Scalable Motor Movement Recognition from Electroencephalography using Machine Learning

Aditi Sharma*[†], Shivee Singh*[†], Brian Wright[†], Alan Perry[†],
Diane Myung-kyung Woodbridge[†], Abbie M. Popa[†]
Data Science Program, University of San Francisco[†]
San Francisco, CA
{*asharma26,ssingh33,bawright3,ajperry2,dwoodbridge,apopa*}*@usfca.edu*

***Abstract*—Electroencephalography (EEG) is a noninvasive and inexpensive means of monitoring brain activity. Because of the low-cost, noninvasive nature of EEG, it may be useful for classification of motor movements when a patient controls a prosthetic device. However, due to the high velocity nature of EEG recordings, the data used in such a classification are often large and may take a long time to process on a local, non-distributed computer. Here we explore the use of a distributed computing architecture for storage and processing of EEG data. We evaluate the classification of EEG recordings during hand movements. We find that processing these data on a distributed system results in much faster classification times (e.g., 726 seconds versus 3925 seconds) without limiting accuracy (e.g., AUC of 0.85).**

***Keywords*-Electroencephalography, EEG, Distributed Processing, Distributed Database, Cloud Computing, Machine Learning, Brain Computer Interface**

I. Introduction

The amputee coalition estimates there are 2.1 million people in the United States who are currently living with limb loss, and an additional 185,000 individuals require an amputation each year [1]. Additionally, limb amputation affects members of the military, with 6,000 amputations per year in the US Armed Forces [2]. Additionally, there are approximately 300,000 individuals in the US living with spinal cord injury impacting upper extremity function, and 12,000 new cases per year [3]. These individuals could also benefit from BCI control of prostheses. Until recently, prostheses were controlled by myoelectric signals, that is, impulses from the muscle tissue neighboring the prosthetic device [4]. However, myoelectric control provides limited dexterity and flexibility of control, whereas brain signals may provide a more precise alternative [4].

Electroencephalography (EEG) uses non-invasive electrodes placed on the scalp of participants to measure signals generated by local field potentials when neurons in the cortex are active [5]. EEG has been used to study healthy individuals, showing, for example, differences in attentional allocation [6] and working memory [7]. There are significant advantages to EEG, primarily because it is non-invasive, (recordings can be performed with electrodes placed on the head) and has high temporal precision [8], [9]. Here, we examine a data set comprised of EEG recordings from participants performing a motor task [10]. Classification of brain activity during motor tasks holds promise as a non-invasive, inexpensive way for patients who use prosthetic hands to control these devices [11]. Here, we benchmark algorithms for classifying EEG during motor tasks using distributed computing architectures. This allows us to examine both accuracy and temporal performance across algorithms and architectures.

Because EEG data is collected at a rapid velocity (generally every 1-4 milliseconds) from multiple sensors the data quickly becomes large for many computer systems [9]. This can lead to long processing times. Storing and processing data in some type of centralized high-performance system poses a novel way to solve this issue. It could, however, present difficulties when data streams come from multiple organizations if the database system suffers read and write reliability issues (e.g., communication slow-down or system failure). Distributed database systems solve reliability issues by splitting data into multiple shards and recovering data from replicated data when there is a failure [12]. This allows for processing the EEG data stream in a relatively short amount of time for rapid classification of intended movement. Distributed computing efficiently completes jobs by distributing tasks to multiple processors and planning and running processes concurrently. Previous work has explored using Hadoop [13] and Beowulf [14]. Our approach uses the modern and popular Spark architecture, which is accessible through Amazon Web Services.

The rest of the paper is organized as follows: Section 2 contains a system overview, Section 3 contains descriptions of the processing algorithms, section for presents the results we obtained, and section 5 concludes with a discussion of our findings.

II. System Overview

To develop a scalable system that stores and processes high frequency EEG data, we stored data in a schemaless distributed database system and processed data using distributed computing. In this study, we used Amazon Web

[*] These authors contributed equally.

