Mode Counting in Acoustic Signals for Spectral Mass Gauging of Unsettled Liquids

Steven Bradley
Software Engineering / Data Science
California Polytechnic State University
San Luis Obispo, United States
stbradle@calpoly.edu

Mateo Ibarguen
Statistics / Data Science
California Polytechnic State University
San Luis Obispo, United States
mibargue@calpoly.edu

Nathan Phillibe
Software Engineering / Data Science
California Polytechnic State University
San Luis Obispo, United States
nphillib@calpoly.edu

***Abstract*—Spectral mass gauging of unsettled liquids is a technique that could aid in human space exploration. Researchers have shown that through the use of Weyl's Law, it is possible to predict the amount of liquid that exists in a container in a low-gravity environment where normal, dry-wet sensors are not adequate. This is done through the counting of peaks caused by acoustic resonances in the liquid when observed over a specific frequency range. Currently, this peak counting is being done manually by a human. In this paper, we explore the automation of this peak counting by using neural networks. We explore many model architectures including convolutional neural networks (CNN), multiple recurrent neural networks (RNN) such as LSTM and GRU, and ensemble models. We show that in a small spectral window, neural networks can count the number of peaks with an accuracy greater than 90%.**

I. Introduction

Researchers at the NASA Ames Research Center are working to solve the problem of measuring the volume of liquid that exists in a container in low-gravity environments. In an environment where there is significant gravity, such as the surface of Earth, the volume of a liquid can be determined by using dry-wet sensors on the walls of the container. However, this approach does not work in a low-gravity environment due to the ability of the liquid to float freely within its container, making it particularly difficult to estimate the volume of the liquid. Through the application of Weyl's Law [1], the volume of liquid in a container can be inferred from analyzing the number of modes that exist in an acoustic response [2].

In this paper, we explore automating the analysis of these acoustic responses using neural networks. We analyze several different network architectures and compare the different results, as well as identify areas of improvement for the future.

II. Related Works

PeakOnly is an approach used for peak resolution in liquid chromatography-mass spectrometry data [3]. This approach utilizes convolutional neural networks (CNN) to detect regions of a signal that contain peaks. The output of these neural networks are fed into another CNN that determines the specific portion of the signal that the peak is comprised of.

Recently, there has been a lot of progress on acoustic signal analysis using neural networks. A research group in Rio uses the Google Speech Recognition dataset to predict English words from 1 second audio clips [4]. Spectral methods are used to preprocess raw waveforms in order to obtain short time Fourier transform (STFT) with a raw waveform. The resulting spectogram demonstrates the frequency distribution and intensity of the audio as a function of time. The neural architecture proposed by this paper is capable of handling command recognition while maintaining a small number of trainable parameters. It uses convolutions to extract short-term dependencies, RNNs and attention to extract long-term dependencies. Figure 1 demonstrates this architecture in more detail [4].

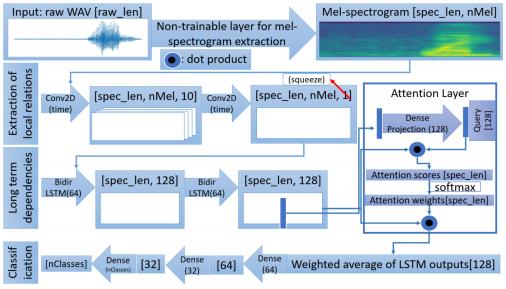

Fig. 1: Referenced Model Architecture [4]

Researchers have also shown that it is possible for a recurrent neural network to learn to count objects [5]. The researchers used an attention model that leverages LSTMs and shows that the model successfully counts objects given three different learning conditions.

In addition to the use of neural networks for acoustic analysis, it has been shown that neural networks can successfully be used for analyzing the signals output by various medical

devices [6], [7]. Researchers show that one-dimensional CNNs are the state-of-the-art solution for classification tasks based on signal analysis [6]. In other explorations, researchers create an ensemble of CNN that makes predictions about a patient's state of mind based on the signals received from various medical devices [7].

III. Dataset Description and Feature Engineering

The dataset used in this paper is a series of spectra that represent acoustic resonances from a tank. Each observation in the dataset is a fixed list of spectra representing readings from multiple actuators at a single point in time. We refer to each single spectra as a channel. For example, a single observation can have ten channels, which equates to ten spectra. A single channel correlates to an actuator/accelerometer pair attached to the tank. While we are working with simulated data – which can have an unlimited number of channels, a real tank will require additional hardware for each additional channel. See Figure 2 for a simple diagram of an example tank's hardware setup of two channels.

