2023 IEEE 24th International Conference on Information Reuse and Integration for Data Science (IRI) | 979-8-3503-3458-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/IRI58017.2023.00045

# A Serverless Electroencephalogram Data Retrieval and Preprocessing Framework

Bathsheba Farrow
*Department of Computer Science*
*Old Dominion University*
Norfolk, VA
bfarrow@cs.odu.edu

Sampath Jayarathna
*Department of Computer Science*
*Old Dominion University*
Norfolk, VA
sampath@cs.odu.edu

***Abstract*—Electroencephalogram (EEG) research continues to rely heavily on data silos used in isolated physical lab environments. However, as a part of the digital transformation, the EEG community has begun its exploration of the public cloud to determine how it can be best utilized to increase collaboration and accelerate research outcomes. The growing number of online repositories for data and tools has provided additional computational resources but the process of downloading data and software along with the installation and configuration requirements is cumbersome and prone to error. To break away from this research paradigm, we present a novel application of cloud technologies to provide reusable EEG data acquisition and preprocessing software as a service (SaaS) that eliminates data and software downloading prerequisites. We utilize the Amazon Web Services (AWS) cloud platform and serverless technologies to create a distributed, highly scalable and extensible solution for EEG signal data preprocessing that is more conducive to effective collaboration and data reproducibility with the potential to expedite neurotechnology breakthroughs.**

***Index Terms*—Electroencephalography, Pipeline, Serverless, Microservices, Software Reuse, PREP, SaaS, AWS**

## I. Introduction

Electroencephalography (EEG) data has proven useful in providing insight into various mental and physical health conditions as well serving as the primary neuroactivity modality for brain-computer interface (BCI) systems [1], [2]. However, EEG data is very susceptible to signal loss or discontinuities as well as contamination from electrical noise, electrode malfunction or misplacement, eye movements, teeth grinding, cardiac activity, and other non-brain artifacts even when methods are taken during research to reduce their occurrence [3], [4]. These artifacts must be removed from the data while minimizing neural signal loss so that it is suitable for subsequent feature extraction and analysis for the various applications. The EEG data cleansing process is very common because of this but still requires considerable domain knowledge to perform this task successfully [5], [6]. The use of more automation for EEG signal data preprocessing and analysis in healthcare and other domains has the potential to accelerate health assessments and research outcomes, support reproducible results [7], and reduce or eliminate bias in comparison to more manual data collection and analysis techniques [8], [9].

There have been multiple software libraries, plugins, and tools introduced to read and preprocess EEG data in a wide range of file formats in hopes to produce more consistent results within the EEG research domain. Such software solutions include EEGLAB [10], Brain Vision Analyzer [11], and MNE suite [12] in addition to other commercial, open source, and custom tools written in programming languages like Python or R. Researchers have also developed pipelines using these tools to standardize processing. These software solutions are usually shared in online repositories such as Github from which users must download the source code, properly configure their environment, and compile and/or install the software before use. Some libraries may even require at least intermediate level programming skills to actually utilize. Tools built on top of software applications like MATLAB may simplify or eliminate programmatic tasks associated with EEG signal data processing and/or automatically generate code [13]. However, the need to properly configure these tools with a wide variety of parameterization options remains, resulting in a significant learning curve, especially for individuals less-experienced in the domain.

Cloud-based software as a service (SaaS) can eliminate the need for users to download and install applications on individual computers. With SaaS, users do not have to concern themselves with this nor debugging potential application and environmental technical issues. There has been some efforts focused on the creation of cloud-based environments to advance EEG research. Unfortunately, most of the cloud platforms that we have come across either are no longer deployed without any available code base to build upon or the platform has not been made publicly available. Yet, with the rapid evolution of cloud technologies, a more unified research approach into the applications of cloud capabilities for EEG data preprocessing and analysis is necessary for the community to keep pace with cloud technology development, encourage technology adoption, and increase collaboration. Some of the most recent trends in cloud computing include significant increases in artificial intelligence (AI) and serverless technologies [14], [15]. This raised the question, how can the serverless technologies be applied to reduce the inefficiencies and burden of the manual tasks associated EEG data preprocessing?

