THE OHIO STATE UNIVERSITY

Peak-Picking SOAE Spectra

Project Category: Research
Physics 5680, Autumn 2024

Author(s): Seth Peacock

December 30, 2024

Abstract

Spontaneous otoacoustic emissions (SOAEs - the sounds an ear makes) provide a readily observable window into the *in vivo* workings of the ear. Identifying and characterizing peaks in SOAE spectra (peak-picking) is essential for subsequent analysis. Traditional approaches relying on maximum picking can be effective, but often require hand-tuning of parameters and struggle with particular types of peaks. Curve fitting techniques that fit a collection of peaks to the experimental spectrum are most effective when given a set number of peaks to fit and initial guesses for their locations. As such, we develop a machine learning model which provides the number and approximate location of peaks in an SOAE spectrum, which are then used for standard nonlinear curve fitting. Our model builds on recently developed networks for spectroscopy and chromatography data with an approach tailored to a dataset of experimentally observed SOAE spectra.

1 Introduction

Spontaneous otoacoustic emissions (SOAEs - the sounds an ear makes) provide a readily observable window into the *in vivo* workings of the ear. Identifying and characterizing peaks in SOAE spectra (peak-picking) is essential for subsequent analysis. For example, if we wish to design a model of an ear, we would like our model to exhibit a spectrum with peaks similar to those observed in experiment; such a comparison necessitates a thorough characterization of the peaks we intend to replicate. Moreover, inter- and intraspecific features of these experimental spectra may inspire novel approaches to modeling and reveal features of the ear's operation (such as why it is that an ear makes sound at all).

The ideal outcome of peak-picking is a peak list with frequency centers, prominence (height above the surrounding noise floor), width (half width half maximum or HWHM), and shape (Lorentzian, Gaussian, etc). Armed with such a peak list, we are then better prepared to hypothesize about the origin of this activity within the ear, compare these peaks within a particular species, compare them with other species to identify shared characteristics, etc. The reconstruction of peak lists from an experimentally observed spectrum (known as deconvolution) is an ill-posed problem, as arbitrarily many peak lists could create the same experimentally observed spectrum due to overlapping. Nevertheless, we would like to do the best we can.

Traditional (non-machine learning) approaches to peak-picking often rely on maximum picking. Due to the inevitable presence of noise, these require some amount of pre-processing to prevent small noise fluctuations

as being counted as maximums. One obvious solution is to smooth the data first, but this raises an obvious question: how much should we smooth the data? Too much smoothing and small peaks may be smoothed out of the data, too little, and noise "peaks" are counted as maxima. Another option is to set a threshold for a minimum peak prominence; again, we must decide on such a threshold, and the appropriate choice for one spectrum may not be appropriate for another. Such hand-tuning is not desirable for large sets of SOAE data; in our case, we have a set of around 4000 experimentally observed spectra. Moreover, there is the issue of "shoulder peaks" – peaks that overlap with a larger peak that manifest as a bump as you travel down the side of the main peak. Figure 1 shows such a peak; there is only one true maximum in this region, so any approach relying on unadulterated maximum picking will be unable to resolve these. However, a human can easily identify this as the sum of two neighboring peaks, and it seems reasonable to expect an algorithm to be able to detect this (at least under favorable conditions).

If the noise floor, number of peaks, and their approximate locations are known, curve-fitting methods that construct a spectrum and vary peak parameters to minimize error between the constructed spectrum and the original can be effective. However, without such initial guesses this approach could result in an extremely large number of peaks chosen to fit every small fluctuation due to noise. Some sort of regularization must be implemented to minimize the number of peaks chosen, but this can easily result in choosing less peaks than needed to appease the regularization constraints. It is also computationally inefficient to repeat fitting for each possible number of peaks. As such, our model will be charged with taking an SOAE spectrum as input and outputting a fixed number of peaks with approximate locations in the spectrum to be used as initial guesses for subsequent nonlinear curve fitting. Since training requires labeled peaks, we generate a synthetic dataset for model training and evaluation via an error metric which takes advantage of the ground truth peak labels. Ultimately, we will evaluate its performance on the real experimental dataset with the mean squared error (MSE) between the reconstruction from nonlinear curve fitting, as well as through visual comparison of a small sample of original vs reconstructed spectra.

2 Related Work

The primary inspiration for our network structure is a model proposed by Schmid et al. for the deconvolution of nuclear magnetic resonance (SMR) spectra [3]. Ten different versions of the input are created via these techniques and fed into ten channels of an "inception"–like convolutional layer (see [4]), which has many filters spread out over several different kernel sizes (2, 4, 8, 32, 64). All of these (filtered) channels go into a "TimeDistributed" dense layer, used (instead of a normal dense layer) to allow for parallel processing of the separate channels. This outputs 64 channels (each channel containing N frequency bins), which then goes into another TimeDistributed layer (outputting 32 x N), and then into a bidirectional Long Short-Term Memory (LSTM) layer which goes along the frequency axis. This is used to further encode relationships between nearby frequency bins. The LSTM then goes to three back-to-back TimeDistributed layers, the last of which is the output layer. There are five nodes for each frequency bin: three probabilities for classification (no peak, narrow peak, broad peak) and two for the narrow/broad peak widths. ReLU activation functions were used for all layers. For a cost function, they equally weight binary cross entropy loss for the classification and MSE for the regression (peak widths). They do not regress the shape (Lorentzian vs Gaussian) or height of the peaks, as they get these parameters through nonlinear curve fitting as described in section 1. When labeling peak positions, they label the true center in addition to some number of peaks on either side (this label width increases with the width of the peak). This is because there are several data points near the true peak that look peak-like enough that the model should be encouraged to pick them as well, especially with the addition of noise. Their training dataset consisted of 250,000 generated spectra, and they trained for 100 epochs with an ADAM optimizer, a batch size of 32, and a learning rate of 0.001.

Schmid et al. implement several other methods which we would like to explore in future finetuning of our model. First, they pre-process the data through a 'dynamic scaling" method: they apply a combination of minimum and maximum filters to the spectrum to enhance local differences (thus amplifying peaks), and then essentially performing min-max scaling with the filtered spectra to normalize the magnitudes to approximately $[0, 1]$. To allow convolutional kernels to capture wider features without the excess of parameters that comes with larger kernel sizes, they also create copies of the spectrum shifted in frequency.

