Classifying Beaked Whale Echolocation Click Images to Species

Taiki Sakai
taiki.sakai@noaa.gov

I. Introduction

Beaked whales are a family of deep diving cetaceans with over 20 distinct species that range across the entire globe. They are difficult to detect using traditional visual survey methods due to their extended time at depth (~60+ minutes), short time at the surface (<5 minutes), and cryptic surface behavior. However, they produce echolocation clicks during much of their dive sequence, and these sounds have species-specific characteristics that can be used to determine the species of the echolocating animal. This makes beaked whales an ideal candidate for studying using passive acoustics. Currently, species determination from acoustic recordings requires experienced technicians to manually examine a set of echolocation clicks using a suite of visualizations. One of the most commonly used visualizations is the Wigner-Ville transform, which creates an image of a click with high resolution in both the frequency and time dimensions. Since these images are already being used by human analysts to classify beaked whale species, we believe it is highly likely that computer vision can also be successful at this task.

The goal for this project is to create a computer vision model to classify beaked whale detections to species using images of individual echolocation clicks. If successful, this has the potential to save hundreds of hours of manual labor, freeing up valuable time for acoustic analysts. Recording equipment is getting cheaper and easier to deploy every year, but without efficient analytical tools most of these recordings do not realize their full value. We simply do not have the resources to hire and train enough acoustic technicians to manually analyze hundreds of hours of recordings. By drastically reducing the time required to identify which beaked whale species are present, computer vision methods have the potential to add immense value to each recording. Passive acoustics is a cost effective way to survey an area, and can even be used to estimate the abundance of certain cetacean species, but this only works if we can correctly identify the species present in each recording.

II. Dataset

The dataset for this project consists of 100,000 clips of individual echolocation clicks of five different beaked whale species. The recordings are from drifting buoys deployed during the PASCAL survey from August to September 2016 off the west coast of the United States ranging from southern California to northern Oregon, totalling over 3000 hours of recordings. Drift deployment locations are shown in Figure 1. The recordings were first processed using PAMGuard acoustic software, which automatically detects echolocation clicks, then sets of clicks were manually organized into events by an acoustic analyst, where an event is a collection of clicks that belong to a single animal or group of animals during a single dive. These events were then classified to species by a team of three highly experienced acoustic analysts over the course of a summer. In total there are 859 events across 27 separate drifts, the distribution of species is shown in Table 1.

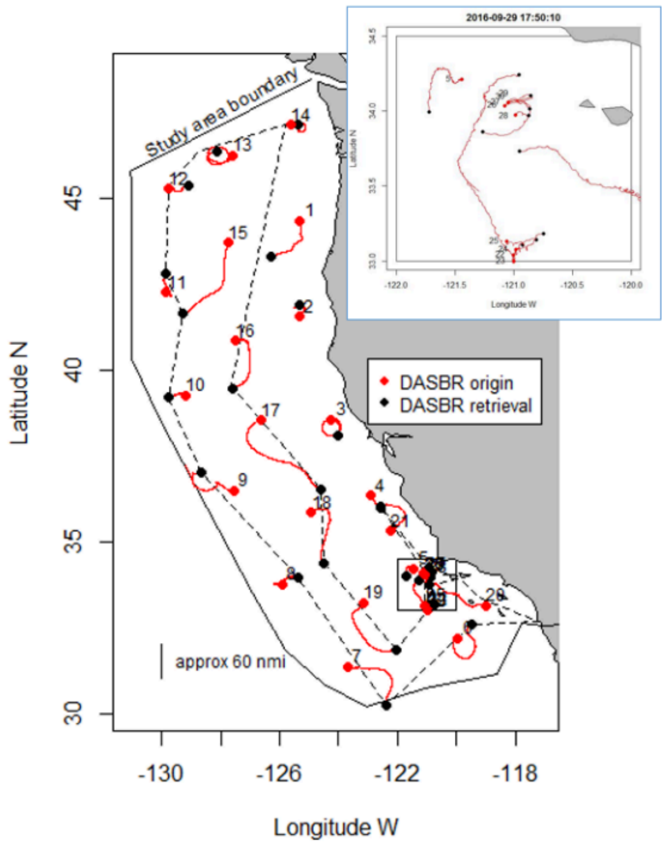

Figure 1: Drift deployment tracks for 2016 PASCAL survey

Individual detections were converted into images by applying the Wigner-Ville transform to shortened clips of the echolocation clicks. Images are created with the following process:

1. Clips are downsampled to a common sample rate of 192kHz
2. A shortened clip of 128 samples is taken centered on the maximum value of the waveform
3. A Hanning window is applied to the shortened clip
4. The Wigner-Ville transform is applied to the windowed clip
5. Values are rescaled to 0-255 range
6. The 128x128 array of transform values is saved as a Numpy array

Each recorder has two channels, so this process is repeated for each channel, resulting in over 200,000 images. Examples of a click image for each species are shown in Figure 2.

Species	Detections (%)	Events (%)
ZC/0 (Ziphius cavirostris)	70018 * 2 (70%)	685 (79%)

BB/1 (Berardius bairdii)	13317 * 2 (13%)	66 (7.7%)
MS/2 (Mesoplodon stejnegeri)	14317 * 2 (14%)	50 (5.8%)
BW43/3 (???)	1376 * 2 (1.4%)	37 (4.3%)
BW37V/4 (???)	1545 * 2 (1.5%)	21 (2.4%)
Total	201158	859