In response and as an alternative to independent, time intensive research to perform repetitive EEG data prepro-

979-8-3503-3458-6/23/$31.00 ©2023 IEEE
DOI 10.1109/IRI58017.2023.00045

cessing tasks, we propose a more simplified and automated method for EEG signal data processing that leverages the accessibility, scalability, and distributed computing nature of the public cloud. We develop an EEG preprocessing capability as a microservice and deploy it using Amazon Web Services (AWS) to provide reusable SaaS that reduces the time and resources that researchers, especially those new to the domain, spend installing, managing, and manipulating software and software applications to replicate these common tasks. Unlike other cloud-based efforts that relied heavily on the use of virtual compute, storage, and networking infrastructure, or infrastructure as a service (IaaS), we developed this cloud based application using microservices and serverless technologies. The serverless technologies used in this work support quicker deployments while reducing operational overhead by leaving provisioning, scaling, and management of the server infrastructure to the cloud service provider.

## II. Related Work

### A. Localized EEG Data Preprocessing

First released in 1997, the MATLAB based EEGLAB toolbox has been well maintained and provides a significant amount of EEG data preprocessing capabilities including high band pass filtering, line noise removal, bad channel rejection, interpolation of removed channels, and rereferencing [10]. Due to the numerous parameterization options that can be especially confusing to those less familiar with the tool, EEG data preprocessing pipelines have been developed that can be run locally to streamline and standardized the usage of EEGLAB and other like tools.

The standardized preprocessing pipeline (PREP) [16] is one well-known solution based on EEGLAB that was designed for large-scale EEG analysis. It was intended to eliminate as much noise as possible while preserving the signal and retaining the resulting dataset in an EEGLAB structure that can be utilized by a variety of applications. The pipeline incorporates the EEGLAB cleanline plugin to remove line noise as shown in Figure 1 as well as the other functions. PREP has been utilized for EEG data preprocessing in a number of research applications, such as EEGNet [17], before any subsequent machine learning or other analysis techniques are applied.

The Harvard Automated Processing Pipeline for EEG (HAPPE) [18] is based on EEGLAB and capable of reading 64 and 128 channel resting-state EEG data from Electrical Geodesics, Inc. (EGI) files. The code used to implement HAPPE could be modified to also import other file types. The pipeline uses both independent component analysis (ICA) with automated component rejection as well as wavelet-enhanced thresholding ICA (W-ICA) to improve the end results [18]. HAPPE also generates quality metrics in a post-processing report unlike many other pipelines created before it [18].

Other pipelines implemented with EEGLAB include Automagic [19] and the Maryland analysis of developmental EEG (MADE) [20], both of which cite improvements over predecessor pipelines. There are also commercial solutions that support EEG processing and analysis like the NeuroPype platform [21]. At the same time, some pipeline developers have steered away from the usage of MATLAB based tools in order to build fully open source solutions. The MNE software libraries for Python and C provide an open source alternative for EEG data processing and analysis capabilities [12]. A Python-based version of the earlier PREP pipeline, pyprep, was built using the MNE Python library, which better enables reuse and automation given that users do not have to purchase any additional software licensing to incorporate it [22]. Nevertheless, all of these localized EEG processing solutions require users to download and install software or incorporate libraries into custom application code. These approaches require users to not only have an understanding of EEG signal data, but also be proficient in properly installing, configuring, programming, and/or invoking function calls in the proper sequence to perform a fundamental step in EEG signal data analysis.

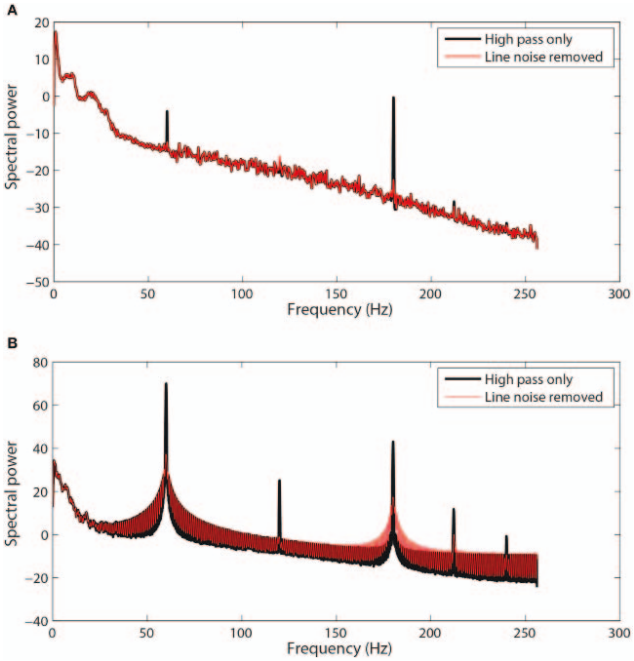

Fig. 1: Selected channel spectra before and after line noise removal using PREP Pipeline [16]

