EMG Classification Using Recurrent Systems

Connor Amorin, Gabriel P. Andrade, Chris Brissette, Matthew Gagnon, Brandon Iles, Jimmy Smith, and Lance Wrobel

University of Massachusetts Amherst

April 30, 2018

Project Overview

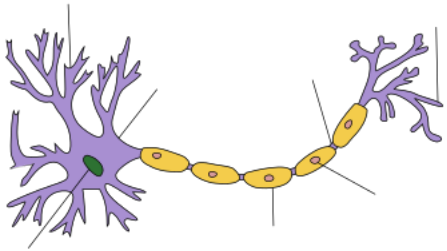
Motivations: Our aim is to compare different methods for classifying EMG data. Performing well on this task is important for many application domains (e.g. prosthetics, remote control of robots), but the current state of the art leaves much to be desired. In what follows we recreate the methodology that achieves this state of the art performance and we explore new methods which might better leverage the time series nature of this data.

Project Overview

- EMG & Methods from the Literature
  - Biological Neurons & Electromyography
  - Data Sets Used
  - Feature Extraction and Popular Classification Methods
- Recurrent Neural Networks
  - Brief Overview to Neural Nets
  - Explanation of our Networks
  - Results & Interpretation
- Reservoir Computing Paradigm
  - Introduction and Brief Overview
  - Explanation of our Networks
  - Results & Interpretation
- Conclusion & Summary of Results

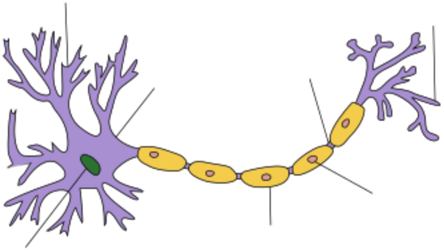
Biological Background: Neurons

- In the past century, computing models have arisen that attempt to analyze and emulate neural activity.
- Electrical activity in neurons is a result of positively and negatively charged ions fluctuating during signal transmission. These fluctuations can be recorded with Electromyography (EMG).



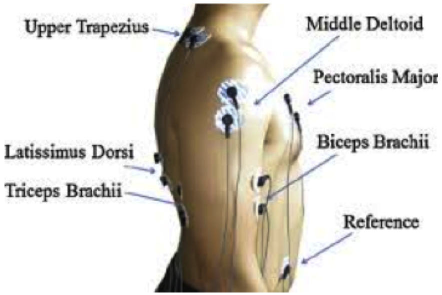
Biological Background: Neurons

- Upon reaching the end of a neuron, the signal will propagate to bordering neurons, creating a unique path for each area of the body that we can track and measure.
- Paths of motor neurons are the most straightforward in the nervous system, and a good candidate for computer analysis.

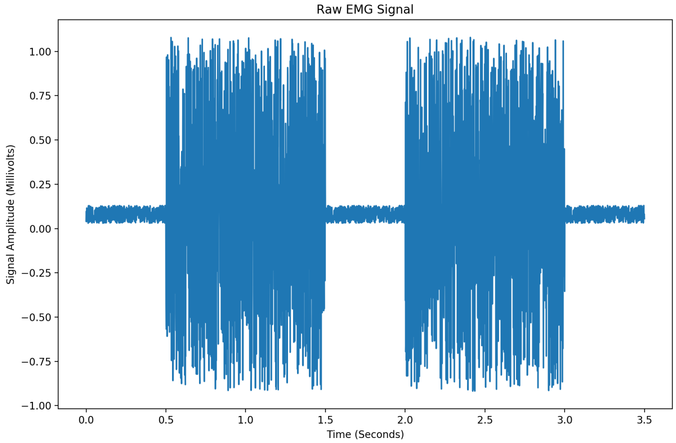


What are EMG Signals?

- Electromyography, better known as EMG, is a technique for evaluating and recording the electrical activity produced by motor neurons in skeletal muscles.
- EMG signals can be measured either by attaching electrodes directly to the skin (known as Surface EMG) or by inserting needle electrodes directly into the muscle (known as Intramuscular EMG).



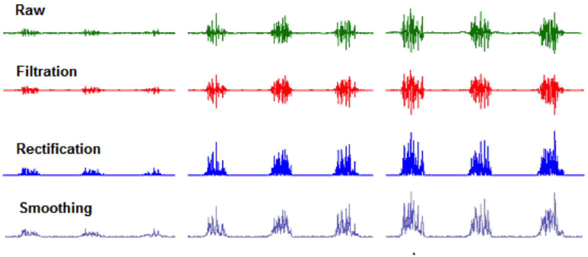
A Visual of a Noisy Raw EMG Signal



- Raw EMG signals are contaminated by noise from electronics and external sources. Inner muscles can also be disrupted by the EMG signals of exterior muscles.

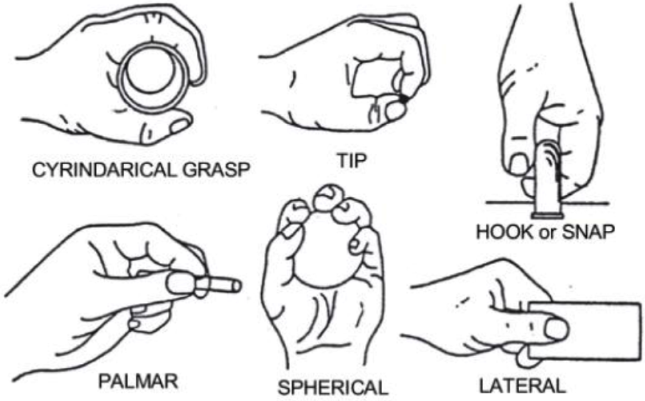
Preprocessing of EMG Signals

- There are preprocessing steps that can be taken to remove some noise and also better prepare an EMG signal for classification purposes.
  - Filtering the signal.
    - ★ Butterworth Filter: $H = 1/\sqrt{1 + \left(w/w_c\right)^{2n}}$
  - Rectification of the signal.
  - Smoothing of the signal.
    - ★ Moving Average: $s_i = \frac{1}{n}\sum_{j=1}^{i+n-1} a_j$ given sequence $(a_i)_{i=1}^{n}$.
  - Segmentation of the signal.