Table 1: Number of detections and events per species

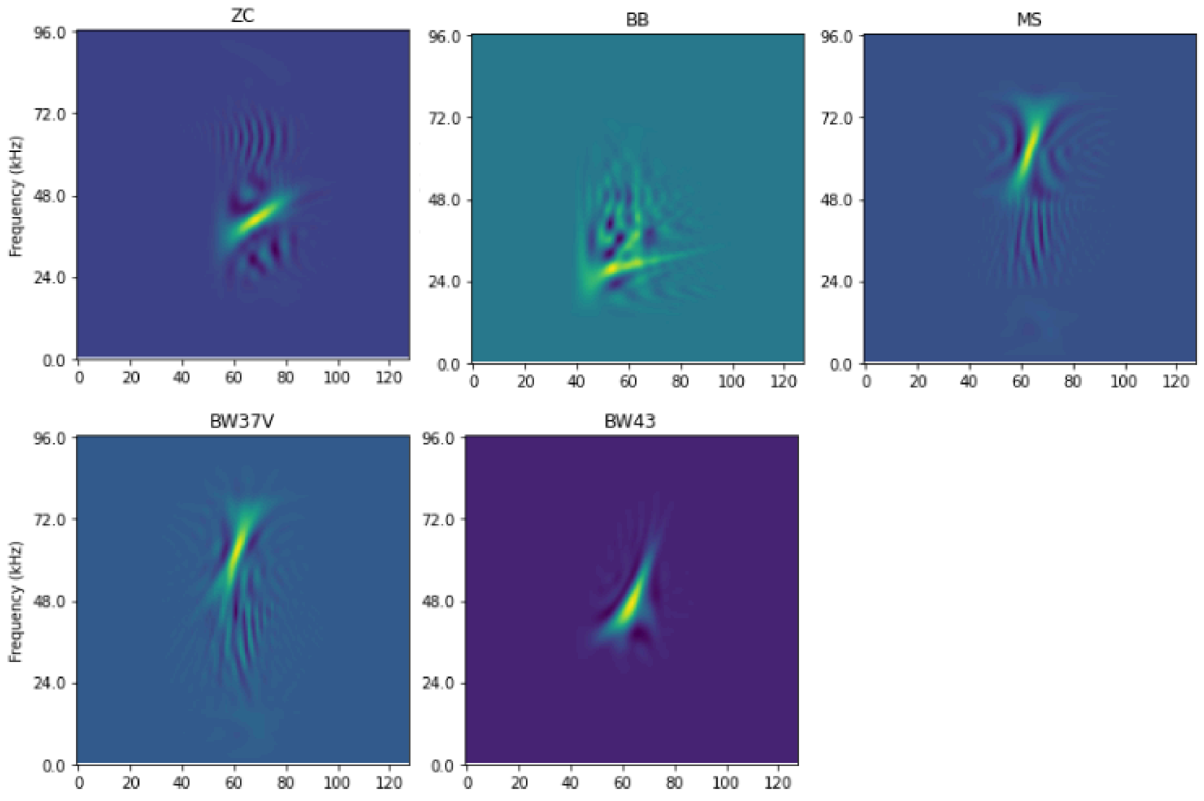

Figure 2: Examples of an echolocation click image for each species

This dataset is split into training, validation, and testing sets with approximately 70%, 15%, and 15% of the data in each split, respectively. The data splitting strategy was chosen to best mimic the anticipated use case for our model. Since we want the model to be able to predict the species of beaked whales in future drift deployments, we will split our data by placing entire drifts into each category. This means that all data in the testing and validation sets will be from times and locations that the training data has not seen, so accuracy on these sets should approximate accuracy in a real-world scenario. However, the distribution of species across our deployments means that this is not straightforward. Figure 3 shows the distribution of species across our drifts. No drift has all 5 species, only one has 4 species, species “BW43/3” and “BW37V/4” are only present on a handful of deployments, and drifts can contain vastly different amounts of data. This means that in any given random split of deployments may not have all 5 species in each of the three splits, and the distribution of events is unlikely to be near the desired 70/15/15. In order to find an appropriate data split, we started by randomly generating 10,000 candidate splits. From these, any that did not have all 5 species in each split are discarded. The remaining candidates are assigned a score by summing the squared difference between the proportions in each split and the desired 70/15/15 proportions. The random split with the lowest score was selected.

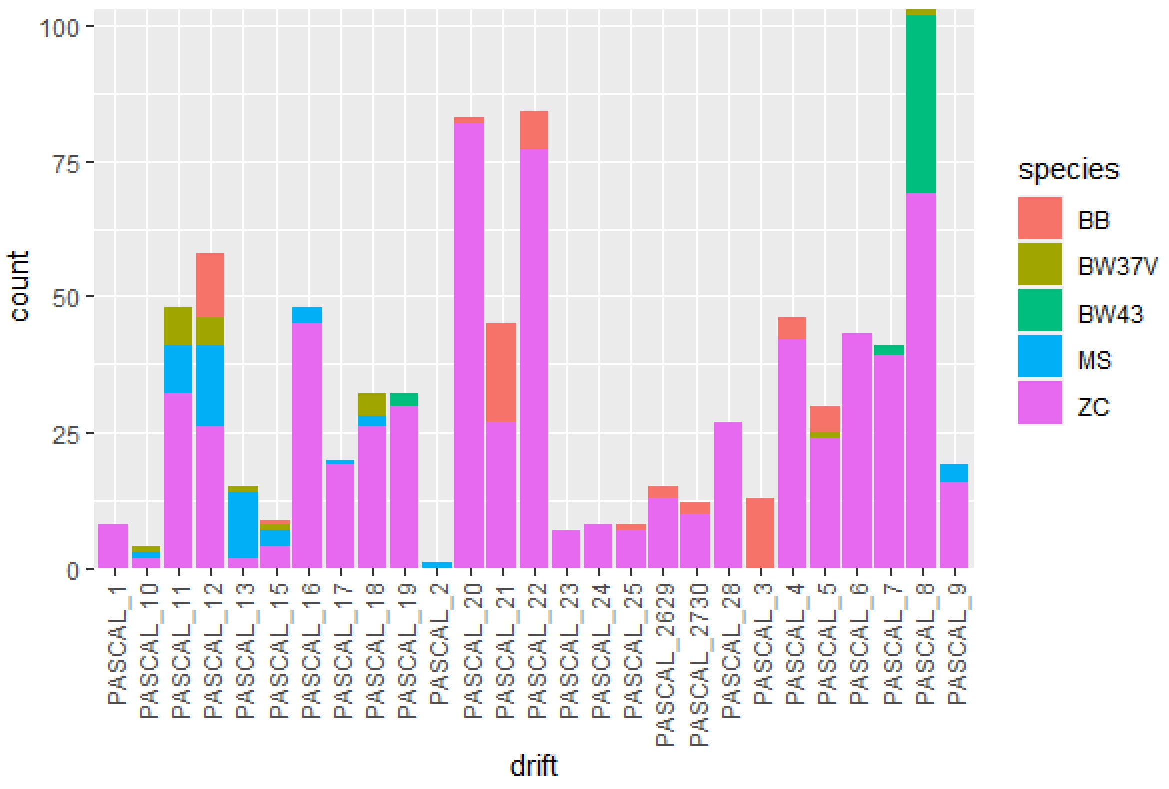

Figure 3: Number of events of each species in each drift

III. Model Development - During CV4Ecology

The first model I tried was a ResNet50 using pretrained weights and cross entropy loss. This was chosen because it is a fairly standard setup for image classification, even though there was some concern that transfer learning from a model trained on the ImageNet dataset might not be appropriate for my transformed audio images. ResNet models expect 3 channel RGB images as input, so the saved Numpy arrays were copied three times to create 128x128x3 arrays and then resized to the 224x224x3 before passing into the model. Initial results were surprisingly good, showing over 90% overall accuracy on the validation set after only a few epochs of training. Since over 70% of my data is from a single class, I was extremely skeptical of these initial results. Figure 4 shows the precision-recall curve for the validation set for each species. From this plot it is obvious that our model is not performing as well as the overall accuracy numbers would indicate. Although performance is better than expected for 4 of the species, performance on “BW43/3” is terrible. This class represents less than 2% of the dataset, so this does show up in the overall accuracy numbers.

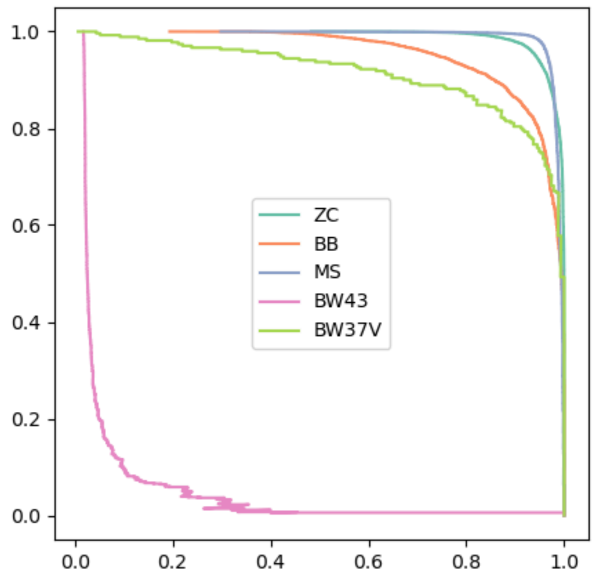

Figure 4: Precision-recall curves for each species in the validation set

Moving forward, two main changes were made. First, the base model architecture was changed to a ResNet18. Performance was nearly identical, and training time was drastically reduced, allowing for more experimentation during the course. Second, class balanced accuracy (the average of the accuracy of each class) was used as the evaluation metric instead of overall accuracy.

Next steps were guided by two main themes. First, to try and improve the extremely poor performance on class “BW43/3”. Second, to address concerns that the model may be learning to associate noise with the class “ZC/0”. Figure 5 shows a plot of 5 images for each class where the model was most confident (highest predicted probability). I had expected these to be loud, clear, high quality images, like the bottom four rows. The predictions for class “ZC/0” look more like what noise would look like (compare to the “ZC/0” image in Figure 2). The way that these detections were manually labeled means that there are likely to be a lot of quiter clicks or even noise included with our labeled data, so this plot suggests that the model was learning to associate this noise with class “ZC/0”.