### B. Cloud-Based EEG Data Preprocessing

Cloud services can provide powerful computing resources that are globally accessible and scalable on demand, which has driven an increase in its interest for EEG research and in other domains [23]. Computing power and storage provided on demand in the cloud also facilitates big data analytics, which require efficient high performance processors to produce timely results. There are some studies that document cloud technology usage for EEG data processing and analysis.

Hosseini *et al.* [24] used AWS infrastructure to establish a cloud-based solution for epileptic seizure prediction. The application used RESTful web services to transfer data from

a local environment to the cloud for processing and analysis [24]. Amazon Elastic Compute Cloud (EC2) virtual servers were incorporated into the solution to perform unsupervised feature extraction and classification in that work [24].

A year later, the BrainBase cloud-based research platform was established to support EEG data management [25]. It is built using AWS and incorporates the use of Amazon EC2 virtual servers, a PostgreSQL database, Amazon Simple Storage Solution (S3), and Amazon Simple Queue Service (SQS) to handle file processing. The initial version also provided clinical file parsing capabilities and a protocol library. However, further details of its implementation and processing capabilities have not been made available to the public.

Rushambwa *et al.* [26] developed a system to perform EEG signal data processing and analysis for epileptic seizure detection. The system completes EEG data acquisition and processing in a local environment before transferring it to a private channel in ThingSpeak.com, an Internet of things (IoT) analytics platform service for aggregating, visualizing, and analyzing live data streams in a cloud environment [26]. Once in the cloud, feature extraction and analysis is performed to classify data sets as epileptic or non-epileptic.

However, in just the few years, the use of serverless technologies has significantly increased in industry [15]. Initially, serverless technologies centered largely around the use of function as a service (FaaS) like AWS Lambda and Google Cloud Functions, which can run code written in multiple programming languages in response to events. Microservices were viewed as an architecture different from the serverless paradigm [27], [28]. However, FaaS often has execution time limitations to prevent run-away functions, making them not appropriate for longer running services and/or more complex applications. The introduction of serverless container options like Amazon Fargate enabled the auto-scaling of microservices and expanded the list of available serverless technologies [29].

## III. Methodology

We selected AWS as the platform on which to deploy our EEG data retrieval and preprocessing software capabilities to produce a highly scalable system. Rather than requiring users to download data and subsequently upload it into another application as often performed during EEG research, the microservices developed in this work retrieve selected data files directly from cloud-based data storage repositories as later described. The Python-based Flask micro web framework is used to create the web applications deployed to AWS. It is a lightweight, scalable option known to be easier to use in comparison to other frameworks. All software development was performed on a laptop running Windows 11 with Python, Visual Studio (VS) Code, Docker Desktop, and AWS Command Line Interface (CLI) installed.

As seen in Figure 2, we use several serverless services including Amazon Fargate, Lambda, and DynamoDB, which are further described in the subsections that follow. Serverless technologies eliminate the need for infrastructure management tasks like capacity provisioning and patching.

### A. Data Sources and Management

EEG signal data is retrieved from Amazon S3 buckets, which is Amazon's blob storage solution. There are two sources of raw data in this initial work. Raw data obtained from a prior study [30] was uploaded to an internal S3 bucket to make it available for use. In addition to this, OpenNeuro provides access to raw EEG data files via its website, its Python library, and an open access Amazon S3 bucket [31]. We programmatically obtain data [32] from the OpenNeuro S3 bucket to demonstrate a consistent method of retrieving raw EEG data between internal and external sources. The OpenNeuro datasets are stored in accordance with the Brain Imaging Data Structure (BIDS) standard for file storage and sharing [31]. While our implementation does not currently enforce strict adherence to the BIDS format, it does expect that a directory for each study is stored in the root of the S3 bucket, with each containing separate sub-directories for every subject in the study. A third S3 bucket was created to store files resulting from completed EEG data preprocessing tasks.