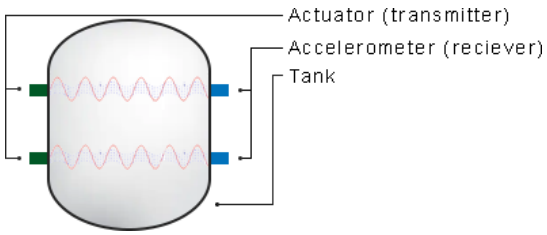

Fig. 2: Example of a Tank Hardware Layout

A. Simulated Data

For this project, we are working with simulated spectra, based on readings from real tank prototypes. Our data is generated via a Matlab script [8], which was given to us by Dr. Khasin. The script supports a number of parameters that can be adjusted to generate different spectra. Since we knew that we wanted to work with Python machine learning packages, we created a Python connector to execute the Matlab data generator.

1) Parameters

Our dataset requires several parameters to be provided. These parameters are defined in Tables I and II.

Table I shows parameters that we must specify in order to create a dataset. For example, Max_Mode specifies the maximum number of modes to be generated in a spectral window.

Table II shows the parameters that each spectra will contain. For example, N is the number of modes in a spectra. N will be no greater than Max_Mode.

TABLE I: Required Parameters for Simulation

Parameter	Description
Max_Mode	Maximum number of modes
Max_Shell_Mode	Maximum number of shell modes
Scale	Width of the window
Omega_Shift	Resolution in angular frequency domain
Delta_Gamma	Variation of gamma
Delta_Gamma_Shell	Variation of gamma for shell modes
Gamma_Amplitude_Factor	Constant factor for peak heights
Shell_Amplitude_Factor	Constant factor for shell noise heights
Epsilon_Noise	White noise factor
Num_Channels	Number of channels per spectra
Num_Samples	Number of spectra to generate

TABLE II: Dependent Parameters per Spectra

Parameter	Description
N	Number of modes
Omega	Frequency of each mode
Gamma	Damping of each resonance

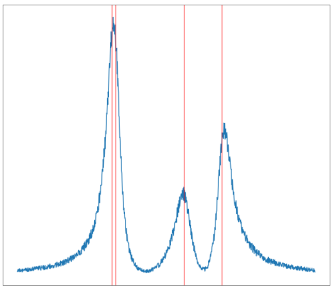

Fig. 3: Example of a Single Channel Spectrum
4 peaks, Delta_Gamma = 0.5, Delta_Gamma_Shell = 1.8, Omega_Shift = 5

2) Shell Modes vs Liquid Modes

In solving this problem, our goal is to count the number of modes in a spectrum. However, there are two types of modes that exist in a spectra. The first kind of mode is caused by acoustic resonances in the liquid within the container. The second type of mode that we see is caused by the acoustic resonances in the container itself. To accurately estimate the volume of liquid, we need to be able to accurately count the liquid modes. Since these modes occur at the same time, we will sometimes see interference in the liquid modes.

Fortunately, some of the characteristics of shell modes make them easily distinguishable. Specifically, shell modes tend to be much wider than liquid modes. Using this information, we can build our model in such a way that it effectively filters out the noise induced by shell modes. In our system, we can tweak the parameters of the shell modes to allow us to build models to cover more difficult cases.

B. Response Variable

The response variable for these spectra is the number of resonance peaks in a spectral window. The number of resonance peaks and their locations will be the same for each channel in an observation. Our dataset also tracks peak

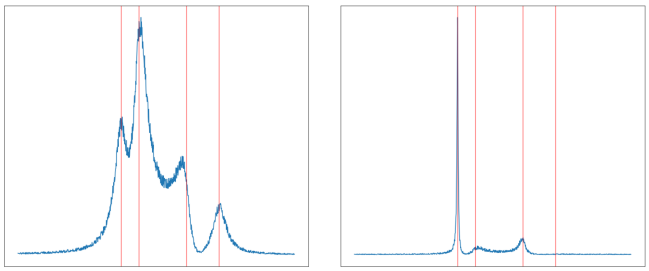

Fig. 4: Left: Gamma = 0.05, Right: Gamma = 5
The plot above demonstrates the same spectrum, with 4 peaks and Delta_Gamma = 1.8, but with different values for Gamma

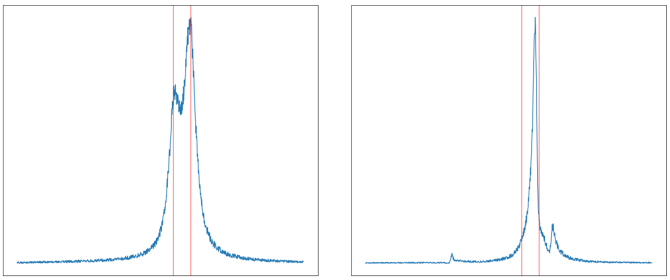

Fig. 5: Left: Delta_Gamma = 0.05, Right: Delta_Gamma = 5
The plot above demonstrates the same spectrum, with 4 peaks and Gamma = 1.8, but with different values for Delta_Gamma

locations, which aren't used in our models, but are useful for analyzing results.