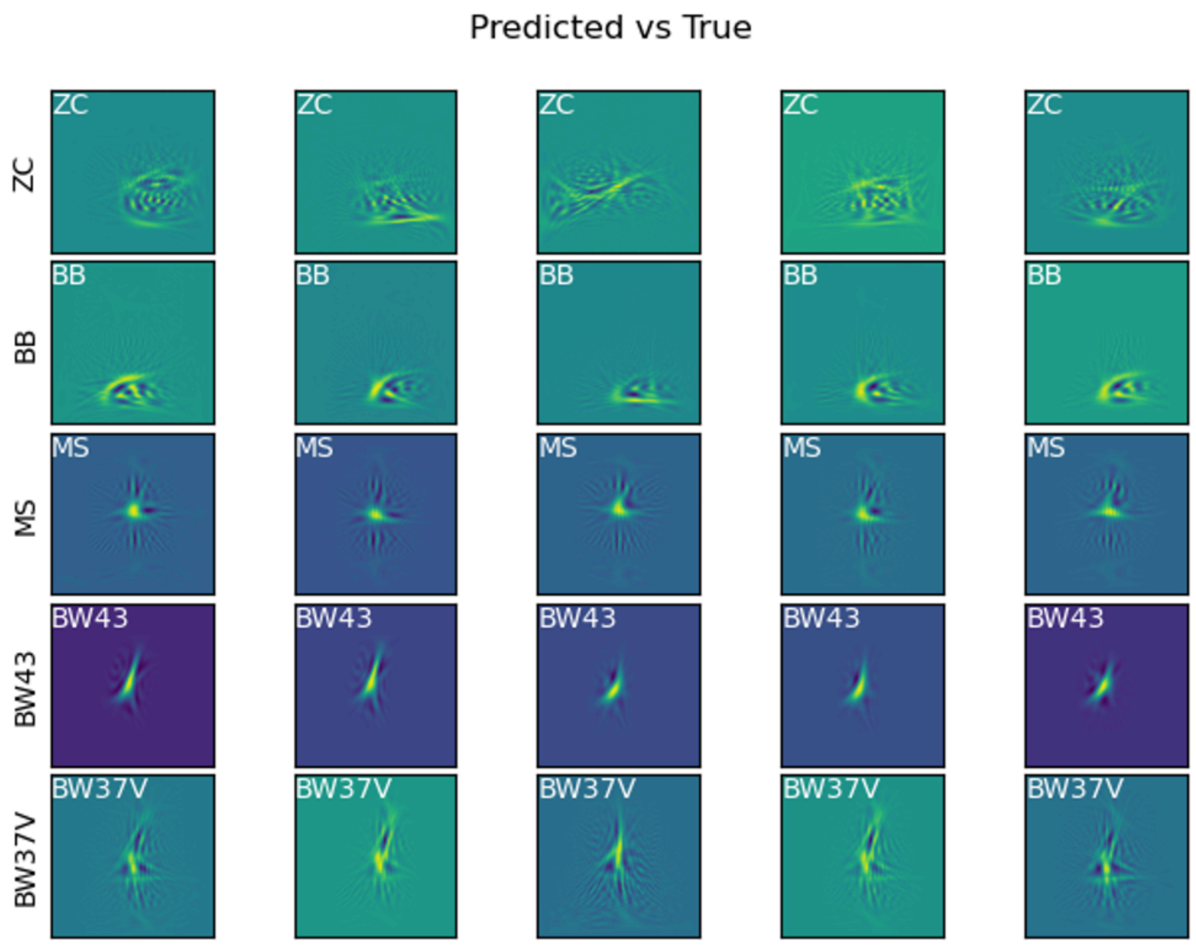

Figure 5: The five images for each class that the model most confidently predicted correctly

Two modifications were made to try and improve performance of under-represented classes. First, a weighted cross entropy loss was used, where each sample loss is weighted by the inverse-proportion of that class during training. This means that an incorrect prediction on an under-represented class is more punishing than for the most common class. Second, a weighted sampling strategy was adopted that creates batches of data with approximately equal class distribution. This means that classes that are more than 20% of the dataset will be undersampled, while classes that are less than 20% of the data will be oversampled. The weighted sampling strategy had no noticeable effect on performance, but the weighted cross entropy significantly increased performance and will be used as a standard option for all models going forward.

Two data augmentations were added to try and improve overall performance. Augmentations must be chosen carefully for acoustic data, since many augmentations common to image classification tasks are not appropriate. The y-axis in these images represents frequency, so any shifts in the vertical direction would change the signal. Rotations and flips are also unacceptable for similar reasons. The first augmentation was a random affine in the horizontal axis, which randomly shifts an image up to 30% to the left or right, filling in the shifted space with zeros. The second augmentation was random gaussian blur. The Wigner-Ville transform is known to create artifacts in the images, so the hope was that adding gaussian blur would stop the model from learning to recognize these artifacts. Neither of these augmentations significantly affected the performance.

Two modifications were made to try and account for noise in the training dataset. First, the loss function was modified by multiplying it by a measure of the signal-to-noise ratio (SNR) of each image. Clicks with lower SNR are more likely to be noise or low quality images, and so were given a lower loss value. Clicks with higher SNR should be more useful for classification purposes, and were given a relatively higher loss value. The hope was that the model would then not try to learn the features of noisy images since they would have lower loss. This did not improve performance. Second, a measure of call "loudness" was added to the model's feature vector before the final classification layer. This did not improve performance, but was mostly done as a learning exercise.

The sum result of these efforts during the course is that model performance did not improve much from the original model. While this was a bit disappointing, learning *how* to implement all these ideas was incredibly valuable. Additionally, examining the model results in a slightly different light reveals a very positive take away. As mentioned in the dataset description, the images we work with belong to discrete events, and we are confident that each event is a single species. Thus we are not necessarily concerned with the species identification of a single click, but rather the event as a whole. By averaging the predicted probabilities for all clicks in an event, we can use this model to generate species predictions at the event level. Figure 6 shows the confusion matrix of these predictions on the validation set. The model is able to predict all events correctly, other than the two "BW43/3" events that it has struggled with from the beginning.

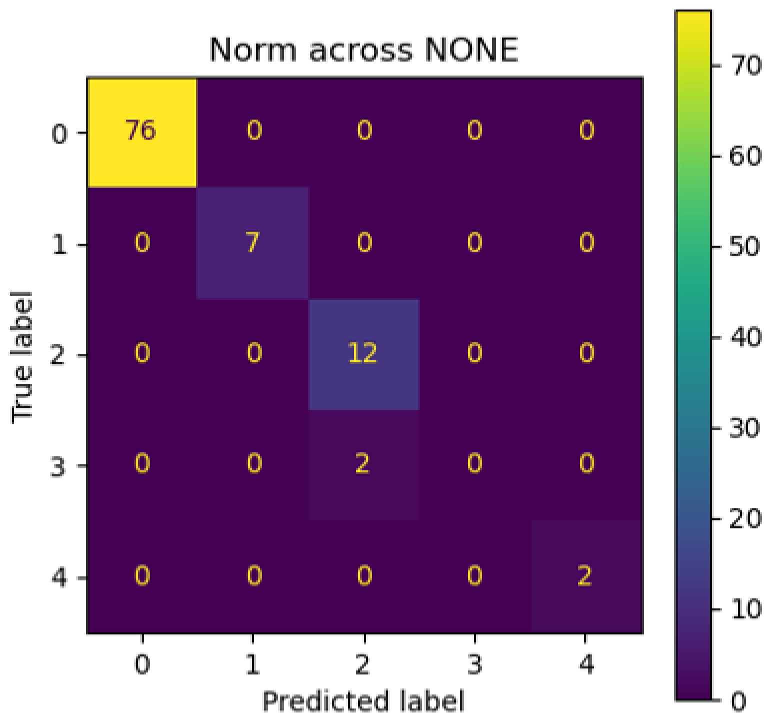

Figure 6: Confusion matrix for event species predictions on validation set

IV. Model Development - After CV4Ecology

The first thing I tried after the workshop was re-processing my data with a smoothed Pseudo-Wigner-Ville transform. This is not a transform that acoustic analysts typically use to classify beaked whale clicks, but it applies some smoothing in the frequency axis. Figure 7 shows an example of the same image processed with the regular and smoothed transforms. I hypothesized that this might help by removing some of the artifacts that are present in quieter click images. Performance with the smoothed images was worse in all metrics.