An Amazon DynamoDB NoSQL database is used as a part of the overall solution. A key-value database table, FileProcessingTask, was created for storing and tracking the status of preprocessing tasks submitted. A user was created through the AWS Identity and Access Management (IAM) service with the AmazonDynamoDBFullAccess AWS managed permission policy as well a custom policy to permit the creation of database access keys. The access keys were assigned to the user to support programmatic interactions. DynamoDB stream service was enabled on FileProcessingTask table through the AWS Management Console. This stream is used to trigger file processing as later described.

Each task in the FileProcessingTask table can currently have one of four statuses assigned to them:

- Pending: Assigned to new EEG file preprocessing task.
- Preprocessing: The task was received by the EEG Preprocessing service.
- Completed: All steps in the pipeline have been completed.
- Error: A critical failure occurred preventing completion.

### B. User Interface

The User Interface (UI) was created primarily using the Flask micro web framework, WTForms, and Bootstrap web design front-end framework. Amazon Elastic Beanstalk was used to deploy it to the cloud. The UI is composed of three main pages: a home/introductory page, a page used to submit file processing requests, and a page for viewing the status of requests and downloading the resulting files. All three of these pages are accessible from the menu at the top of each page. The Process File page permits users to select the data source, the internal S3 bucket or OpenNeuro, and subsequently select a study then one or more subjects from the study to be processed. Additional user parameter options include the montage, which is the arrangement of EEG channels used during the recordings, and signal filtering settings.

When a file processing request is submitted, a record is immediately added to the FileProcessingTask DynamoDB

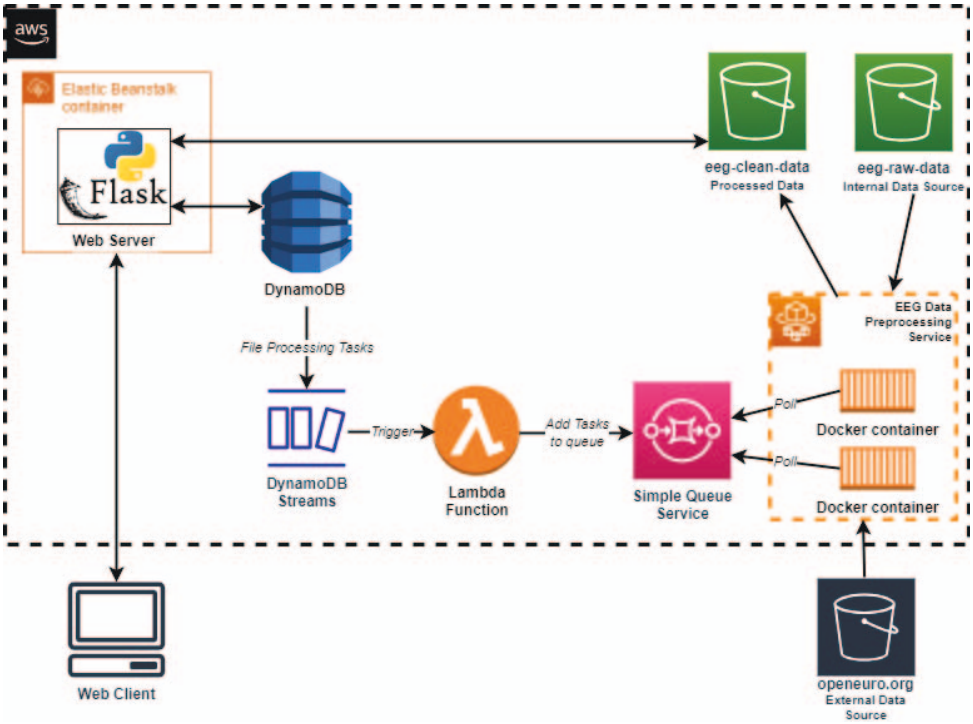


Fig. 2: Serverless EEG Data Preprocessor Framework

table with a unique identifier created for the request. The Amazon SDK for Python, Boto3, is used to perform basic create, retrieve, update, and delete (CRUD) operations on DynamoDB tables. Amazon API Gateway could be used to handle the authentication of requests and database updates but was not included in this simplified implementation. The user is immediately notified that the request has been successfully captured in the database and can submit additional requests or navigate to other pages while waiting for the file processing to complete. With none of their local computing resources impacted by the cloud-based data processing, users could also continue with other localized work.