They also point out that when labeling their synthetic data, "Employing the peak parameters used to create the synthetic spectra is not a sensible option" since the final generated spectra could have been created with many different peak lists, and the model would be penalized for missing peaks we could never have hoped for it to identify. For example, two thin peaks side by side may appear for all intents and purposes as a single, wider peak. To address this, they take the pre-noise version of the synthetic spectra, *shrink* the widths of the peaks by a factor dependent on the signal-to-noise-ratio (SNR), and then pick out the local maximums. This automatically "distinguishes between what should be detected and what not" – a peak with high SNR is less likely to be obscured by nearby peaks, and so they sharpen this to make it more likely to be identified by the local maximum finder. The final peak width and height labels are then calculated using a system of equations.

Another model for deconvolution of NMR spectra is proposed by Li et al. in [2]. Their model consists of seven convolutional layers, the output of which is then split into two paths. One path goes through max pooling and a softmax activation function which then provides the output probabilities for no peak, Class 1, or Class 2 peak. Their class distinction is between peaks which dominate and peaks which are dominated by their neighbor (the latter being the aforementioned "shoulder peaks"). They label each peak center and the bin on either side, and their ultimate peak label is chosen via a non-maximum suppression algorithm. To deal with overlapping peaks, they use nonlinear curve fitting with a single peak curve to determine how well a region with two peaks could be explained by a single peak: "when the maximal difference in amplitude for all points of the original and the fitted peak is $< 3\%$ of the peak amplitude, a synthetic peak pair is excluded from the training set." The other path is used to regress peak amplitude, width, shape, and sub-pixel peak position relative to the on-grid points (for data points predicted to be a peak). They also use MSE for the regressor and cross-entropy for the classifier, and trained for 4000 epochs with an Adam optimizer and a learning rate of 0.002.

Finally, [1] develop a model for peak detection in reversed-phase liquid chromatography. Their model breaks up the spectrum into small equally spaced regions. They then output predictions for whether or not there is a peak in each region, as well as the center location of the peak and the area of the peak (if there is a peak identified). The structure of their network is a series of convolutional blocks, each with a convolutional layer, ReLU activation, batch norm, and max pooling. They note that the sharing of weights in a convolutional layer (each kernel is applied in the same way to each part of the spectrum) leads to a translationaly invariant network, implying that "simulated chromatograms do not need to be realistic from a global perspective, but more so from a local perspective." This is also true for much of the network we will use (and the network proposed by Schmid et al.), since the TimeDistributed layers share weights for each of the N frequency bins. Thus most of the processing on each bin involves only the bin itself and its neighbors (up to the width of the largest kernel in the inception layer). The exception for our model is the transformation performed by the bidirectional LSTM, which travels along the entire frequency axis and is able to detect and take advantage of relationships on an arbitrarily global scale.

3 Dataset

To obtain an SOAE spectrum, a microphone is placed in an animal's ear and set to record what comes out. We have just under 4000 samples from over a decade of experiments. There are data from various animals, though we will only use data from lizards and humans (which make up the vast majority of the set). The samples are audio signals recorded at 44.1kHz which then chopped up into windows each with 32768 points. Fourier transform magnitudes are found for each window, followed by averaging over all windows to minimize noise. The final spectra is cropped to 8192 points (there is rarely significant activity beyond this) and plotted on a dB scale.

To generate a synthetic spectrum, we first need a noise floor. To obtain noise floors at least roughly similar to the experimental dataset, we approximate a noise floor from each of the samples in the dataset. First, we perform a LOWESS fit with a fraction parameter of 0.3 (medium-broad) which allows us to "skate" under high peak regions. We then take the minimum of the fitted curve and the original spectra at each bin, and smooth with a Gaussian filter (standard deviation of 150). This approach resulted in a noise floor that appeared consistently too low for the low frequency part of the spectrum (the LOWESS fit was too low,

causing the minimum to be as well) and so the first 300 bins are replaced with the original spectrum values before smoothing.

For each noise floor, we then generate two synthetic samples. The first makes up our "transfer" dataset, and is meant to closely replicate a typical experimental sample of the relevant species (not the specific profile which yielded the noise floor). Many parameters were finetuned here to yield spectra that looked the most like the original spectra (though the messy low frequency range of the spectra was not attempted to be replicated, as this will be likely be too difficult to deconvolve and will be ignored in the end). Quantitative parameter distributions can be found in the GitHub repository, but we give a rough account here. Positions were drawn from a chi-square distribution centered about 2/5 down the spectrum. If the original noise floor was from a human, $\approx 1-5$ peaks were chosen with a 90% chance to be thin (HWHM uniformly in [3, 10] Hz) and 10% chance to be wide (HWHM uniformly in [10, 100] Hz). If the original noise floor was from a lizard, there are $\approx 5-15$ peaks chosen; this time, 90% are wide and 10% are thin (same uniform ranges as before). For lizards, an initial center position was drawn from the chi-square, and the peaks were drawn from a uniform distribution spanning this center frequency $\pm$1500Hz to reflect tighter clustering observed in lizard spectra. Peak heights were slightly higher for humans, but both were chi-square distributions with the most density in $\approx [3, 12]$. The purpose of the chi-square distributions was to concentrate within an rough experimentally observed range, while still having a reasonably heavy tail for higher number peaks/heights. The peak shape is Lorentzian. To minimize overlapping peaks, peak centers are at least 50Hz apart (this still results in plenty of overlapping/shoulder peaks for the network to learn).

For each noise floor, another sample is generated for a "general" dataset, whose purpose is to give the network a variety of peak regions to encourage generalization to nonstandard spectra. Peak positions are drawn uniformly across the whole frequency range, and many more peaks are picked (uniform [20, 35]) since they are no longer concentrated to a smaller region. Each peak has a 50% chance to be thin or wide, and the peak heights are picked from the same distribution as the transfer lizard samples.