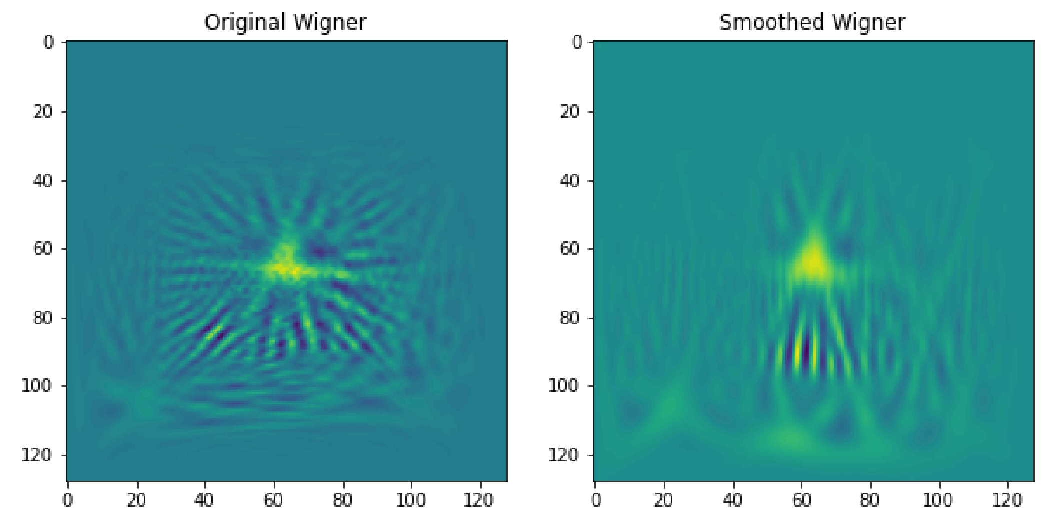

Figure 7: Comparison of the same image with the regular and smoothed Wigner-Ville

Next, I wanted to add another year of data from the 2018 CCES survey to our model to boost the number of events for the underrepresented classes. While processing the CCES survey data and trying to combine it with the existing PASCAL data, I discovered a number of issues. First, some of the PASCAL data had been processed with a 10kHz highpass filter, while others had been processed with a 4kHz filter. All of the CCES data had a 10kHz filter, so all PASCAL data with a 4kHz filter had to be reprocessed with a 10kHz filter. Second, I discovered that a small subset of both PASCAL and CCES deployments were made with a different model of recording device, and this device had significant noise issues. These noisy recorders will not be used in future surveys, so all data from these drifts are discarded (note - the two “BW43/3” events that the original model struggled to classify correctly were *not* part of the discarded noisy data). Third, some of the CCES data had already been decimated to a lower sample rate, but this was not accounted for in the original processing workflow, so they had to be reprocessed with the adjusted sample rate. I had originally planned on training a model solely on one survey then predicting on the other survey as a test of our future use case, but after discarding drifts with noisy recorders there were no longer a sufficient number of drifts in each survey with each species to do this. The two survey datasets were combined, and new train/val/test splits were created following the same process as the original split. This combined dataset is used for all models going forward. Figure 8 shows the confusion matrix for the validation set of the combined data. Performance is similar to the original PASCAL-only models in that it generally performs well except for one class, but the class with poor performance has changed from “BW43/3” to “BB/1”.

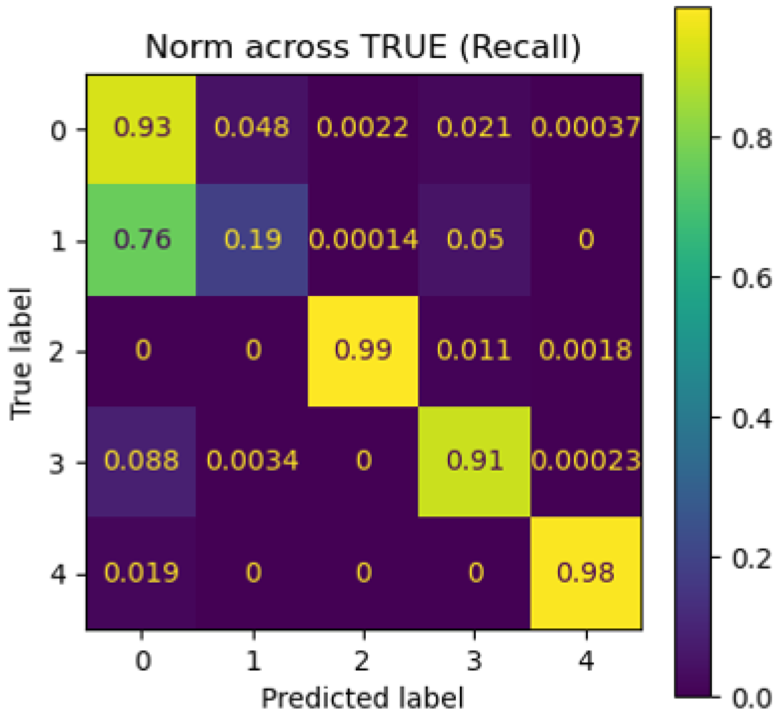

Figure 8: Confusion matrix for validation set of combined CCES and PASCAL data

Another issue I wanted to investigate was the relationship between signal-to-noise ratio (SNR) and the error rates of the model. Figure 9 shows a histogram of the distribution of SNR in the validation set, colored by prediction error. There is a clear relationship that calls with lower SNR have higher error rates. Calls with SNR 5 or less have 40% or higher, while calls with SNR higher than 5 have an error rate of 31% or less. I decided to filter out all calls with an SNR lower than 5 and train a new model. We can easily measure SNR in a consistent way for any new data, and this will hopefully prevent the model from learning characteristics of low quality quiet signals. This resulted in small performance improvements for all classes, and is used for all models going forward.

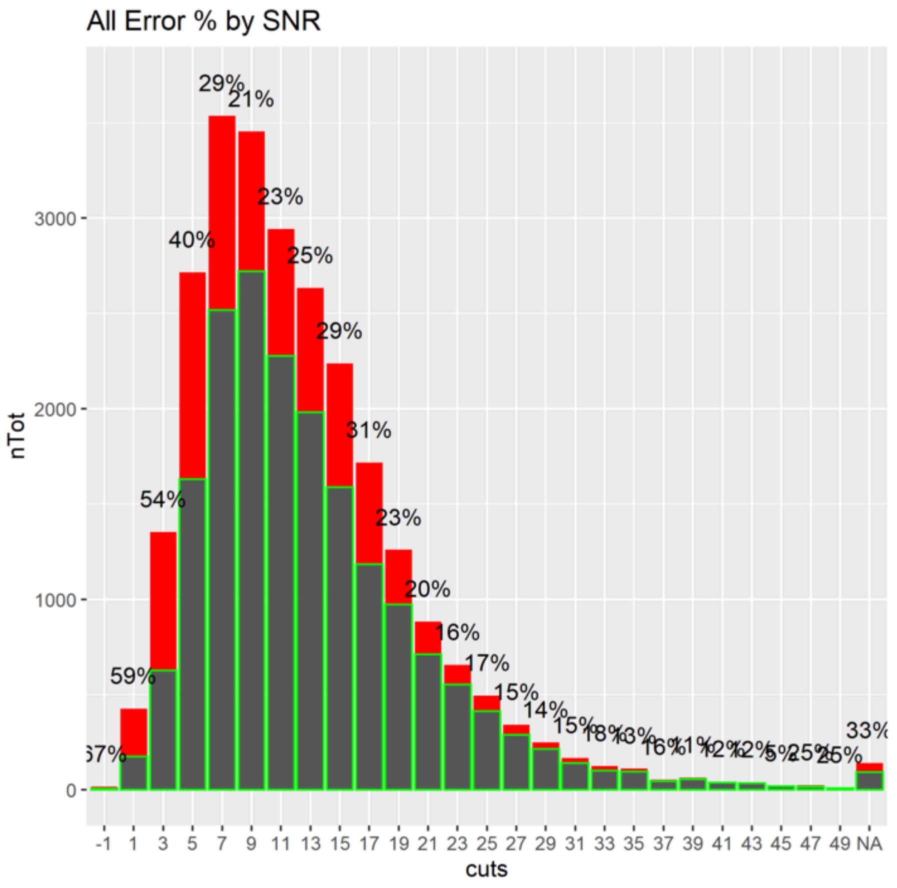

Figure 9: Distribution of predictions in the validation set by signal-to-noise ratio (SNR), green and gray bars are accurate predictions and red are inaccurate predictions. The percent error in each bin is listed above each bar.

