Decoding Brainwave Data using Regression

Justin Kilmarx: The University of Tennessee, Knoxville
David Saffo: Loyola University Chicago
Lucien Ng: The Chinese University of Hong Kong
Mentor: Dr. Xiaopeng Zhao

Introduction

- Brain-Computer Interface (BCI)
- Applications
  - Manipulation of external devices (e.g. wheelchairs)
  - For communication in disabled people
  - Rehabilitation robotics
  - Diagnosis and prediction of diseases (e.g. Parkinson’s disease, Seizure, Epilepsy)
  - Games
- Invasive vs Noninvasive
  - Electrocorticography
    - Fifer et al. (2012)
  - Electroencephalography
    - Mcfarland & Wolpaw (2011)

Background

Invasive

Noninvasive

Sensorimotor Rhythms (SMR)

Steady-State Visual Evoked Potential (SSVEP)

Imagined Body Kinematics

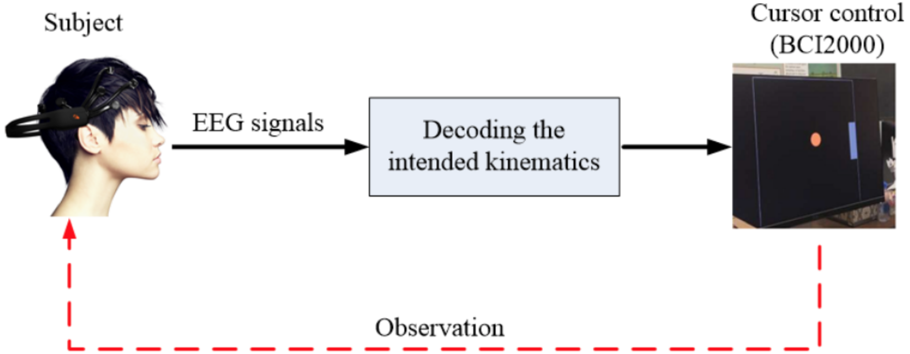
- Continuous decoding the kinematic parameters during imaginary movements of one body part
- Short time of training
- Natural imaginary movement
- Smoother controller system
- Possibility of developing a generalized decoder
- Eliminating Subject dependency

Research Objective and Setup

Objective: The goal of this project was to improve the prediction accuracy for a previously developed BCI model that used linear regression to predict cursor velocity from a subject's thoughts by testing new methods and nonlinear models.

Setup

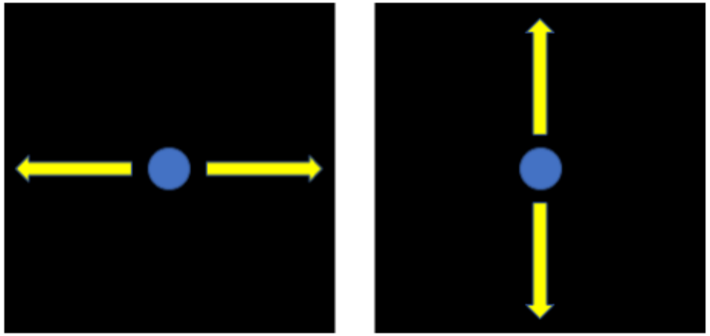
- Emotiv EPOC for recording EEG signals
- BCI2000 for cursor visualization and data collectection
- Matlab/Python for processing



Training

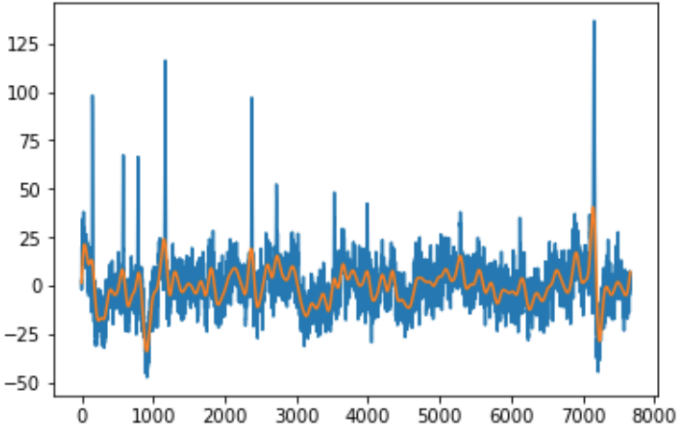
Automated cursor movement on computer monitor in 1D
Subject imagines following movement with dominant hand
10 trials
- 5 horizontal
- 5 vertical

1 minute each
Cross validation between trials
33 Subjects



Filtering

Raw EEG signals contain a lot of noise
4th order Butterworth lowpass filter with cutoff at 1 Hz
Attempted using bandpass over Mu, Alpha, and Beta bands, but these did not contain useful information for imagined body kinematics