Finally, each spectrum is converted to a linear scale, additive noise is generated, the spectrum is converted back to dB, and min-max scaling is performed (using the min/max of each spectrum, not the overall dataset). The additive noise is normally distributed with mean zero and a standard deviation matching the original experimental spectrum. To determine this standard deviation, we take a peakless region of the spectrum (8-11kHz) in a linear scale and do a LOWESS fit (fraction = 0.1) to find the center of the noise floor. We then take the difference between data points in this region and the LOWESS fit. As seen in Figure 5, these data points very accurately sit in a normal distribution; the standard deviation of the additive noise for the synthetic spectra (applied to the *entire* spectra) is the estimated standard deviation of this experimentally observed distribution. Interestingly, the estimated standard deviation itself appears to sit in one of several normal distributions, presumably corresponding to different lab setups.

We produce about 7500 samples, of which 70% become our training set, and 15% each become validation and testing (half in each are from the general set and the other half are from transfer, stratified by species). See Figure 2, Figure 3, and Figure 4 for examples of each kind of synthetic spectra with the experimental spectrum that provided the noise floor underneath.

4 Methods

4.1 Network Structure

The structure of our peak-picker network is very similar to the one developed by [3]. The input is a spectrum with $N = 8192$ bins, which first enters an "Inception"-like (see [4]) set of parallel convolutional layers. Our kernel widths and the number of filters for each are similar to those used in [3], with two changes. First, we chose kernel widths with odd numbers of bins to be symmetric around the single bin at the top of a peak. Second, since we did not implement their spectrum shifting method (which increased the receptive field of the convolutional layers), we simply added wider kernels to capture more large scale features. Our choices were (kernel width, number of filters): (3, 4), (5, 8), (9, 16), (15, 32), (31, 32), (55, 32), (71, 16), (101, 8), (149, 4), and (201, 2). Each of these kernels are swept across the spectrum and perform weighted averages (the weights being trainable parameters), outputting N bins with 154 channels each (one per filter).

The 154 channels from each of these N bins is then fed into N identical (shared weights) fully connected Dense layers connected to 64 nodes. This is accomplished by a TimeDistributed layer wrapper, which applies the same transformation to each of the N bins. This $N \times 64$ data is directed into another TimeDistributed Dense layer with 32 nodes, meaning we have a single set of weights fully-connecting 64 to 32 nodes, and the channels from each the N bins goes through this set of weights. This $N \times 32$ data enters a Bidirectional Long Short Term Memory (LSTM) layer (though we will test versions of the model with or without this layer). This is a type of recurrent neural network which was (in part) designed to solve the vanishing gradient problem. It travels along the frequency axis and related information between bins; "bidirectional" indicates that it goes both directions along the axis (from low frequencies to high and vice versa). This is fed into another TimeDistributed Dense layer (32 nodes $\times N$ bins), then another (16 nodes $\times N$ bins), and finally an output layer (1 node for each of the N bins). Rectified linear unit (ReLU – 0 for negative numbers, identity otherwise) activation functions are used in all layers except the output. Here, a sigmoid function $\frac{1}{1+e^{-x}}$ is used to obtain peak probabilities $\in (0, 1)$.

4.2 Peak Labeling

Similar to [3], we label not just the center of the peak itself but also bins on either side, depending on the width of the peak. Due to the effect of noise, it is quite common that the true center is not even the highest bin, especially for wider peaks. By visual inspection we determined that the thinnest peak that deserves this treatment has a HWHM of 5Hz, and that our widest peaks (100Hz HWHM) should have an additional 15 bins on either side. Along with our minimum distance of 50 Hz between any peaks, this also guarantees that there will always be non-peak labeled bins in between any two adjacent peaks (which will be important in subsection 5.1). Thus we chose the number of additional bins to label (on either side – that is, the "half width") to be a linear function of HWHM that is 1 at 5Hz and 15 at 100Hz (see Figure 6).

4.3 Training

Our loss function is binary cross entropy with two weighting factors. First, we would like to weight our loss function for peak bins by a factor $\in [0, 1]$ dependent on the prominence. To determine this, we will set a prominence level k under which we will be more tolerant if the model is unable to resolve the peak. This will encourage the model to prioritize the taller peaks. We'd like a function that is 0.5 at k, falls off quickly towards zero below k, and approaches 1 quickly above k. A shifted sigmoid $\frac{1}{1+e^{k-x}}$ has exactly this property. In our dataset, we've included peaks with prominences arbitrarily close to zero (though these are rare), as these may still be detectable under good conditions (no nearby peaks, low width, etc). We found a prominence of $\approx$ 3dB to be a reasonable cutoff for being able to detect peaks in noisy/crowded regions by eye, so we initially set $k = 3$.

Note that now all non-peak bins (which make up the vast majority of a spectrum) have a weight of 1, while peak bins are all weighted by a factor < 1. This weight differential (along with the relative abundance of non-peak bins) encourages the network to prioritize minimizing false positives over false negatives. To some extent this is good, as we don't want to over-pick peaks (as discussed in section 1). However, we found this caused under-picking to an unsatisfactory degree, and so included another weight parameter $PE \geq 1$ which is multiplied by the weight function of all (true) peak bins (PE = "Peak Encourage"). Multiple combinations of PE and k were tested, and their interplay is explored in subsection 5.3.

Following [3], we use an ADAM optimizer with a learning rate of 0.001 and a batch size of 32. Training was carried out for 25 epochs for each hyperparameter combination and the model that gave the lowest error (see subsection 5.1) was chosen to represent that combination.

5 Results/Discussion

5.1 Error Metric

Ultimately, we will test our model by its ability to reconstruct a spectrum given a set number of peaks and just initial guesses for the peak positions, performing a nonlinear curve fitting to find the exact locations, widths,

and heights. Therefore, our top priority is to determine the number of peaks, along with their *approximate* locations. Since every peak receives a range of labels, and we only care about approximate locations, we want to assign our model full marks if it only identifies (say) the middle 3 marked bins around a peak but does not get the tails of peak labels on either side. We therefore use the following accuracy metric. First, set a probability threshold (discussed shortly) and mark all bins whose predicted peak probability exceeds this threshold as 1 and the rest as 0. Each chunk of contiguous 1s is then taken to predict a single peak, with these peak regions interrupted by chunks of contiguous 0s. Then for each of the M regions of contiguous 1s or 0s, we compare the predicted number of peaks ($y^m_{\text{pred}} = 1$ for regions of 1s and 0 for regions of 0s) to the actual number of peaks in that region y^m_{true}. The final error is calculated as $\sum_{m=1}^{M}(y^m_{\text{pred}} - y^m_{\text{true}})^2$.