Next I wanted to try and get a better understanding of how the model is learning to classify images by creating saliency maps for each image. This is a technique that uses the gradients of the classifier to show areas of an image that have the highest impact on the classification score. Ideally these would show that areas of the image that have characteristics specific to a particular species are important to the classification score. Another possibility is that they could highlight cases where the model is potentially learning to associate background noise with a certain, or some other undesired outcome. Three different varieties of saliency maps were implemented. First, a basic saliency map that just plots the maximum value of the gradient (based on this post). Second, a method called Deep Feature Factorization (DFF) that segments an image into regions, and shows the predicted probability of each class in those regions (based on this repo). Third, a method called GradCAM that highlights regions of an image that were important to the chosen classification based on the average gradient magnitudes (based on same repo as DFF). Figure 10 shows an example of these images for one click of each species that was correctly classified. The saliency maps seemed to indicate that the model was focused on the background and not the click itself, but this may be because this technique is overly simplistic. The DFF images show that both the click and the area immediately around the click are important for classification. It also shows that the background regions with no information are generally classified as “ZC/0”, our most popular class. The GradCAM show that the model is generally focused on the regions of the image with click information, but the “BB/1” image is an interesting exception to this. It is highly focused on the blank region above the click. This could be because “BB/1” are the lowest frequency call of the five species in the model, so the model has learned that if there is no information in high frequency areas then the species should be “BB/1”. The overall takeaways from these images is that the model is generally performing as expected, and that there still may be some issues with noisy or quiet data.

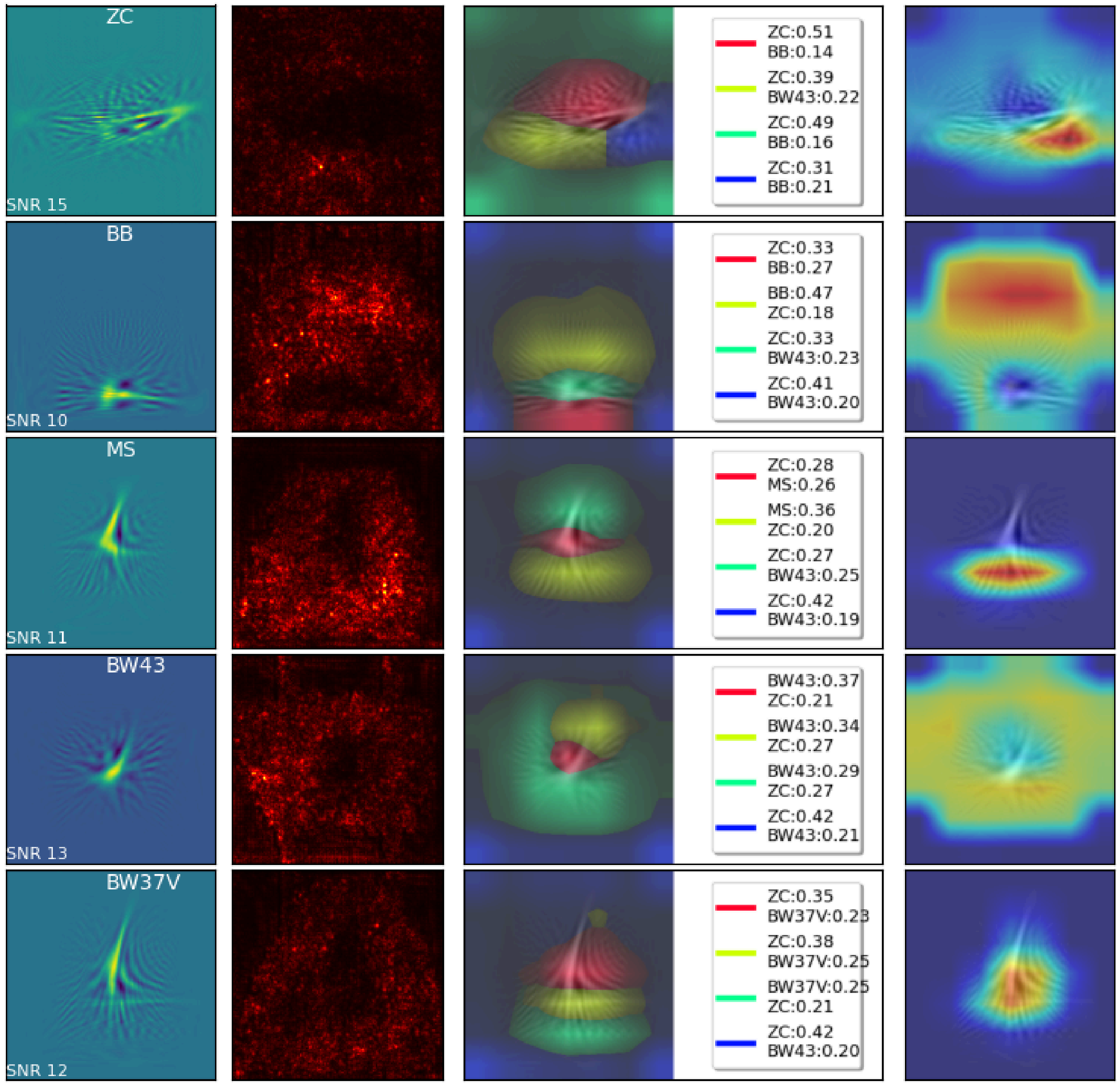

Figure 10: Click image (first column), saliency map (second column), Deep Feature Factorization (third column), and GradCAM (fourth column) images for one click of each species that was correctly classified.

Next I wanted to see if adding inter-click interval (ICI) to the feature vector would help improve model performance. Difference species emit clicks at different ICI values, and this is a key piece of information that acoustic analysts use to identify species. Since there may be clicks that are not detected or extraneous clicks included in each event, ICI is measured by taking the modal value of the time difference between each click in an event. Figure 11 shows the distribution of ICI values for each species in the training dataset. While there is some overlap between some of the species, there are also clear distinctions. I first implemented this by appending the ICI value for each detection to the feature vector before the final classification layer. This resulted in no notable change in model performance, which was surprising since it is such a valuable piece of information used by acoustic analysts when classifying beaked whale species. After a discussion with Eli, he suggested that I try multiplying the ICI values by 10 before appending them to the feature vector to force the model to give them more consideration. This improved the results, although in a bit of a surprising way. Many overall metrics at the detection level declined, but performance at the event level improved.

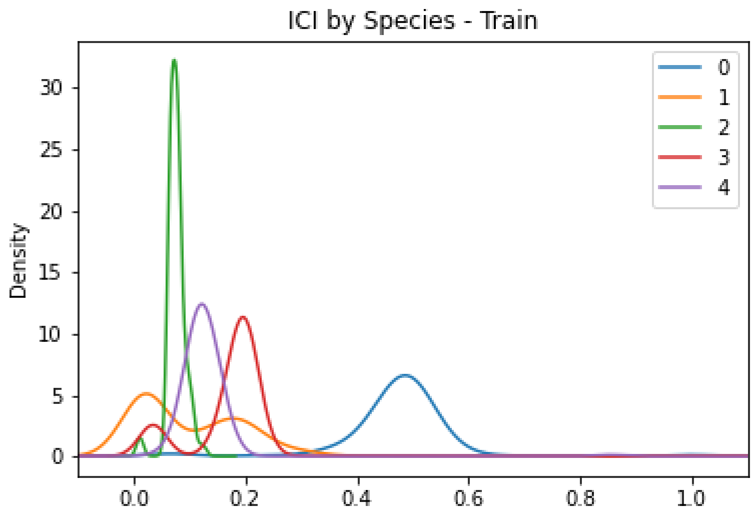

Figure 11: Distribution of measured inter-click interval (ICI) values in the training dataset colored by species

The improved results of the model including ICIx10 are even more encouraging when looking at the prediction confidence of the model. Figure 12 shows the distribution of predicted probabilities for each event in the model without ICI (top row) and with ICI (bottom row). Events with incorrect predictions are marked in red. Adding ICI to the model has not only reduced the number of incorrect predictions in the validation and test datasets (note that the test set is now included because this was intended to be the last set of model runs before GPU access was cut off), but it has also reduced the model's confidence in the few incorrect predictions that remain (except for one high confidence event in the test set). The real world use case for this model would never be to simply apply it to new data and trust the results. Rather we would want to predict and then verify a subset. These plots show that if we manually verified the 10% of events with the lowest prediction confidence from the model we would catch *all but one* of the errors the model makes! This is an incredible level of accuracy and incredible savings in the amount of manual labor that needs to be done.