Regression

Predict cursor velocity from EEG data

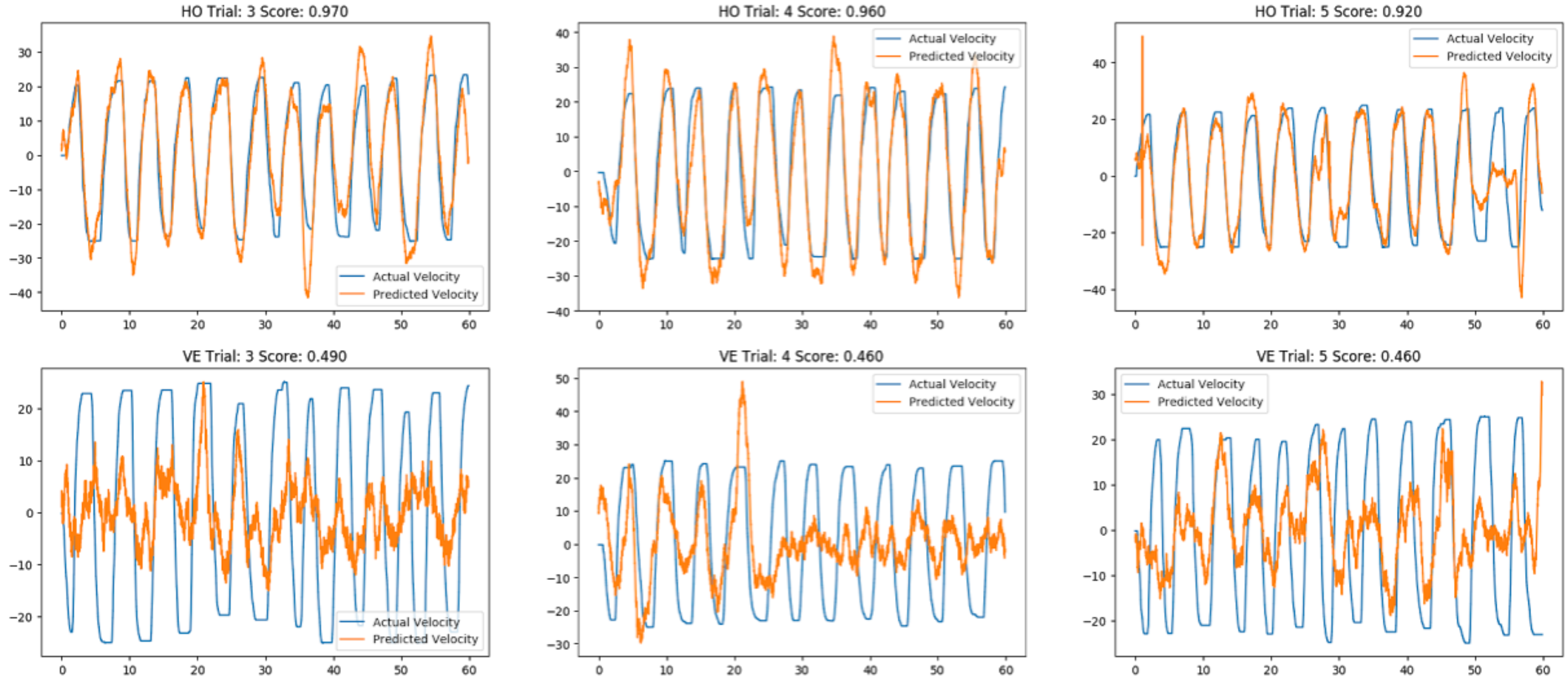
12 previous points in memory as features

Trial wise cross validation

Average prediction accuracy using goodness of fit on linear regression model

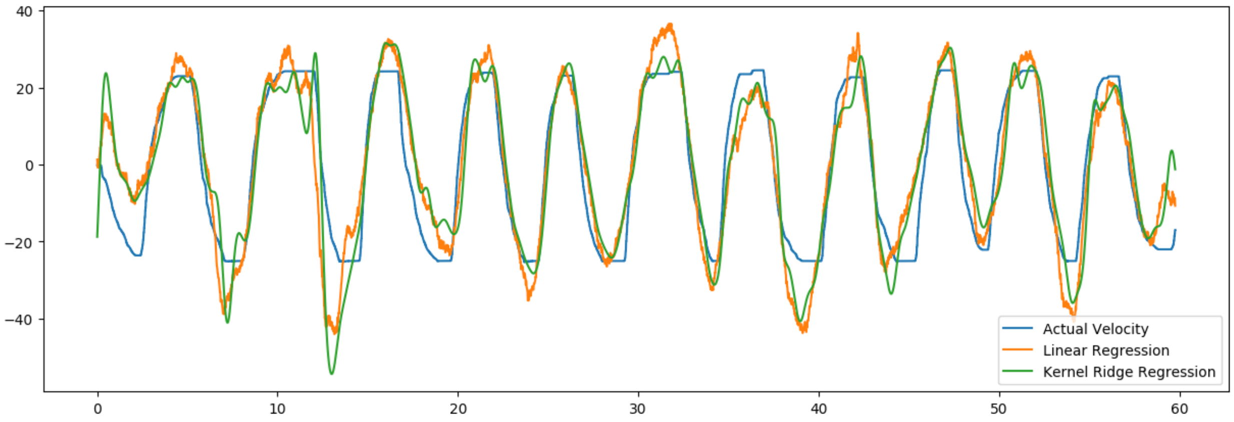
- Horizontal: 70.77%
- Vertical: 44.67%

Results


HO Trial: 3 Score: 0.970
HO Trial: 4 Score: 0.960
HO Trial: 5 Score: 0.920
VE Trial: 3 Score: 0.490
VE Trial: 4 Score: 0.460
VE Trial: 5 Score: 0.460
Actual Velocity
Predicted Velocity