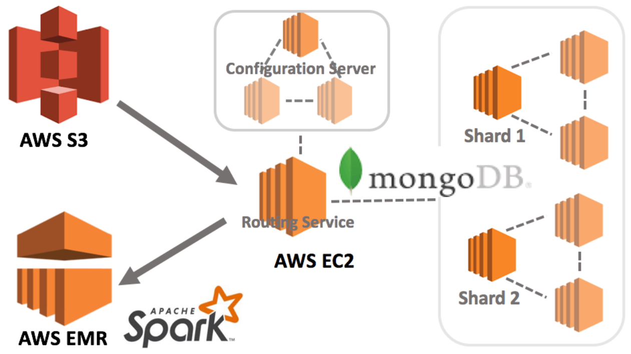

Figure 1. Data Pipeline

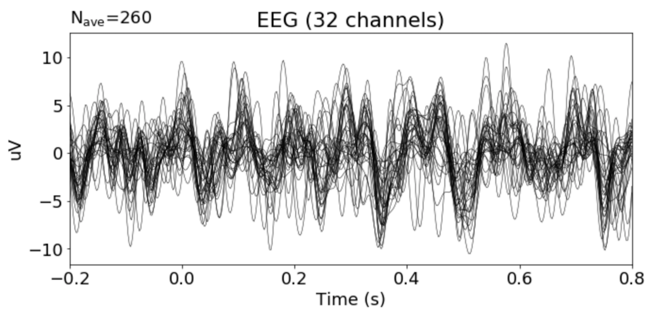

Figure 2. EEG Signal for Subject 1 where event type is Replace following 60 Hz notch filter.

Service Simple Storage Service (AWS S3), MongoDB, a document database, and Apache Spark, a distributed cluster-computing framework for data storage, management and processing respectively (Figure 1).

AWS S3 is a cloud storage system, adopting a "pay-as-you-go" charging model, offering limitless storage capacity. AWS S3 was selected for its replicated hosting in multiple data-centers that enables 99.99% data availability and low data access latency. S3 also provides ease of interoperability with other AWS cloud services including Elastic Compute Cloud (EC2) and Elastic MapReduce (EMR) [15].

MongoDB stores data without restricting to have either constant or fixed schema, which suits datasets from various institutes that may collect data in a different format or use a different numbers of channels. In addition, unlike traditional database management systems, MongoDB is built as a distributed data storage which can handle a large volume of high frequency data. By storing data in different partitions and maintaining its replica sets, a database management system is scalable while reliable and free from single point failure [16]. For MongoDB, we launched two nodes for dividing the entire dataset into two partitions. We had additional instances for saving a replica set of each partition's meta data and configurations. When the primary node fails or is not available, MongoDB will choose one of the secondary nodes as a new primary node and replicated data in it will be provided to the user. MongoDB also maintains a configuration server and its replica set to store state and organization of data, and data partitions and replications. Data partitions, replications and configuration servers were all launched on AWS EC2.

Pre-processing was completed locally in order to use the highly regarded MNE package [17]. Also, computation of these linear features was not sufficiently time-intensive to be a limiting factor in the processing pipeline, even when computed locally.

For machine learning model development and application, we chose Apache Spark, which uses traditional MapReduce concepts. Hadoop MapReduce is a programming paradigm for processing a large volume of data in parallel by dividing a task into a set of subtasks, and processing them in parallel on a cluster of multiple machines. For MapReduce, users design map and reduce steps, where a map function, such as filtering, processes key-value pairs in parallel. A reduce function takes the outputs of the map function as input from multiple machines and executes a summary operation, returning a single answer to a driver [18]. Apache Spark enhanced efficiency by using its directed acyclic graph-based job scheduling and in-memory data processing [19], showing 100 times faster than Hadoop MapReduce when processing data in memory. Apache Spark is a highly efficient model and is used in both research and industry [20], [21], [22]. For Apache Spark, we launched instances on AWS EMR that provides a fully managed Hadoop framework hosted on AWS EC2 instances [23]. In the launched EMR cluster, a master node distributes tasks to slave nodes for processing data in parallel. EMR automatically provisions hardware resources, installs the required software and provides an accessible monitoring dashboard [24].

III. Algorithms

A. Feature Processing

The EEG data were originally presented as the raw signal from the 32 channels of the standard 10-20 recording montage, the data were pre-processed by band-pass filtering from 0.016-1,000 Hz and down-sampled to 500 Hz. We further applied a 60 Hz Butterworth notch filter to remove electrical noise [8] (Fig. 2). We chose to decompose the signal into 4 low frequency bands and 1 mid-frequency band, as this has shown promise in previous work decoding brain signals associated with motor tasks [25]. We decomposed these frequency bands into four common spatial components that may be associated with different sources in the brain, this allows us to preserve information about local variations in the frequency bands while reducing data more than keeping five bands at 32 channels [26]. This method has been shown to be robust in single-trial analyses [27] and multi-class classification problems [28]. We processed the data into these 20 features (5 filters by 4 sources) using the MNE package [17] for Python [29]. Each epoch was binned beginning 200 ms before event-onset to 800 ms following event onset (Algorithm 1).