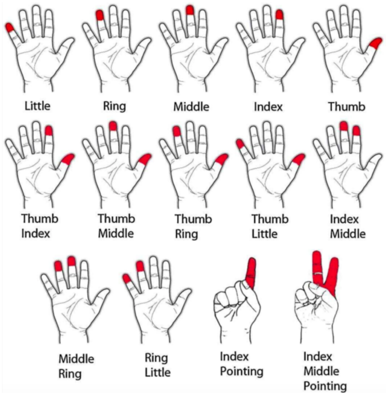
The First Dataset: Hand-Movement Data

- The first dataset we are using is hand-movement data from the UCI Machine Learning Repository. This dataset contains EMG signals of 5 subjects (3 female, 2 male) performing 6 basic hand movements for 30 trials each 6 seconds long.



The Second Dataset: Digit Data

- The second dataset we are using is finger-movement data from Dr. Rami Khushaba of the University of Technology, Sydney. This dataset contains EMG signals of 8 subjects (6 male, 2 female) performing a variety of finger movements such as flexion of individual fingers and pinching of the thumb with different fingers, for three trials each movement, 20 seconds each trial.



Feature Extraction on EMG Signals

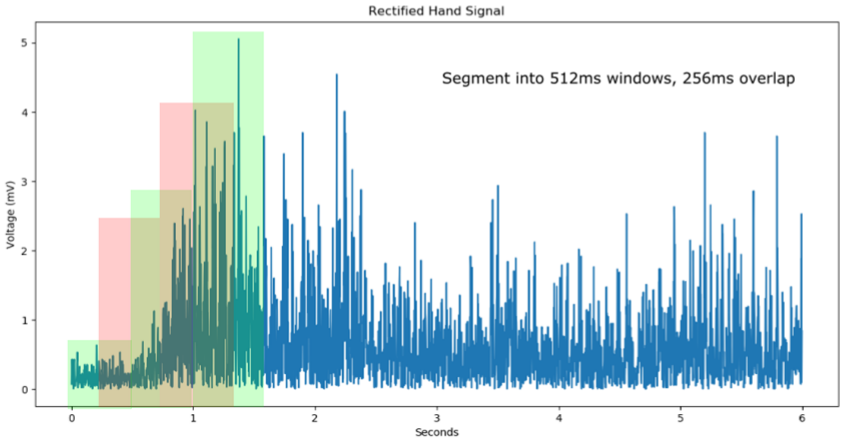
- EMG signals can be cumbersome to work with in their raw form, so extracting features can allow us to identify underlying information as well as reduce dimension for classification
- EMG signals are non-stationary time series and so their frequency and amplitude can change through time; so features are calculated on different parts of each signal called windows which are either adjacent or overlapping
- A set of features extracted from a signal provides a compact representation of the signal

Some EMG Features we Used

Name	Domain	Formula
Root Mean Square	Time	$RMS = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}$
Mean Absolute Value	Time	$MAV = \frac{1}{n}\sum_{i=1}^{n} \lvert x_i \rvert$
AR Coefficients $(\beta_1, ..., \beta_p)$	Frequency	$X_t = c + \sum_{i=1}^{p} \beta_i X_{t-i} + \epsilon_t$
Waveform Length	Time	$WL = \frac{1}{n}\sum_{i=2}^{n} \lvert x_i - x_{i-1} \rvert$
Wilson Amplitude	Time	$WAMP = \sum_{i=1}^{n-1} f(\lvert x_{i+1} - x_i \rvert)$, $f(x) = \begin{cases} 1 & x \geq threshold \\ 0 & otherwise \end{cases}$

Hand and Digit Data Processing and Feature Extraction

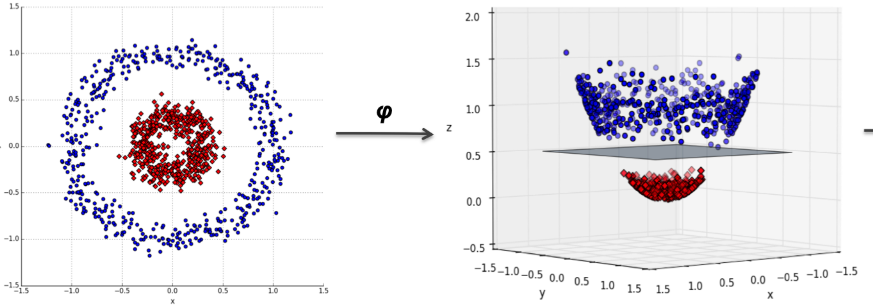
- We rectified the signals and segmented as follows:
  - Hand Data: 512ms windows, 256 ms overlap between windows
  - Digit Data: 128ms windows, 64 ms overlap between windows
- Features were extracted on each segment and appended together afterwords to form the feature vector
- Several different combinations of time-domain features were used along with the auto-regressive model coefficients.


