. Each kernel in this layer has the same size $W \times H \times D$. The value $w_{i,j,k,z}$ is the weight of kernel z at position (i, j, k), while b_z is a bias factor for kernel z. The values of the w and b parameters are set during training, so that each kernel responds to a particular pattern. There are often two or more adjacent convolutional layers in the network. Much like in the animal visual cortex, each subsequent layer is trained to detect more abstract and high-level patterns.

The width and height of the layers tend to decrease the further one gets into the network. To counter this, the input to a layer is sometimes padded with 0s around the outside. More complicated convolutional layers might also make use of a *stride*, which determines the amount of overlap between adjacent receptive fields. In the equation given above, the stride is 1×1. This means that adjacent values in a convolutional layer are formed by moving the kernel over by one square, in both the x and y directions. Increasing the stride decreases the overlap.

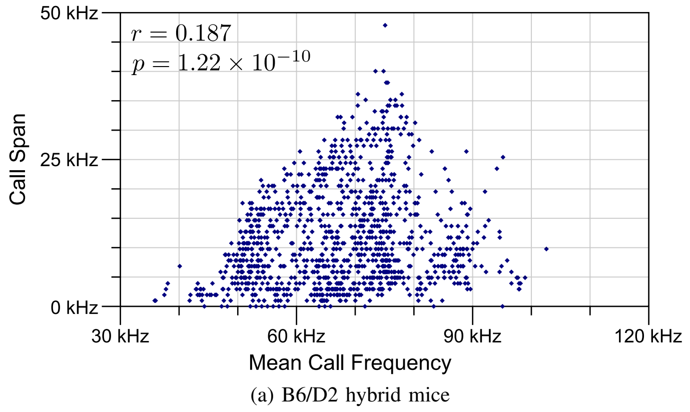
CNNs also commonly make use of *pooling layers*, in order to downsample the data for the purpose of preventing overfitting. As shown in Figure 5, a pooling layer has the same depth as the input layer, but smaller width and height. Each value is derived from a region in the input layer—often the maximum value or average value. For a stride of 1×1, $W \times D$ max pooling is defined as:

$$o_{x,y,z} = \max_{i \in [0,\ldots,W-1], j \in [0,\ldots,H-1]} p_{x+i,y+j,z}. \tag{6}$$

However, stride is usually higher. For a stride of $W \times H$ or more, adjacent values in the pooling layer will not be derived from overlapping regions. The figure illustrates 3×3 pooling, with a stride of 2×2. This necessarily downsamples the $7 \times 7 \times 2$ input layer into a $3 \times 3 \times 2$ pooling layer.

Importantly, a convolutional layer cannot learn a function that could not be learned by two layers of a traditional neural network of large enough size. However like the visual cortex, a convolutional layer has a strong bias toward identifying patterns in adjacent elements of the input data. It is for this reason that they have been so successful in solving computer-vision problems. Further, traditional networks often fail to detect important patterns when they exist in unforeseen places. Because of the convolutional nature of the kernels, sliding against the input data while calculating a dot product with the same parameters, they are able to find familar patterns in novel locations. This is refered to as "translation invariance".

Thus, the procedure we are suggesting is shown in Figure 8. First a Morlet wavelet transform creates a scalogram from the original audio data. Then, a convolutional neural network can be used to create a map showing the location of each call.