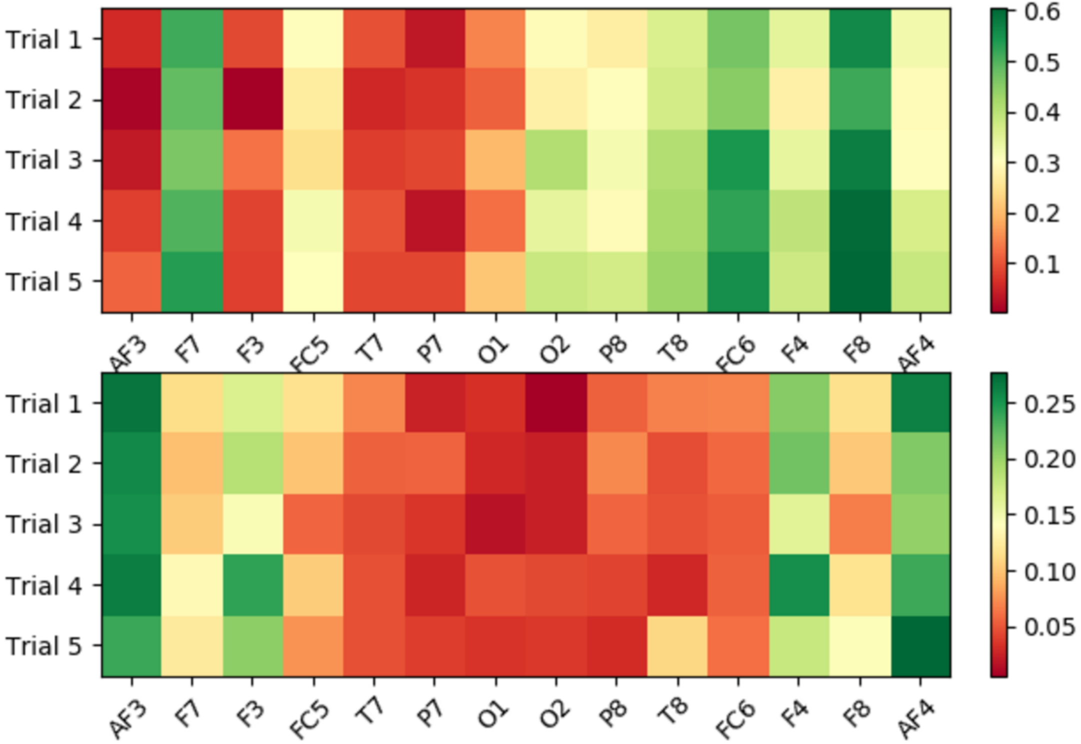
Results

Other models did not show significant improvements and were more computationally expensive (adaboost regression, ridge regression, kernel ridge regression, support vector regression, and multilayer perceptron)

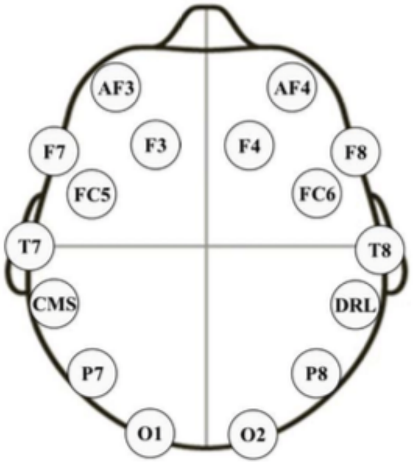


Channel Importance

- Channel-wise identification
- Horizontal (top)
    - F7 and F8
    - Right hemisphere
- Vertical (bottom)
    - AF3 and AF4
    - F3 and F4
- Clear pattern between horizontal and vertical
- Right hemisphere controls left body



Results

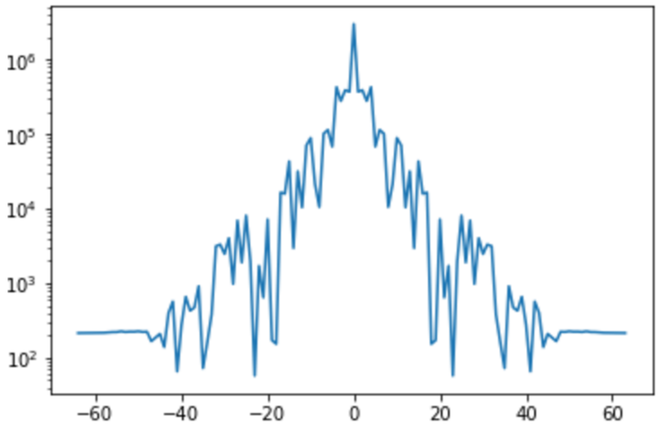


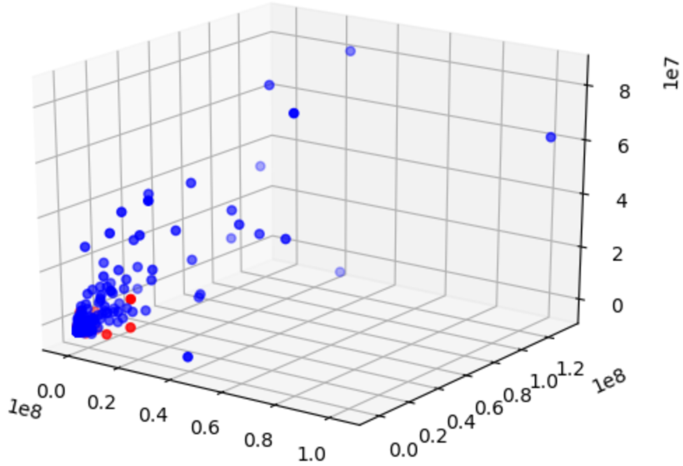
Channels	Horizontal Accuracy	Vertical Accuracy
All Channels	70.77%	44.67%
F7, 02, P8, T8, FC6, F4, F8, AF4	71.03%	41.68%
F7 and F8	69.93%	25.64%
F7, FC5, T8, FC6, F4, F8	72.73%	36.98%
AF3 AND AF4	41.93%	30.29%
AF3, F3, F4, and AF4	49.21%	33.09%
AF3, F3, F7, F8, F4, and AF4	69.97%	41.61%

Classification

Horizontal vs. Vertical
FFT analysis across 14 channels in 1 second samples
224 total features from 4 bands: Theta (4-7 Hz), Alpha (8-15 Hz), Beta (16-32 Hz), and Gamma (32-40 Hz)
- Mean PSD, median PSD, min PSD, max PSD

