Automatic Detection of Whale Lunges: A Deep Learning Approach

Hugo Valdivia, William McCloskey

December 14, 2018

1 Task Definition

1.1 Problem Statement And Motivation

Understanding whale lunges is a key area of current whale research; researchers use this information to further conservation efforts and to more deeply understand their impact on the ecosystems they inhabit [1]. As the largest animals that have ever inhabited this planet continue to be endangered, this research is of vital importance [2]. Currently, researchers parse through hundreds of hours of sensor data from tagged whales to label lunges by hand. Not only does this inefficiency take researchers' time away from more important tasks, but it also limits the amount of data they can collect. Indeed, it does not make sense to collect data more quickly than it can be labeled, and length of tag deployment is limited by the amount of memory in the tag. An automated lunge detection algorithm would eliminate the need for hand-labeling and allow for online labeling to free up memory for long-term tag deployments. Thus, we seek an automated method for detecting whale lunges.

The task is set up as follows. We have a finite input time series sampled at 10 Hz $\{x^{(t)} : t \in \{0, 0.1, 0.2, \dots T\}, x^{(t)} \in \mathbb{R}^5\}$ of five tag measurements per sample. These quantities are speed, pitch, roll, heading, and jerk. We also have a set of ground truth lunge times $Y = \{t_1, \dots, t_k\}$ hand labeled by the researchers. Using the measurements $x^{(t)}$, we want to output a collection of lunge labels $\hat{Y} = \{\hat{t}_1, \hat{t}_2, \dots, \hat{t}_\ell\}$ corresponding to times when the whale lunges.

1.2 Evaluation Metrics

To evaluate the performance of the models, we use a few metrics. Because we cannot expect to output the true times $t_1, \dots, t_k$, we define a lunge as correct based on an error tolerance E. Note that we use an error tolerance $E = 5$ seconds in this paper. Basically, an estimated lunge time is correct if it is within E seconds of a true lunge. For a given error tolerance E, we define the true positive rate tp_E and false positive rate fp_E as follows:

$$tp_E = \frac{|\{t_i : \min_{\hat{t}_j \in \hat{Y}} |t_i - \hat{t}_j| \leq E\}|}{|Y|}$$

$$fp_E = \frac{|\{\hat{t}_j : \min_{t_i \in T} |t_i - \hat{t}_j| > E\}|}{|\hat{Y}|}$$

We note that we do not need to worry about duplicate labels because our algorithms correct for these. Based on the quantities tp_E and fp_E, we can define the $F1$ score analogously to the setting of binary classification:

$$F1_E = \text{Harmonic Mean}(tp_E, 1 - fp_E).$$

For us, we focus on optimizing the $F1$ score for $E = 5$ seconds as labeling within 5 seconds is sufficient for our task, and both missing true lunge times and outputting correct labels are important.

2 Literature Review

To our knowledge, researchers working in this domain have applied only classical machine learning techniques to this problem. The most recent development that we found from 2016 used decision trees, which incorporate the extensive domain knowledge that biologists hold; it is with such approaches that algorithms can be specifically tailored to fit particular problems (e.g., predicting lunges on specific whale species) [3]. The results from this paper ultimately are incomparable with ours, as they covered the more difficult case of shallow feeding but it appears they did not test their model on unseen data. Nonetheless, it did guide our choices for evaluation metrics and error tolerance thresholds. In conclusion, these classical methods have been shown to work reasonably well, but as larger amounts of data have been collected and as neural network models have been found to succeed on similar tasks, there is huge potential to achieve better results by moving away from such machine learning algorithms.

Indeed, the literature has found tremendous success in applying deep learning techniques to time-series data, as in [4]. This paper greatly influenced our design choices in our iterative progression through different architectures. In particular, the researchers found that on datasets very similar to ours, the ResNet architecture consistently ranked among the best. Indeed, out of all of the models that we tried with our dataset, our ResNet model achieved the best results per our evaluation metrics and is a key part of our final model.

We expand on the literature by appending a lunge correction model to our lunge prediction architectures. The basic idea is as follows: given a positive lunge prediction from the original detection model, we aim to improve the precise placement of such a prediction. In order to fully automate this task and to aid researchers, we believe this to be an essential step.