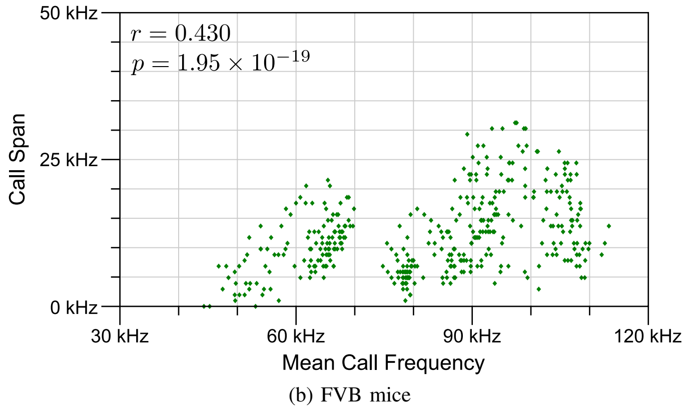
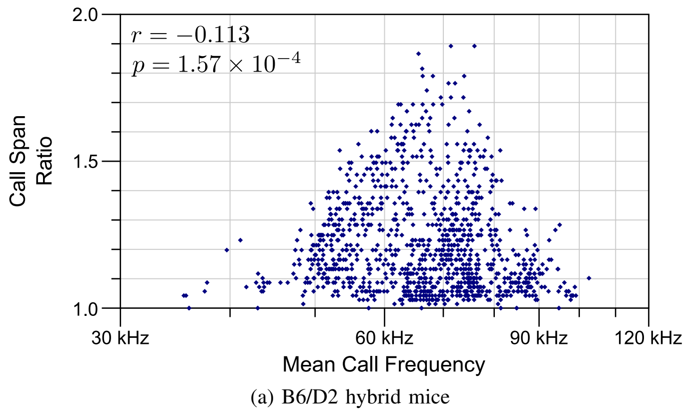


Fig. 6. Mean call frequency vs. call span (high frequency minus low frequency) for B6/D2 hybrid mice and for FVB mice. As expected, there is a clear and significant correlation between the two.



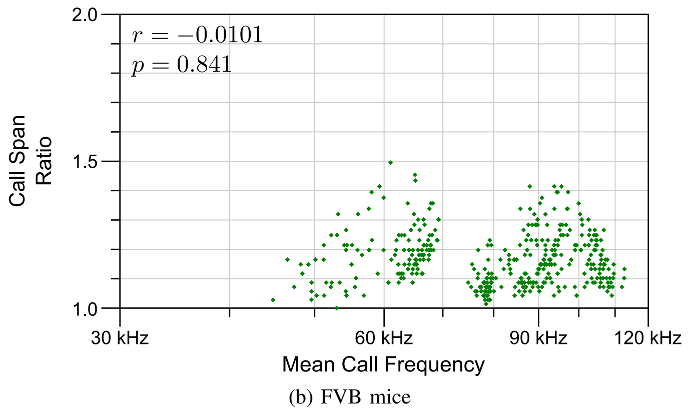


Fig. 7. Mean call frequency vs. call span ratio (high frequency divided by low frequency) for B6/D2 hybrid mice and for FVB mice. There is significant negative correlation for the B6/D2 mice, and no significant correlation for the FVB mice. Mean call frequency is graphed on a logarithmic scale.

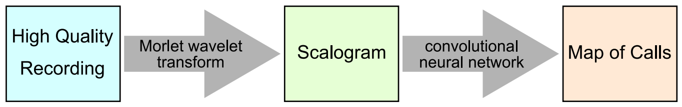


Fig. 8. Data flow diagram for our suggested call-extraction procedure.

III. Experiments

Here we detail the experiments we have done, along with their results. All audio recordings were downloaded from MouseTube [35]. We did not conduct any new animal research. We wrote all our original programs in Python 3.5.

Comparing Call Ratio and Range

If mice do perceive pitch logarithmically, we would expect a call's frequency range to increase linearly with the call's base frequency. This would be consistent with calls being perceived as having similar modulation, despite occurring over a range of frequencies. To find this, we graphed the frequency of 1110 different calls of B6/D2 (C57BL/6J and DBA/2J) hybrid mice and 399 calls of FVB (FVB/NJ) mice, against the calls' frequency ranges (high frequency minus low frequency) and frequency ratios (high frequency divided by low frequency). All mouse calls were identified by human from audio files, and a "call" in this case indicates a continuous region of high value within a spectrogram or scalogram, such as those shown in Figure 1. Discontinuous regions during the same time period are considered two separate calls.