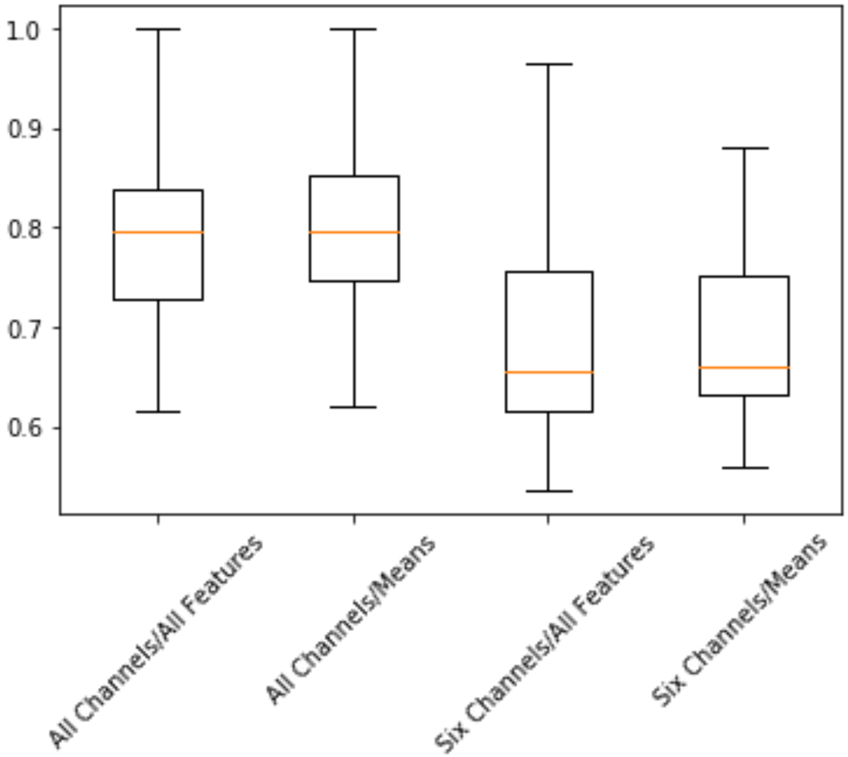
Model trained with Random Forest





Results

Channels/Features	Average Accuracy Score
All Channels/All Features	79%
All Channels/Means	80%
Six Channels/All Features	68%
Six Channels/Means	69%



Serial vs. Distributed

Results were generated using distributed computing

Dask Distributed Library

Cluster setup on Comet at the San Diego Supercomputer Center

Serial	Adaboost	04:30:00
Distributed	Adaboost	00:4:29

EEG-Based Control of a Computer Cursor with Machine Learning

Lucien Ng(The Chinese University of Hong Kong),

Justin Kilmarx (University of Tennessee),

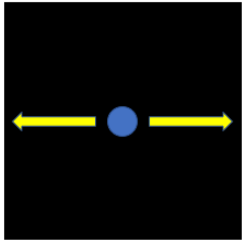
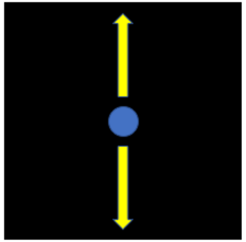
David Saffo (Loyola University Chicago)

EEG-Based Cursor Movement Classification

In the sense of machine learning, this is a supervised multiclass classification. The specification as follow:

- **Input**
  EEG data (time series) with 128 Hz and 14 channels
-

Vertical	Left	Right	No Movement
Horizontal	Up	Down	No Movement

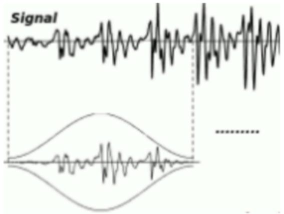
ire   n time point

Objective

- To classify the user indenting cursor movement by using EEG signal with high accuracy, and
- To accelerate the process to acceptable speed