Rectified Hand Signal
Segment into 512ms windows, 256ms overlap
Voltage (mV)
Seconds
0
1
2
3
4
5
6

Support Vector Machines

- SVMs split the feature space of the training data with hyperplanes which maximize the margin between the two closest points on either side of the hyperplane

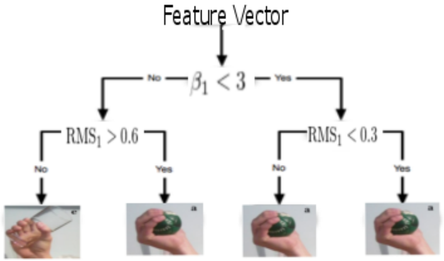
- If data is not linearly separable, we can use a kernel function ϕ that maps data to a higher-dimensional space where separation can be done



Classification Trees: CART Algorithm

- CART is a greedy algorithm which splits the feature space of the training data recursively in order to create a decision tree which can classify testing data points
- Gini Impurity $I_G(p) = \sum_{i=1}^{N}(p_i(1-p_i)) = 1 - \sum_{i=1}^{N} p_i^2$ is what defines a best split at each node (N classes, p_i is the fraction of items labeled with class i in the node p)

Simplified Tree For Hand Data



Random Forest and Extra Trees

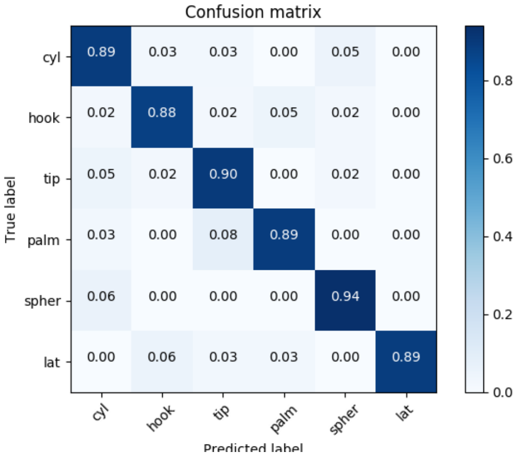
- These methods can be used to prevent overfitting of classification trees as well as reduce variance
- Tree Bagging (Bootstrap Aggregating): Take n training samples with replacement for the training data and train a classification tree on each sample; then an observation falls in the class predicted by a majority of the trees
- Random Forest: Uses tree bagging but selects a random sample of features at each split in the trees.
- Extra-Trees (extremely randomized trees): Similar to random forest but random values from the training set range of each feature are selected for the cut points

Some Hand Data Results

Feature Set	Classifier	Accuracy Random at 16.7%
AR(6)+RMS	SVM, rbf kernel	50% +/- 4%
	QDA	74% +/- 3%
	Random Forest	91% +/- 3%
	Extra-Trees	92% +/- 3%
	Gradient Boosting	94% +/- 3%
AR(11)+MAV	Random Forest	82% +/- 4%
	Extra-Trees	84%+/- 3%
MAV + RMS + ZC + SSC + WL + WAMP + AR(11)	Random Forest	89% +/- 2%
	RF + PCA(10)	70% +/- 7%

Note: 8-fold cross-validation was used for each classifier

Hand Data Confusion Matrix For Random Forest



True \ Predicted	cyl	hook	tip	palm	spher	lat
cyl	0.89	0.03	0.03	0.00	0.05	0.00
hook	0.02	0.88	0.02	0.05	0.02	0.00
tip	0.05	0.02	0.90	0.00	0.02	0.00
palm	0.03	0.00	0.08	0.89	0.00	0.00
spher	0.06	0.00	0.00	0.00	0.94	0.00
lat	0.00	0.06	0.03	0.03	0.00	0.89

Here, random sampling of 75% training data, 25% testing data

Some Digit Data Results

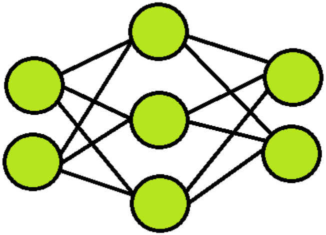
Feature Set	Classifier	Accuracy Random at 6.7%
AR(11)+RMS	QDA	20.2% +/- 7%
	SVM, rbf kernal	23.8% +/- 6%
	Random Forest	37.7% +/- 7%
	Extra-Trees	39.2% +/- 6%
AR(6)+MAV+WAMP	QDA	21% +/- 6%
	Random Forest	38.6% +/- 7%
	Extra-Trees	43.8% +/- 7%

Note: Only first five seconds of each movement was used (20000 signal points). 8-fold cross-validation was used for each classifier.

Neural Network Overview

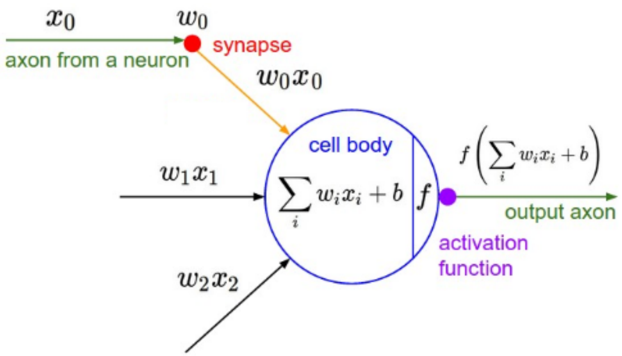
- Neural Networks are function approximators loosely based on connections in the brain.
- Commonly used for applications with high dimensionality and sample size (Computer Vision, Natural Language Processing, Time Series Classification and Forecasting.)
- Recent success due to improvements in hardware and larger datasets. (GPUs, crowdsourced labeling, optimizations in training)

Parts of a Neural Network



- Three main components:
  - Input Layer
  - Hidden Layers
  - Output Layer
- Each layer consists of neurons which do some nonlinear function, e.g. sigmoid, tanh, softmax, ReLU, etc.
- These functions are connected by synapses which do simple multiplication.