C. Training and Test Sets

When developing a neural network, it is important to divide a set of data into different subsets. Those familiar with machine learning may already be familiar with the terms: training set and test set. The training set is the bulk of the data, in our case we used an 85% split. The other 15% of the data is saved as a test set. A test set is a subset of the data that the model is not trained with and never sees. This is extremely useful and necessary when evaluating a model's accuracy.

In machine learning, it is possible that a model can become "overfit" to a set of data. This essentially means that the model has "memorized" your specific dataset and will most likely perform poorly on new data. The test set, which the model has never seen, allows us to determine if a model is performing well, or just overfitting.

IV. Model Architecture

A. Classification vs Regression

The problem of peak counting can be considered to be either a regression problem or a classification problem. In this paper, we focus on solving the local problem of peak counting in which we are looking at a relatively small spectral window. In this local case, the number of peaks is relatively limited and a classification based approach is appropriate. However, the volume of liquid is not based on just this portion of the spectral window, but rather the entire range of frequencies.

When looking at the global problem, a classification based approach is less appropriate. In this case, a regression model is more appropriate as the number of peaks in the spectrum covers a much larger range.

Another consideration that needs to be made when building a regression model is the fact that the number of peaks is a discrete value and regression models predict continuous values. Due to this, when building a regression model, prediction rounding thresholds would need to be carefully chosen.

Since we focus on solving the local problem in this paper, we use a classification based approach.

B. Loss Function

In machine learning, a loss function is the method of evaluating how well specific algorithm models the given data. There are numerous different loss functions to choose from.

The problem of selecting an appropriate loss function is closely tied to the decision of treating the problem as regression or classification.

We considered a few different loss functions when building our models. These included categorical crossentropy, macro averaged mean absolute error, and poisson loss.

1) Categorical Crossentropy

Categorical crossentropy is a loss function used for single label categorization – when each object can only be of one class. This is acceptable in our case if we consider each label to be a number of peaks (ex: 1 peak, 2 peaks, 3 peaks, etc.). This loss function does not value certain errors over others. For example, a prediction of 1 peak on a spectra of 5 peaks is no worse than a prediction of 4 peaks.

2) Macro Averaged Mean Absolute Error

Statistically, mean absolute error (MAE) refers to the difference between two continuous variables. In our case, our variables are not really continuous, but they are ordinal values. Macro averaged MAE (MA-MAE) will return a loss that reflects the degree of error. For example, a prediction of 1 peak on a spectra of 5 peaks would accrue more loss than the 4 peak prediction.

3) Poisson Loss

The Poisson loss function is used for regression when modeling count data. We thought this might apply to our situation since we are counting peaks. Currently, our model is set up as a classification, but it may be valuable to experiment with a regression model using this loss metric.

C. Types of Model Architecture

In this section, we explain the different model architectures that we used in order to predict the number of peaks per spectrum. We began the process of determining a suitable model architecture by using a convolutional neural network. After obtaining a moderate success with convolutional neural networks, we explored the use of recurrent neural networks which further improved our validation results. Our final model architecture is a mix of convolutional neural networks and

recurrent neural networks which provided a substantial improvement of our validation results over our previous model architectures.

1) Convolutional

We began the process of choosing a model architecture by experimenting with convolutional neural networks, as was suggested by previous research [6]. Convolutional neural networks are a class of deep neural networks that have been used in image recognition, recommendations, and natural language processing. A convolutional neural network consists of convolutional and subsampling layers, followed by fully connected layers.

2) Recurrent

Recurrent neural networks (RNN) architectures are adept at recognizing what parts of the spectrum are important towards predicting the number of modes. In addition, LSTM and GRU architectures are able to carry information while still controlling the problem of vanishing and exploding gradients.

A standard LSTM traverses the data and keeps track of past data to get an understanding of context. An extension on the LSTM is the bidirectional LSTM which traverses the data both forward and backwards. Bidirectional LSTM are able to better understand context since they keep track of both past and future data. However, using bidirectional LSTMs increases the time it takes to train a model.

