HARVEY MUDD COLLEGE

**Computer Science & Math Clinic**

Midyear Update for
*Auburn / USDA*

# Time-series Modeling, Analysis, Interface, and Insight from Entomological Electropenetrography

December 7, 2024


**Team Members**

Mehrezat Abbas (Team Lead)
Milo Knell
Devanshi Guglani
Lillian Vernooy
Zachary Traul

**Advisor**

Prof. Gabriel Hope

**Liaisons**

Dr. Elaine Backus
Dr. Anastasia Cooper
Dr. Kathryn Reif

# 1 Introduction

Manually labeling feeding behaviors on electropenetography (EPG) waveforms is an lengthy process for entomologists. As labeling waveforms takes away time that could be spent on lab work, manuscript preparation, and other tasks, it becomes a bottleneck on research throughput. The importance of removing this bottleneck has risen in recent years due to climate-driven changes in arthropod ranges that threaten agriculture in both the United States and worldwide. EPG is a powerful tool for studying how parasitic arthropods interact with their hosts, so improving the speed at which EPG data can be interpreted is critical for understanding and combating threats to agriculture by problematic pests.

According our funding specifications as provided by the National Bio and Agro-Defense Facility and National Science Foundation, our project's focus is on blood-sucking arthropods such as mosquitoes and ticks, which threaten humans and livestock. Our goal is to use machine learning to develop a software that largely automates the process of labeling EPG waveforms. As our sponsors' entomologists lack the computing expertise to do so themselves, they are seeking the assistance of Harvey Mudd College's Clinic for development of this tool. Our team is currently developing a machine learning model capable of labeling feeding phases and behaviors of bloodsucking arthropods in EPG waveforms with a software interface for its use by entomologists. If successful, our tool will improve EPG research output and help to modernize EPG for combating the current and future pest threats faced within the agriculture industry.

In this report, we summarize the results of our work to date in processing EPG data, developing machine learning algorithms for waveform labeling, and creating a user interface for interaction with these algorithms.

# 2 Progress So Far

## 2.1 Data Cleaning

We have several datasets available to us. Our primary set consists of recordings made on *Culex tarsalis* (Western encephalitis mosquito), described in Cooper et al. (2024). The recordings themselves (voltage time-series data) are in the form of WinDaq (`.WDQ`) files made with the WinDaq data acquisition software. They are accompanied by plaintext CSV files and Excel spreadsheets containing transition times between waveform types, as well as other recording metadata such as excitation voltage level and amplifier input impedance. To more easily work with these data, we converted the recordings into densely labeled CSV files. WinDaq writes a binary file format. However, the format is documented, and we were able to use an existing Python library from Perry (2024) to import the voltage data. More recently we have also gained access to recordings made on *Aedes aegypti* (yellow fever mosquito), and we are in the process of cleaning these to be used as well. We also have access to a set of recordings made on aphids, but have not yet done a significant amount of work with these as our primary focus is mosquitoes.

Files recorded with the existing EPG system contain two channels, `pre_rect` and `post_rect`. The `pre_rect` signal is sampled from the output of the amplifier, and the `post_rect` signal is additionally passed through a rectifier and low-pass filter. We orig-

inally began working with the `pre_rect` signal, as we observed instances of clipping in the `post_rect` signal. However, this proved to be problematic. The dataset consists of a mix of recordings made with DC and AC excitation signals, and the AC excitation signal is at 1 kHz. Both signals are only sampled at 100 Hz, meaning the Nyquist frequency is 50 Hz. This results in the excitation signal being aliased down to around 10 Hz. While the amplitude is relatively small, this still obscures the waveform signals we are interested in, especially if we are doing spectral analysis. For the `post_rect` signal, the low-pass filter serves as an anti-aliasing filter, filtering out the excitation signal before it is sampled. This is shown in Figure 1. Moving forward, we plan to use the `post_rect` signal when possible. This may be challenging in certain cases, as mentioned previously we have observed instances of clipping in the `post_rect` signal (presumably on the rectifier voltage rails).