Amazon Lambda, which provides FaaS, is used here to handle events as they are captured on the DynamoDB stream. The Lambda function will determine if an event on the DynamoDB stream is a new record insert signifying a new file processing task. If so, the Lambda function will send the file processing task to the application load balancer (ALB) of the EEG Data Preprocessing Service, as later described, to initiate the EEG data preprocessing. The status of the task is updated to *Preprocessing* in DynamoDB upon successful initiation.

### C. EEG Data Preprocessing Capability

An event-driven microservice was created using Python to provide the signal data preprocessing capability. The service expects an HTTP PUT request with a JSON body containing the task identifier, selected S3 bucket, study identifier, subject identifier(s), and other parameters entered by the user. The application retrieves the selected file(s) from its source S3 bucket using Boto3. `MNE` is used to read data from various EEG file formats, perform downsampling as needed, and generate visualizations [12]. The `pyprep` library is used to clean the EEG signal. Pyprep simplifies the process of filtering, removing line noise, bad channel rejection, and interpolation as shown in Figure 3 [22]. Some initial feature data is also generated using `MNE`, `numpy` and `scipy` libraries.

Fig. 3: Code snippet for PREP pipeline execution

```
prep = PrepPipeline(raw_copy, prep_params, montage)
prep.fit()
# raw_eeg contains processed data if fit() invoked
return prep.raw_eeg
```

While other output formats could be supported, the application currently produces its clean output data file in CSV format. The output file, time and frequency based feature data, and time-domain and power spectral density (PSD) plots are added to a zip file and saved in the `eeg-clean-data` S3 bucket. Object creations in the S3 bucket automatically trigger another Lambda function that is responsible for updating the status of the file processing task to *Completed* in DynamoDB. Once marked as completed, users can download the zip file from S3 using icons on the *View Task* page of the UI.

### D. Service Container Orchestration

All of the microservice's dependencies were saved to a `requirements.txt` file and a `Dockerfile` was added

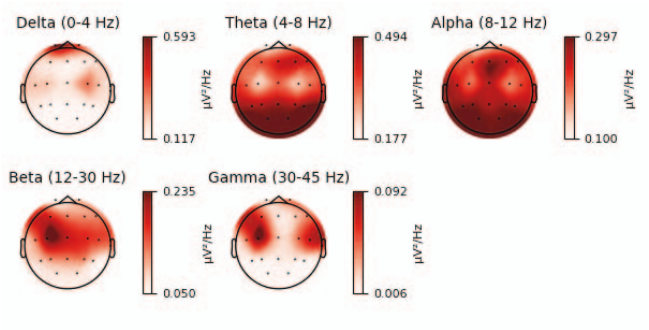


Fig. 4: A PSD plot generated by the EEG Data Preprocessing service

to the program directory to support the creation of a Docker container image. The Python official Docker image is used, instead of building an image from a Linux distribution, since it already has all the tools and packages needed to run a Python application. Commands to build the Docker image, tag it, and push it to a private Amazon Elastic Container Registry (ECR) repository were executed in a VS Code terminal window.

Amazon Elastic Container Service (ECS) is an AWS container orchestration service used here to create a serverless solution. A new ECS cluster managed by AWS Fargate was created. A new Fargate task definition was then created that referenced the URI of the Docker image previously pushed to the ECR repository. In this task definition for the containers, 2 virtual CPU (vCPU) and 4 GB of memory was reserved. An ALB was also created to distribute incoming application traffic across multiple containers. The inbound traffic rules for the security group assigned to the ALB were modified to permit traffic on the HTTP port. The task definition and load balancer were used to configure and deploy a new service in the ECS cluster. The service was configured to have a minimum of 3 containers running with autoscaling up to a maximum of 8.