Due to the sequential nature of LSTMs, contextual information becomes weaker as the amount of data increases. This can be thought of as the model forgetting old information as new information becomes available. To combat this, researchers have developed a mechanism called attention which is able to assign weights to the context vectors generated in LSTMs. By weighing the values in the context vectors, the model learns to pay more attention to certain features and helps stymie the loss of context.

3) Combination

Our final proposed neural architecture is able to combine the benefits of convolutions and RNNs. We use convolutions to extract short-term dependencies and RNNs with attention to extract long-term dependencies [4]. See Figure 6 for an architecture diagram.

4) Ensemble Model

Ensemble models combine several base models in order to produce one optimal predictive model. This method allows the creation of better predictive performance compared to a single model. There are different methods to create a ensemble model, such as bagging and random forest. As more models are produced, it may be beneficial to experiment with using them in an ensemble model.

Hierarchical models are a type of ensemble model that consist of neural networks that feed into other neural networks. We implemented a hierarchical model made up of several binary classifiers. The first binary classifier labels a spectra as having 1-2 peaks or 3-4 peaks. Depending on how the classifier identifies the spectra, it is piped to the appropriate binary classifier to classify the final number of peaks. We found this ensemble of three networks to perform much better than a single categorical classifier and hypothesise it may be easier to tune each network to a narrower range of peaks.

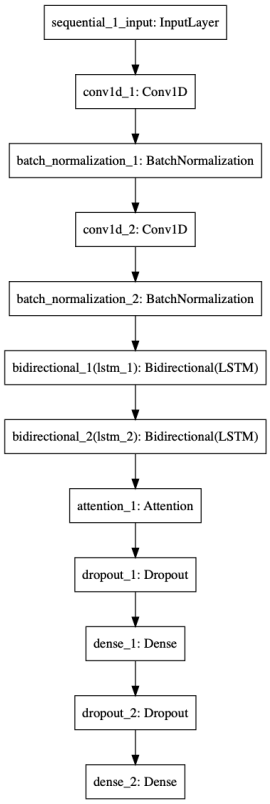

Fig. 6: Combination Model Architecture

V. Results

To evaluate our models, we ran a number of experiments to determine which model performed the best. In addition to this, we ran experiments to show how the best model performs under different circumstances. The first experiments we ran consisted of comparing the performance of models with different architectures. After deciding which architecture performed the best, we ran more specific experiments to see how the model performance was affected by changes in various parameters. Specifically, we looked at how changing parameters in dataset generation and how changing the number of channels in a spectra affected the model's performance. In addition to this, we performed an experiment to show how the model performs when missing data (i.e. a sensor breaks).

A. Evaluating Model Architectures

Over the course of this project, we developed several different architectures and methods of counting peaks in the spectral data. We will compare the following models:

1) LSTM

The LSTM was one of the first models we developed and tested. Figure 7 shows the architecture of our LSTM model.

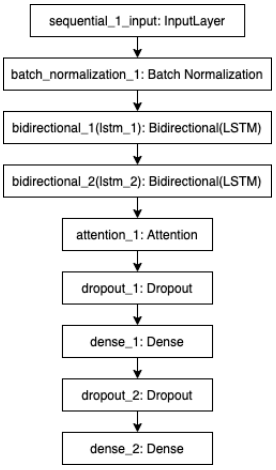

Fig. 7: LSTM Model Architecture

TABLE III: Generated Dataset Parameters for Model Comparisons

Param	Value
Max_Mode	4
Max_Shell_Mode	5
Scale	1.0
Omega_Shift	10.0
Delta_Gamma	0.5
Delta_Gamma_Shell	0.5
Gamma_Amplitude_Factor	4.0
Shell_Amplitude_Factor	5.0
Epsilon_Noise	0.05
Num_Channels	50
Num_Samples	200,000

After training the LSTM network on the data defined in Table III, we evaluated on an unseen test set. Table IV shows the results.

TABLE IV: LSTM Result Metrics

Metric	Result
Test Accuracy	81.103%
Test Loss	0.401
Test MAE	0.118
Test MSE	0.064

Our LSTM model was able to classify a spectra as having 1-4 peaks with 81.103% accuracy.

Figure 8 shows the confusion matrix comparing predicted labels vs actual. The true label is shown across the rows, and the predicted label is shown across columns. For example, the model predicted 1 peak when there was actually 1 peak 100% of the time. The model incorrectly predicted 2 peaks when there were 3 peaks 38 times. The values along the diagonal represent correct classifications.

2) 1D CNN + LSTM

This model utilizes two 1-dimensional convolutional layers before the LSTM layers. An architecture diagram can be found in Figure 6.