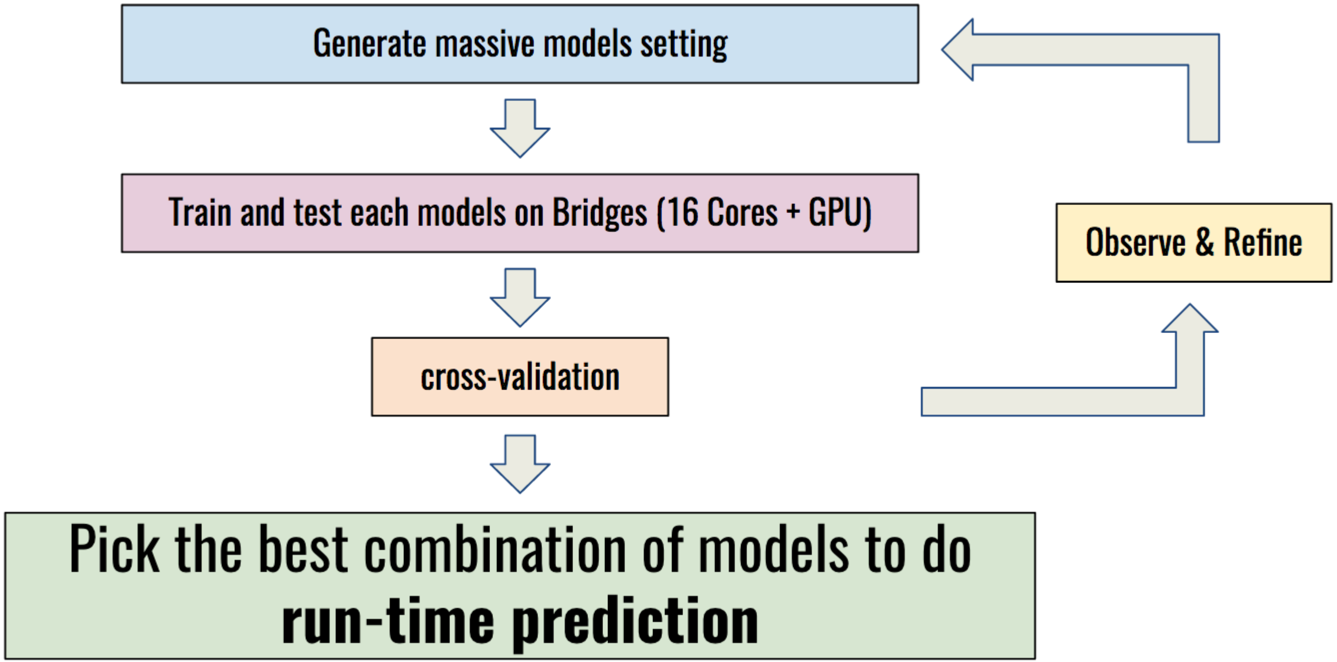
Overview of Models

Raw Data: Current and Past Sample



Preprocessing	Basic Models	Ensemble Models	Prediction
Filter Bank	Logistic Regression	Gradient Boosting	Up/Right
Event-related Potential	Neural Network		Down/Left
Neural Oscillation			No movement

Workflow


Generate massive models setting
Train and test each models on Bridges (16 Cores + GPU)
cross-validation
Pick the best combination of models to do **run-time prediction**
Observe & Refine

Feature Extraction: Filter Bank

Left low pass filter: Only past time points were used to train and test the mo

Low freq.

Applied

1 Hz, 2

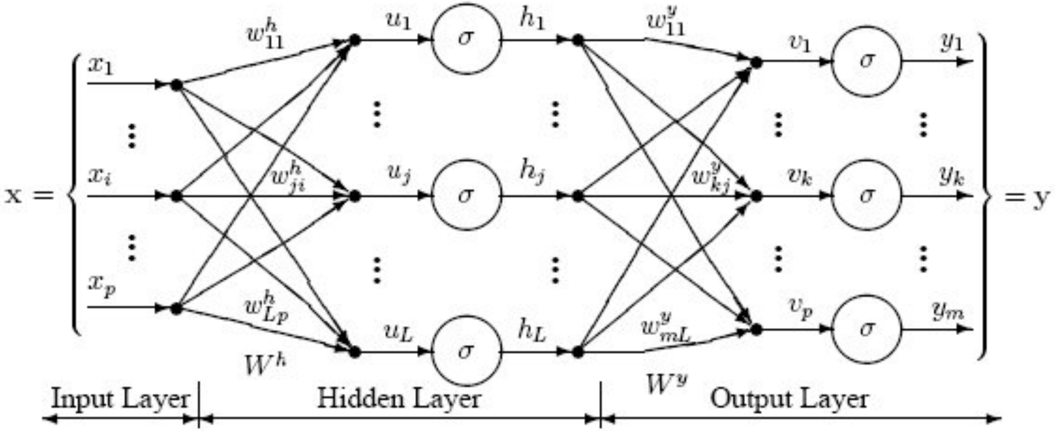
Psychological or Physiological State	Changes in EEG Waves
Deep sleep	Predominance of the delta wave
Concentrated	Suppression of the alpha wave
Vigilant	Generation of beta wave
Recognition of sensory stimuli	Changes in gamma wave

High freq.

7 Hz, 9 Hz, 15 Hz, 30 Hz

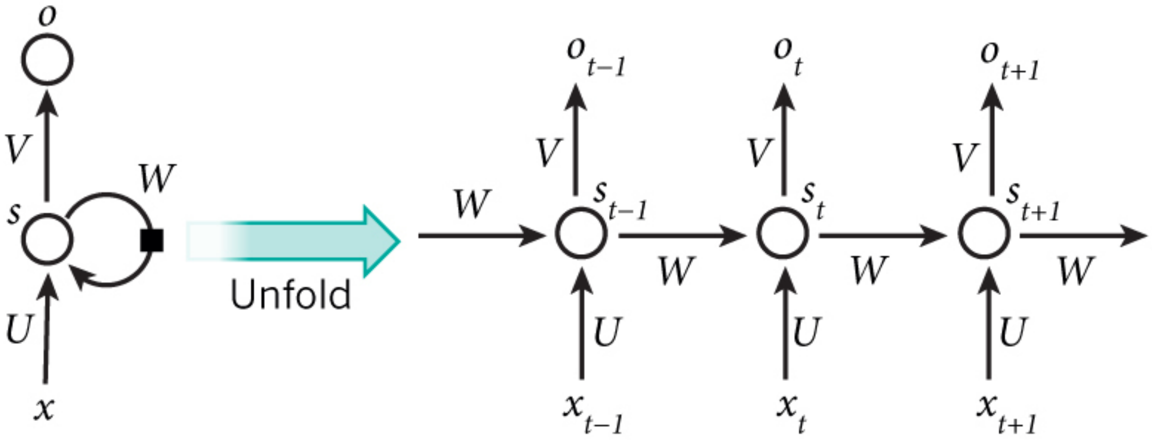
KatarzynaBlinowska, Piota Durka. ELECTROENCEPHALOGRAPHY (EEG) [Online]. Available: http://users.rowan.edu/~polikar/CLASSES/ECE504/EEG.pdf