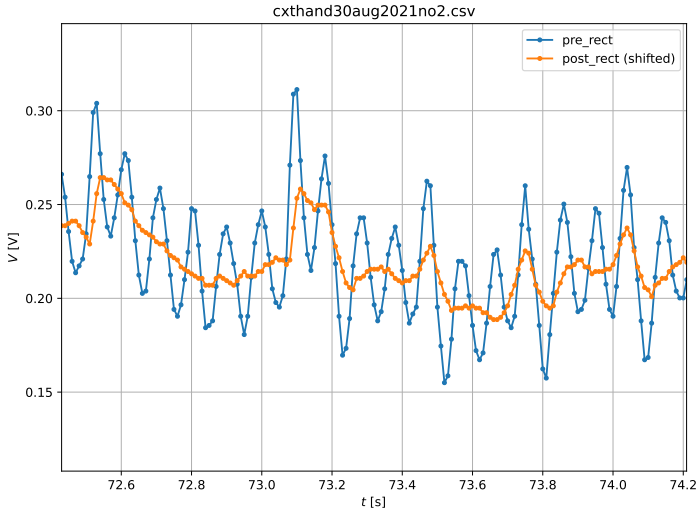


**Figure 1** `pre_rect` showing excitation signal aliased at about 11 Hz and `post_rect` with excitation signal filtered out. (The `post_rect` signal has been shifted to the same DC level as `pre_rect` for visualization purposes.)

## 2.2 Machine Learning

So far, we have made use of a variety of machine learning models and techniques. Each section below gives a description of a model and a characterization of its performance in a holistic sense. A summary of the models with numerical performance measures can be found in Table 1, along with implementation details about whether data is being classified densely or in windows. We compare each model based on accuracy, precision, recall, and F1 score. Accuracy measures the overall proportion of correct predictions, precision evaluates how often the model correctly predicts a class, recall assesses the proportion of actual positives correctly identified, and F1 score provides a balanced average of precision and recall. F1 score is the most indicative of actual model performance, but the other metrics are also useful when comparing models.

Note that these metrics are not directly comparable across different model architectures

| Model Name | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Random Forest | 0.849 | 0.51 | 0.39 | 0.40 |
| TCN | 0.715 | 0.57 | 0.56 | 0.52 |
| Unet | 0.773 | 0.41 | 0.45 | 0.40 |
| TCN+STFT | 0.711 | 0.57 | 0.53 | 0.51 |

**Table 1** Relevant Model Performance Summary

because the way that they make predictions is different. They examine different data windows, based on the unique computational needs of the different models. So while these metrics can be a useful way to understand relative performance, it is not necessarily a fair comparison. Nevertheless, these metrics will be helpful as a baseline measure of performance for next semester's models.

### 2.2.1 Random Forest

The Random Forest approaches implements the work of Willett et al. (2016) and applies it to our mosquito-specific use case. While a more in-depth description of this model can be found at the citation, at a high level this model divides each recording into one-second segments, uses a Fast Fourier Transform to identify the six most prominent frequency components in each segment, and then uses a Random Forest model to classify each segment based on these frequency features in combination with other features like time since start of recording and the settings used while taking the recording. We found that this approach performed about as well on our data as it did in the original paper in terms of overall accuracy, correctly labeling 85 percent of the segments. However, on a more granular level, it only performed well on the M, L, and NP waveform types and struggled on recordings with multiple probes.

Poor performance on J, K, N, and Z waveform types likely stems from their scarcity in the dataset as a whole and their short durations. Other drawbacks of this method stem from its narrow field of view and inability to incorporate information from around each segment when they are classified. This results in "barcoding" where the classified alternatively labels sections between two different waveform types in an interval of the recording. That is, it is unable to account for the fact that such rapid changes are not possible because this model does not look at the waveform as a whole when training and inferring waveform type, which can be seen in Figure 2. While this method acts as a good starting point, it would need to be paired with both better pre- and post-processing to reach the level of performance we desire. Specifically, performance would likely improve by segmenting probes before labeling and by correcting for barcoding in a post-processing smoothing step as described 3.1.1.

### 2.2.2 Deep Methods

We believe that deep learning, primarily based on convolutional neural networks (CNNs) is a good choice for this task. CNNs learn filters automatically during the training process, which can help them understand low-level patterns in our data by considering relative

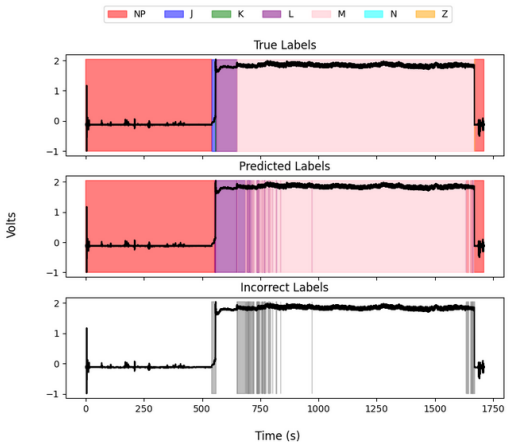


**Figure 2** Random Forest “Barcoding" Example

temporal features. There is a lot of existing research in applying CNNs to time series problems more effectively.