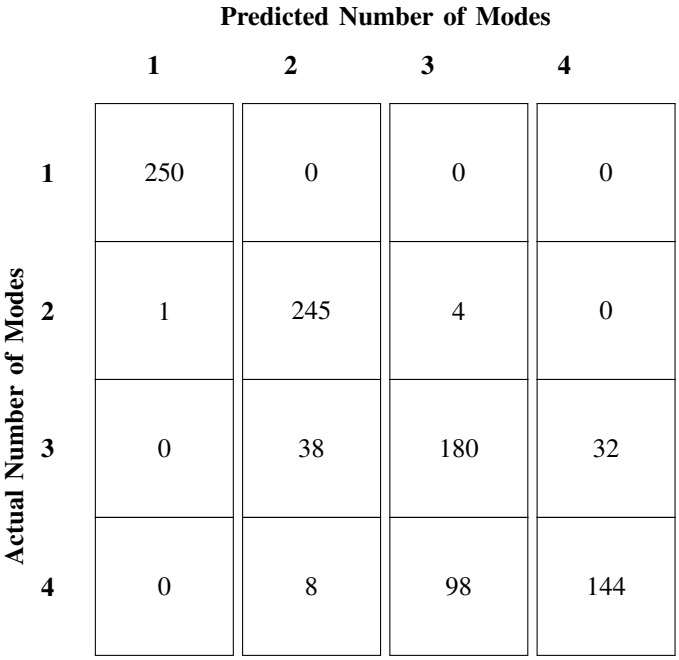

Fig. 8: LSTM Evaluation Confusion Matrix

After training the 1D CNN + LSTM network with the data defined in Table III, we evaluated on an unseen test set. Table V shows the results.

TABLE V: 1D CNN + LSTM Result Metrics

Metric	Result
Test Accuracy	82.315%
Test Loss	0.659
Test MAE	0.092
Test MSE	0.073

Our 1D CNN + LSTM model was able to classify a spectra as having 1-4 peaks with 82.315% accuracy. Figure 9 shows the confusion matrix comparing predicted labels vs actual.

Interestingly, our model performs very well on low numbers of peaks, and more poorly on high numbers of peaks. This observation inspired us to create a hierarchical model composed of several networks. We hypothesized that we could train each network to perform better on a specific number of peaks, rather than trying to capture all in one model.

3) 1D CNN + LSTM Binary Ensemble

This hierarchical model is made up of 3 networks – each producing a binary classification. In order to classify a spectra, it is fed into an initial network, and then piped to the appropriate sub-network based on the initial classification.

Figure 10 depicts a flow chart outlining the relationships between the networks.

To evaluate this ensemble of networks we will examine the metrics of each network separately. These metrics can be found in Table VI.