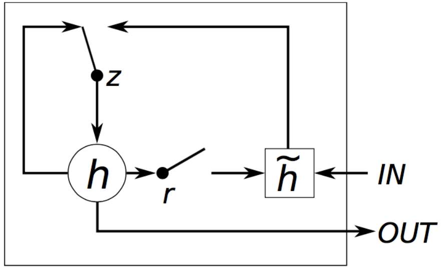
Classifying : Multilayer perceptron



K + TensorFlow

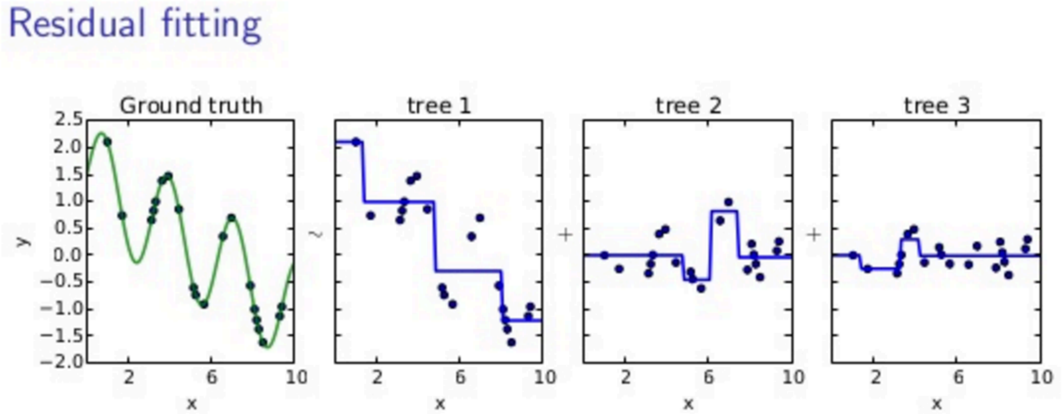
Classifying: Recurrent Neural Network

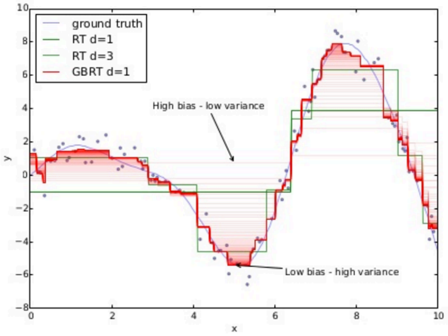




Recurrent Neural Networks Tutorial. Available:
http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Models Ensembling: Gradient Boosting





Gradient Boosted Regression Trees in scikit-learn. Available: https://www.slideshare.net/DataRobot/gradient-boosted-regression-trees-in-scikitlearn

Experimental Setup

- 12 Subjects' data were used
- Each of them has 5 trials about horizontal / vertical movements
- Validation:

Trials	1^{st}, 2^{nd} and 3^{rd}	4^{th}	5^{th}
Basic Models	Training Data	Validation	Validation
Ensemble Models	-	2-fold Valid	2-fold Valid

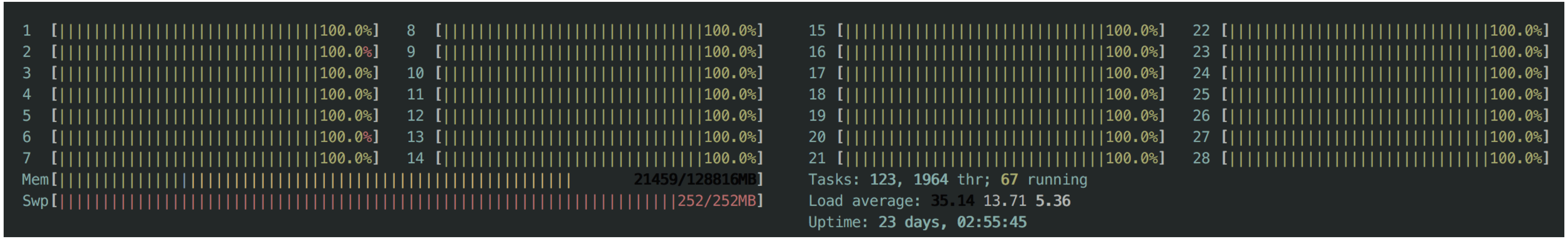
- Cr
  co
- The experiment ran on XSEDE-bridges with 16 CPU-cores and GPU (P100)

Multithread

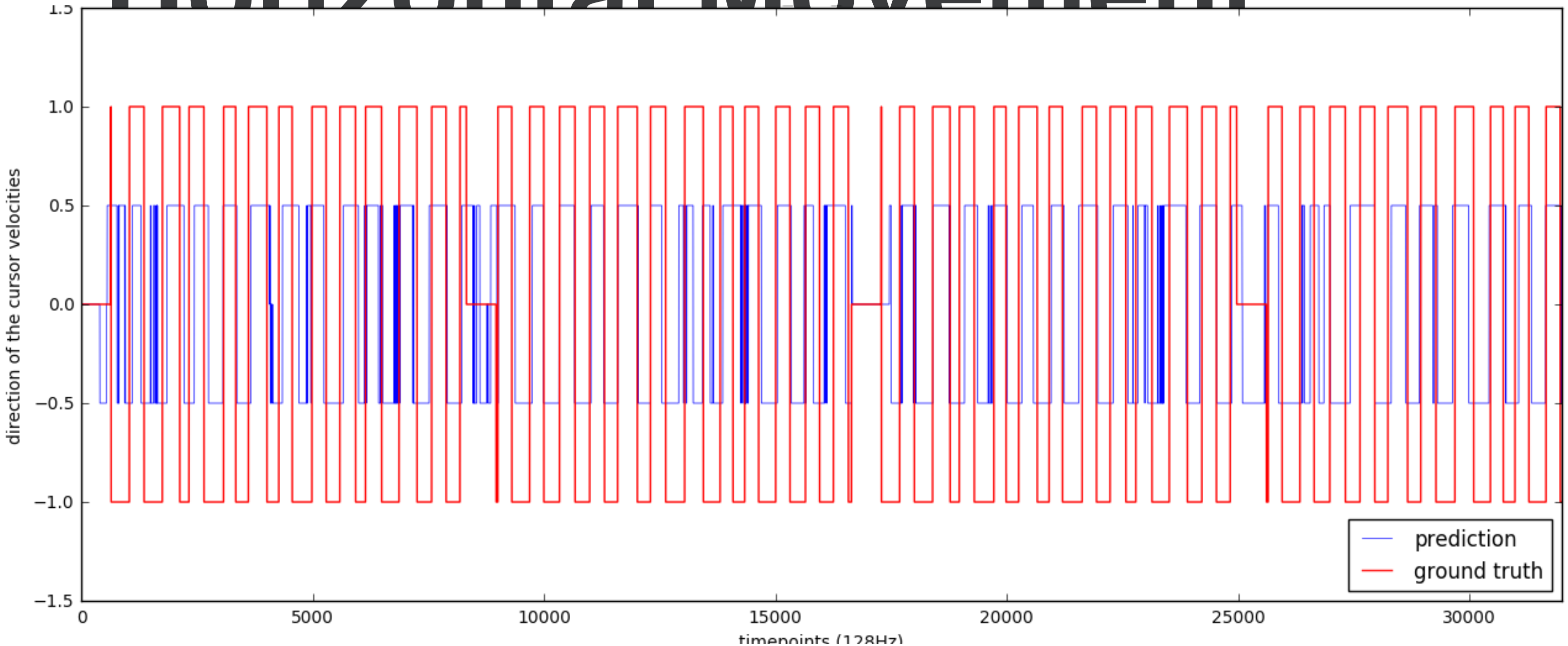
All the train-validation sets can run independently.