3 Dataset

Our data comes from the Goldbogen Lab at the Stanford Hopkins Marine Station. Researchers observe whale behavior far below the ocean's surface with an accelerometer tag, and obtain a time series consisting of a whale's depth, speed, jerk, pitch, and roll. Biologists use the deviations in speed, jerk, or roll, in particular, to indicate lunges. The lunges in our data have been hand-labeled by the lab. In total, our dataset consists of 29 distinct tag

deployments on blue whales. The time, length, and number of lunges of each deployment are contained in the table below.

Deployment Number	Hours	Lunges
0	4.9	84
1	4.8	87
2	5.6	114
3	4.7	62
4	0.9	6
5	28.4	369
6	13.1	244
7	30.7	699
8	5.0	8
9	18.9	65
10	1.1	21
11	3.3	40
12	8.8	213
13	4.2	15
14	12.5	213
15	2.6	17
16	3.3	54
17	10.1	230
18	16.1	395
19	6.9	147
20	15.8	276
21	1.14	2
22	8.1	160
23	37.5	383
24	21.5	294
25	26.4	314
26	0.8	10
27	19.1	214
28	3.3	32
Total	319.7	4768

Measurement characteristics vary across deployments due to inconsistencies in tag deployment and whale behavior. To mitigate the effect of this inconsistency on our models, we standardized each measurement for each deployment. For example, we took the speed measurements from a given deployment and normalized them to have zero mean and unit variance. By standardizing per deployment instead of using universal quantities on the whole training set, we account for individual differences between deployments.

We randomly selected three deployments for our validation set and three deployments for our test set (provided the validation sets had roughly 300-600 lunges). As a result, our validation set was deployments 6, 9, and 10 for a total of 330 lunges, and our test set was deployments 3, 17, and 20 for a total of 568 lunges.

4 Approach

To obtain an algorithm that automatically labels a deployment, we use a sliding window approach. Given a twenty second window of features $X^{(t)} = x^{(t-10)}, x^{(t-9.9)}, \ldots, x^{(t)}, \ldots, x^{(t+9.9)}, x^{(t+10)}$, we label the window as $y^{(t)} = 1$ if there is a lunge in the middle six seconds of the window, i.e. if $Y \cap \{t-3, t-2.9, \ldots, t+3\}$ is nonempty, and as $y^{(t)} = 0$ otherwise. We train our networks to predict these labels $y^{(t)}$, so the output layer of the network has sigmoid activation.