**TCN** The Temporal Convolutional Network (TCN), as suggested by Lea et al. (2016), is similar to a more traditional stack of convolutional layers. TCNs differ from regular CNNs in their ability to examine larger chunks of time with a large receptive field. A receptive field is defined as the set of data points in the sequence that can be used by the model in its classification of any point. A unique aspect of TCNs is that their receptive field grows exponentially with the number of layers (rather than linearly, like a regular CNN). This means we can use far fewer TCN layers to achieve the same receptive field, giving us fewer parameters and a model that is less prone to overfitting. This large receptive field is obtained by exponentially growing the dilation, a parameter that controls the spacing between elements that are fed into the convolutional layers. By growing the dilation over time, the model is allowed to extract local fine-grained patterns and combine them to find long-range dependencies.

TCNs can be sensitive to hyperparameter choices, such as the dilation factor and receptive field size. They can also be parameter intensive, even though the scaling is exponential a large receptive field still requires many layers. Overall, these are a strong choice and empirically have the highest F1 score (although note, as above, that these are not directly comparable between different model architectures).

Additionally, TCNs (as well as a stack of convolutional layers) need to "chunk" the data into several second long chunks and predict the median label within that chunk. Because a TCN has such a large receptive field, it can be fed additional chunks before and after the "target" chunk that we are trying to predict.

**UNet** The UNet architecture as suggested by Ronneberger et al. (2015), modified for 1D segmentation, uses an encoder-decoder structure with skip connections to keep low-level features while combining them to find time-dependent connections. The UNet gets its name from its distinctive U shape, where we downsample the inputs in the encoder and

then upsample them in the decoder step, forming a U-shape. We add skip connections between levels in the encoder and decoder to pass the low-level information along to the higher level. These architectural choices force the model to learn abstract features and reconstruct them. An example architecture for a UNet is pictured below, in figure 3.

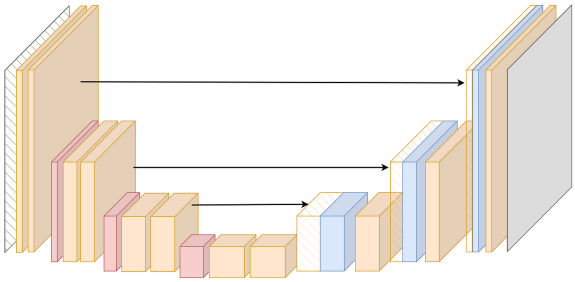

**Figure 3** Diagram of a standard 2D UNet

Another important advantage to a UNet is its ability to output a prediction for every label (rather than the median of a chunk). There are also no chunk boundaries, which has the potential to cause issues with lost information at the chunk boundaries and errors in a chunk that contains a similar number of points of two classes.

The main downside of a UNet is its computational cost, especially with long time series, and there are many potential concerns with loss function and pre- and post-processing. Because it makes dense predictions the barcoding problem is more severe than usual and requires post-processing to fix. This is described in more detail later in section 3.1.1. We believe this model has the potential to be the best among what we have suggested, however the difficulty of designing a successful UNet requires a lot of iteration.

**TCN+STFT** When the TCN is extended to include short-time Fourier transform (STFT) frequency data as additional channels, it can leverage the frequency-domain data that we know is important for mosquitos. However, this introduces greater computational overhead and adds more potential for overfitting.

### 2.2.3 Unsupervised Segmentation

Unsupervised segmentation attempts to divide the recordings into segments according to the waveform type in each segment without making use of training data. Of note is that this approach does not actually classify segments of the waveform as the above methods do and instead returns the location where a change in waveform type likely occurred. Therefore all of the methods that follow would also need to be paired with something like one of the above classifiers. Despite this shortcoming, unsupervised segmentation algorithms have potential to augment the above approaches because they do not depend on having training data and can thus achieve good performance on sparse datasets. Additionally, from a more holistic perspective they more closely mimic the approach taken by humans on the same task, who first look for where transitions between waveform types occur and then classify them accordingly.