For frequency range, we calculated spectrograms of each audio file using Praat [15], using a Gaussian window of length 2 ms and time step of 0.5 ms, with maximum frequency of 120 kHz and a frequency step size of 1 kHz. After calls were identified by human, each call's average frequency was calculated as a weighted average of all time-frequency pairs ("pixels") within that call weighted by coefficient, and the frequency range was simply the highest frequency present minus the lowest. We calculated the correlation r between frequency and range, and calculated the p-value p by assuming a t distribution. The results are shown in Figure 6.

The process for frequency ratio is similar. In this case we calculated the scalogram for each audio file, with frequencies varying over two octaves from 30 kHz to 120 kHz, with 50 voices per octave, a time-decay value of 1×10^{-6}, and a time resolution of 0.5 ms. We then calculated dominant frequency as before, though using a geometric average of frequencies rather than an arithmetic average (the difference was very slight). Call ratio is a call's highest frequency divided by its lowest frequency, and we calculated r and p as before. The results are shown in Figure 7.

For both strains we found a highly significant correlation between frequency and range, adding weight to our hypothesis. Interestingly, the correlation is negative when comparing frequency and ratio for the B6/D2 mice, though the significance is not nearly as great as it was in the range comparison. High-frequency calls simply appear to have very little modulation in these calls. For the FVB mice, which vocalize at significantly higher frequencies, no significant correlation was found between frequency and ratio.

Evaluating Convolutional Neural Networks

We attempted to construct a convolutional neural network that would be able to identify calls within fifteen scalograms created from fifteen ten-second recordings of B6/D2 hybrid mice (using the same parameters as above). We used Google's TensorFlow library [36]. The network we created is able to answer, for each coefficient ("pixel") of a scalogram, whether or not it is part of a call. The input to the neural network is the 17×17 area around the pixel in question, as well as the normalized logarithm of the pixel's frequency. Each scalogram was normalized by taking the logarithm of each value, and then scaling so that the mean value is 0 and the greatest value is 1.

To find the CNN's architecture, we started with a very basic convolutional layer followed by two traditional fully connected layers. At this point we manually performed a search, adjusting one aspect of the network at a time (e.g. number of layers, number of nodes within a layer, relative position of two layers) and retesting it, keeping changes resulting in a higher performing model. To evaluate a network we used leave-one-out cross-validation, training on fourteen scalograms at a time and testing on the fifteenth. We used human-labeled data as our gold standard. During the training process we used AdaGrad stochastic gradient descent [37] with the learning rate set to 0.1. We trained using 20,000 epochs of 100 labeled pixels from a scalogram. Each training pixel was chosen randomly from the training set, and had a 25% chance to be chosen from a call, 25% from non-call pixels within 2 pixels of a call, 25% to be from human-identified "false calls" (non-calls of high value that we thought might be problematic), and 25% from among all pixels. The convolutional layers use rectified linear units [23], while the fully connected layers use the more traditional sigmoid units [33]. Finally, all found calls consisting of fewer than five adjacent pixels were filtered out.

Our goal was to maximize the "pixelwise" F_1 score. This is the harmonic mean of the precision (fraction of pixels declared to be parts of calls, that really are) and the recall (fraction of pixels that are parts of calls that are identified as such). We also calculated the "timewise" F_1 score, that only determined whether a time step contains a call, without trying to locate it in frequency.

The network we identified is shown in Figure 9. Its performance is shown in Figure 10, along with that of two control methods. The first control is a traditional two-layer fully connected neural network. This had 290 inputs: the flattened 17×17 region around a pixel, and the normalized logarithm of the pixel's frequency. It had 50 hidden units, and two output units: one for "part of a call" and the other for "not part of a call". (We tried other numbers of hidden units as well, with less success.) We trained the network in the same manner we did the CNN. The other control we tested was the Song filter. Here we used the filter described in Section II to identify times containing calls, and then all pixels above a second threshold are determined to be part of a call. We determined the thresholds by using training data, picking first the Song threshold and then the pixel threshold that maximized the pixelwise F_1 score in the training set. For both controls, we eliminated all calls with fewer than five pixels. Our CNN achieved significantly higher F_1 scores than both controls using a one-tailed paired Student's t test, using both the pixelwise and timewise approaches.

