***Abstract***

***Problem***

Monitoring and forecasting the health of our lakes grows more important as global temperatures rise, local ecologies becomes more unpredictable, and environmental water protections are relaxed (Stafford et al, 2013). The ecology of these aquatic ecosystems is heavily dependent on the composition and volume of nutrients flowing into and out of them. With an overloading of nutrients, particularly phosphorus and nitrogen compounds, the state of an ecosystem can become unstable. The structure of the food web, balance of nutrients, and population of each species can shift rapidly in what is called a regime shift, a transition from one ecological equilibrium to another. One such transition in lake ecosystems is from an oligotrophic state with low-biomass to a eutrophic state with high plant and phytoplankton biomass (Carpenter, 2003). This transition is what occurs in cyanobacterial algal blooms that produce toxins and endanger humans, animals, and the long-term health of the ecosystem (Kidwell, 2015). The ability to predict if and when these regime shifts will occur would prove a useful tool for mitigating harmful ecological transitions like these (Ontario Ministry of the Environment and Climate Change, 2017).

While they occur on the relatively short timescale of a few months, regime shifts in lake ecosystems are accompanied by pretransition indicators that can act like early-warning systems in ecological timeseries (Carpenter et al, 2011). There are three theoretical features that indicate an increased likelihood of transition: critical slowing down, flickering, and frequency multimodality. In the first of these, the system equilibrates from a perturbation more slowly because an underlying instability in the ecosystem. Existing techniques use an increase in temporal autocorrelation to measure this feature. Next, in flickering, an ecosystem transiently “flips” into another regime because of stochastic fluctuations or patchiness This is marked by an increase in both autocorrelation and signal variance. Lastly, frequency multimodality is found when characteristic oscillations of other regimes begin forming underneath the current regime (Scheffer et al, 2012). This requires spectral analysis. Autocorrelative methods to detect trends in multimodality have tended to produce false positives, weakening their predictive power (Andersen et al, 2008; Hsieh, 2007).

These early warning features and their corresponding metrics are insufficient to provide a comprehensive understanding of an ecological regime and its stability. There is currently no single metric that reliably evaluates the presence of all three pretransition characteristics. Additionally, measures like autocorrelation and variance are calculated in a fixed-length moving window across the timeseries; they cannot measure across the diverse timescales characteristic to aquatic ecology (Cazelles et al, 2008).

We have deficient statistical methods for predicting how aquatic ecosystems will progress at a point when climate change, anthropogenic over-enrichment, and hydrological modifications threaten the health and stability of our lakes and waterways (Burford et al, 2019). Aquatic ecosystems that are most threatened are also those that provide the most economic and social benefit. Harmful regime shifts like algal blooms can kill local aquatic life, endanger the health of residents and tourists, and cost millions to the economy (Wilson et al, 2018). Controlled experiments by Pace et al (2017) have shown that these harmful algal blooms can be prevented if early-warning signs are detected and acted on. This remains impossible if we cannot predict real-world regime shifts.

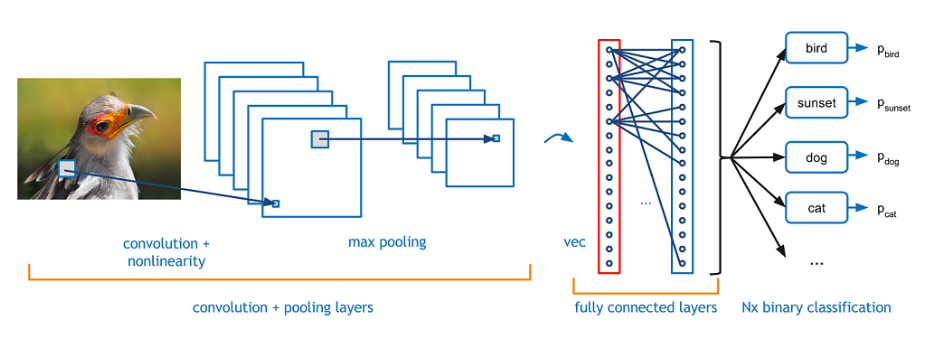
***Background***

Key terms: Convolutional Neural Network, Wavelet Analysis, Regime, Regime Shift

**Convolutional neural networks** (CNNs) are powerful tools for image analysis and feature identification. They work by applying layers of interconnected filters and operators to an image until the output is an encoded piece of information about that image, like a classification.

