# This week

* Sequence Decimating Network - Description document
* Sequence Decimating Network - CPU implementation
* Numerical gradient Python slicing surprise!

def SetWeightVector(self, raW):

# Keep this or you'll be sorry!

raW = numpy.copy(raW)

iBase = 0

for iLayer in range(self.iLayers):

self.oaLayers[iLayer].raaW = numpy.reshape(raW[iBase:iBase+self.oaLayers[iLayer].raaW.size],self.oaLayers[iLayer].raaW.shape)

iBase += self.oaLayers[iLayer].raaW.size

self.oaLayers[iLayer].raB = numpy.reshape(raW[iBase:iBase+self.oaLayers[iLayer].raB.size], self.oaLayers[iLayer].raB.shape)

iBase += self.oaLayers[iLayer].raB.size

# Plans

* Clean up Sequence Decimating Network code & implement slight API modification (output patterns)
* Complete Sequence Decimating Network GPU Implementation
* Complete the workflow for submission preparation

# Pseudocode

Kaggle Submission Process

preprocess data(target sample rate)

for each individual...

split training and validation subsets

preprocess the raw data files(target sample rate)

split training and validation subsets(training fraction)

load all sample file names

shuffle them

save shuffled file names to a csv

preprocess raw data

for each sample file

load mat file

detrend

lowpass filter & decimate (using numpy resample)

save as pickle file

train network(validation fraction, decimation ratios, layer sizes)

for each individual...

train a network(validation fraction)

assess network(validation fraction)

compute test sample predictions()

combine test sample predictions

upload test sample predictions

assess network()

apply network(training samples)

apply network(validation samples)

compute roc curve(training sample)

compute roc curve(validation sample)

plot roc curves

compute area under roc curves

compute roc curve(samples)

load validation samples and class targets

compute validation predictions

sort predictions & class targets by prediction

for each prediction score compute false positive rate and false negative rate

compute validation auc(training fraction)

plot curve connecting [0,0] [fp0,fn0], [fp1,fn1],... [1,1]

compute sum of areas under each line segment (fp1+fp0)\*(fn1-fn0)/2

compute test sample predictions(max files)

while test files remain

load up to max files test files

for each file

for each offset

add a row

compute all activations

for each file

average all offsets to form prediction