When comparing hyperparameter combinations, it may be the case that one version of the model is better suited for a higher or lower threshold. Moreover, it may be the case that while training loss continues to decrease for later epochs, an earlier epoch had a better error. Therefore for each epoch of each model we sweep through a variety of thresholds ($T \in \{0.1, 0.2, ..., 0.8, 0.9\}$) and pick the one which leads to the lowest error on the validation set ($\equiv T_0$). After training, we revert the model to the epoch which had the lowest error. This allows us to minimize the effect of overfitting on the training set, as well as the very real (and observed) possibility that minimizing *loss* (a relatively arbitrary metric) may not correspond to the minimum *error* (a concrete metric we actually care about). Finally, to reduce the probability that a particular threshold for a particular model was particularly effective on the validation set by chance, the final error for that model is assigned as the error on the *test* set using T_0. Since we are using the test set to choose the best hyperparameter combination, a truly unbiased evaluation of the model would require a fourth set of synthetic data. However, recall that our true goal is to apply this model to our real experimental SOAE data. This data is unlabeled, so we cannot apply the same metric used for the synthetic data. We discuss a method to test our model's performance on the experimental data in section 6.

5.2 Hyperparameter Results

We varied three hyperparameters: $k \in \{0, 3\}$ (the peak height threshold for which the weighting function output was 0.5, rapidly approaching 1/0 for peaks higher/lower), $PE \in \{1, 5, 10, 15\}$, and including/not including LSTM. Models were trained with all possible combinations of these parameters. The best epoch / peak prediction threshold on the validation set was recorded along with that epoch and threshold's error on the test set; results can be found in Table 1.

5.3 Discussion

All models with the LSTM performed better than models without. It is possible that this is simply due to an increase in model complexity; adding the LSTM took the number of trainable parameters from 20,159 to 26,431 and took significantly longer to train. In future work, we would like to compare the model with the LSTM against a model without the LSTM but with an additional dense layer so that the number of trainable parameters is similar. However, it makes sense that facilitating the model's ability to relate information between bins more distant than the kernel widths would increase performance. For example, for shoulder peak scenarios (Figure 1) the bin at the center of the shoulder peak doesn't look very peak-like until you also know that there is another (taller) peak nearby.

Four out of the top five models had $k = 0$, implying that setting $k = 3$ (being more lenient with peaks whose height was below 3) was largely not beneficial. This likely indicates that we underestimated the model's ability to resolve short peaks; if the $k = 3$ models performed better, that would have indicated that those short peaks would be too difficult to resolve and it would be beneficial to prioritize the higher peaks to make sure it can at least get those. The question remains whether different values of k would perform even better (including $k = -\infty$, corresponding to not implementing prominence-dependent-weighting at all).

The top model was $k = 0$ and $PE = 10$, indicating that $PE = 10$ was the closest to the "correct" amount to encourage peak finding to offset the relative abundance of non-peak bins. Note that higher k implicitly lowers the weights on all peak bins (just some more than others), while higher PE directly raises the weights on all peak bins (equally). The second best model was $k = 3$ and $PE = 10$; the combination of raising k (lowering

peak bin weights) and lowering PE (raising peak bin weights) may have had the effect of maintaining this balance.

Out of the top four models, three had "reasonable"/expected best peak prediction thresholds near 50%. However, almost all models with $k = 3$ had much lower thresholds near 10%. The only other model with such a low threshold was $k = 0$ and $PE = 1$ (that is, the minimum PE peak bin weighting out of all $k = 0$ models). The common thread here is relatively more weight being placed on non-peak bins than peak bins. This makes sense if we interpret the model's priorities as shifting from identifying peaks as 1s to the goal of *identifying non-peak bins as 0s*. It must keep non-peak predictions as low as possible, making it more difficult to make sure peak predictions are as high as possible. Then it is reasonable to expect the best threshold to be low: non-peak bins are all low, so we don't need a high threshold to clear those, but peak bins are all over the place so we must set the threshold as low as possible to ensure we get all peak predictions. Curiously, the second best model $k = 3$ and $PE = 15$ also had a very low best threshold; this may indicate $k = 3$ is enough to cause this effect even with such a high uniform peak weighting $PE = 15$, and/or that other factors are relevant here.

In future work, we would also like to explore the effect of PE and k on the speed of convergence. For example: do different choices ultimately converge to the same/similar model performance after enough epochs, they just do so at different speeds, prioritizing different things along the way?

6 Conclusions/Future Work

Overall, the model performed well on the synthetic spectra. A visual representation of the algorithm's performance on a sample is displayed in Figure 7, which shows the predicted peak probability for each bin as a color. With a final peak counting error near 2 (a little over a single false positive or false negative per sample, due to the square in the error calculation), we can be confident that our model obtained almost all of the peaks and their approximate positions in the synthetic spectra, even those with very low prominence. However, recall that the ultimate goal was to apply the model to our experimental SOAE spectra. Further, we would like to completely characterize the peaks in the spectra, not just find the number and approximate locations. To do this, we will implement a nonlinear curve fitting algorithm to fit a function which is the sum of a fixed estimated noise floor and fixed L (the number of peaks estimated by the network with the threshold that performed best on the synthetic set) Lorentizan peaks with configurable HWHM, prominence, and precise location. Initial guesses for the positions will also be determined by the network, which we will likely choose as the bin in a contiguous chunk of 1s with the highest peak probability (another option is to take the middle of each chunk). To find an estimate for the true noise floor of an experimental spectrum, we will develop another neural network that accomplishes this, training it using our synthetic data for which we have the ground truth noise floors. Finally, to test the performance of our system on the experimental data, we will use a combination of qualitative visual inspection (do the reconstructed spectra look about the same?) and quantitatively via the MSE between the original spectrum and the reconstructed spectra.