All the event classifiers can be trained independently

Lets utilize all the CPU-cores!



Visualized results: An Example of Prediction on Horizontal Movement

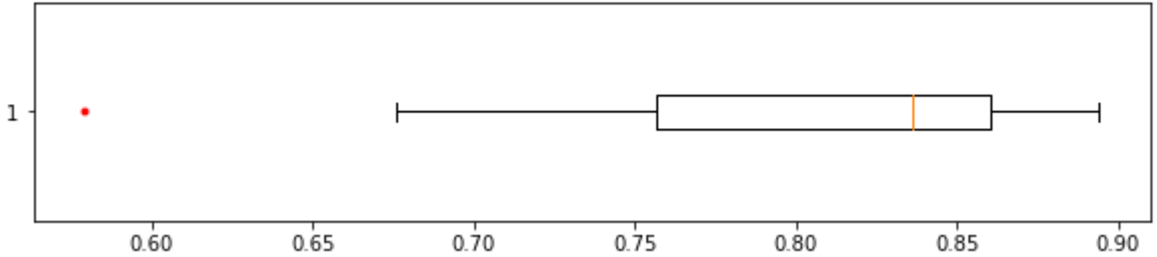


Results: Horizontal

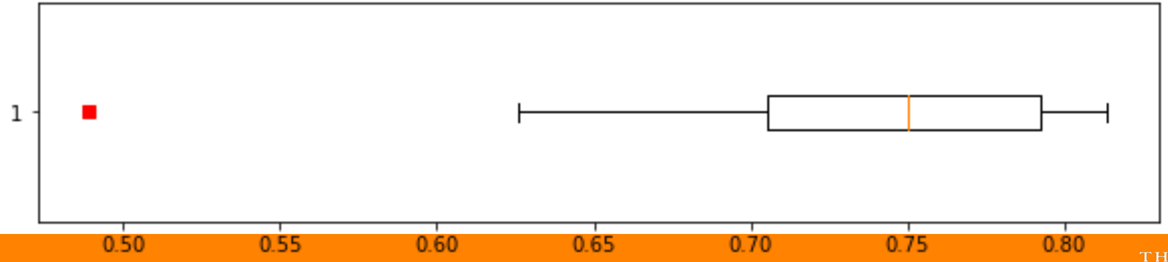
The best basic model:
- Preprocessed by filter bank
- Neural Network with 32 hidden units

The Accuracy/AUC of subjects

AUC



Accuracy



Results

Prediction	AUC	Accuracy	Total Time
Horizontal	0.91	80%	10.5 hours
Vertical	0.71	60%	10.5 hours

Time for a training process = 10 minutes

Acceleration: Magma-DNN

MAGMA-DNN: Toward a More Flexible DNN Framework for Low-Level Implementation

Magma is a large, well-supported software package designed for computations in algebra, number theory, algebraic geometry and algebraic combinatorics

The main operation in neural network is matrix multiplication.

Lets try to use Magma to build a neural network!

Advantage

Open-source

Flexibility: Free to implement any mathematical function for both CPU and GPU with Magma

Fast

Benchmark: MNIST dataset

Number of input siz

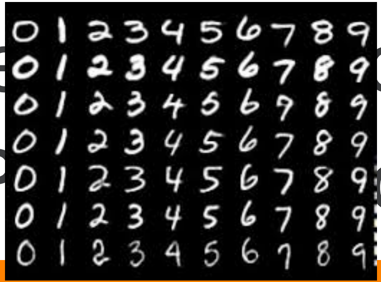
Number of Hidden u

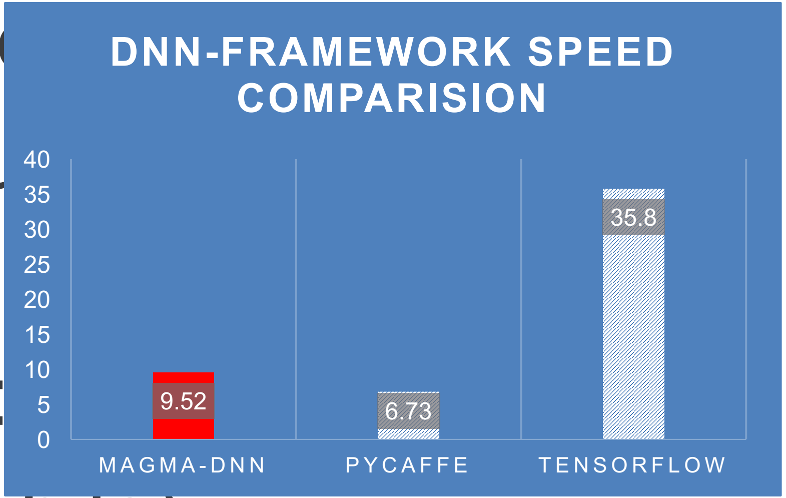
Batch size: 100

Number of iteration:

