**CNN Classification Algorithm Steps**

The following steps are to classify a single ten second recording. The recording is split into 2 second recordings with 50% overlap (so 10 total windows). We will end up storing a binary decision for each of these 2-second windows (abnormal/normal).

Preprocessing

1. Separate incoming signal into 2-second recordings with 1sec (50%) percent overlap. The input vector should have a sampling rate of 8KHz and should be the denoised signal.
2. Input vector is **X**=[16000x1]. It should be normalized to values between [-1,1] by dividing by max of **X**

and zero centered by subtracting by the mean of **X**

1. Calculate the **auditory spectrogram** representation with output matrix of **Y**=[250x128] samples: 250 8msec time steps (x axis) and 128 frequency bands (y axis). The filter bank and coefficients model the cochlear response.
   1. **COCH\_x.txt** and **p\_order.txt** filescontain the filter coefficients needed to create the auditory spectrogram. 
      1. **p\_order.txt** file gives you the order of each of the 128 filters as well as a starting 129th filter in the last column.
      2. **COCH\_B.txt** file gives the 8-tap coefficients **B** in each column for each of the 128 filters as well as a starting 129th filter in the last column.
      3. **COCH\_A.txt** file gives the 8-tap coefficients **A** in each column for each of the 128 filters as well as a starting 129th filter in the last column.
   2. An ARMA filter is designed as an IIR filter with coefficients **B** and **A** from above.
   3. Starting with the last filter (the 129th filter), pass **X** via the ARMA filter with the last columns of **B** and **A** with filter order 7 (129th column of **p**)**.** Store the filtered signal as . This value is used for as the starting point for the lateral inhibitory network in the cochlear model.
   4. Move to filter 128. Pass **X** via the ARMA filter with the 128th columns of **B** and **A** with filter order in the 128th column of **p**. Store filtered signal as .
   5. Reduce the filtered signal by subtracting hair cell signal from filtered signal:

which is called lateral inhibitory masking.

* 1. Store as the new.
  2. Half-wave-rectify i.e. set any negative values to 0. Keep half-wave rectified signal as
  3. Apply a fixed 2-Tap Moving Average filter to half-wave-rectified ; B =1, A coefficients store in **moving\_average\_filter\_coeffs.txt.**
  4. Down sample to 250 total coefficients (take a sample every 8ms, every 64 samples). This is all called a leaky integration process.
  5. Store the 250-point vector in the 128th column of **Y**.
  6. Repeat steps **d)**-**i)** for filters 127, 126, down to filter 1.
  7. Output should create **Y**=[250x128].

CNN Architecture

1. The auditory spectrogram for each 2 second segment is then used as the “image” input to the convolutional neural network. The network architecture is currently made up of 30 layers. The layers are described in **CNN\_layers.txt**.
2. The weights and biases for all of the filters in each channel are in the **Weights and Biases by Layer.zip** folder. In this folder, there are 7 layers that require weights and biases (the convolutional layers and the fully connected layer.) All filter sizes are 3x3. By layer:
   1. Layer 2- Convolutional Layer [3x3x1x4]
      1. 1 Channels
      2. 4 Filters
   2. Layer 6- Convolutional Layer [3x3x4x8]
      1. 4 Channels
      2. 8 Filters
   3. Layer 11- Convolutional Layer [3x3x8x16]
      1. 8 Channels
      2. 16 Filters
   4. Layer 15- Convolutional Layer [3x3x16x16]
      1. 16 Channels
      2. 16 Filters
   5. Layer 20- Convolutional Layer [3x3x16x16]
      1. 16 Channels
      2. 16 Filters
   6. Layer 24- Convolutional Layer [3x3x16x16]
      1. 16 Channels
      2. 16 Filters
   7. Layer 28- Fully Connected Layer [2x448]
      1. Input Size 448
      2. Output Size 2

Final Decision

1. If any of the 2-second recordings were classified as abnormal, then classify the whole recording as abnormal. (This might eventually change in the next few weeks and we might do some sort of scoring based on each window to give a better over all decision or make the decision from one window affect the input to the next. This should be pretty easy to implement if everything else is completed.)