6.1 Additional Model Features

There are several features we would like to incorporate in future work. First, we assumed that the shape of our peaks in SOAE spectrum are Lorentzian, but this may not be the case in the experimental spectra. To this end, we will generate synthetic data with peaks that have Voigt profiles (a mix of Lorentizan and Gaussian). Second, to prevent confusing the model by expecting it to resolve two peaks right on top of each other (which to a human would look like a single peak), we simply baked in a hard minimum distance of 50Hz. This is both too wide (two thin peaks with a 5Hz HWHM and sufficient prominence could easily be resolved when less than 50Hz apart) and too thin (two wide peaks with a 100Hz HWHM with 50Hz between their centers still look like a single peak, despite being labeled as two). Therefore we will eventually implement one of (or a combination of) the approaches described in section 2, such as shrinking the pre-noise spectrum by a factor dependent on peak prominence and then maximum picking to determine where we should fairly label the peak centers. Rather than calculate the new prominence and width as in [3], we would likely just replace any double peaks that still had a single maximum after shrinking with a single peak whose prominence and width was some combination of the original two. Finally, we would like to add extra

output nodes to each bin in our current model to regress estimates for the widths and prominences of the peaks. These can then be used as starting guesses for the nonlinear curve fitting. This should increase the speed (if not the accuracy as well) of the curve fitting, which is desirable since – compared to the time it will take for our trained network to provide predictions for a single sample – the curve fitting process will be the primary bottleneck in performance speed when peak-picking a spectrum.

7 Code

All code is provided at https://github.com/seththepeacock/SOAEpeaks

8 Figures

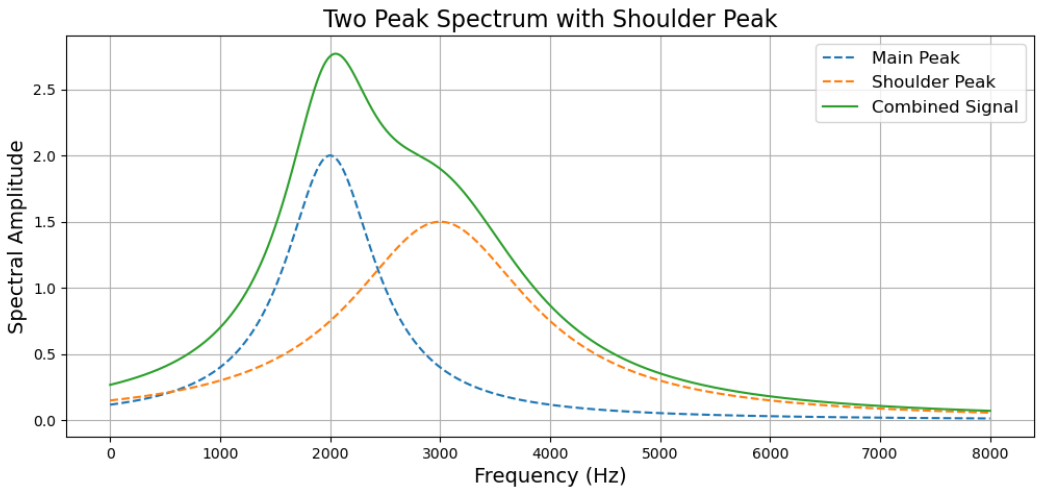

Figure 1: Example of a "shoulder peak" which cannot be resolved by simply maximum finding. Clearly, there's not just one peak in the combined signal, but there's only one maximum.

LSTM	k	PE	Test Error	Best Val Threshold	Best Epoch
Yes	0	10	1.854	0.5	24
Yes	3	15	1.955	0.1	23
Yes	0	15	1.971	0.6	25
Yes	0	5	2.564	0.4	23
Yes	0	1	2.686	0.1	24
Yes	3	5	3.195	0.1	25
Yes	3	10	3.646	0.1	20
Yes	3	1	4.696	0.1	15
No	0	5	4.741	0.8	24
No	0	1	5.021	0.4	24
No	3	5	5.051	0.1	25
No	0	10	5.253	0.7	24
No	3	10	5.565	0.3	15
No	3	15	5.751	0.6	22
No	0	15	5.944	0.9	21
No	3	1	6.487	0.2	24

Table 1: Model types and their metrics (sorted by error on the test set). LSTM indicates whether or not the model had a Bidirectional Long Short Term Memory layer, k was the peak prominence threshold for which the weighting function output 0.5 was applied (approaching rapidly to 1 and 0 for peaks higher/lower), and PE ("peak encourage") was a fixed weight applied to all peak bins to encourage peak finding.