IV. Discussion & Conclusion

We used a complex Morlet wavelet transform in order to produce scalograms of laboratory mouse vocalizations. Scalograms are intuitively similar to spectrograms, but output frequency on a logarithmic scale while minimizing artifacts due to the Heisenberg uncertainty principle. In addition, we used a deep neural network, including convolutional layers, to automatically extract the vocalizations from the generated scalogram. Convolutional layers are especially useful for classifying visual information, and our CNN extracted the vocalizations on a pixelwise level with greater precision and recall than previous methods. However, timewise we only obtained the best F_1. The Song filter obtains better timewise precision than our CNN (though not significantly), but much worse recall.

Our program takes more time to calculate a scalogram than most commerical and open-source programs take to calculate a spectrogram. However, we do not anticipate this to be a problem. Firstly, the scalogram can be calculated ahead of time and saved. Secondly, we made no attempt to parallelize our computation. Because the convolutions are independent of each other (given the wavelet and the data) and because each frequency can be calculated separately, there is ample opportunity to parallelize the computation and speed it up by an order of magnitude on modern multi-core computers.

The FVB data is strongly in line with our hypothesis that a call's range should correlate with its base frequency, and thus using a tool with a logarithmic scale should help prevent erronous conclusions. Surprisingly however, the B6/D2 mean call frequency has a significant negative correlation between call frequency and call span ratio, where we expected there to be no correlation. It may be that this is a genuine pattern: that these mice simply do not make highly modulated calls at higher frequencies. Figure 7 seems to confirm this idea. If this