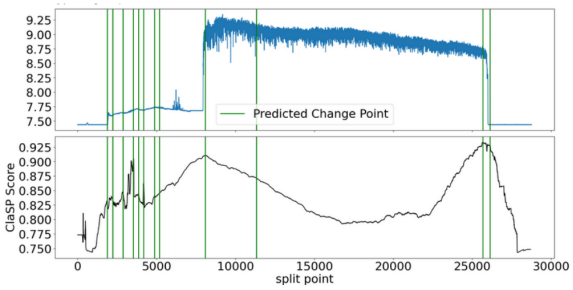


**Figure 4** ClaSP Segmentation of a probe (upper) and the corresponding ClaSP score profile used for determining segmentation locations (lower). Based on conversation with our liaisons, predicted change points correspond well with actual change points (not shown).

**Change point detection** Our first attempt at unsupervised segmentation made use of the python change point detection library Ruptures from Truong et al. (2020). At a high level, this algorithm works by finding the best time to split a recording so as to maximize the likelihood that the data on each side of the segmentation follows a normal distribution, each with a different mean and standard deviation. It then repeats this process on each side of the split until a stopping condition is reached. We used a stopping condition of ten total splits per probe as that was the maximum observed in our training data. Intuitively, this method just finds places where the baseline and amplitude of the input signal changes, which we observed is generally associated with changes in waveform type. In terms of performance, this method did well on observation. However, as it relies on assumptions about the mean and standard deviation being constant within a waveform type which are not met, we believe we could get better segmentation performance with a model that does not make those assumptions.

**ClaSP** The second unsupervised segmentation algorithm we explored was the Classification Score Profile (ClaSP) algorithm from Ermshaus et al. (2023). The main advantage of this algorithm is that it was designed with time series that exhibit periodicity within states in mind, matching what we have observed in EPG data across species in that most waveform types are periodic in nature. Again at a high level, this algorithm works by identifying the time point in the recording at which it is best able to classify whether a window of time points in the recording happened before or after that chosen point. In theory, the time at which it is easiest to tell apart the data that came before and after that point is when the waveform type changes. The data is then split at this point and the algorithm is applied recursively to each side until a given number of splits is reached. Performance for this algorithm was very good. In consultation with our liaisons, we found that it was able to segment recordings at both the waveform type and occasionally sub-type level, although sub-type segmentation is not a priority of this project. An example of a good segmentation along with the corresponding ClaSP score can be seen in Figure 4. While this algorithm has good performance, it of course needs to be paired with another algorithm for determining the waveform type between segmentations. Nevertheless, it shows promise for being a component in our final algorithm.

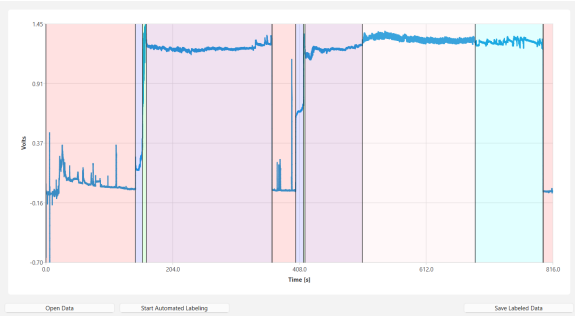


**Figure 5** GUI Prototype

**Hidden Markov Models (HMMs)** HMMs are particularly useful in determining the most accurate assignment of class labels for a time series. They utilize first order Markov chains– a modeling technique that assumes that the probability of each next state only depends on the current state.

Our objective is to uncover hidden states beyond those directly observed in the data and extract new state transitions from the same dataset. So far, a baseline HMM framework has been implemented, leveraging the scikit-learn library to improve state prediction accuracy. This included experimenting with various state configurations, random walk techniques, and HMMs to better capture state durations. Additionally, we developed a state transition probability matrix and implemented a Gaussian HMM with two components that enable the prediction of hidden states.

HMMs are particularly efficient models due to the strong assumptions they make about the structure of the data. Additionally, HMMs can handle inputs of variable length, making them the most flexible generalization of the EPG time-series data. They are also locally learnable, so when a researcher wants to add or remove a label, the model can update itself accordingly. However, since an HMM evaluates one state at a time, it cannot capture dependencies between non-adjacent states. As we continue to develop the HMM, we will keep this consideration in mind.

## 2.3 Graphical User Interface

For any of the above algorithms to be usable by entomologists, a graphical user interface (GUI) for running them and adjusting their outputs is also necessary. With that in mind, we have begun development of such an interface. Progress so far has focused primarily on the window in which users adjust model outputs, which can be seen in Figure 5. Each color corresponds to a different waveform type. The black transition lines can be moved, added, and deleted using mouse gestures. Finally, the data window is capable of panning and zooming. In terms of user interaction, this window constitutes the bulk of the work necessary to make our models usable for researchers.