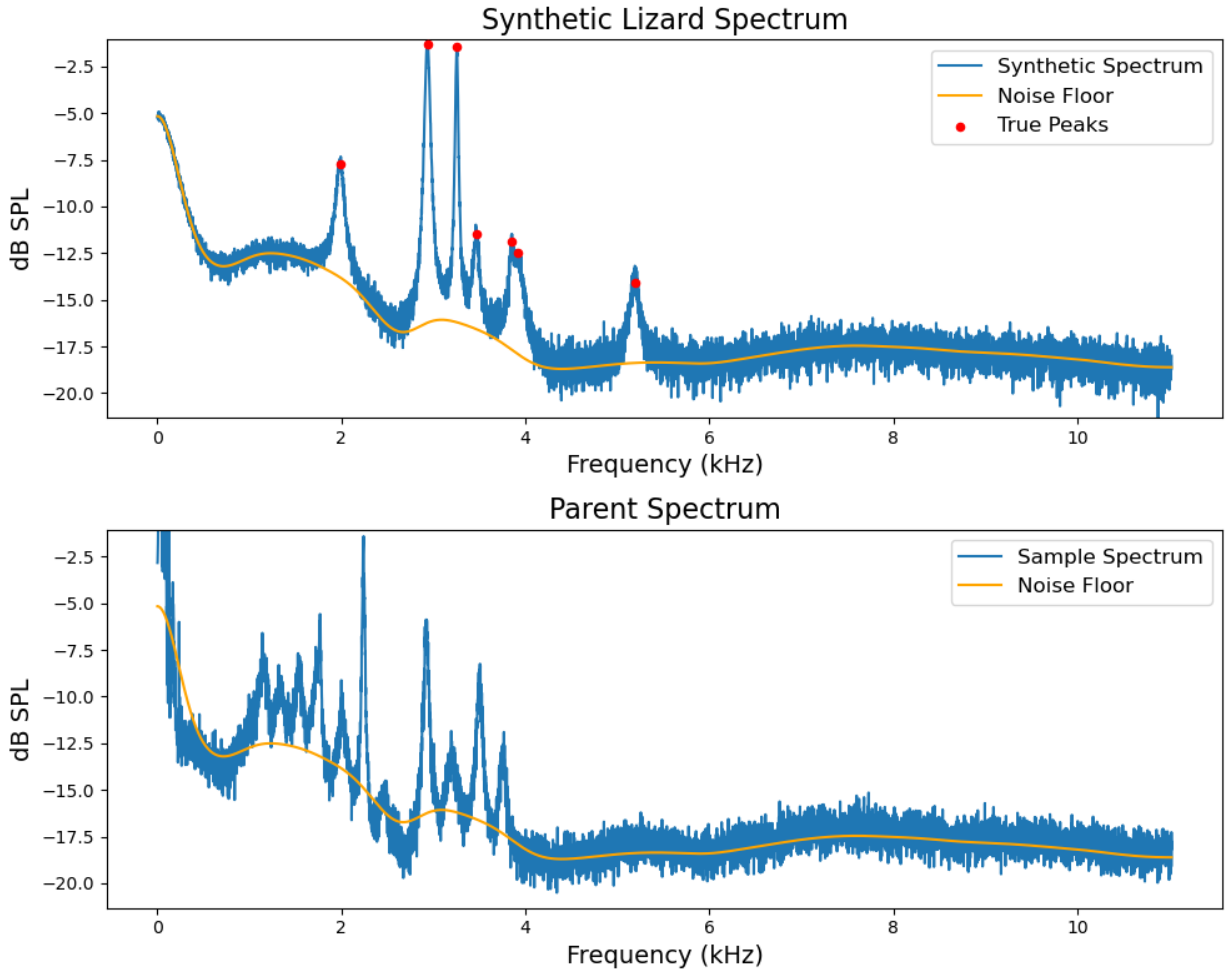

Figure 2: Example synthetic spectrum emulating a typical lizard sample, along with the parent lizard experimental spectrum which provided the noise floor.

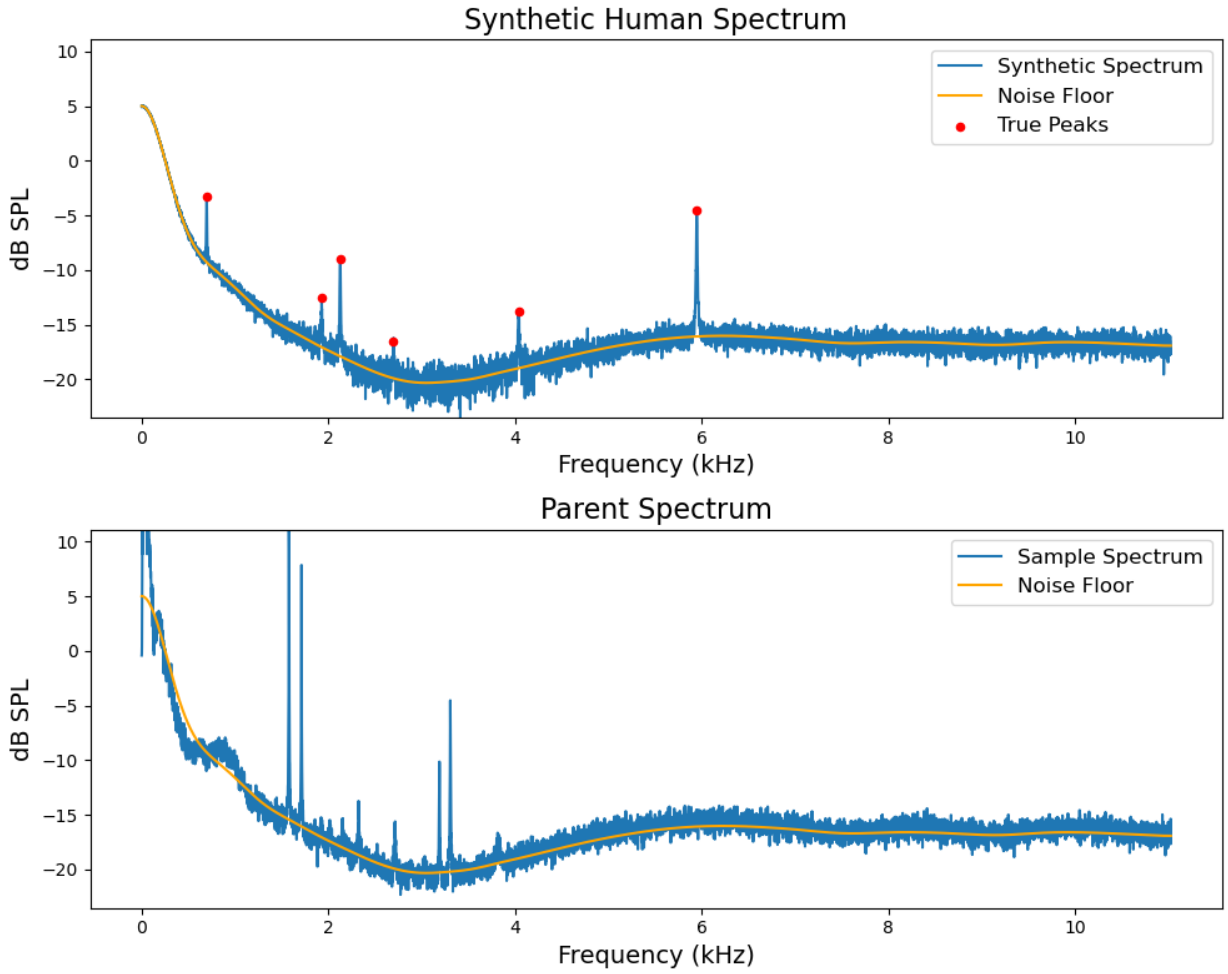

Figure 3: Example synthetic spectrum emulating a typical human sample, along with the parent human experimental spectrum which provided the noise floor.