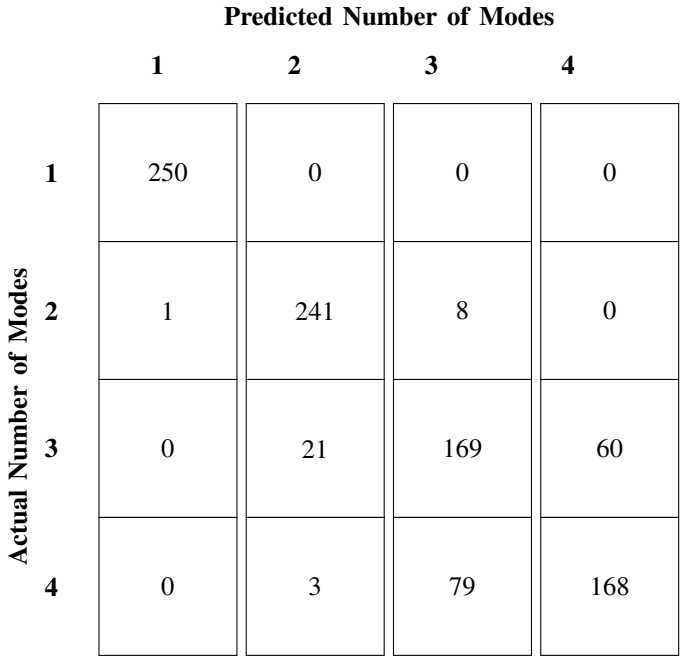

Fig. 9: 1D CNN + LSTM Evaluation Confusion Matrix

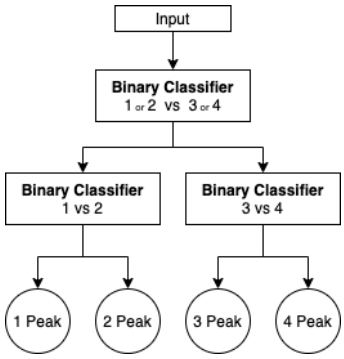

Fig. 10: 1D CNN + LSTM Binary Ensemble Data Flow

TABLE VI: 1D CNN + LSTM Binary Ensemble Result Metrics

Network	Metric	Result
1-2 vs 3-4	Test Accuracy	96.73%
1-2 vs 3-4	Test Loss	0.092
1-2 vs 3-4	Test MAE	0.040
1-2 vs 3-4	Test MSE	0.026
1 vs 2	Test Accuracy	99.73%
1 vs 2	Test Loss	0.011
1 vs 2	Test MAE	0.004
1 vs 2	Test MSE	0.002
3 vs 4	Test Accuracy	73.36%
3 vs 4	Test Loss	1.028
3 vs 4	Test MAE	0.303
3 vs 4	Test MSE	0.182

It is worth noting that the metrics in Table VI are each obtained individually so that each sub-network can be evaluated independently. Accurate subsets of data were fed to the sub-models, which wouldn't always be the case in a real implementation. In a real application, the accuracies of the sub-models will be slightly lower depending on the accuracy of the root model.

The results in Table VI follow the trend observed in previous experiments where we saw lower numbers of peaks being more accurately predicted compared to those of several peaks.

B. Model Evaluation with Different Shell Mode Parameters

In this section, we aim to measure how robust our model's predictions are with data that it may not have seen before. In order to do so, we will use a model trained with a value of $Delta\_Gamma\_Shell = 0.5$ and values of $Delta\_Gamma\_Shell$ between 1 and 5. We will use this same model to predict the number of liquid modes with data generated with a wider range of these parameters.

TABLE VII: Number of Shell Modes Experimentation

Num_Shell_Modes	Precision	Recall	F1 Score
1	0.79	0.79	0.78
2	0.74	0.74	0.73
3	0.78	0.76	0.76
4	0.74	0.71	0.70
5	0.75	0.74	0.74
6	0.85	0.80	0.81
7	0.82	0.80	0.81
8	0.80	0.80	0.80
9	0.84	0.84	0.84
10	0.76	0.75	0.73

In table VII, we measure the performance of our model with data containing between 1 and 10 shell modes. It is worth noting that we kept the same value of $Delta\_Gamma\_Shell = 0.5$ and that each row of this table was evaluated with about 100 instances. We notice that, even though our model was trained with a maximum of 5 shell modes, the performance of our model does not appear to decrease when we increase the number of shell modes.

TABLE VIII: Shell Delta Gamma Shell Parameter Experimentation

Delta_Gamma_Shell	Avg.Precision	Avg.Recall	Avg.F1-Score
0.025	0.82	0.82	0.82
0.050	0.81	0.81	0.81
0.075	0.82	0.82	0.82
1.00	0.81	0.80	0.81
1.25	0.84	0.84	0.84
1.50	0.81	0.81	0.81
1.75	0.82	0.82	0.82
2.00	0.69	0.69	0.68
2.25	0.66	0.65	0.64

In table VIII, we measure the performance of our model with data generated with different values of $Delta\_Gamma\_Shell$. Each row of this table was evaluated with about 475 instances. In this table, we notice that the performance of our model remains relatively consistent with an average accuracy of about 0.81 until we reach a value of

$Delta\_Gamma\_Shell = 2.00$ in which case the accuracy drops to 0.69.

C. Effect of Variable Channels

Each spectra sample contains a number of channels that represent readings from sets of sensors at a single point in time. One of the first questions we had after developing a rudimentary model was: what effect does number of channels per spectra have on the performance of it?

To compare the effect the number of channels per spectra in the dataset has on model performance, we generated the following datasets shown in Table IX:

TABLE IX: Generated Dataset Parameters for Channels Experiment

Param	Value
Max_Mode	4
Max_Shell_Mode	5
Scale	1.0
Omega_Shift	5.0
Delta_Gamma	0.5
Delta_Gamma_Shell	1.8
Gamma_Amplitude_Factor	4.0
Shell_Amplitude_Factor	5.0
Epsilon_Noise	0.05
Num_Channels	1, 3, 7, 10, 15, 20, 35, 50
Num_Samples	20,000

For each of the eight datasets, the samples are split into a 15% test set and an 85% training set. We trained one of our models with each dataset and observed the validation accuracy.