Detailed Neuron



- x_i is the input
- w_i are the learned weights
- b the learned bias
- f is the nonlinear activation function (ReLU, tanh, sigmoid, etc.)

Training - Cross Entropy

$$H(p,q) = -\sum_x p(x)\,\log q(x).$$

- We take the score (output) of the network and define a cost function.
- In our case we use cross entropy which calculates the average number of bits needed to identify an event from our set.
- This is closely related to the Kullback-Leibler Divergence between our networks probability distribution and our underlying probability distribution.

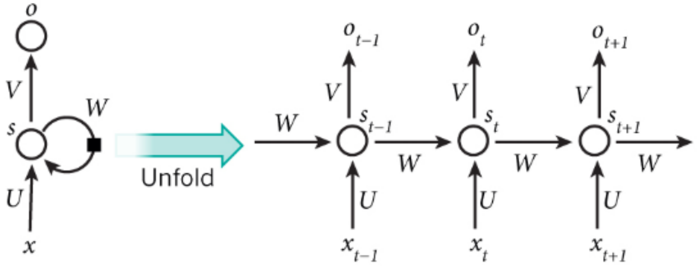
$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i)\,\log\frac{P(i)}{Q(i)}$$

Training - Gradient Descent

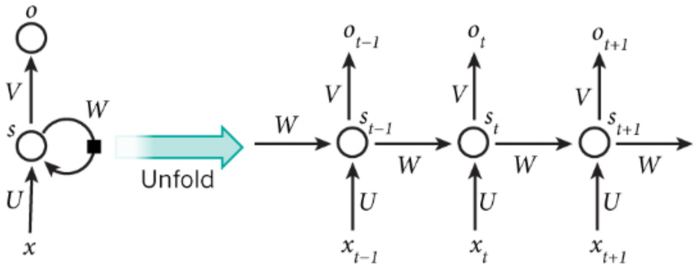
- While there are many methods such as stochastic gradient descent, conjugate gradient descent, and hessian free optimization they all are based off of the well known idea of basic gradient descent.
- During training, costs are calculated over many different inputs. This is known as the forward pass.
- Then we can then figure out the derivative of this with respect to our weights via a method known as back-propagation.

Why Recurrent Neural Networks?

- Feed forward networks and convolutional networks are poor at identifying correlation between events in time.
- To overcome this we introduce memory to the network, and this is an RNN.
- The idea is to feed the output from the hidden layers back into the network so that previous calculations persist in some way.



Problems with training RNNs



- As with all neural networks the issue of vanishing gradient is a plague.
- However, now with an RNN it is far more prominent since an RNN is essentially a very deep feed forward network with connections in time.
- This allows for far more possibilities of near zero gradient hindering optimization.

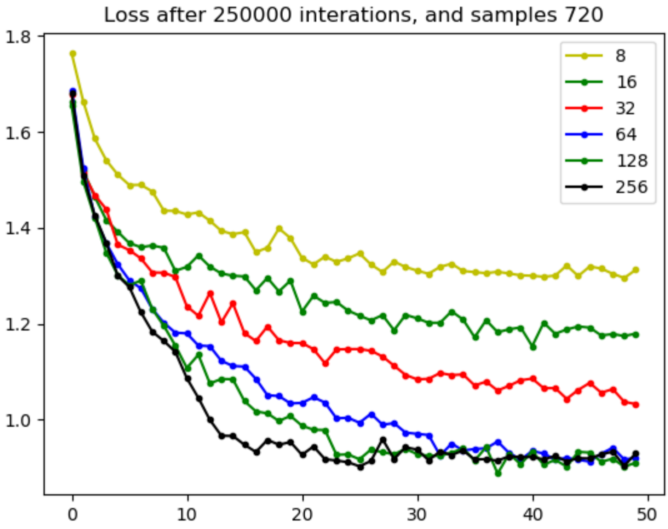
Our Recurrent Networks

- Basic RNN (Used on Hand Data, built in Pytorch)
    - Uses a simple network, that takes in the two channel data from the Hand Data set, after AR Coefficient feature extraction.
    - Takes in a variable input size, and uses CrossEntropy loss and Stochastic Gradient Descent (SGD)
    - Trains on random sets of data for 250,000 iterations
- Basic LSTM (Used on Digit Data, built in Keras)
    - Two hidden layer network trained on raw Digit Data signals.
    - Takes in samples of 1000, uses Cross Entropy loss and SGD

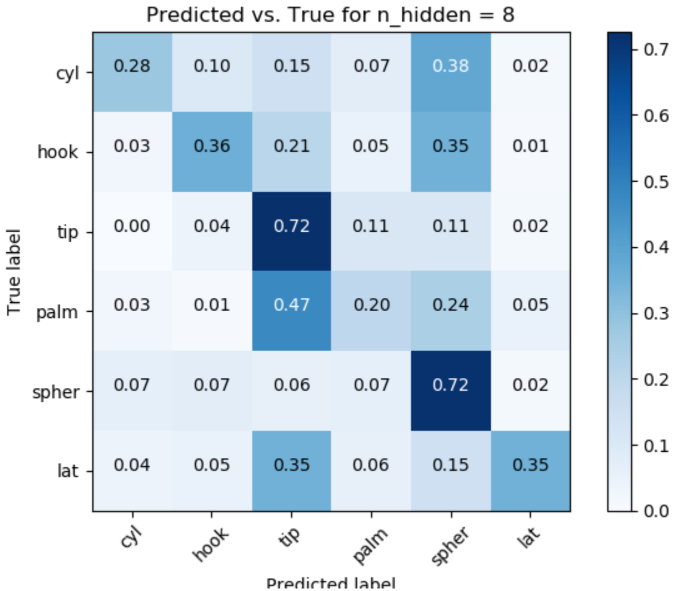
Simple Recurrent Network: Results

- Hidden Layer Size
  - By changing this size for different training sessions, we can see that there is a heavier loss decrease over the increasing size of the hidden layer.
  - This increase in size gives the model a bigger matrix for linear operations in the network. This corresponds to a longer training time, but also a possibility of more important information being captured in the hidden layer.