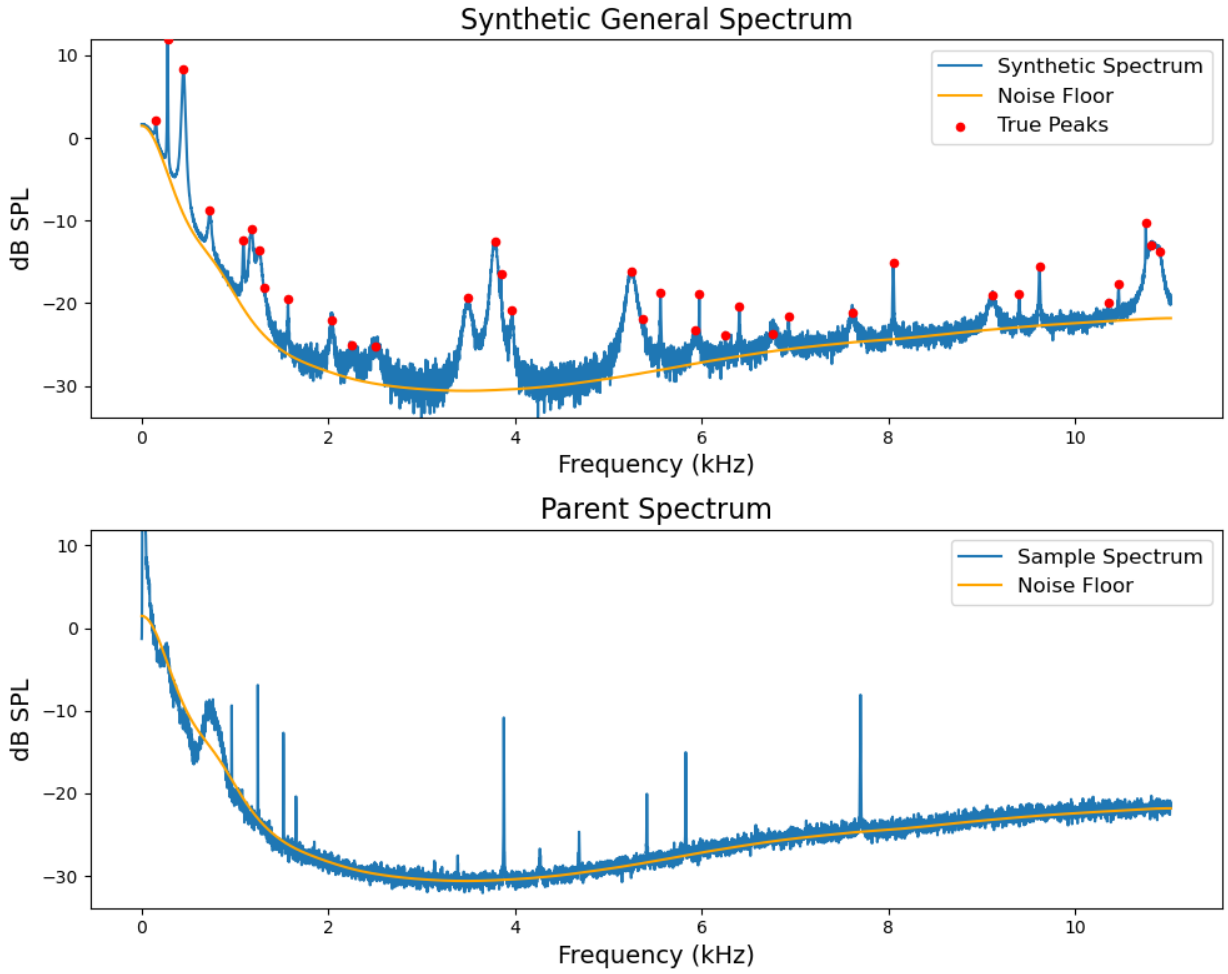

Figure 4: Example synthetic spectrum sample from the "general" dataset, along with the parent experimental spectrum which provided the noise floor.

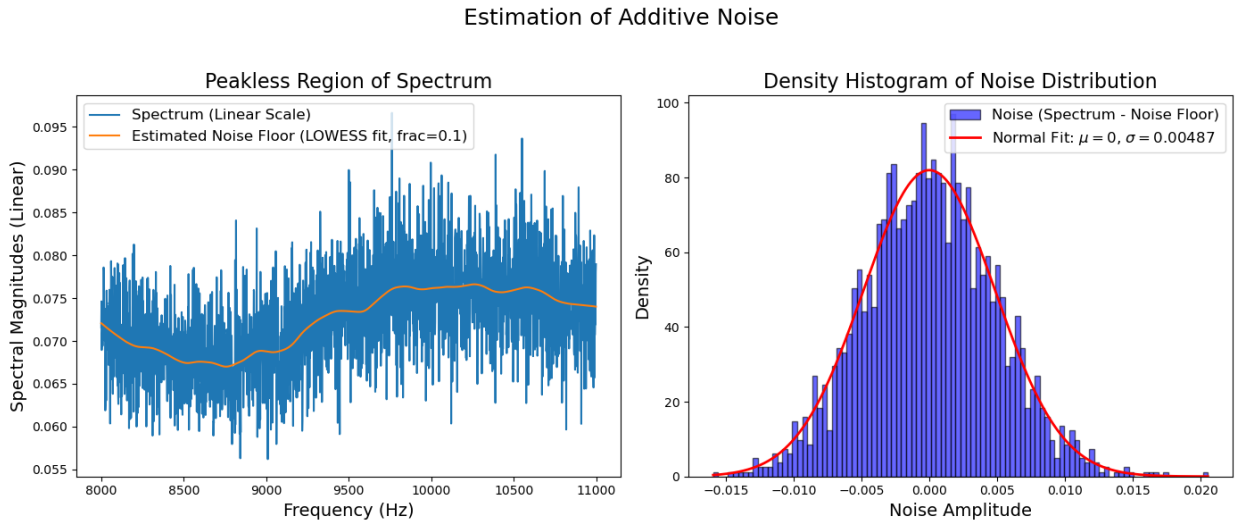

Figure 5: To estimate the normal distribution of noise, a LOWESS curve was fitted to the peakless region of each experimental spectra (in the linear domain), which is taken as an estimate for the noise floor. The distribution of the difference between each datapoint and the estimated noise floor was then fitted to a normal distribution of mean zero; the standard deviation of this distribution was used to generate additive noise (in the linear domain) for the two synthetic data samples corresponding to each experimental spectrum.