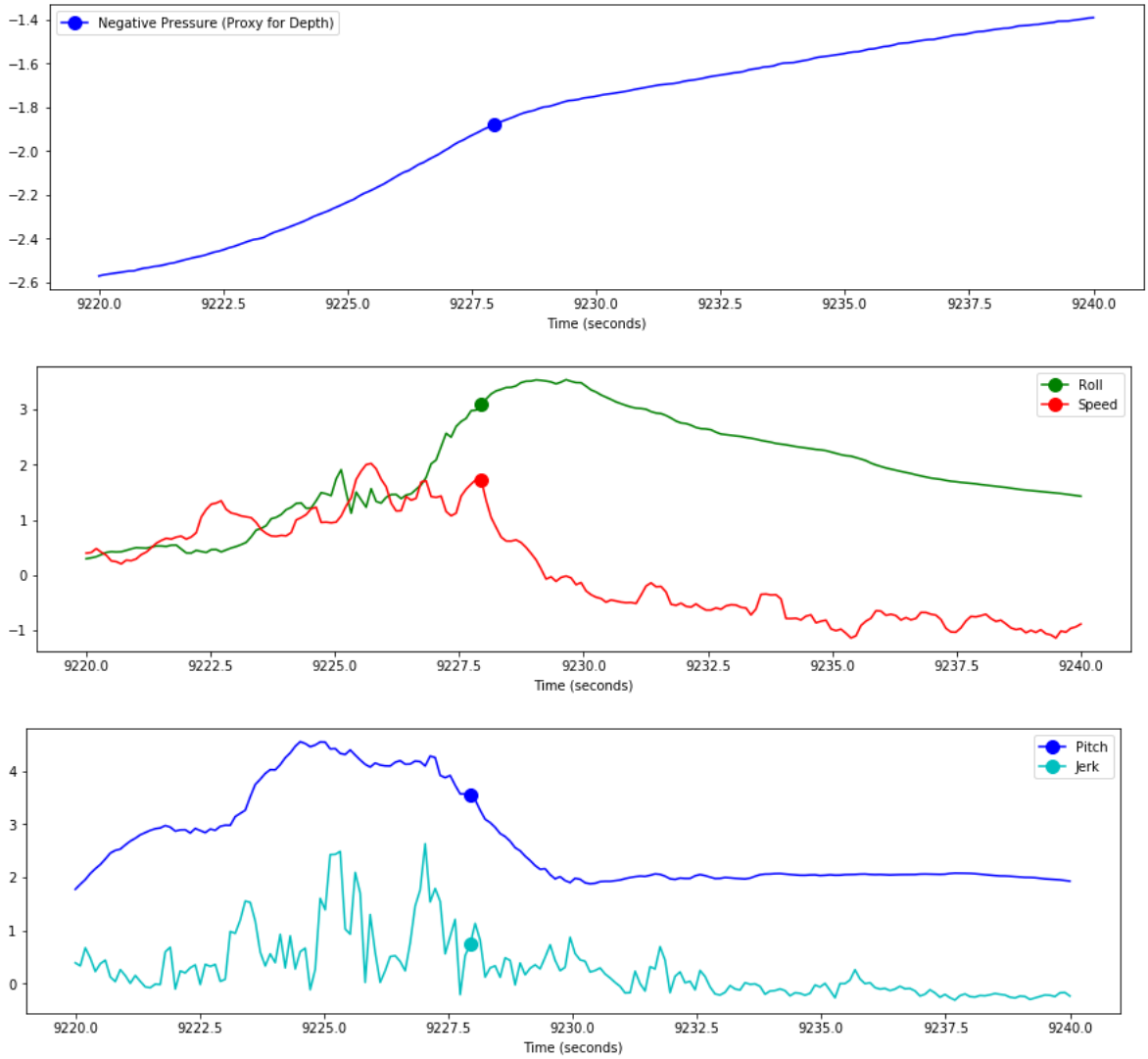


Figure 1: An example of a window $X^{(9230)}$ with a positive label. Notice that the lunge occurs within the middle six seconds of the window.

Given a model $\mathcal{M}$ that predicts on a twenty second window, we output a sequence of predictions every second by sliding the window by one second at a time across the entire deployment to get an output sequence $\hat{s}^{(t)} = \mathcal{M}(X^{(t)})$, where $t = 10, 11, 12, \ldots T - 10$. We then smooth the prediction window by a moving average filter $\hat{y}^{(t)} = \frac{1}{5}\sum_{i=-2}^{2} s^{(t+i)}$ to mitigate noisy predictions.

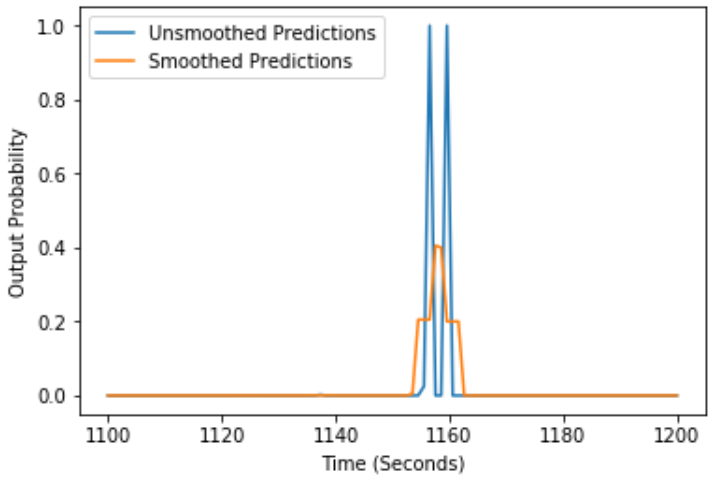


Figure 2: Example prediction signal before smoothing ($\hat{s}^{(t)}$) and after smoothing ($\hat{y}^{(t)}$).