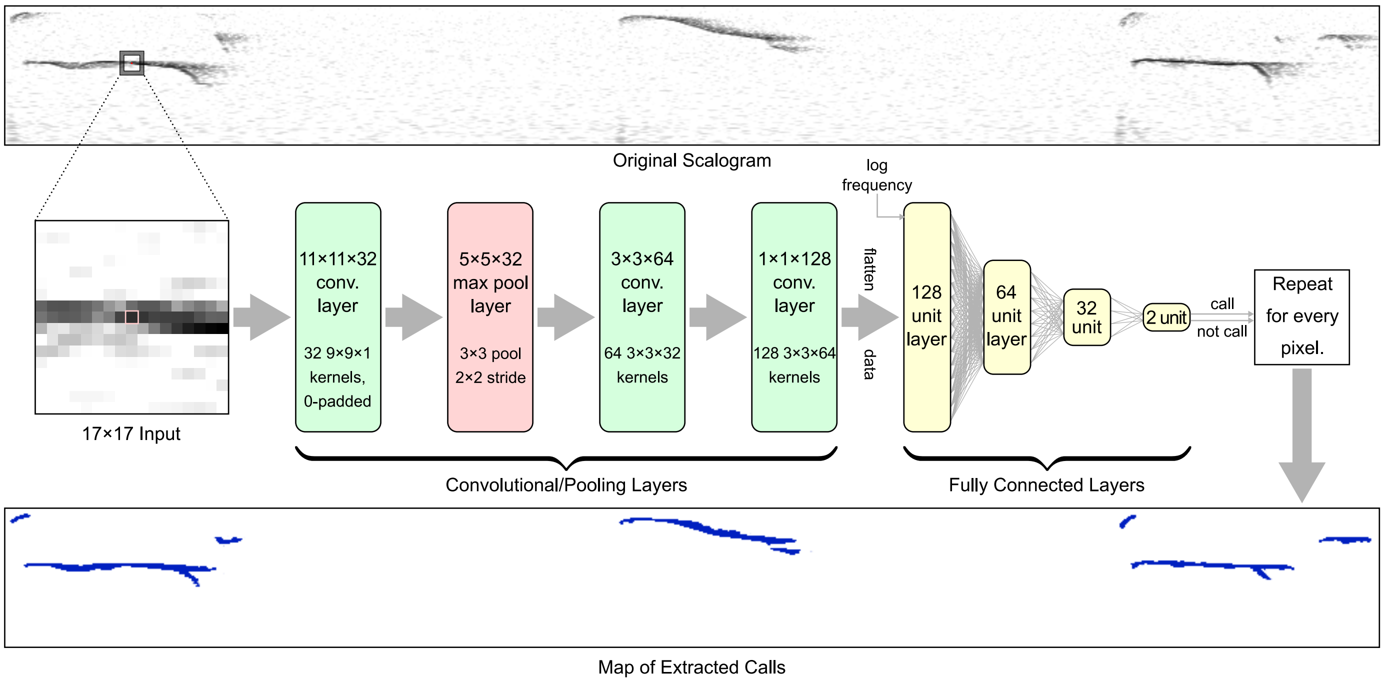


Fig. 9. Structure of a convolutional neural network, to determine if a single pixel in a scalogram is part of a call. The input is the 17×17-pixel area around the pixel The data goes through multiple convolutional layers with one pooling layer, and then into four traditional fully connected layers. The output is a single "yes" or "no". Repeated for every pixel, we generate a map of the calls' locations.

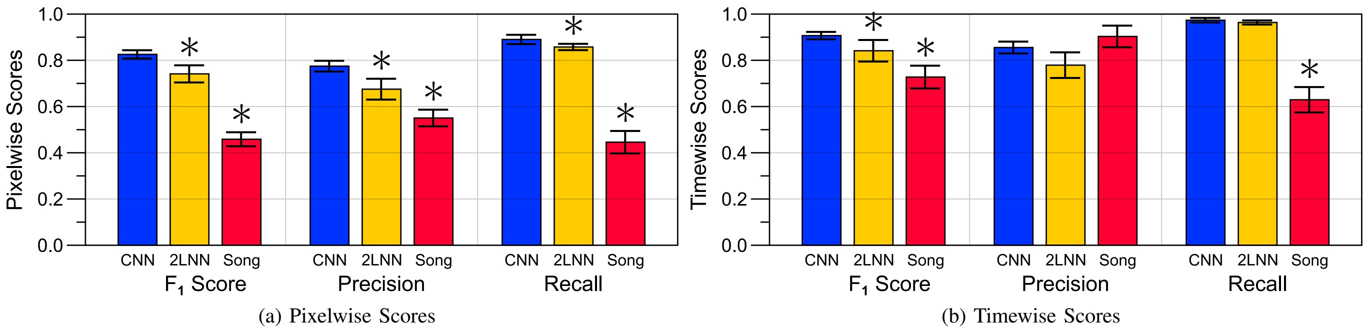


Fig. 10. Pixelwise and timewise F_1 score, precision, and recall for our CNN method versus the conventional two-layer neural network and thresholding using the Song filter. The $*$ indicates that the control method performed significantly differently than our CNN using a one-tailed paired Student's t test ($p < 0.5$). Error bars show the standard error of the mean.

is the case, partitioning calls based on span (as one often does when employing a spectrogram) could obscure this pattern.

Another curious factor is that where our CNN makes mistakes, it makes the same mistakes that a human being would. The vast majority of false-positive pixels are within one or two pixels of a human-labeled call. Since there is no accepted hard rule about which pixels are really part of a call (we relied on our best judgment, and our decisions are admittedly inconsistent), we believe that this error is acceptable. Obtaining a higher F_1 score may thus be very difficult without developing hard and fast (and arbitrary) rules about which scalogram elements are part of a call.

Our process to find the best CNN architecture was admittedly ad hoc: it is very likely that better architectures exist. However, it is difficult to proceed without more high-quality human-labeled data. Any successful architecture we might develop is vulnerable to overfitting on our small data set. Although MouseTube is an excellent source of data, many mouse biologists are reluctant to place their data online for others to analyze. Further, labeling this data in order to train the CNN is time consuming; in our case it took several days of tedious labor. More data would surely improve the quality of our CNN, and help mitigate the overfitting issue. It is important to note that different strains of mice vocalize differently, and some strains also produce more nonvocal noise during vocalizations. A high quality extraction algorithm should be robust to this issue, but it will need a wide variety of training data to achieve this.