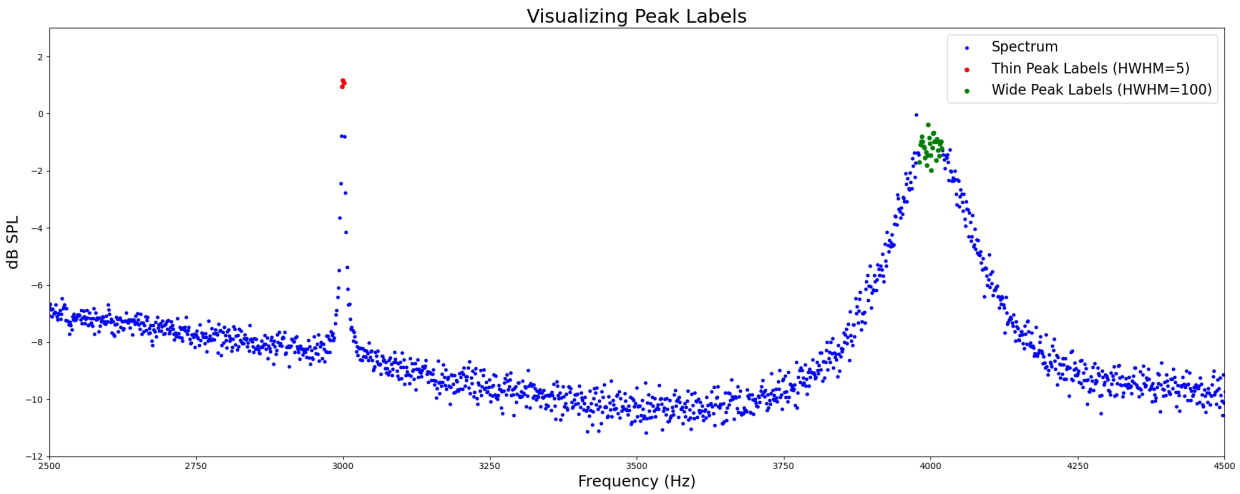

Figure 6: Due to noise and the width of the peak, more than just the center of the peak will look nearly equally peak like. Therefore we label the center of the peaks $\pm L$, where L is a function of the half width. Pictured is the thinnest peak for which L is nonzero (HWHM=5) and the widest peak (HWHM=100) where $L = 15$.

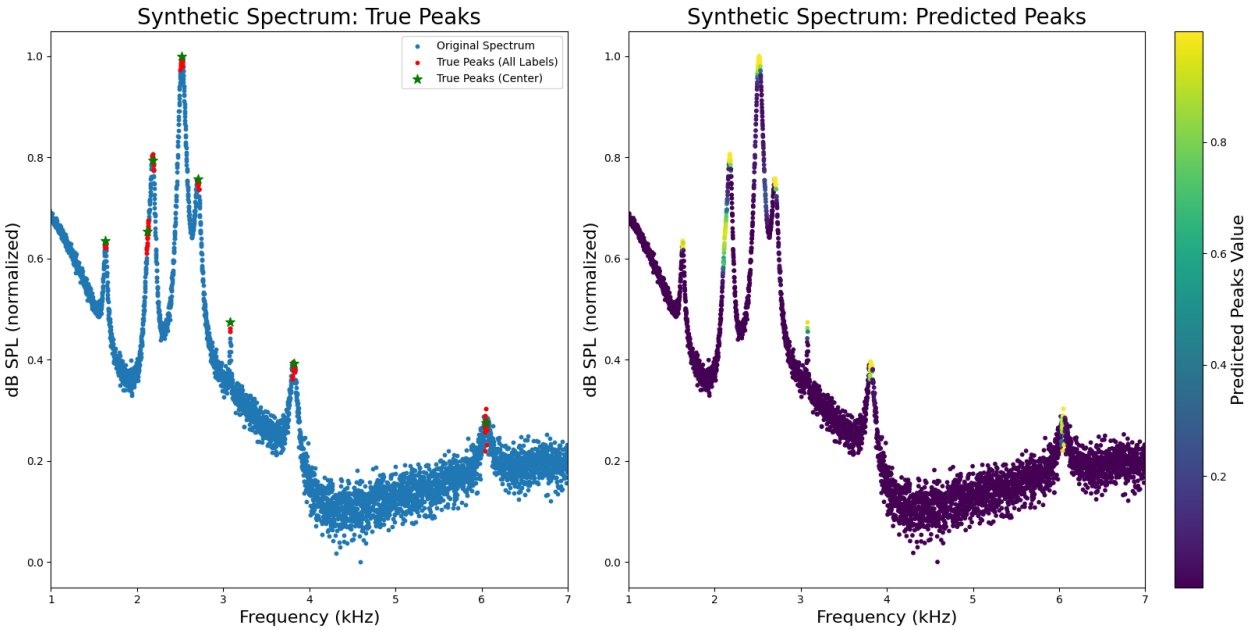

Figure 7: Synthetic spectrum colored by peak prediction value (using the best model yes LSTM, $k = 0$, and $PE = 3$) next to the true peak labels (both the true center in green and all bins that were labeled as peak in red) for this spectrum.

9 Contributions

Thank you to Professor Christopher Bergevin at York University for the SOAE data and guidance along the way!

References

[1] Alexander Kensert, Emery Bosten, Gilles Collaerts, Kyriakos Efthymiadis, Peter Van Broeck, Gert Desmet, and Deirdre Cabooter. Convolutional neural network for automated peak detection in reversed-phase liquid chromatography. *Journal of Chromatography A*, 1672:463005, 2022.

[2] Da-Wei Li, Alexandar L Hansen, Chunhua Yuan, Lei Bruschweiler-Li, and Rafael Brüschweiler. Deep picker is a deep neural network for accurate deconvolution of complex two-dimensional nmr spectra. *Nature communications*, 12(1):5229, 2021.

[3] Nicolas Schmid, Simon Bruderer, F Paruzzo, G Fischetti, Giuseppe Toscano, Dominik Graf, Michael Fey, Andreas Henrici, V Ziebart, B Heitmann, et al. Deconvolution of 1d nmr spectra: A deep learning-based approach. *Journal of Magnetic Resonance*, 347:107357, 2023.

[4] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.