Pre   oat (32 bits)

GP   ce GTX 1050Ti

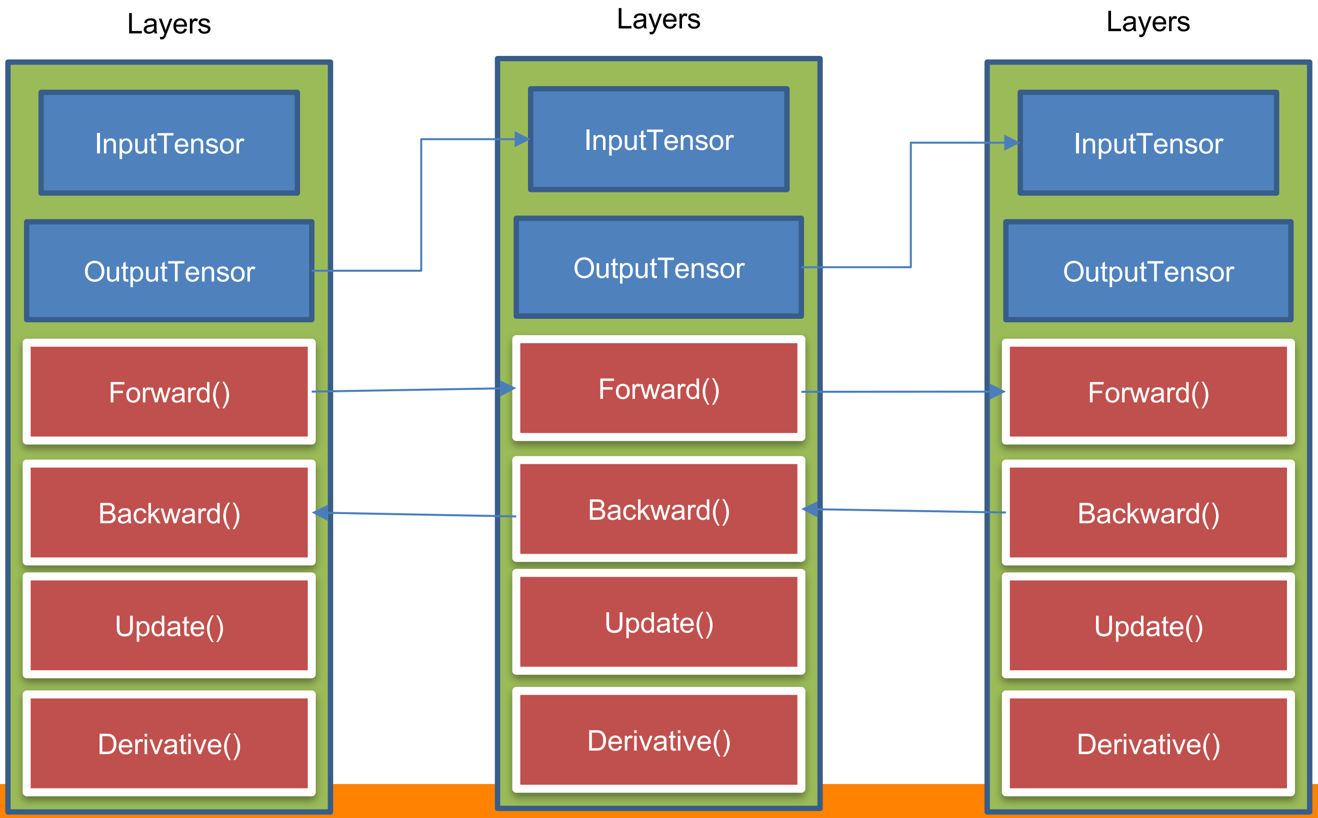




Comparison with other DNN frameworks

	MAGMA-DNN	Caffe	TensorFlow
Speed	Fast	Fast	Relatively Slow
Input Data Format	Support Native Pointer Array	HDF5 Only	NumPy
Dependency	MAGMA	Protobuf, HDF5, CUDA, BLAS, OpenCV, Boost…	CUDA, NumPy …

Architecture



Code Example

Layers Initialization

```
InputLayer<float> inputLayer(inputMat);
FCLayer<float> FC1(&inputLayer, n_hidden_units);
ActivationLayer<float> actv1(&FC1, SIGMOID);
FCLayer<float> FC2(&actv1, n_output_classes);
OutputLayer<float> outputLayer(&FC2, labelsMat, BIN_CROSSENTROPY_WITH_SIGMOID);
```

Network construction

```
std::vector<Layer<float>*> vec_layer;
vec_layer.push_back(&outputLayer);
vec_layer.push_back(&FC1);
vec_layer.push_back(&actv1);
vec_layer.push_back(&FC2);
vec_layer.push_back(&outputLayer);
```

Training

```
for (int i = 0; i < (int) vec_layer.size(); i++) vec_layer[i]->forward_gpu();
for (int i = vec_layer.size() - 1; i >= 0; i--) {
    vec_layer[i]->update();
    if (i >= 2) vec_layer[i]->backward_gpu(); // fc1 doesn't need to backward
}
```

Future Works

- Explore the EEG data by apply more machine learning techniques on it
- Implement Convolutional Neural Network and Recurrent Neural Network on MAGMA
- Apply MAGMA-DNN on the EEG data analysis