Loss

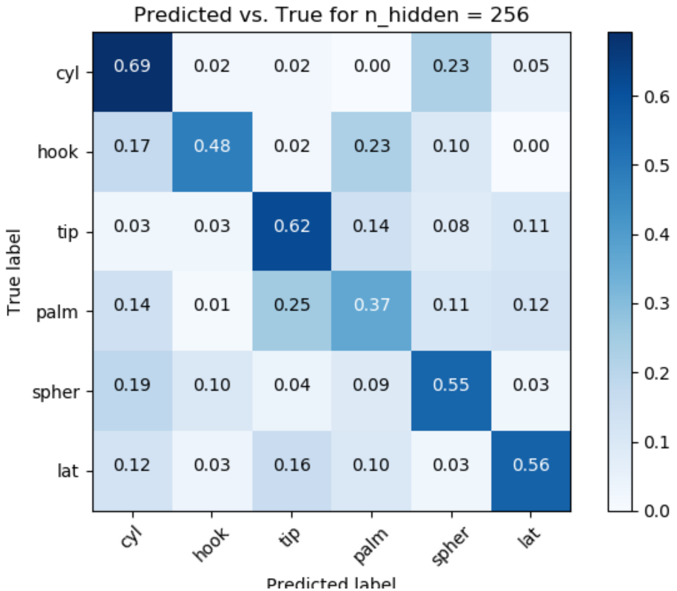

Loss after 250000 interations, and samples 720
1.8
1.6
1.4
1.2
1.0
0
10
20
30
40
50
8
16
32
64
128
256

Accuracy Over Changes



True \ Predicted	cyl	hook	tip	palm	spher	lat
cyl	0.28	0.10	0.15	0.07	0.38	0.02
hook	0.03	0.36	0.21	0.05	0.35	0.01
tip	0.00	0.04	0.72	0.11	0.11	0.02
palm	0.03	0.01	0.47	0.20	0.24	0.05
spher	0.07	0.07	0.06	0.07	0.72	0.02
lat	0.04	0.05	0.35	0.06	0.15	0.35

Accuracy Over Changes



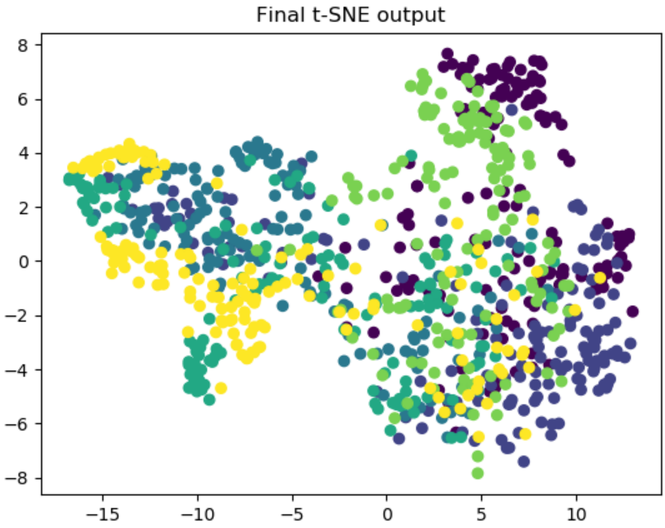
	cyl	hook	tip	palm	spher	lat
cyl	0.69	0.02	0.02	0.00	0.23	0.05
hook	0.17	0.48	0.02	0.23	0.10	0.00
tip	0.03	0.03	0.62	0.14	0.08	0.11
palm	0.14	0.01	0.25	0.37	0.11	0.12
spher	0.19	0.10	0.04	0.09	0.55	0.03
lat	0.12	0.03	0.16	0.10	0.03	0.56

Visualizing with t-SNE

- t-SNE (t-Distributed Stochastic Neighbor Embedding)
  - The purpose of this technique is to take a set of points in n-dimensional space and visually represent them in 2 or 3 dimensions, while still retaining the separation information from the higher dimensions.
  - Uses the conditional probabilities of each point in space (shown below) to each other point, and those of the points in 2 or 3 dimensions. These coordinates try to converge in x-many iterations.
- Using t-SNE
  - Using the outer layer of the network, which is the size of the class set, t-SNE was used to give not only a visualization of the RNN results, but hopefully can represent the prediction results better than a softmax function would.