Algorithm 1 Feature Engineering

```
Data: EEG Data

Function mne_raw_object(filename, read_events=TRUE) is
    Get data file, channel names, montage from MNE
    if read_event = TRUE then
        Get events file, event names
        Concatenate event file and data file
    end
    Create and populate MNE info structure
    return RawArray(data, MNE info structure)
end

Function create_epochs() is
    for each subject do
        for filename of the subject do
            Append mne_raw_object(filename) to raw
        end
        Pick EEG signal using raw info
        Filter data for low- and mid-frequency bands using
          Butterworth filters
        Initialize list y for class for each epoch
        for each event_i do
            Create epoch_i for event_i where time = -0.2 to
              0.8
            Extend y for class for each epoch
        end
        Train Common Spatial Pattern (CSP)
        Create training features using CSP filter and rectify
          signals
        Concatenate training features
        Get labels from raw data
        Concatenate labels and features and save to disk
    end
end

def main:
    Initialize Butterworth filters
    create_epochs()
```

B. Modelling Approach

We aimed to classify the type of hand movement from the extracted features. The type of hand movement fell into six different possible classes: Hand Start (when the hand begins moving), First Digit Touch (when the hand first contacts the object to be grasped), Both Start Load Phase (when the hand begins to support the weight of the object), LiftOff (when upward movement begins), Replace (when the object is returned to the surface), and Both Released (when the thumb and fingers are no longer in contact with the object). Notably, these six classes were not mutually exclusive. This means that the hand movement could belong to two categories, i.e., a hand movement could be both HandStart and LiftOff. To deal with this, we built six different models, one for each label, and then aggregated the results into a tuple of six binary outcomes. By building a model for each motion class separately, we treated classes as independent events, which they are definitely not. An alternative approach would have been to incorporate all possible combinations of the six labels into one class with 64 (two to the power of six) possible labels. The problem with this method, is several of those 64 labels are not going to be represented at all (e.g., one cannot simultaneous pick up and set down the object). Thus, we felt that aggregating across binary outcomes would better mitigate bias from having too many zeros in the outcome array.

We trained our data using an 80%, 20% train/test split on participant.

We test four different model types: Logistic Regression Classifier [30], Random Forest Classifier [31], Linear Support Vector Machine Classifier [32], and Gradient Boosted Trees [33].

The logistic regression classifier (LR) makes a soft prediction by fitting a sigmoid function to the relationship between the predictive features and the outcome (movement type). The output of the logistic regression is an odds ratio for each movement class. The hard prediction was movement class with the highest odds ratio. [30]

The random forest classifier (RF) is an ensemble model of decision trees, where each decision tree makes recursive binary partitions of feature space and test observations falling in a leaf node are predicted to be the majority class of the leaf. Further, to avoid overly correlated trees, each tree is built on a subset of the predictive features (for further discussion see [31]).

The linear support vector machine classifier (SVC) works by iteratively identifying the hyperplane which best separates classes of data. For our problem, each class was modelled in a one versus all fashion to determine the hyperplane with the largest margin between classes by optimizing for lowest hinge loss. [32]

Finally, gradient boosted trees (GBT) are similar to RF in that they take the vote of an ensemble of trees, however trees are trained iteratively on the residuals from the previous tree. Thus, subsequent trees aim to reduce the error for the points where the previous trees made poor predictions, in our case indicated by higher log loss. [33]

For each method we tested a small set of hyper-parameters using a grid-search and the best hyper-parameters were used to calculate our final results.

IV. Experiment Output

A. Data

Our dataset [10] consists of EEG data from 12 different participants. Each participant completed 10 trials, for a total of 120 trials. The stimulus channels were preprocessed into six binary movement columns, where a 1 represents that movement being executed, and a 0 represents the lack of that specific movement. During each trial the weight or texture of the object used to complete the motor tasks changed, in order to allow classification of the same six movements under different conditions. This is important because if the classification were used to control a prosthetic hand it would need to correctly classify these motions across a wide variety of objects and environments.

B. Experiment Setup

1) Distributed Database

In this work, we set up MongoDB on 10 EC2 instances with 1 GB Memory and 6 CPU cores. We launched one instance for service routing, three for configuration meta data storage where one was a primary and two was a replica. In addition, we launched three instances for the first shard and another three for the second shard. For each shard, one was a primary node and the other two were replication nodes. Setting up three instances helps satisfy a read and write quorum in case read and write inconsistency happens in a distributed system [34].