## IV. Results and Discussion

Amazon CloudWatch collects performance and operational data in the form of logs and metrics from ECS, DynamoDB, and Lambda in this application. To evaluate performance, we examined CPU utilization and memory utilization metrics for the EEG Data Processing service while executing several tasks. We also examined the total processing time for each task. In this case, each task will equate to processing files for one subject from a study selected from the OpenNeuro database [32].

Metrics were initially collected after processing the 30 MB data file for one subject from the selected study. The study used a sampling rate of 500 Hz with 10uV/mm resolution [32]. The end-to-end processing was completed within 55 seconds and the service had an average CPU utilization of 2.31% with a maximum CPU utilization of 63.44% as seen in the first spike in Figure 5. We then processed five data files simultaneously. The application took 128 seconds to complete all five tasks as configured. However, CPU utilization spiked to 99% during the execution and remained there for several minutes. Changing the task definition to reserve 4 vCPU and 8 GB of memory of each container lowered total execution time to 103 seconds for the five tasks. We expect that additional performance tuning of the microservices and cloud environment could further improve results.

Although autoscaling was turned on and CPU utilization spiked, CPU utilization was low on average. Therefore, the average CPU utilization threshold had to be lowered significantly before it would trigger the creation of additional containers using that metric. Also, due to the amount of processing time required and/or the amount of time it takes to launch a new container, there appeared to be data loss between the Lambda function and the service when the number of tasks processed simultaneously was increased to 10. To initially combat this issue, the batch size and batch window was increased for the Lambda function that sends data to ECS. However, long running batches could cause the Lambda function to reach its timeout threshold.

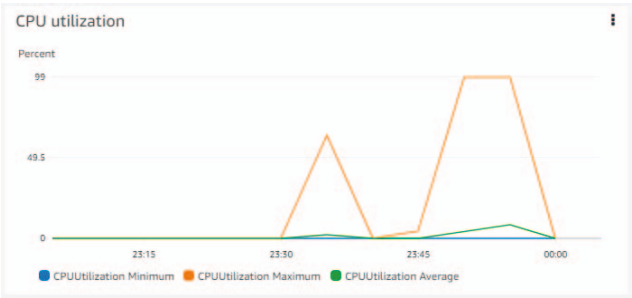


Fig. 5: EEG Data Preprocessing Service CPU Resource Utilization

As a result of the observations, the framework was modified so that the Lambda function sends new tasks to an SQS queue instead of directly to ECS, thereby eliminating the data loss previously encountered. The code comprising the worker container images was modified to long poll the queue for new file processing tasks to complete. Step autoscaling, based on the number of visible tasks in the queue, replaced the CPU based autoscaling. While the EEG preprocessing service containers were initially configured to long poll the queue constantly, current system usage remains low in this developmental stage. Therefore, cloud service costs are reduced by having these workers stop polling the queue if no tasks are found after a several attempts until they are triggered to resume polling again by the Lambda function.

## V. Conclusion and Future Outlook

In this work, a serverless EEG data retrieval and preprocessing framework was developed and examined. Through it, we eliminated data download requirements via direct interfaces with public cloud data repositories. We demonstrated the use of AWS ECS on AWS Fargate and Lambda functions to produce a scalable EEG data preprocessing capability. The initial design using the ALB worked well with a small number

of tasks. However, we found using SQS with polling better handled higher numbers of long running tasks. We showed how increases in computing power in terms of vCPU and memory reduced processing time but could also increase costs. Additionally, we demonstrated how autoscaling can be configured to handle workload increases by creating more worker containers in response to changes in CPU utilization, memory utilization, or even custom metrics like queue size.

We continue to develop enhancements to this framework to include to security improvements, additional error handling, further decoupling of services as well as additional user parameter options for algorithm performance optimization. We also plan to incorporate additional services into the ECS cluster to expand the system's EEG analysis capabilities. Further evaluation of the resulting system in terms of system performance and usability through an approved user study will be completed in the near future.

## ACKNOWLEDGMENT

This work was supported in part by NSF award 2219634. Any opinions, findings, conclusions and/or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsors.

## REFERENCES

[1] V. Gandhi, G. Prasad, D. Coyle, L. Behera, and T. M. McGinnity, "EEG-Based Mobile Robot Control Through an Adaptive Brain–Robot Interface," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 9, pp. 1278–1285, 2014.