We still have multiple predictions for each true label $t_i \in Y$. To resolve this, we take the set $\{\hat{y}^{(t)} > 0.5\}$ and mod out by the transitive closure equivalence relation $\hat{y}^{(t_1)} \sim \hat{y}^{(t_2)}$ if $|t_1 - t_2| < 10$. Essentially, we cluster together positive predictions if they are less than 10 seconds apart. For each equivalence class C, we output the time with the highest prediction $\arg\max_{t:\hat{y}^{(t)} \in C} \hat{y}^{(t)}$ if that prediction exceeds 0.9. These are essentially our predicted lunge times $\hat{Y}$.

Since the model $\mathcal{M}$ only predicts based on whether a lunge was in the middle six seconds of the window, these predictions are not as close in time to the true label as they could be. So we also train a model $\mathcal{N}$ that takes in a predicted time $\hat{t} \in \hat{Y}$, the surrounding features within 8 seconds $x^{(t-8)}, \ldots, x^{(t+8)}$ and predicts the offset from the true label t. We can thus use $\mathcal{N}$ to make our predictions more precise.

4.1 Preprocessing

To train the model $\mathcal{M}$, we collected windows from the training set deployments. Keeping in mind our data imbalance problem, we included every possible positively labeled window from the training deployments in our training set. Thus, for each label $t_i \in Y$, we had sixty-one possible windows $X^{(t-3)}$, $X^{(t-2.9)}$, ..., $X^{(t+3)}$. Originally, we tried keeping only twice as many negative windows as positive windows per deployment, but we found that our models had a high false positive rate. After much tuning, we eventually settled on twenty times as many negative windows as positive windows. One one-hundredth of these windows were randomly sampled near-misses – they had a lunge in the window but not in the middle six seconds (this is a fifth as many positive windows). The rest of the negative windows were randomly sampled from the deployments.

To train the model $\mathcal{N}$, we synthetically generated incorrect labels. For each true label t_i, we generated thirty false labels $t_i + U$ where U is drawn uniformly from the interval $[-5, 5]$. We then got windows $x^{(t_i+U-8)}, \ldots, x^{(t_i+U+8)}$ and we train the model $\mathcal{N}$ to predict U.

4.2 Models

In order to gauge how well we could do expect to do on this task, we asked researchers at the Goldbogen Lab roughly what percentage of candidate lunges they would unanimously agree on. Their answer depended on species and feed. In particular, for our blue whale task, they said they could agree on upwards of 95% of the lunges. Given that the humans labeling this data could reach this upper bound, this oracle set our expectations for what our more advanced neural network models, which can be very good at mimicking human performance, could potentially achieve.

Conversely, in order to get a lower bound on our performance, for our baseline, we chose a logistic regression model. Our original baseline was framed as a simple binary classification problem for each time step. Thus, for this model, we completely ignored the time-series component - that is, our input was a 5-dimensional vector of features for a single time step and our output was whether a lunge occurred or not. We did not apply any of the techniques discussed above (no windows, smoothing, lunge correction, et cetera). On our validation set, this model produced an F1 score of 0.72 on such a binary classification task.

However, with our approach described above, our task is not quite as simple as binary classification. Thus, we decided to re-run our baseline; for this, we trained a single layer feed forward network with a single sigmoid output, as the two models are equivalent. This allowed us to take advantage of the structure we had built for our neural network models, as opposed to treating it as a simple binary classification task at every time step.

We then moved on to a deeper feed-forward network. We implemented a 3-layer feed-forward neural network, where the number of hidden units of the layers is [200, 50, 1]. In order to run our data through this model, we had to flatten our input to create a (1000,1) input vector. We used ReLU activations for all of the hidden layers except for the output one, where we used a sigmoid activation. We trained our network using an Adam optimizer with a learning rate of 0.001 and a binary cross-entropy loss. We found that a network this deep was enough to overfit on the training set. Since this model was expressive enough to overfit, we tried two different forms of regularization: ℓ_2 regularization and dropout; however, we opted for early stopping as that required less tuning. During our initial training, we found that, while this model could do quite well on the training set, we believed more specialized architectures could better generalize to our validation set.