Ultimately, we believe that using these tools will aid in the quality of the analysis of laboratory mouse USVs. Employing a scalogram rather than a spectrogram will help avoid misclassifying calls based on their frequency modulation. And using a convolutional neural network to identify the calls will help save labor, as a computer could do the majority of call identification with little oversight from a human technician.

Acknowledgment

Thanks to the McCormick Scholar award at the University of Puget Sound, that funded this research. Past training was facilitated by Grant T32 DA007262 from the National Institute on Drug Abuse (NIH). The TensorFlow library is the property of Google, Inc., and all mouse data is courtesy of MouseTube, and the laboratories of Jonathan Chabout and Joe Dougherty.

References

[1] G. P. Lahvis, E. Alleva, and M. L. Scattoni, "Translating mouse vocalizations: prosody and frequency modulation," *Genes, Brain and Behavior*, vol. 10, no. 1, pp. 4–16, 2011.

[2] S. Jamain, K. Radyushkin, K. Hammerschmidt, S. Granon, S. Boretius, F. Varoqueaux, N. Ramanantsoa, J. Gallego, A. Ronnenberg, D. Winter, J. Frahm, J. Fischer, T. Bourgeron, H. Ehrenreich, and N. Brose, "Reduced social interaction and ultrasonic communication in a mouse model of monogenic heritable autism," *Proceedings of the National Academy of Sciences*, vol. 105, no. 5, pp. 1710–1715, 2008.

[3] M. L. Scattoni, S. U. Gandhy, L. Ricceri, and J. N. Crawley, "Unusual repertoire of vocalizations in the BTBR T+tf/J mouse model of autism," *PLoS ONE*, vol. 3, no. 8, p. e3067, 08 2008.

[4] M. L. Scattoni, J. Crawley, and L. Ricceri, "Ultrasonic vocalizations: A tool for behavioural phenotyping of mouse models of neurodevelopmental disorders," *Neuroscience & Biobehavioral Reviews*, vol. 33, no. 4, pp. 508 – 515, 2009, risk Factors for Mental Health: Translational Models from Behavioral Neuroscience.

[5] C. Menuet, Y. Cazals, C. Gestreau, P. Borghgraef, L. Gielis, M. Dutschmann, F. Van Leuven, and G. Hilaire, "Age-related impairment of ultrasonic vocalization in tau. p3011 mice: possible implication for progressive language disorders," *PloS one*, vol. 6, no. 10, p. e25770, 2011.

[6] J. B. Panksepp, K. A. Jochman, J. U. Kim, J. J. Koy, E. D. Wilson, Q. Chen, C. R. Wilson, and G. P. Lahvis, "Affiliative behavior, ultrasonic communication and social reward are influenced by genetic variation in adolescent mice," *PLoS ONE*, vol. 2, no. 4, p. e351, 2007.

[7] J. Chabout, P. Serreau, E. Ey, L. Bellier, T. Aubin, T. Bourgeron, and S. Granon, "Adult male mice emit context-specific ultrasonic vocalizations that are modulated by prior isolation or group rearing environment," *PLoS ONE*, vol. 7, no. 1, p. e29401, 01 2012.

[8] T. E. Holy and Z. Guo, "Ultrasonic songs of male mice," *PLoS Biol*, vol. 3, no. 12, p. e386, 11 2005.

[9] J. Nyby, C. J. Wysocki, G. Whitney, G. Dizinno, and J. Schneider, "Elicitation of male mouse (Mus musculus) ultrasonic vocalizations: I. Urinary cues." *Journal of Comparative and Physiological Psychology*, vol. 93, no. 5, p. 957, 1979.

[10] H.-S. Mun, T. V. Lipina, and J. C. Roder, "Ultrasonic vocalizations in mice during exploratory behavior are context-dependent," *Frontiers in behavioral neuroscience*, vol. 9, 2015.