# 3 Spring Goals

## 3.1 Machine Learning

### 3.1.1 Post-processing

In working with many of our models, we have noticed that their outputs may be generally correct at a high level but they suffer from poor performance in specific categories and do not always produce results that are biologically possible. Therefore, we will look to implement the following post-processing steps to our model outputs to enhance their predictions.

1. We can apply "smoothing" by re-labeling sections based on statistics about their neighbors. For example, if the model predicts one hundredth of a second of waveform as type “A" and the full seconds before and after it as type “B", it would be reasonable to assume that type “A" prediction was incorrect. We believe that applying this reasoning to our data with an automated tool would lead to more stable predictions and would also make it easier for our liaisons to edit the output of these models. If we predict hundreds of transitions, fixing these manually would be much more difficult than using this automated approach.

2. We can use our knowledge of which sequences of waveform types are biologically reasonable to perform a second kind of post-processing. This would consist of creating a rules-based parser that runs over the predicted waveform types and changes them to follow valid transitions.

### 3.1.2 Combining existing models

There are two main ways we could combine the output of these various models.

1. We can take the output of one of the unsupervised models like the HMM or ClaSP-based segmentation model and use a more complex (probably CNN-based) model to predict the waveform type for each segment. We think this approach could improve results because predicting a single waveform type for a segment should be an easier problem than making dense predictions.

2. We can also ensemble the outputs of several prediction models together. By ensemble, we mean taking a weighted average of the predictions from several models to get a single prediction. These ensembles have the advantage of reducing the variance and often bias, which should hopefully have the effect of improving overall performance.

### 3.1.3 Exploring new models

**CRF** The conditional random field (CRF) model, as suggested by Sutton and McCallum (2010), can be thought of as combining the graphical structure of a HMM with the complex non-linear relationships modeled by a neural network. This will allow us to encode our beliefs about the relationships between states without giving up the representative

power of a neural network. This seems like a very natural choice for this kind of problem, since it combines the best parts of a HMM and UNet.

However, this is a less-studied model and there are fewer resources making this a more labor-intensive project and current likely not worth the time investment. As we complete our other modeling goals, we hope to find the time to look into this interesting and unique model.

**Hidden Semi-markov Model** The Diversified Hidden Markov Models (DHMMs) for sequential labeling as outlined by Qiao et al. (2019), can help us ensure reliable labels for predictive applications. The logical next step for developing the HMM approach we presently have are DHMM. DHMMs extend traditional Hidden Markov Models by emphasizing diversity in state transitions, making them particularly suited for complex temporal patterns like those found in EPG signals.

### 3.2 GUI

Our current iteration of the user interface is generally functional aside from its pending connection to a machine learning model. Next semester, we will complete this first GUI prototype by selecting one of the models we have developed this semester and setting it to run upon the press of our "Begin Labeling" button. This will put us in position to test the comprehensive software with a placeholder model while we continue to develop our final model.

Once we have finished our first prototype, we will move fully into the user feedback and iteration stages. We hope to focus on user experience feedback during our site visit in February or March, and our ultimate goal is to ensure that our software provides the tools and ease of use best fit for our audience of scientists (represented during testing by our liaisons). We have already presented our interface and received some suggestions including:

- User preferences for initial zoom, label colors, etc.
- Improvement in comment viewing, iterating on WinDaq's comment side panel system
- Statistical summaries of waveform types including time per waveform type, amplitude, peak count, etc.

We will begin work on these items next semester along with any other features and changes based on feedback from our liaisons.

### 3.3 Publication

We are presented with the opportunity to publish a paper about our research and models. We need to discuss whether or not we will pursue this with our liaisons at the beginning of next semester, and will make way for this progress if we decide to continue with publication.

| Task | Area | Due Date |
|---|---|---|
| Apply a neural network to output from unsupervised segmentation and write a memo summarizing results | Machine Learning | February 3 |
| Apply hidden and semi-hidden markov models to data set and write a memo summarizing results | Machine Learning | February 3 |
| Final decision on whether to pursue publication | Publication | February 3 |
| Develop and test postprocessing filters for model output and write a memo summarizing results | Machine Learning | February 10 |
| Finish cleaning Aedes aegypti dataset and write memo summarizing model performance on this new dataset | Machine Learning | February 10 |
| Create back end interface for use of leading model in GUI | Machine Learning | February 17 |
| Finish prototype of GUI and share with liaisons for feedback | GUI | February 20 |
| Write memo based on literature review of EPG labeling algorithms. Identification of remaining algorithms to try | Machine Learning | February 24 |
| Deadline to receive GUI feedback from liaisons | GUI | February 27 |
| Write memo on progress of implementing above methods | Machine Learning | Mid-March |
| Incorporate feedback from liaisons into GUI and share again for feedback | GUI | Mid-March |
| Write memo on results of latest set of models. Create backend interface for leading model for inclusion in final product | Machine Learning | April |
| Finalize software package | Machine Learning + GUI | April |
| Submit Poster | Clinic | April 12 |
| Submit Final Report | Clinic | May 9 |