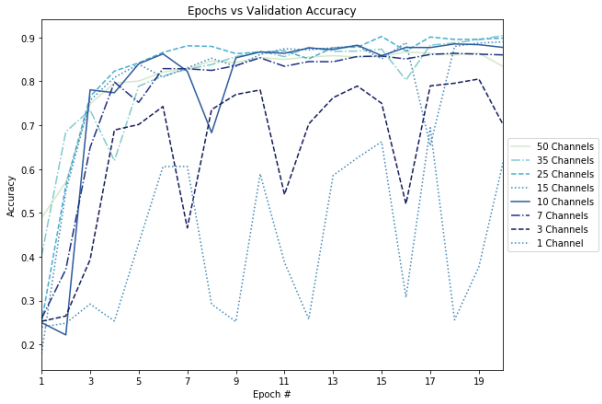

Fig. 11: Epochs vs Validation Accuracy for Different Numbers of Channels

In Figure 11 we can see that it's pretty clear the number of channels makes a large difference when the number of channels is at or below 10. After 10 channels, these preliminary results suggest that there are diminishing returns, especially considering the increase in data size and training time with each channel. In the future, we will continue to explore this effect through more robust experimentation.

D. Evaluation of Channel Padding

Even though our best performing model was trained using 50 channels, we may want to use this model to obtain predictions for data with a smaller number of channels. Originally, we would require each sample to have the same number of channels that was used to train the model. Therefore, if we would want to use a model to obtain predictions with data containing a fewer number of channels, we would have to train an entirely new model. In order to address this problem, we recommend padding the original number of channels in order for it to fit the number of channels used to train the model. Padding is the process of replicating the available data such that we are able to increase the number of channels per sample. We tried various padding strategies including: "reflect" in which the reflection of a channel vector is mirrored on the first and last values of a vector along each axis, "symmetric" in which the reflection of the vector is mirrored, and "constant" in which we filled missing channels with a constant value of "0".

In order to measure the performance of this strategy, we used a model originally trained with 50 channels. We then used data with different numbers of original channels to evaluate the model's performance (starting with 1 original channel and 49 padded channels and ending with 50 original channels and 0 padded channels). If we reference figure: 13, we can notice

TABLE X: Validation Metrics for Different Numbers of Channels

NC	Max Val Acc	Min Val Loss	Min Val MAE	Min Val MSE
1	0.6953	0.6720	0.1901	0.0998
3	0.8167	0.4640	0.1254	0.0677
7	0.8730	0.3232	0.0859	0.0433
10	0.8927	0.2982	0.0734	0.0420
25	0.9127	0.2301	0.0602	0.0327
35	0.9089	0.2443	0.0645	0.0349
50	0.8927	0.2654	0.0703	0.0420

Predicted Number of Modes

Actual Number of Modes	1	2	3	4
1	7452	16	0	0
2	56	7326	140	3
3	1	329	6893	296
4	0	16	1110	6362

Fig. 12: Confusion Matrix of Predicted Test Data for 10 Channels

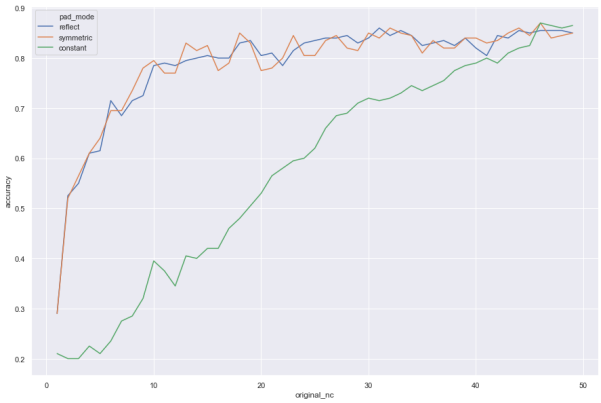

Fig. 13: Number of Original Channels vs. Validation Accuracy

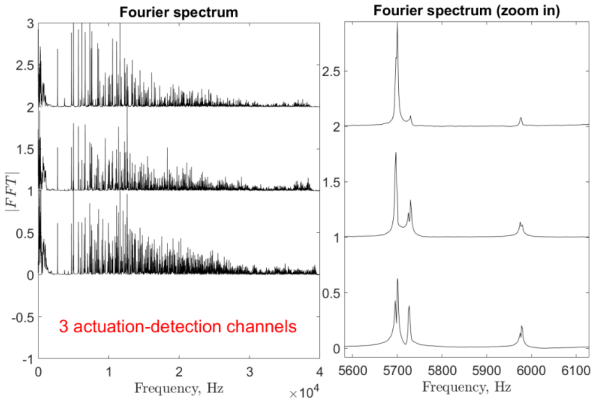

Fig. 14: Epochs vs Validation Accuracy for Different Numbers of Channels

that the model has a validation accuracy of about 80 percent with "reflect" and "symmetric" padded channels, even when we only have about 20 original channels.