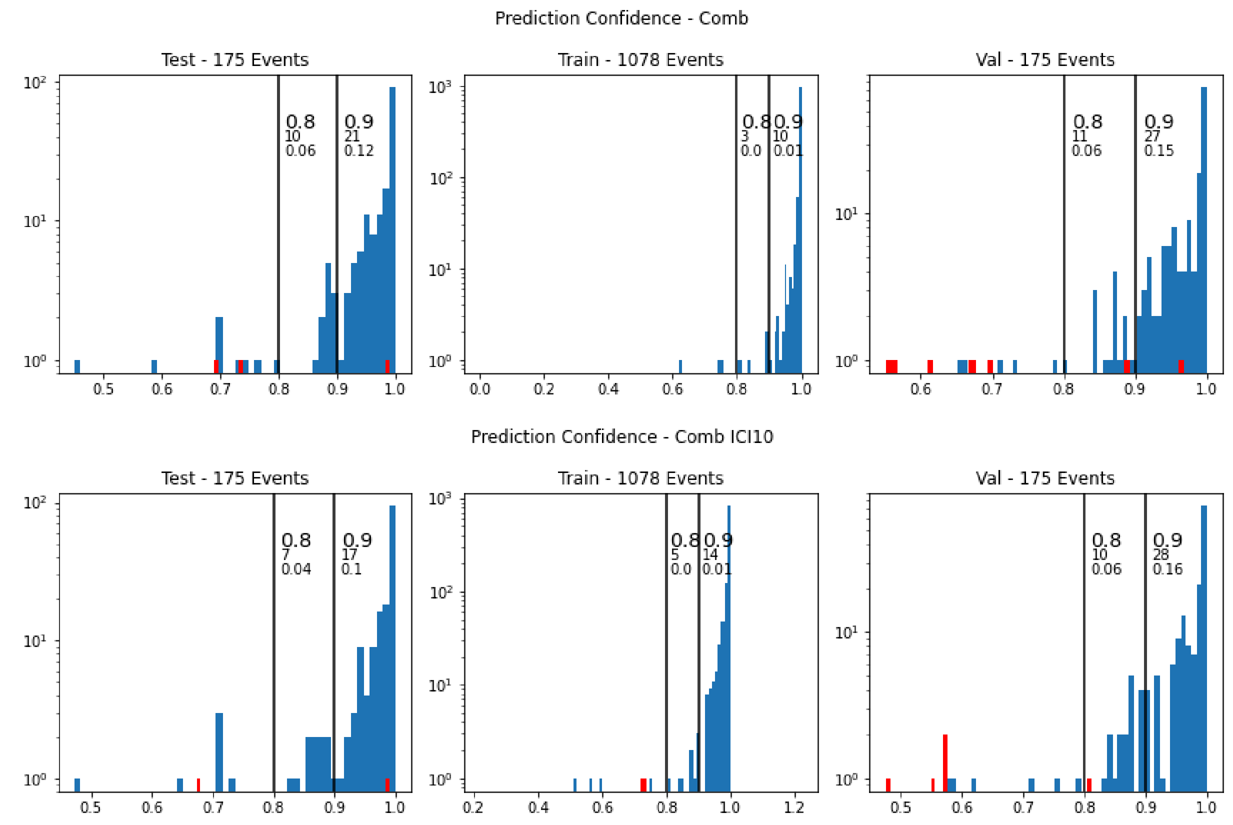

Figure 12: Distribution of average prediction confidence for each event for the model without ICI (top) and with ICI (bottom). Blue bars are correct predictions, red bars are incorrect predictions. Black lines are drawn at 0.8 and 0.9 confidence values, with numbers below showing the number and percentage of events below that confidence value.

With these results in hand, I wanted to compare this computer vision (CV) model to an existing model framework we use at our lab called BANTER. BANTER is a hierarchical random forest based classifier developed at the Southwest Fisheries Science Center that has shown strong performance in a number of classification tasks. It relies on calculating a set of features for each click, so its success is highly dependent on figuring out how to calculate an informative set of features. In order to make a fair comparison, I trained a BANTER model on the same exact training data used for the CV model, and also included ICI as an extra variable. Figure 13 shows event level confusion matrices for both the CV and BANTER models for both the validation and test data sets. The CV model outperforms the BANTER model in almost every way. The BANTER model is particularly prone to misclassifying species “ZC/0” as species “BB/1” or “BW43/3”, reducing the precision for those classes. BANTER models had previously been state of the art for this dataset and have been very successful in many other applications, so the fact that I was able to create a computer vision model that surpassed its performance in a relatively short amount of time is very exciting.

The model was also able to identify a mistake in our labeled data. Once model selection was finished, I worked with an acoustician in our group to double check a few of the events that the model was getting wrong. One event in the test set (the high confidence error in Figure 12) was incorrectly labeled as “ZC/0” when it should have been “MS/2”, which is what the model predicted. The success of this project means that our lab will likely try to find other use cases for computer vision in our work.

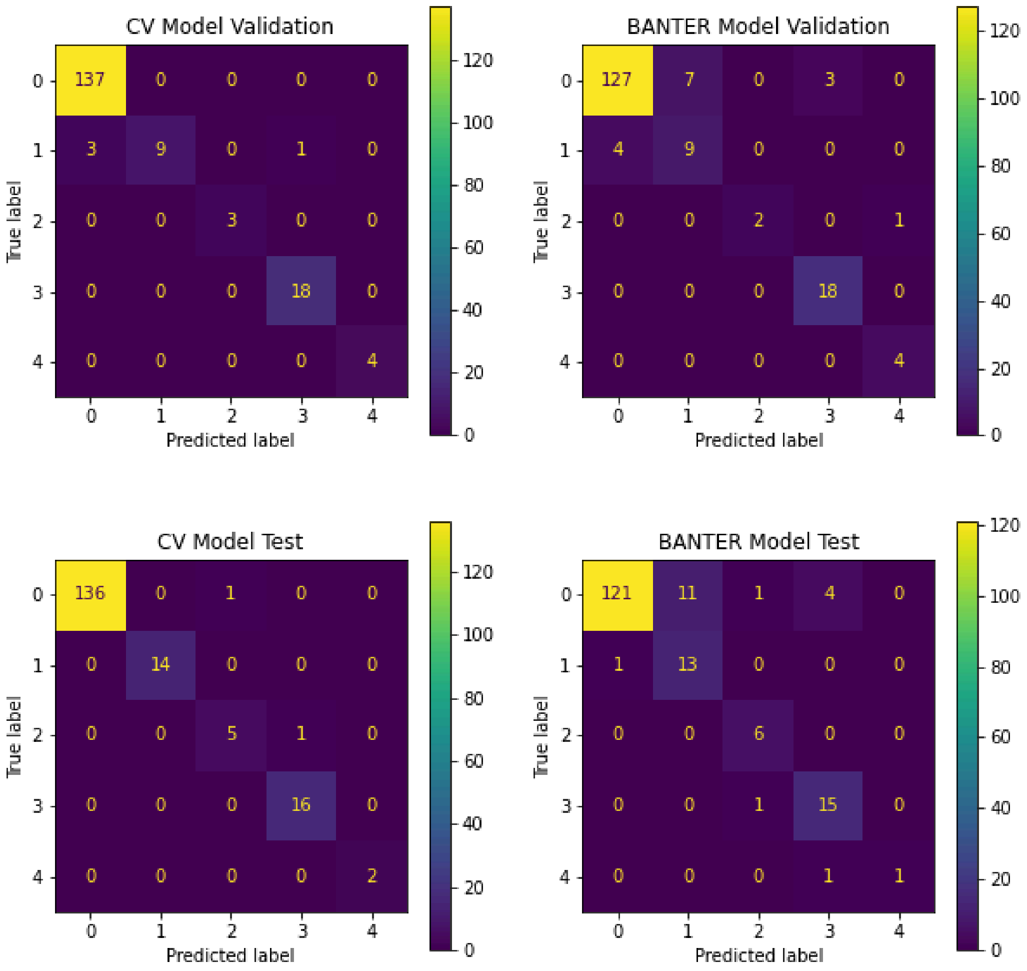

Figure 13: Event level confusion matrices for the computer vision model with ICIx10 (left column) and BANTER model with ICI. Models were trained on the same training data, then predictions made on the same validation (top row) and test (bottom row) sets