**Table 2** Spring Schedule

## 4 Spring Plans

Table 2 summarizes our timeline for each of our main tasks (machine learning and GUI) in the Spring. As our focus is largely on the machine learning and the GUI is nearing a complete state, machine learning tasks are given in more detail. Should the team and our liaisons decide to pursue a publication, we will integrate that in to our schedule as well.

## 5 Reflections

We are happy to report that we have completed our prototype GUI in expected time. We also expected to have chosen a machine learning method by the end of this Fall, to be honed in on for improvements during the Spring. We have found, however, that the solution

space for this problem is quite large. While we have pursued many potential options, we think it would be prudent to continue exploring machine learning methods in the Spring semester. We have found during our experience so far that the greatest points toward model performance improvements come from implementing totally different machine learning frameworks. Adjustments to these are easy to complete, but committing focus to them too early will provide less substantial progress than continuing to build our understanding of a wide variety of machine learning methods. We are building our GUI in such a way that we can plug in any machine learning model for use during our site visit, even if we have not perfected our final approach.

# 6 Conclusion

The high-level goal of our project was to produce an automated system to label arthropod (with mosquitoes as the primary focus and ticks and biting midges as a secondary focus) EPG voltage data with waveform names. To this end, we have made substantial progress, having targeted and implemented 5 different machine learning models.

We have made great progress towards our goal to create a user interface for researchers without computer science experience to use this technology on their own data. We have produced a functional GUI, and we will continue to build on this going forward to affirm that our software is both intuitive and accessible to users.

## References

Cooper, Anastasia M. W., Samuel B. Jameson, Victoria Pickens, Cameron Osborne, Elaine A. Backus, Kristopher Silver, and Dana N. Mitzel. 2024. An electropenetrography waveform library for the probing and ingestion behaviors of culex tarsalis on human hands. *Insect Science* 31(4):1165–1186. doi:https://doi.org/10.1111/1744-7917.13292. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1744-7917.13292. https://onlinelibrary.wiley.com/doi/pdf/10.1111/1744-7917.13292.

Ermshaus, Arik, Patrick Schäfer, and Ulf Leser. 2023. Clasp: parameter-free time series segmentation. *Data Mining and Knowledge Discovery* 37(3):1262–1300. doi:10.1007/s10618-023-00923-x. URL http://dx.doi.org/10.1007/s10618-023-00923-x.

Guo, Chuan, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. URL https://arxiv.org/abs/1706.04599. 1706.04599.

Lea, Colin, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. 2016. Temporal convolutional networks for action segmentation and detection. URL https://arxiv.org/abs/1611.05267. 1611.05267.

Perry, Sam. 2024. windaq3. https://github.com/sdp8483/windaq3.

Qiao, Maoying, Wei Bian, Richard Yida Xu, and Dacheng Tao. 2019. Diversified hidden markov models for sequential labeling. *CoRR* abs/1904.03170. URL http://arxiv.org/abs/1904.03170. 1904.03170.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. URL https://arxiv.org/abs/1505.04597. 1505.04597.

Sutton, Charles, and Andrew McCallum. 2010. An introduction to conditional random fields. URL https://arxiv.org/abs/1011.4088. 1011.4088.

Truong, Charles, Laurent Oudre, and Nicolas Vayatis. 2020. Selective review of offline change point detection methods. *Signal Processing* 167:107,299. doi:https://doi.org/10.1016/j.sigpro.2019.107299. URL https://www.sciencedirect.com/science/article/pii/S0165168419303494.

Willett, Denis S., Justin George, Nora S. Willett, Lukasz L. Stelinski, and Stephen L. Lapointe. 2016. Machine learning for characterization of insect vector feeding. *PLOS Computational Biology* 12(11):e1005,158. doi:10.1371/journal.pcbi.1005158. URL https://app.dimensions.ai/details/publication/pub.1014451934.