A CNN is composed of multiple sub-layers convolution and pooling. These sublayers typically include a convolutional layer, a thresholding layer, and a pooling layer. The first layer applies a windowed filter or *kernel* to the input through element-wise multiplications, a convolution. The second layer applies nonlinearity, setting a threshold for passing the value into the next layer. The third layer is a pooling layer which decreases the size and resolution of its input.

These convolution and pooling layers are stacked on top of each other to increase the complexity of feature recognition. The first layer may recognize a simple line, the second layer groups of lines, and the third a collection of lines that makes the beak of a bird (see Figure 1). Filters at each layer of convolution are trained to identify their own particular patterns.

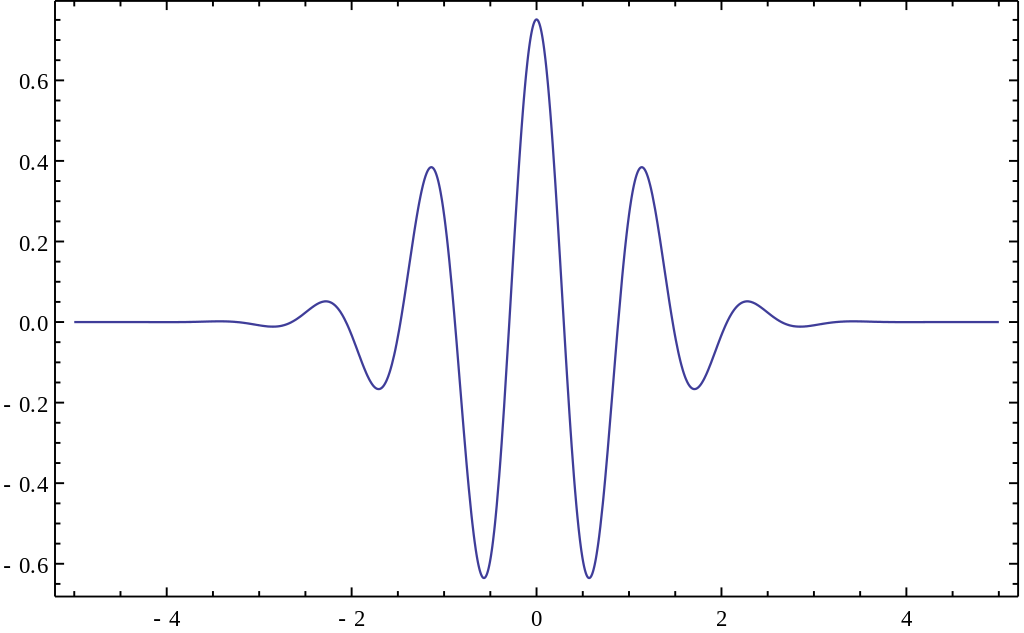
**Figure 1** *– Structure of a Convolutional Neural Network – Image from Adit Deshpande (2016)*

The output from an arbitrary number of convolutional layers is encoded into classifications or feature identification by fully connected layers (see Figure 1). Fully connected layers help translate the convolutional output into easier understand classifications.

Labeled data, which associate images with their classification, are required to train a convolutional network. During training, filters and layer connections are tuned to reduce the error between network output and the data label. (Desphande, 2016).

Once a network is trained, new label data from the same dataset is passed through the network to test its accuracy. This testing allows the network to be validated as effective, since it has not been able to train on the testing data.

**Wavelet Analysis** is a technique for spectral analysis of a timeseries, through which frequency-based features can be extracted across time. A wavelet transformation is similar to a Fourier transformation, which can be thought of as a convolution of sine waves with a timeseries. The waves used are moved across the timeseries and stretched to different frequencies to produce a two-dimensional transformation. Wavelet analysis is very similar, but the waves it uses are more localized wave than a sine wave (see Image 2).