[11] I. Branchi, P. Campolongo, and E. Alleva, "Scopolamine effects on ultrasonic vocalization emission and behavior in the neonatal mouse," *Behavioural Brain Research*, vol. 151, no. 12, pp. 9 – 16, 2004.

[12] H. Wang, S. Liang, J. Burgdorf, J. Wess, and J. Yeomans, "Ultrasonic vocalizations induced by sex and amphetamine in M2, M4, M5 muscarinic and D2 dopamine receptor knockout mice," *PLoS ONE*, vol. 3, no. 4, p. e1893, 04 2008.

[13] L. A. Holmstrom, L. B. Eeuwes, P. D. Roberts, and C. V. Portfors, "Efficient encoding of vocalizations in the auditory midbrain," *The Journal of Neuroscience*, vol. 30, no. 3, pp. 802–819, 2010.

[14] E. G. Neilans, D. P. Holfoth, K. E. Radziwon, C. V. Portfors, and M. L. Dent, "Discrimination of ultrasonic vocalizations by CBA/CaJ mice (Mus musculus) is related to spectrotemporal dissimilarity of vocalizations," *PLoS ONE*, vol. 9, no. 1, p. e85405, 01 2014.

[15] P. Boersma and D. Weenink, "Praat: doing phonetics by computer (version 6.0.29)," May 2017.

[16] R. Specht, *Avisoft-SASLab Pro (Version 5.2.10)*, Avisoft Bioacoustics, Berlin, Germany, June 2017.

[17] M. Van Segbroeck, A. T. Knoll, P. Levitt, and S. Narayanan, "Mupet-mouse ultrasonic profile extraction: A signal processing tool for rapid and unsupervised analysis of ultrasonic vocalizations," *Neuron*, vol. 94, no. 3, pp. 465–485, 2017.

[18] J. M. Grimsley, J. J. Monaghan, and J. J. Wenstrup, "Development of social vocalizations in mice," *PloS one*, vol. 6, no. 3, p. e17460, 2011.

[19] V. B. Deecke and V. M. Janik, "Automated categorization of bioacoustic signals: Avoiding perceptual pitfalls," *The Journal of the Acoustical Society of America*, vol. 119, no. 1, pp. 645–653, 2006.

[20] M. Müller, K. von Hünerbein, S. Hoidis, and J. W. Smolders, "A physiological placefrequency map of the cochlea in the cba/j mouse," *Hearing Research*, vol. 202, no. 12, pp. 63 – 73, 2005.

[21] N. Y. Song, J. Nicon, B. Min, R. C. C. Cheung, M. A. Amin, and H. Yan, "Noise filtering and occurrence identification of mouse ultrasonic vocalization call," in *2013 International Conference on Machine Learning and Cybernetics*, vol. 03, July 2013, pp. 1218–1223.

[22] G. Kaiser, *A friendly guide to wavelets*. Boston: Birkhäuser, 1994.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[24] Y. LeCun, L. Jackel, B. Boser, J. Denker, H. Graf, I. Guyon, D. Henderson, R. Howard, and W. Hubbard, "Handwritten digit recognition: Applications of neural network chips and automatic learning," *IEEE Communications Magazine*, vol. 27, no. 11, pp. 41–46, 1989.

[25] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[27] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[28] I. Wallach, M. Dzamba, and A. Heifets, "Atomnet: a deep convolutional neural network for bioactivity prediction in structure-based drug discovery," *arXiv preprint arXiv:1510.02855*, 2015.

[29] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[30] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang ***et al.***, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[31] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot ***et al.***, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[32] D. E. Rumelhart, J. L. McClelland, P. R. Group *et al.*, *Parallel distributed processing*. MIT press Cambridge, MA, USA:, 1986, vol. 1.

[33] T. M. Mitchell, *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[34] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[35] N. Torquet, F. De Chaumont, P. Faure, T. Bourgeron, and E. Ey, "mousetube–a database to collaboratively unravel mouse ultrasonic communication," *F1000Research*, vol. 5, 2016.

[36] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

[37] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.