V. Model Development - Bonus GPU Time

Originally we were scheduled to lose access to our Azure GPUs just a few weeks after the workshop ended, but that ended up not being the case. That means that I got a chance to pursue another side project with this dataset. I wanted to try and train a classifier with a rejection option. Normally a classifier has no choice but to assign one of the species IDs it was trained on to a new input. A model with a rejection option is instead allowed to abstain from making a prediction in ambiguous cases. In the context of our problem, we know that there are more species of beaked whale present in California than the five we trained the model on, we just didn't have enough training data for any of the other species. Additionally, we know that there will be a number of noisy or low quality clicks included with each labeled event, or even clicks that were just created from ambient noise. A classifier with rejection could potentially address both of these issues by first learning to reject noise (and thus not learn that features of noise are important for the classification task), and also abstaining from making a prediction on a click from a different species.

After some research I decided to try and build a model based on SelectiveNet, mostly because it looked relatively straightforward compared to some other options (and this is the most complicated thing I've tried to implement) and because they had some code available on GitHub. SelectiveNet works by simultaneously training a second model that decides whether or not to keep an input image, outputting a score *q* from 0 to 1. The loss for an input is also scaled by *q*, so that if the selection model decides not to keep an image (low *q*), then the loss for that image is reduced. In order to stop the model from simply deciding to minimize loss by setting all *q* values to zero, there is an additional loss component tied to a value called coverage. The target coverage is a hyperparameter selected by the user, and roughly corresponds with the percentage of inputs you want the model to keep. So target coverage of 0.9 means you want around 10% of inputs to be discarded. Empirical coverage is the average of the *q* values, and the squared difference between empirical and target coverage (multiplied by a scaling factor *lambda*) is added as an additional loss term.

Fine tuning the SelectiveNet model was difficult. Examining the results of the first iteration of the model revealed that it was deciding to discard entire events purely based on class. I realized that since I was using a class weighted loss function the model was likely learning that discarded under-represented classes with higher class weights was most beneficial. I tried changing from a weighted loss function to a weighted sampling strategy to get around this and still account for class imbalance. The results showed that the selection model had a more even distribution of selection/discard throughout events (no longer discarding entire events with any obvious bias), but accuracy within the selected images was lower than accuracy in the discarded set.

Since the balanced sampling strategy did not work well for the selection model, and a weighted loss function performed better on my previous models, I wanted to try and modify the SelectiveNet loss to better incorporate class weights. I changed both the loss function and the empirical coverage measures so that each individual loss / coverage is multiplied by the class weight, and the sum is divided by the sum of the weights (equation 1).

$$Loss = \frac{\Sigma\, w_i\, q_i\, Loss_i}{\Sigma\, w_i} + \lambda\,(Cov_{emp} - Cov_{tar})^2,\ Cov_{emp} = \frac{\Sigma\, w_i q_i}{\Sigma\, w_i} \quad (1)$$

$$Loss = \frac{1}{n}\sum q_i Loss_i + \lambda\,(Cov_{emp} - Cov_{tar})^2,\ Cov_{emp} = \frac{1}{n}\sum q_i$$

This model seemed to perform better, but a closer look at the images that were not getting selected revealed strange results. Figure 14 shows the images in the training set with the lowest *q* values. They are all from the same species, look very similar, and they are all images that should be very useful for classification, not the quiet or noisy images I was expecting or hoping that the model would choose to discard.

figures/SelNetV2_Train_LowSelection

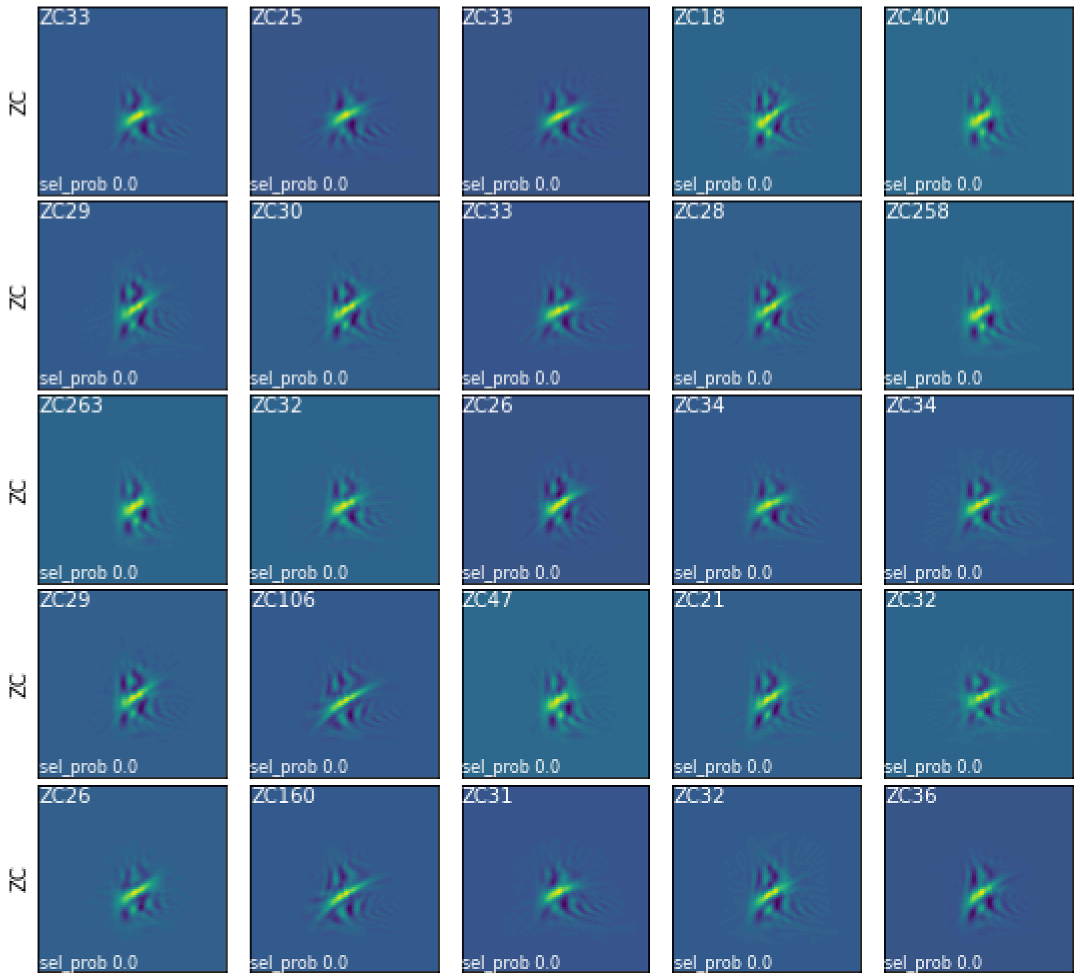

Figure 14: Images with the lowest selection value *q* in the training set for early implementation of SelectiveNet style model

I hypothesized that the selection model might not be sophisticated enough to effectively learn which image to discard. The initial architecture for this model presented in the SelectiveNet code takes the feature vector as input, passes it to a linear of size 512, then passes this to a linear layer of size 1 as the output. I wanted to add an additional linear layer, so I modified this structure to take the feature vector as input, then a linear layer of size 128, then a linear layer of size 64, then a linear layer of size 1 as output. In all cases each hidden linear layer has batch normalization and a ReLU activation, while the output layer has a sigmoid activation to get outputs from 0 to 1. The numbers of 128 and 64 for my modified architecture were chosen arbitrarily because I have no experience designing model architecture. The results of this modified model were terrible. The selection model was no longer choosing to discard any inputs, all *q* values were above 0.6. I changed the sizes of the hidden linear layers from 128 and 64 to 512 and 512, and the results were greatly improved. The distribution of *q* values was much more reasonable, and accuracy within the selected group was higher than in the discarded group. Figure 15 shows images in the training set with the lowest *q* values, and they have significantly changed from the earlier selection model implementation. These are much more in line with what I would expect the model to discard since they are all mostly noisier images, or images of clicks that are more ambiguous and would be harder to classify. Model performance improved in almost all metrics, most notably the recall for images of class "BB/1" increased from 28% to 41% with the inclusion of the selection model, resulting in one additional "BB/1" event being predicted correctly.