2) Distributed Processing

For processing data in parallel using Apache Spark, we used EMR with a master and multiple slave nodes and compare with a local machine (Table I).

Table I
Instance Specifications for Distributed Data Processing

Cluster Configuration	Master		Slave	
	CPU Cores	Memory (Gb)	CPU Cores	Memory (Gb)
Cluster 1 1 master, 4 slaves	8	32	4	16
Cluster 2 1 master, 3 slaves	8	32	8	32
Local	4	6	-	-

C. Results

In order to benchmark these models, we used the metric area under the receiver operating characteristic curve (Area Under the Curve: AUC). The AUC is useful in a multiclass classification problem because higher values represent more true positives without a higher rate of false positives. After fitting and transforming our models with the Spark ML package, we found that the Random Forest model outperformed the others, with an average AUC score of 0.85. Next was Gradient Boosted Trees, (AUC of 0.83) followed by Logistic Regression (AUC of 0.57) and Linear SVC (AUC of 0.52) (Fig. 3). Both of our clusters were much faster than running the analysis locally, and Cluster 1 was slightly faster than Cluster 2 (Fig. 4). In addition to being the highest accuracy, the Random Forest model was the second fastest (Cluster 1: 726 seconds), surpassed only by the Logistic Regression model (Cluster 1: 150 seconds). In both cases the speed was significantly improved on the cluster versus a non-distributed (local) computer, where the two models completed in 3952 seconds (1.09 hours) and 2400 seconds (40 minutes), respectively.

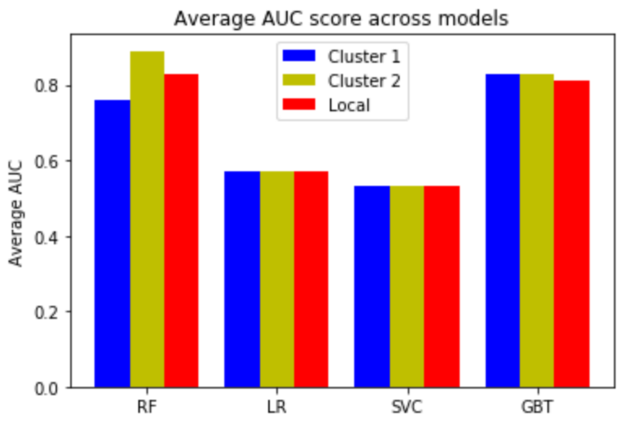

Figure 3. Average AUC for each model and architecture.

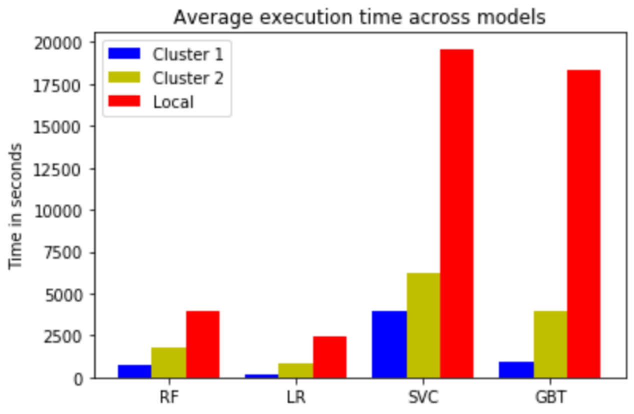

Figure 4. Average time for each model and architecture.

V. Conclusion

A. Discussion

Regardless of the modelling algorithm we used running the training on a cluster rather than a local machine was far faster. Cluster 1 was faster than Cluster 2, and because of that we recommend more smaller slaves as opposed to fewer larger slaves when designing a cluster for this type of analysis. An additional advantage to this is that Cluster 1, in addition to performing faster, was also cheaper at \$0.28 per hour (while Cluster 2 cost \$0.48 per hour).

Random forest had the highest accuracy, with a nearly 150% improvement over logistic regression. While logistic regression was the fastest, taking 20% as long on Cluster 1, 47% as long on Cluster 2, and 61% as long as random forest when run locally. Despite this faster rate of training time, we advocate for the use of random forest models in classifying EEG data. Patients will place a high value on their prosthesis performing the action they want it to, so the additional training time would pay-off in valuable accuracy.