Because of this, we implemented a Bidirectional RNN LSTM model. We thought this model would work well, as a model that could take advantage of the previous time steps as well as look into the future would be a good fit for our problem. Without an extensive hyper-parameter search, however, this model did not perform as well as the feed forward model. With limited computational resources, we opted to focus our efforts on our next model.

For our last model, we trained a ResNet on our data, making use of the implementation in [5]. In tuning this model, we found the biggest gains in tuning the proportion of negative samples to positive samples that we fed into the network. This brought our evaluation numbers quite close to Bayes' error, which suggested that our earlier models could also benefit from this increase in data.

Lastly, for our correction model problem, we chose a 4-layer feed-forward network with, where the number of hidden units of the layers was [32, 20, 5,1]. We used ReLU activations

for all of the hidden layers except for the output one, where we used a tanh activation. This model surprisingly provided better predictions than the ResNet architecture, which could have been too deep for the task. Since this was a regression problem, we used a mean squared error loss function and used an Adam optimizer with a learning rate of 0.001

4.3 Final Results

The table below summarizes our results on the validation and test sets using both our lunge prediction and correction models under a 5 second tolerance.

Model	Val TP	Val FP	Val F1	Test TP	Test FP	Test F1
Logistic Regression	0.930	0.264	0.822	0.963	0.090	0.945
Feed Forward	0.973	0.027	0.973	0.982	0.049	0.966
ResNet	0.964	0.036	0.964	0.972	0.030	0.971

Surprisingly, the logistic regression did quite well on the test set albeit not as well as the neural network models. Since we elected to refrain evaluation on the test set until the very end after all of our models had been tuned, we conjecture that the test set could have been easier and that the validation numbers are closer to reality. We also observed that logistic regression converged to a much higher loss than the other models on the training set. We believe that our methods of smoothing and thresholding to make our final predictions also cause the model's specific performance to matter less in the final metrics.