**Figure 2** *– Morlet wavelet – Image from Jon McLoone (2012)*

Wavelets, being more time-localized, provide a clearer picture of frequency phenomena at a more precise point in time. Wavelet analysis can be used to characterize autocorrelation, variance, and multimodality of a timeseries, all of which are indicators of regime shifts (Rouyer et al, 2008). The technique also allows for non-stationary timeseries to be analyzed while maintaining evaluation at large timescales (Torrence & Compo, 1998).

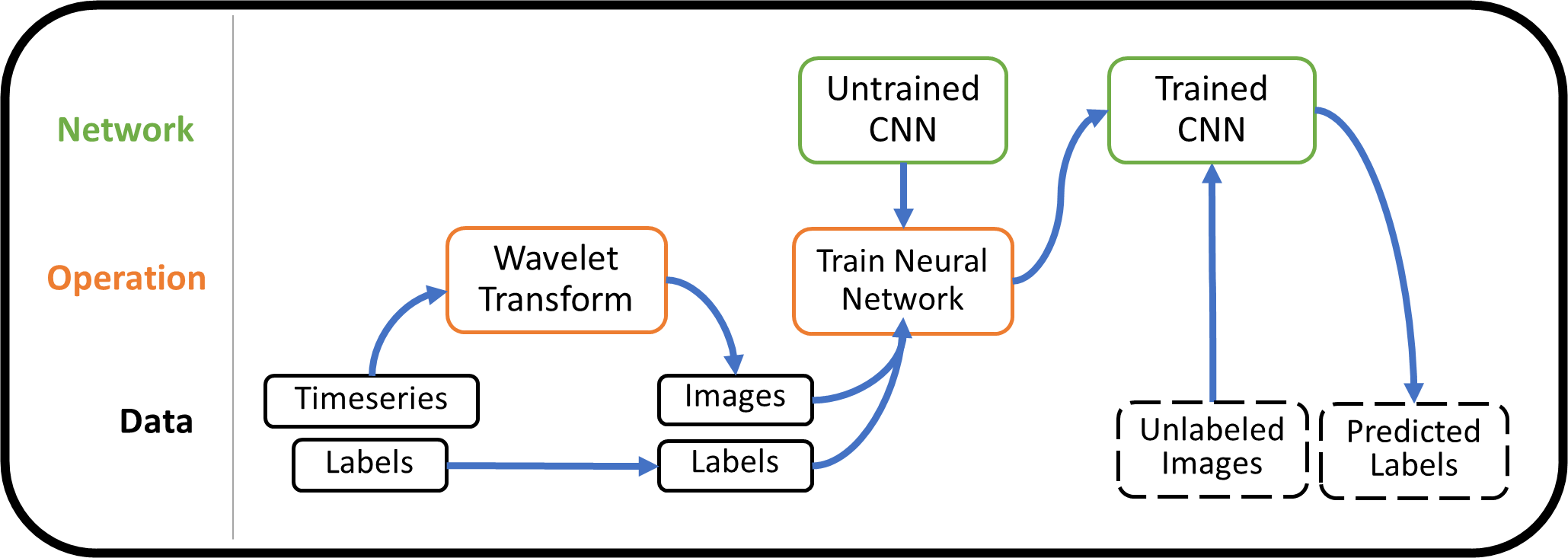
The fundamental transformation in wavelet analysis is from a one-dimensional timeseries of a variable over time to a two-dimensional image of wavelet power across time and frequency. This transformation allows for a one-variable timeseries to be analyzed with a convolutional neural network setup to process one-channel images.

***Approach***

Our timeseries analysis tool employs a wavelet transform convolutional neural network (wtCNN). Timeseries are analyze using wavelet transforms to produce images. These images are then passed through a convolutional neural network for classification. This composite approach allows for the power of CNNs to be applied to timeseries by amplifying important characteristics with wavelet transforms.