In addition to showing the highest accuracy, in examining the AUC scores for random forest we see more variability than in our other models 3. This is likely because the maximum depth for the trees in the random forest we arrived at was 10. Generally random forest uses deep trees to decrease the variance of the model, future research could examine the use of deeper trees to reduce this variance, though this hyper-parameter was likely successful in our grid-search because the shallower trees reduced over-fitting.

B. Future Directions

One approach that could improve our model would be to train parameters for each participant, rather than for the group as a whole. The model may perform better if we kept the ID as a feature, because the human brain is quite different across person to person [35]. This approach, however, would reduce the generalizability of our model to new participants. By building a more general model new patients would likely receive higher accuracy even without their personal EEG recordings being incorporated into training data. Considering our data was split on participant we demonstrate how powerful this machine learning approach is to classifying movement of new individuals.

Further, we could have stacked multiple models. Although we note that while the stacking approach improves accuracy [36], it likely greatly slows down the time needed to train the model, and more importantly, classify actions taken by a patient using a prosthesis. While training time can be a little longer, if it takes a long time to classify actions with a trained model this will impact the patient's quality of life. Most patients will not want to wait for every action they take to be classified, and would prefer a more responsive prosthesis.

Finally, we may have been able to improve our model further by using more advanced methods of hyper-parameter tuning such as Bayesian optimization.

C. Summary

Overall, regardless of the machine learning model chosen, these findings are important to patient outcomes. Being able to classify motor intent from EEG quickly is critical to patient care, since prostheses (or other treatments) must be deployed in a timely matter. We urge future research into EEG classification to use distributed data systems for storage and analysis, as we have shown this improves processing time markedly.

References

[1] Amputee-Coalition, "Limb Loss in the USA," *Inside Track*, vol. 3, 2016.

[2] F. O'Donnell, "Medical Surveillance Monthly Report," Silver Springs, MD, 2012.

[3] N. S. C. I. S. Center, "Spinal cord injury facts and figures at a glance," Birmingham, AL, 2013.

[4] R. E. Lee, "Reassessing myoelectric control: Is it time to look at alternatives?" *Medical Science News*, vol. 136, pp. 467–469, 1987.

[5] G. Buzsáki, C. A. Anastassiou, and C. Koch, "The origin of extracellular fields and currents-EEG, ECoG, LFP and spikes," *Nature Reviews Neuroscience*, vol. 13, no. 6, pp. 407–420, 2012. [Online]. Available: http://dx.doi.org/10.1038/nrn3241

[6] R. Sawaki, J. J. Geng, and S. J. Luck, "A common neural mechanism for preventing and terminating the allocation of attention." *The Journal of neuroscience : the official journal of the Society for Neuroscience*, vol. 32, no. 31, pp. 10 725–36, aug 2012. [Online]. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3488698{&}tool=pmcentrez{&}rendertype=abstract

[7] S. Dong, L. M. Reder, Y. Yao, Y. Liu, and F. Chen, "Individual differences in working memory capacity are reflected in different erp and eeg patterns to task difficulty," *Brain Research*, vol. 1616, pp. 146 – 156, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0006899315003881

[8] S. J. Luck, *An Introduction to the Event-Related Potential Technique*, 2nd ed. Cambridge, MA: Massachusetts Institute of Technology, 2014. [Online]. Available: http://mitpress.mit.edu/sites/default/files/titles/content/9780262621960{_}sch{_}0001.pdf

[9] M. X. Cohen, *Analyzing Neural Time Series Data: Theory and Practice*. MIT Press, 2014.

[10] M. D. Luciw, E. Jarocka, and B. B. Edin, "Multi-channel EEG recordings during 3,936 grasp and lift trials with varying weight and friction," *Scientific Data*, vol. 1, p. 140047, nov 2014. [Online]. Available: http://www.nature.com/articles/sdata201447

[11] F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, and B. Arnaldi, "A review of classification algorithms for EEG-based braincomputer interfaces," *Journal of Neural Engineering*, vol. 4, no. 2, pp. R1–R13, jun 2007. [Online]. Available: http://stacks.iop.org/1741-2552/4/i=2/a=R01?key=crossref.1f25f98b3df5ef0ed57b44ec26f48caa

[12] J. R. Lourenço, V. Abramova, M. Vieira, B. Cabral, and J. Bernardino, "Nosql databases: A software engineering perspective," in *New Contributions in Information Systems and Technologies*. Springer, 2015, pp. 741–750.