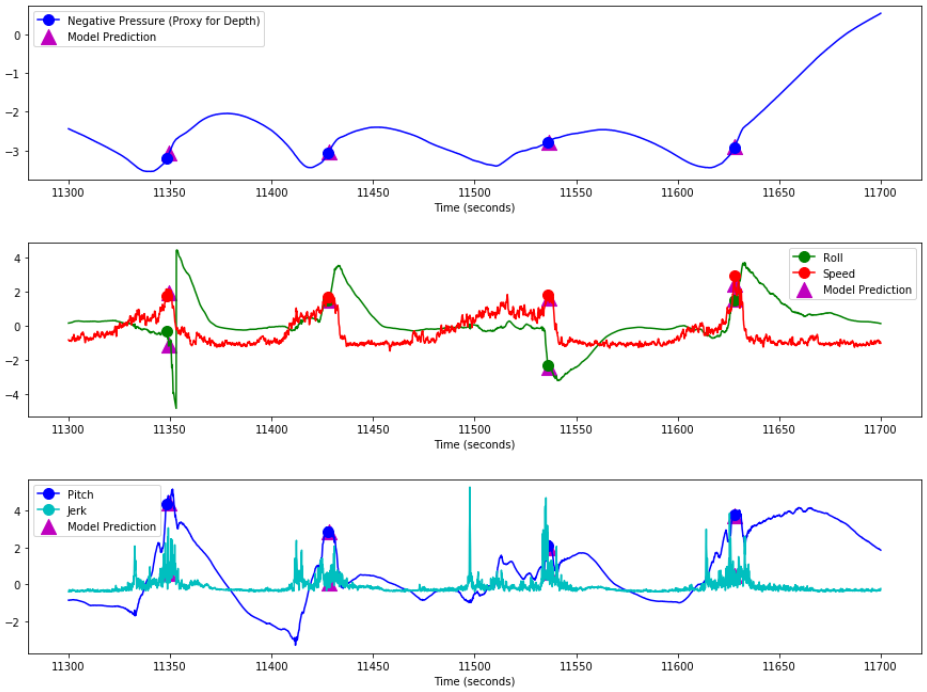
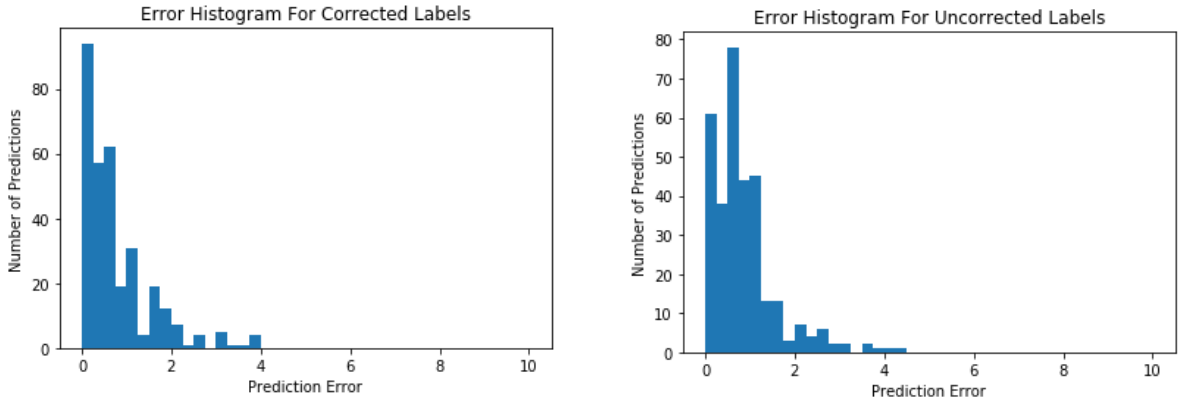


Figure 3: Example output predictions for the feed-forward model with correction. (Circle denotes a ground truth lunge and triangle denotes our model's prediction.)



(a) Errors For Corrected Model

(b) Errors For Uncorrected Model

Figure 4: Histogram of errors (in seconds) for the Feed Forward Model on the validation set with and without the correction model. The corrected model has an average prediction error of 0.743 seconds while the uncorrected model has an average prediction error of 0.837 seconds.

5 Error Analysis

When we were training on a lower ratio of negative examples to positive examples, our models had a high false positive rate. (Around 28% on the validation set.) To diagnose our problem, we plotted a few examples of the incorrect outputs. We found that the incorrectly labeled lunge times did not appear close to a lunge. The reason for this is that the negative class (absence of a lunge) has a huge variety of elements. Since the model learns a class by looking at representative samples, we found that we needed much more negative examples, eventually switching from a 2:1 negative window:positive window ratio to a 20:1 ratio. Nonetheless, the few false positives that the current model outputs still do not look like lunges, though there are substantially less false positives with the 20:1 ratio.

These errors indicate that our model may be able improve even more. Indeed, though we are close to Bayes error for our positive predictions, it may be possible to further lower our false positive. Our model is likely to improve the false positive rate with an even higher number of negative examples. (Though the result may include a lower true positive rate.) However, when we tried training on a 30:1 negative window:positive window ratio, our model never learned to do better than constantly output zero. We think that some tricks with data augmentation or starting to train on a 20:1 ratio and finishing on a 30:1 ratio may improve the model even more.