VI. Future Work

A. More Experimentation

1) Peak Counting vs. Object Detection

In this paper, we focus on the peak counting approach rather than the peak detection approach. We made this decision after getting good initial results from basic peak counting models. Further research can be done to explore the peak detection approach in more detail.

2) Effect of Different Dataset Parameters

We believe that it would be beneficial to explore how different dataset parameters impact the overall accuracy of our models. We want to make sure that our models are able to perform well under a wide variety of circumstances and are not over-fitted to perform well with certain dataset parameter values but poorly on other dataset parameter values.

3) Comparing Padded Channels

The strategy of padding channels so that they are able to fit a model trained with a larger number of channels appears to have promising results. However, more research is required in order to determine whether it is preferable to use a model trained with a larger number of channels and pad the remaining channels over simply training a new model with the number of channels that we have available.

B. Global Problem

In this paper, we propose solutions to the local peak counting problem. However, we cannot accurately estimate the volume of liquid in a container by solving the local problem as the spectral windows we looked at were only a small portion of the entire spectrum. The difference between the global resonances and the local resonances can be seen in Figure 14.

Throughout this project, we did not want to lose sight of the global problem. Based on the work we've done in solving the local problem, we currently have two approaches in mind for solving the global problem.

1) Spectral Window Slicing

In the first approach, we would approach the global problem as a combination of many local problems. This would consist of slicing the global spectral window into smaller windows that our local solution would work for. This is the most simple approach that we are considering.

One of the drawbacks to this approach is the necessity to understand some of the physical properties of the spectral windows. Specifically, we would need to know if there is a theoretical maximum number of peaks that can exist in a spectral window of a given size. Due to the fact that our models perform classification, they are limited in the number of peaks that they can predict. However, if there is a physical property that limits the number of peaks in a given frequency range, then it would be possible to split the global window into smaller local problems.

2) Regression Based Approach

The second approach would involve changing how our models behave. In this approach, we would need to convert our models to regression rather than classification. By converting our models to regression models, we would be able to predict any number of peaks.

This solution does not seem ideal as it would require us to almost completely re-architect our models. In addition to this, in our initial testing of models, we were unable to get reasonable results with a regression model. However, we believe that the poor performance can be partially attributed to the fact that we were predicting discrete values with a continuous model. To remedy that, we would need to devise a rounding strategy that would allow us to round predictions in a way that yields good results.

VII. Conclusion

In order to be able to measure the volume of liquid in a zero-gravity environment, we must be able to count peaks in an

acoustic wave – ignoring shell interference and noise. There are several different approaches to counts peaks in acoustic signals, each with their own strengths and drawbacks. In this project, we have explored deep learning architectures such as CNNs, LSTMs, and combinations of the two. We explored training our model on data generated with different parameters, such as a different number of channels. We were also able to show that our model can perform well in situations where there is missing data. Through our experiments and findings, We have demonstrated that this problem can be automated through the use of neural networks with high accuracy.

REFERENCES

[1] H. Weyl, "Das asymptotische verteilungsgesetz der eigenwerte linearer partieller differentialgleichungen (mit einer anwendung auf die theorie der hohlraumstrahlung)," *Mathematische Annalen*, vol. 71, pp. 441–479.

[2] J. Feller, A. Kashani, M. A. Khasin, C. B. Muratov, V. V. Osipov, and S. Sharma, "Spectral mass gauging of unsettled liquid with acoustic waves," 2017.

[3] A. Melnikov, Y. P. Tsentalovich, and V. V. Yanshole, "Deep learning for the precise peak detection in high-resolution lc-ms data," *Analytical chemistry*, 2019.

[4] D. C. de Andrade, S. Leo, M. L. D. S. Viana, and C. Bernkopf, "A neural attention model for speech command recognition," 2018.

[5] M. Fang, Z. Zhou, S. Chen, and J. L. McClelland, "Can a recurrent neural network learn to count things?" in *CogSci*, 2018.

[6] S. Kiranyaz, T. Ince, O. Abdeljaber, O. Avci, and M. Gabbouj, "1-d convolutional neural networks for signal processing applications," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8360–8364, 2019.

[7] S. Chakraborty, S. Aich, M. il Joo, M. Sain, and H. Kim, "A multichannel convolutional neural network architecture for the detection of the state of mind using physiological signals from wearable devices," *Journal of healthcare engineering*, vol. 2019, p. 5397814, 2019.

[8] M. Khasin, "Spectra generator – matlab source code," 2019.