[13] L. Wang, D. Chen, R. Ranjan, S. U. Khan, J. Kołodziej, and J. Wang, "Parallel processing of massive EEG data with MapReduce," *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, pp. 164–171, 2012.

[14] Y. Yao, J. Chang, and K. Xia, "A Case of Parallel EEG Data Processing upon a Beowulf Cluster," in *2009 15th International Conference on Parallel and Distributed Systems*. IEEE, 2009, pp. 799–803. [Online]. Available: http://ieeexplore.ieee.org/document/5395346/

[15] Amazon Web Service . (2018) Amazon s3. [Online]. Available: https://aws.amazon.com/s3/

[16] MongoDB. (2019) Mongodb for giant ideas. [Online]. Available: https://www.mongodb.com/

[17] A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, L. Parkkonen, and M. S. Hämäläinen, "MNE software for processing MEG and EEG data," *NeuroImage*, vol. 86, pp. 446–460, feb 2014. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/24161808

[18] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[19] Apache Spark, "Apache spark: Lightning-fast cluster computing," 2019. [Online]. Available: http://spark.apache.org

[20] J. Ma, A. Ovalle, and D. M.-k. Woodbridge, "Medhere: A smartwatch-based medication adherence monitoring system using machine learning and distributed computing," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2018, pp. 4945–4948.

[21] A. Howard, T. Lee, S. Mahar, P. Intrevado, and D. Woodbridge, "Distributed data analytics framework for smart transportation," in *IEEE 16th International Conference on Smart City*. IEEE, 2018, pp. 1374–1380.

[22] C. Dong, L. Du, F. Ji, Z. Song, Y. Zheng, A. Howard, P. Intrevado, and D. Woodbridge, "Forecasting smart meter energy usage using distributed systems and machine learning," in *IEEE 16th International Conference on Smart City*. IEEE, 2018, pp. 1293–1298.

[23] P. Deyhim, "Best practices for amazon emr," *Technical report*, 2013.

[24] Amazon Web Service. (2018) Overview of amazon emr architecture. [Online]. Available: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview-arch.html

[25] J. Jiaen Liu, C. Perdoni, and B. Bin He, "Hand movement decoding by phase-locking low frequency EEG signals," in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 2011. IEEE, aug 2011, pp. 6335–6338. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/22255787http://ieeexplore.ieee.org/document/6091564/

[26] Z. J. Koles, M. S. Lazar, and S. Z. Zhou, "Spatial patterns underlying population differences in the background EEG." *Brain topography*, vol. 2, no. 4, pp. 275–84, 1990. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/2223384

[27] B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, and K. R. Müller, "Optimizing spatial filters for robust EEG single-trial analysis," *IEEE Signal Processing Magazine*, vol. 25, no. 1, pp. 41–56, 2008.

[28] M. Grosse-Wentrup and M. Buss, "Multiclass Common Spatial Patterns and Information Theoretic Feature Extraction," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 8, pp. 1991–2000, aug 2008. [Online]. Available: http://ieeexplore.ieee.org/document/4473042/

[29] A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, and M. Hämäläinen, "MEG and EEG data analysis with MNE-Python," *Frontiers in Neuroscience*, vol. 7, p. 267, dec 2013. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fnins.2013.00267/abstract

[30] S. H. Walker and D. B. Duncan, "Estimation of the Probability of an Event as a Function of Several Independent Variables," *Biometrika*, vol. 54, no. 1/2, p. 167, jun 1967. [Online]. Available: https://www.jstor.org/stable/2333860?origin=crossref

[31] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[32] C. Cortes, C. Cortes, and V. Vapnik, "Support-Vector Networks," *MACHINE LEARNING*, vol. 20, pp. 273—-297, 1995. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.9362

[33] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. Springer, 2009.

[34] D. K. Gifford, "Weighted voting for replicated data," in *Proceedings of the seventh ACM symposium on Operating systems principles*. ACM, 1979, pp. 150–162.

[35] J. Stastny, P. Sovka, and A. Stancak, "EEG signal classification," in *Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2001, pp. 2020–2023. [Online]. Available: http://ieeexplore.ieee.org/document/1020628/

[36] A. Barachant and R. Cycon, "alexandrebarachant/Grasp-and-lift-EEG-challenge: Code and documentation for the winning sollution to the Grasp-and-Lift EEG Detection challenge," 2016. [Online]. Available: https://github.com/alexandrebarachant/Grasp-and-lift-EEG-challenge