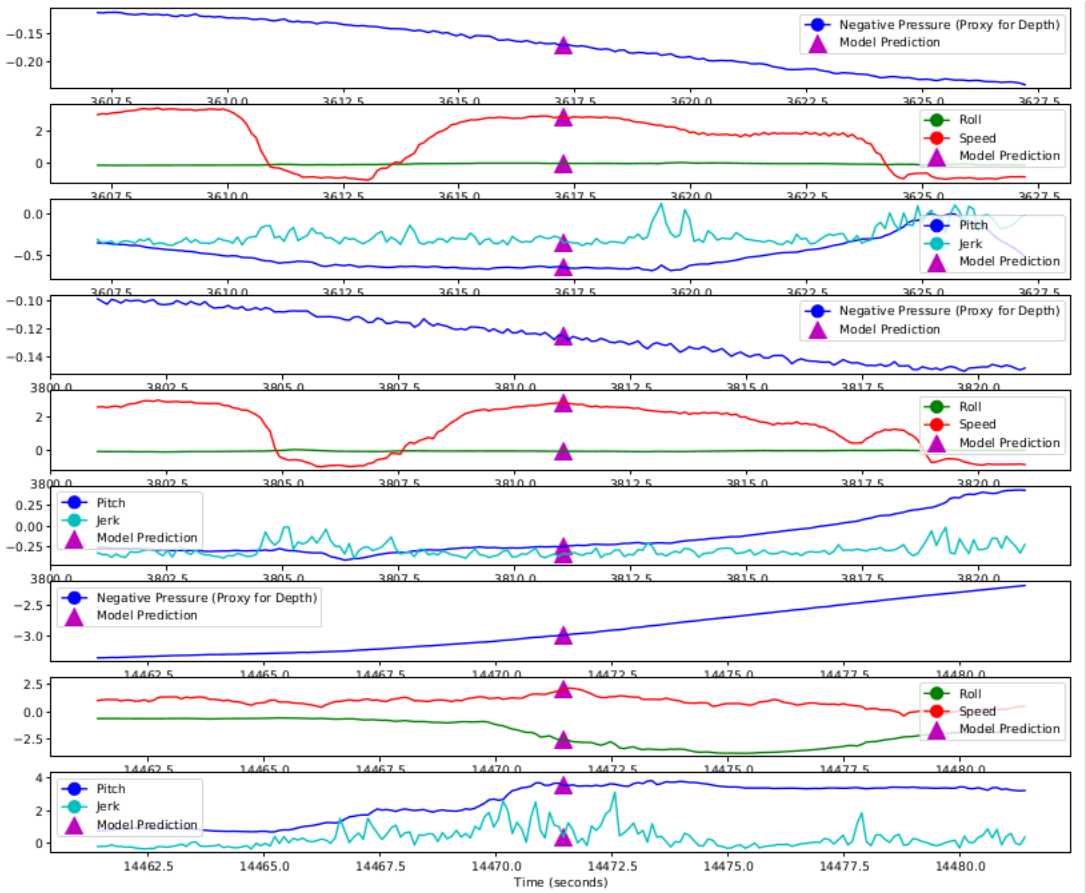


Figure 5: Incorrect labels of the Feed Forward Network with a 20:1 negative window:positive window ratio.

6 Future Work

We can conclude that our models work quite well for this particular blue whale task; however, there is still much opportunity for continued development. For instance, a natural extension of this work that [3] included and is of interest to the whale community is to develop models for different whale species or for shallow feeding. Blue whale lunges are the easiest case to label, since most of their lunges follow a very stereotypical pattern. Humpbacks feeding on fish, by contrast, have very erratic time series measurements and many lunges are ambiguous even to well-trained researchers. Detecting shallow lunge feeding is also difficult, as the measurement signals can contain plenty of noise due to bubble-net feeding and other shallow water behavior. We are optimistic that our approach here can be generalized to those cases, by transfer learning or by other techniques.

References

[1] Stanford researchers reveal details about the unique feeding habits of whales. https://news.stanford.edu/2016/09/22/unique-feeding-habits-whales-come-light/. Accessed: 2018-12-14.

[2] Blue whale. https://www.nationalgeographic.com/animals/mammals/b/blue-whale/. Accessed: 2018-12-14.

[3] Ann N Allen, Jeremy A Goldbogen, Ari S Friedlaender, and John Calambokidis. Development of an automated method of detecting stereotyped feeding events in multisensor data from tagged rorqual whales. *Ecology and evolution*, 6(20):7522–7535, 2016.

[4] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *arXiv preprint arXiv:1809.04356*, 2018.

[5] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. https://github.com/hfawaz/dl-4-tsc.