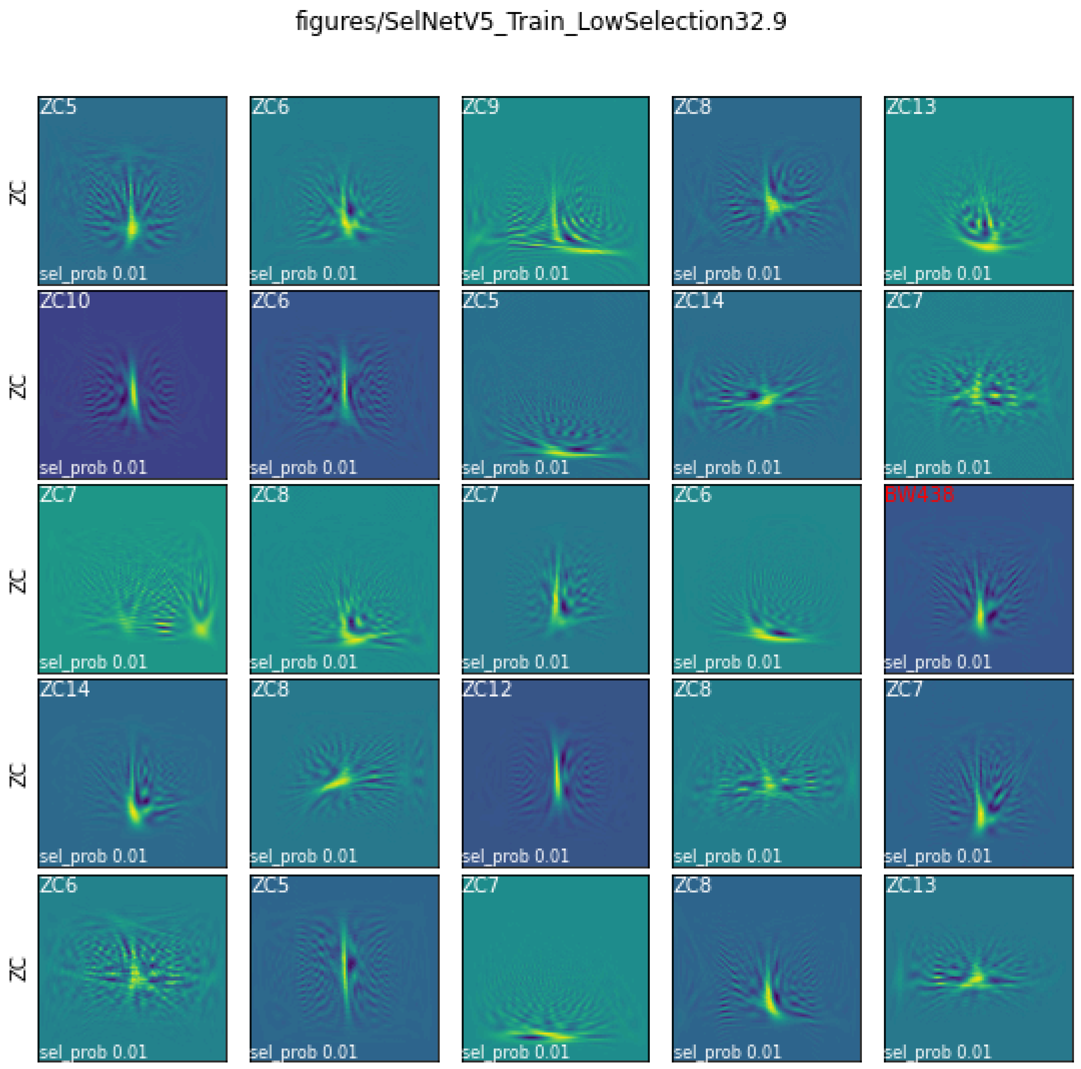

Figure 15: Images with the lowest selection value *q* in the training set for improved implementation of SelectiveNet style model

As a final test of the SelectiveNet model, I wanted to see what the selection scores would be for data from species other than the five I trained the model on. I have additional labeled data for 11 more classes which include other beaked whale species, beaked whale detections that cannot be confidently labeled to a specific species, other cetaceans, and ship noise. Figure 16 shows the average selection scores across all data for every class. The blue bars are the five classes the model was trained on, with a dashed blue line marking the lowest average selection score for these classes. Purple bars are for other beaked whale species, and red bars are for other cetaceans and ship noise. While selection scores are mostly lower than for the classes the model was trained on, there are many cases where they are similar. Thus it is unlikely that this model will reliably discard images belonging to classes outside of our training set. One notable takeaway from this image is that the class with the lowest average score represents ship noise, suggesting that the selection model is useful for filtering out some anthropogenic sounds.

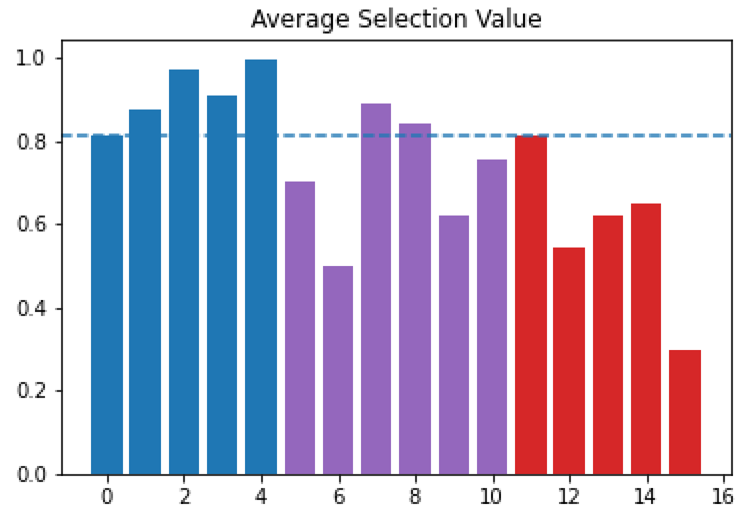

Fig 16: Average selection score values for all 16 classes that I have data for. Blue bars represent classes in the original training set, purple bars represent other beaked whale species, and red bars are other cetaceans and ship noise.

VI. Model Development - Future Goals

The next step for this particular project would likely be to try a different style of classifier with rejection to see if it can more reliably reject species that were not part of our training data. There are a number of very rare beaked whale species that we may have on future recordings, so any model used in a real world setting would need to account for this.

The success of this project means that we are also likely to find other ways we can use computer vision models in our lab's work. We chose this beaked whale dataset for the summer school because we believed it had a strong chance to be successful, and we had an existing baseline model (BANTER) to compare performance to. If we couldn't get a computer vision model to work for this dataset, then we didn't think it would be worth pursuing them as an option for future harder tasks. We are currently in the process of starting a Zooniverse project that will provide some labeled spectrogram images of humpback whale song and ship noise, and we also have a dataset of labeled fin whale detections. Both of these are strong candidates for a next computer vision project, although a timeline for starting either of them is unclear.