*Core Functionality*

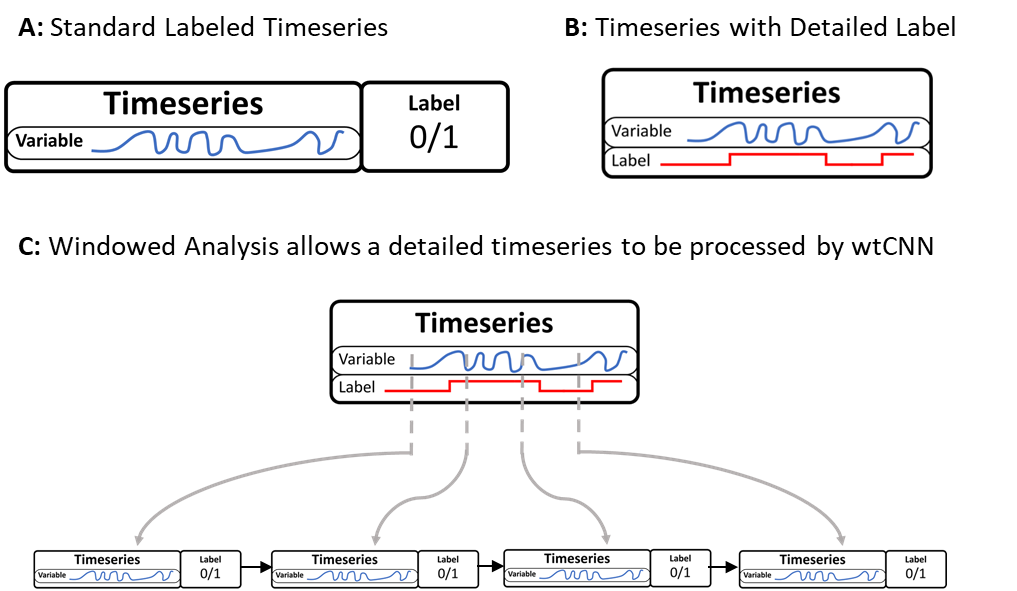
To create a trained CNN that can identify features in a timeseries, we start with a labeled timeseries and apply a wavelet transformation to produce images, still with their labels. We then use those labeled images to train a convolutional neural network.



**Figure 3** – Core data flow of our package

A trained CNN is the pivotal tool for monitoring ecological timeseries in the real world. Timeseries without labelled features can be wavelet transformed and put through the trained CNN to give predictions of feature presence. These lead to image classification according to how the network was originally trained.

*Windowed Analysis*



***Figure 4*** – Input Timeseries and Labeling

To process timeseries and their labels, we employ two approaches. The first is to classify the entire timeseries with a single label (see Figure 4A). This label can indicate a particular domain behavior or the presence of a regime shift, as long as there is a one-to-one relationship between the sets of labels and timeseries. In our package, labels are binary.

The second approach is what we called a **windowed analysis**, which breaks down the timeseries into smaller segments of time before processing them. After breaking them down, the timeseries subsets are sent through the same wtCNN process as the whole timeseries. In order to appropriately label the windows, each point of the timeseries must have a corresponding label before windowed analysis begins. This means that the labelled data must have a *variable* timeseries as well as a *label timeseries* of the same length (see Figure 4B). Each window is then output with a label corresponding to the majority key within its range of values. With each window from a timeseries labeled with a single key, they can be processed in sequence (see Figure 4C).

Why employ windowed analysis? Windowed analysis allows a non-stationary timeseries to be analyzed as data comes in, making it ideal for real-world application. By analyzing a smaller window of timeseries behavior, the wtCNN output pertains to a more localized region. This allows for the larger-scale timeseries classification to lead into **timeseries** **forecasting**.

*Outlier Detection and Forecasting*

In analyzing the most recent timeseries window, model output can be compared to outputs from all previous windows, allowing us to evaluate if the current window is an outlier. If the CNN used was trained to classify domains or regime shifts, an outlier sample is a useful indicator that the system is moving out of the previous domain space.

* Talk about event2D