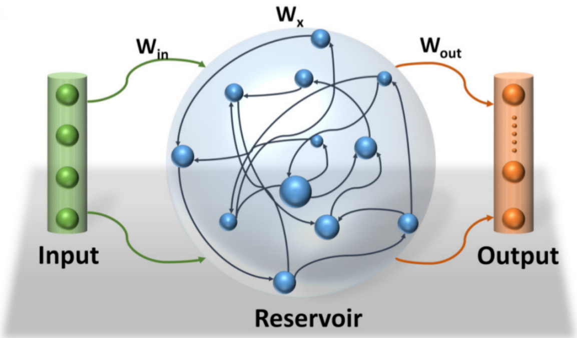
$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)},$$

Our t-SNE



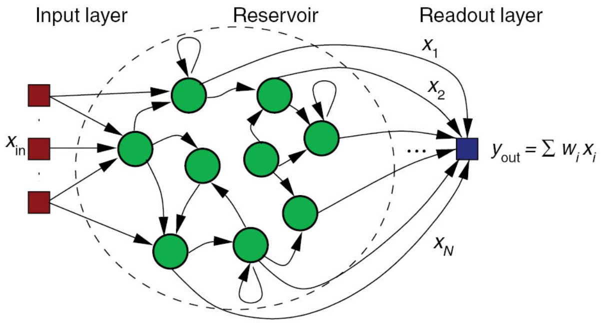
Reservoir Computing

- Paradigm that tries to strike a balance between linear models and neural networks
- Generalizes RNNs, but typically refers to an Echo State Network (ESN) or Liquid State Machine (LSM)
- The basic idea is to use an RNN as a nonlinear, time-dependant filter for the data and to feed the "states" of the RNN into some classifier



Reservoir Computing Models

- Have an input Layer, a reservoir layer, and a readout layer
- Weight matrices for the input and reservoir layers are held fixed
- Only the weights between the readout layer and the classifier are trained
- Rely on the fact that a reservoir is a highly coupled dynamical system with unique solutions; treats datum as perturbations that drive this system



Designing Reservoirs

In general, to build a reservoir we need to decide on:

- Which “neurons” we use
- The network structure
- How we assign weights to edges
- The classifier and what we use for the readout layer

Best practices for designing a reservoir is currently a open question, but it has been shown in the literature that reservoirs perform best when they:

1. have the “echo state property”
2. map the input into a much higher dimensional space

Our Reservoirs

In this presentation we will focus on results from two specific reservoir topologies:

1. A fully connected network
2. A "minimalist", cyclic-graph-like network

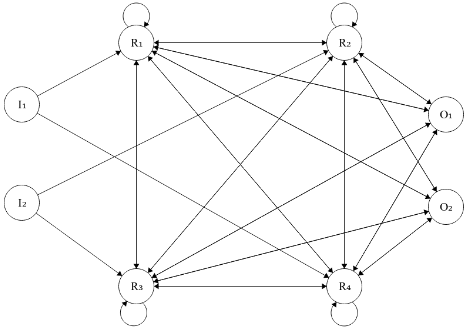
It should be noted that, for all results we are showing,

- the classifier is a simple one vs. rest logistic regression
- each channel of the input is assigned a subset of nodes that only that channel "drops" data into (with a weight of 1)
- at time t the state of a neuron $x(t)$ is defined by

$$\tanh\left(\left(\sum_{v \in N(x)} v \cdot w_{v,x}\right) + I(t)\right)$$

  where $N(x)$ is the open neighborhood of x, $w_{v,x}$ is the weight from v to x, and $I(t)$ is the input data at time t

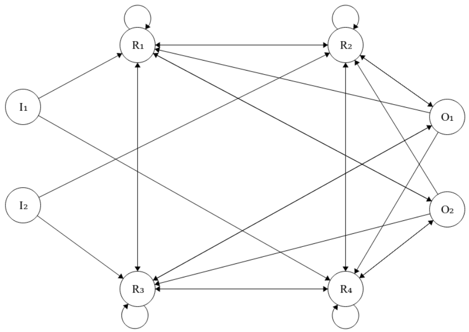
Our Reservoirs



Fully connected reservoir:

- We don't directly drop data into readout neurons and reserve as many neurons for readout as there are class labels
- We sample weights for each edge i.i.d from $\mathcal{N}(0, 0.1)$ for the digit data and from $\mathcal{N}(0, 0.5)$ for the hand data

Our Reservoirs

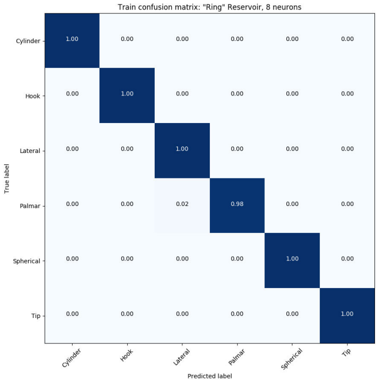


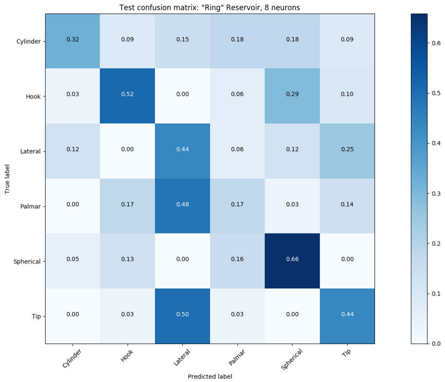
"Minimalist", ring topology:

- Again, we don't directly drop data into readout neurons and we reserve as many neurons for readout as there are class labels
- We sample weights for each edge i.i.d from a Uniform$(-1, 1)$ distribution

Ring Reservoir on Hand Data: 8 Neurons

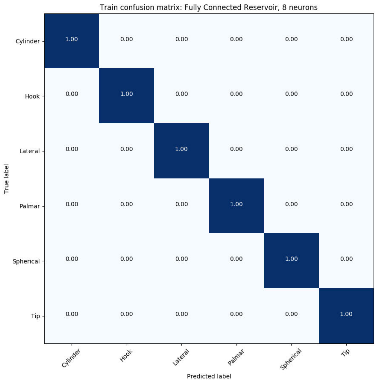
Accuracy: Training= 99.7%, Testing≈ 43.3%

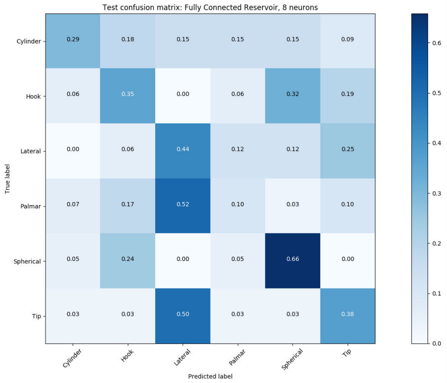




Fully Connected Reservoir on Hand Data: 8 Neurons

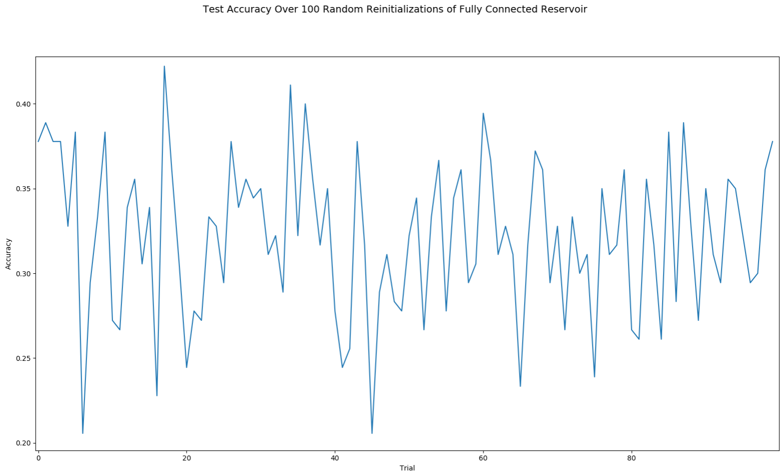
Accuracy: Training$=$ 100%, Testing$\approx$ 37.7%





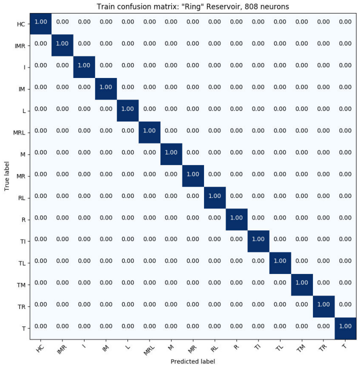
Fully Connected Reservoir on Hand Data: 8 Neurons

Accuracy: Training$=100\%$, Testing$\approx 37.7\%$

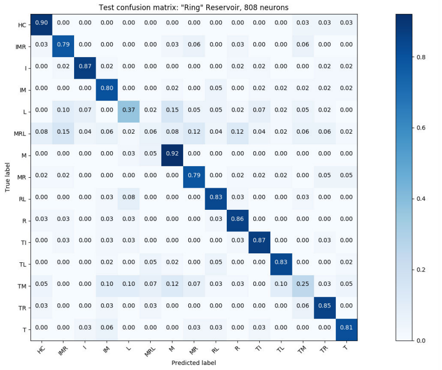


Ring Reservoir on Digit Data: 808 Neurons

Accuracy: Training= 100%, Testing≈ 70.3%



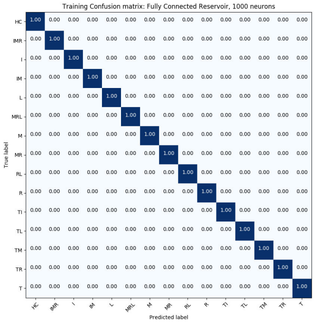
	HC	IMR	I	IM	L	MRL	M	MR	RL	R	TI	TL	TM	TR	T
HC	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
IMR	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
I	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
IM	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
L	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MRL	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
M	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
RL	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
R	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
TI	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
TL	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
TM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
TR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
T	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00



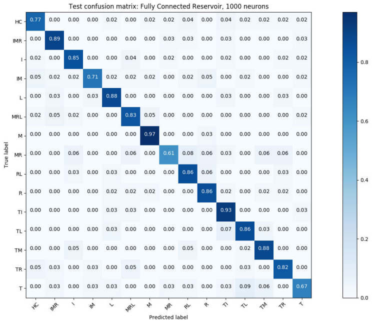
	HC	IMR	I	IM	L	MRL	M	MR	RL	R	TI	TL	TM	TR	T
HC	0.90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.03	0.03
IMR	0.03	0.79	0.00	0.00	0.00	0.00	0.03	0.06	0.00	0.03	0.00	0.00	0.06	0.00	0.00
I	0.00	0.02	0.87	0.02	0.00	0.00	0.00	0.00	0.00	0.02	0.02	0.00	0.00	0.02	0.02
IM	0.00	0.00	0.00	0.80	0.00	0.00	0.02	0.00	0.05	0.00	0.02	0.02	0.02	0.02	0.02
L	0.00	0.10	0.07	0.00	0.37	0.02	0.15	0.05	0.05	0.02	0.07	0.02	0.05	0.02	0.00
MRL	0.08	0.15	0.04	0.06	0.02	0.06	0.08	0.12	0.04	0.12	0.04	0.02	0.06	0.06	0.02
M	0.00	0.00	0.00	0.00	0.03	0.05	0.92	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MR	0.02	0.02	0.00	0.00	0.00	0.00	0.00	0.79	0.00	0.02	0.02	0.02	0.00	0.05	0.05
RL	0.00	0.00	0.00	0.03	0.08	0.00	0.00	0.00	0.83	0.03	0.00	0.00	0.03	0.00	0.00
R	0.03	0.03	0.00	0.03	0.03	0.00	0.00	0.00	0.03	0.86	0.00	0.00	0.00	0.00	0.00
TI	0.00	0.03	0.00	0.03	0.03	0.00	0.00	0.00	0.00	0.03	0.87	0.00	0.00	0.03	0.00
TL	0.00	0.00	0.00	0.02	0.00	0.05	0.02	0.00	0.05	0.00	0.00	0.83	0.00	0.00	0.02
TM	0.05	0.00	0.00	0.10	0.10	0.07	0.12	0.07	0.03	0.03	0.00	0.10	0.25	0.03	0.05
TR	0.03	0.00	0.00	0.03	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.85	0.00
T	0.00	0.00	0.03	0.06	0.00	0.00	0.00	0.03	0.03	0.00	0.03	0.00	0.00	0.00	0.81

Fully Connected Reservoir on Digit Data: 1000 Neurons

Accuracy: Training= 100%, Testing≈ 82.5%


Training Confusion matrix: Fully Connected Reservoir, 1000 neurons

True label \ Predicted label	HC	IMR	I	IM	L	MRL	M	MR	RL	R	TI	TL	TM	TR	T
HC	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
IMR	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
I	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
IM	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
L	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MRL	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
M	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
RL	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
R	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
TI	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
TL	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
TM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
TR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
T	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00


Test confusion matrix: Fully Connected Reservoir, 1000 neurons

True label \ Predicted label	HC	IMR	I	IM	L	MRL	M	MR	RL	R	TI	TL	TM	TR	T
HC	0.77	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.04	0.00	0.04	0.02	0.02	0.02	0.02
IMR	0.00	0.89	0.00	0.00	0.00	0.00	0.00	0.03	0.03	0.00	0.03	0.00	0.00	0.00	0.03
I	0.02	0.00	0.85	0.00	0.00	0.04	0.00	0.02	0.00	0.00	0.02	0.00	0.00	0.02	0.02
IM	0.05	0.02	0.02	0.71	0.02	0.02	0.02	0.00	0.02	0.05	0.00	0.02	0.00	0.00	0.02
L	0.00	0.03	0.00	0.03	0.88	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.03	0.00
MRL	0.02	0.05	0.02	0.00	0.00	0.83	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
M	0.00	0.00	0.00	0.00	0.00	0.00	0.97	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00
MR	0.00	0.00	0.06	0.00	0.00	0.06	0.00	0.61	0.08	0.06	0.03	0.00	0.06	0.06	0.00
RL	0.00	0.00	0.03	0.00	0.03	0.00	0.00	0.00	0.86	0.06	0.00	0.00	0.00	0.03	0.00
R	0.00	0.00	0.00	0.00	0.02	0.02	0.02	0.00	0.00	0.86	0.02	0.00	0.02	0.02	0.00
TI	0.00	0.00	0.00	0.00	0.03	0.03	0.00	0.00	0.00	0.00	0.93	0.00	0.00	0.00	0.03
TL	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.07	0.86	0.03	0.00	0.00
TM	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.02	0.88	0.00	0.00
TR	0.05	0.03	0.00	0.03	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.82	0.00
T	0.00	0.03	0.00	0.03	0.03	0.00	0.00	0.03	0.03	0.00	0.03	0.09	0.06	0.00	0.67

Summary of Results on Hand data

Model	Features	Accuracy
True Random	N/A	16.6%
FC Reservoir	None	37.7%
Ring Reservoir	None	43.3%
RNN	ARMA(1,1)	54.5%
Extra-Trees	AR(6) + RMS	92%[1]

[1]Validation accuracy, but test accuracy was comparable

Summary of Results on Digit data

Model	Features	Accuracy
True Random	N/A	6.6%
Extra-Trees	AR(6)+MAV+WAMP	43.8%[2]
LSTM	None	60.7%
Ring Reservoir	None	70.3%
FC Reservoir	None	82.5%

[2]Validation accuracy, but test accuracy was comparable

Conclusions

Feature Extraction and Popular Models

- Feature extraction with methods like random forest perform well on EMG data when there are a small number of classes and multiple trials for each movement. Different data can cause these models' accuracy to suffer and, though we might be able to slightly increase results by trying different feature combinations, it is unlikely that results will significantly improve.

Conclusions

Recurrent Neural Networks

- Shallow RNNs achieve reasonable accuracy given minimal features or even raw data. Accuracy can likely be boosted by carefully choosing features or by using deeper architectures, but training is expensive both in terms of computational complexity and data required.

Conclusions

Reservoir Computing Models

- Reservoirs can perform very well with minimal engineering involved, but due to the probabilistic and complex nature of the model it is currently impossible to know if you are getting the “best” results. Furthermore, as we saw with the Hand data, these models can require careful consideration of the dynamics in the data being used itself and are sensitive to the relative magnitude of datum.

Conclusions
Overall

- Feature extraction methods in conjunction with simple models are good when computational resources are limited and interpretability is key. RNNs are good when there isn't a human capable of tuning a model and when data becomes abundant. Reservoir computing models share the pros and cons with both of these approaches, but it essentially has less of the pros and also less of the cons.
- Classification of EMG data can certainly be done with high accuracy **when we want to be able to classify the movements of individuals**. Generalization to new people had very poor results, but it is unclear if this is due to the data gathering methodology, fundamental differences in neuronal responses between people, or simply something in the model being used.

Future Work

- Exploration into methods for finding combinations of features and channels that maximize accuracy
- Thoroughly searching the parameter space and using feature extraction to improve RNN performance
- An analysis of EMG signals' “timescale” to choose spectral properties of reservoirs and quantizing input to stop large input from saturating the system
- Finding larger sets of data so that we can generalize to new people's EMG recordings

Questions?

Thank you to Professor Chen and everybody in the department who has helped us throughout this project.