[2] S. M. Hosni, M. E. Gadallah, S. F. Bahgat, and M. S. AbdelWahab, "Classification of EEG signals using different feature extraction techniques for mental-task BCI," in *2007 International Conference on Computer Engineering & Systems*. IEEE, 2007, pp. 220–226.

[3] S. Thapaliya, S. Jayarathna, and M. Jaime, "Evaluating the eeg and eye movements for autism spectrum disorder," in *2018 IEEE international conference on big data (Big Data)*. IEEE, 2018, pp. 2328–2336.

[4] D. Haputhanthri, G. Brihadiswaran, S. Gunathilaka, D. Meedeniya, S. Jayarathna, M. Jaime, and C. Harshaw, "Integration of facial thermography in eeg-based classification of asd," *International Journal of Automation and Computing*, vol. 17, no. 6, pp. 837–854, 2020.

[5] G. Brihadiswaran, D. Haputhanthri, S. Gunathilaka, D. Meedeniya, and S. Jayarathna, "Eeg-based processing and classification methodologies for autism spectrum disorder: A review," *Journal of Computer Science*, vol. 15, no. 8, 2019.

[6] D. Haputhanthri, G. Brihadiswaran, S. Gunathilaka, D. Meedeniya, Y. Jayawardena, S. Jayarathna, and M. Jaime, "An eeg based channel optimized classification approach for autism spectrum disorder," in *2019 Moratuwa Engineering Research Conference (MERCon)*. IEEE, 2019, pp. 123–128.

[7] Y. Jayawardana, V. G. Ashok, and S. Jayarathna, "Streaminghub: interactive stream analysis workflows," in *Proceedings of the 22nd ACM/IEEE Joint Conference on Digital Libraries*, 2022, pp. 1–10.

[8] Y. Jayawardana and S. Jayarathna, "Streaming analytics and workflow automation for dfs," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, 2020, pp. 513–514.

[9] B. Farrow and S. Jayarathna, "Technological advancements in post-traumatic stress disorder detection: a survey," in *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE, 2019, pp. 223–228.

[10] A. Delorme and S. Makeig, "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis," *Journal of neuroscience methods*, vol. 134, no. 1.

[11] "Brainvision analyzer (version 2.2.2) [software]," 2021.

[12] 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, 2014.

[13] I. MathWorks, "Machine Learning with MATLAB," 2021. [Online]. Available: https://www.mathworks.com/solutions/machine-learning.html

[14] S. Jayarathna, Y. Jayawardana, M. Jaime, and S. Thapaliya, "Electroencephalogram (eeg) for delineating objective measure of autism spectrum disorder," in *Computational Models for Biomedical Reasoning and Problem Solving*. IGI Global, 2019, pp. 34–65.

[15] B. Marr, "The 5 biggest cloud computing trends in 2022," Nov 2022. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2021/10/25/the-5-biggest-cloud-computing-trends-in-2022/?sh=2f2f955e2267

[16] N. Bigdely-Shamlo, T. Mullen, C. Kothe, K.-M. Su, and K. A. Robbins, "The PREP pipeline: standardized preprocessing for large-scale EEG analysis," *Frontiers in neuroinformatics*, vol. 9, p. 16, 2015.

[17] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance, "EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces," *Journal of neural engineering*, vol. 15, no. 5, p. 056013, 2018.

[18] L. J. Gabard-Durnam, A. S. Mendez Leal, C. L. Wilkinson, and A. R. Levin, "The Harvard Automated Processing Pipeline for Electroencephalography (HAPPE): Standardized Processing Software for Developmental and High-Artifact Data," *Frontiers in Neuroscience*, vol. 12, p. 97, 2018.

[19] A. Pedroni, A. Bahreini, and N. Langer, "Automagic: Standardized preprocessing of big EEG data," *Neuroimage*, vol. 200, pp. 460–473, 2019.

[20] R. Debnath, G. A. Buzzell, S. Morales, M. E. Bowers, S. C. Leach, and N. A. Fox, "The Maryland analysis of developmental EEG (MADE) pipeline," *Psychophysiology*, vol. 57, no. 6, p. e13580, 2020.

[21] "Release notes," 2020. [Online]. Available: https://www.neuropype.io/docs/release

[22] S. Appelhoff, A. Hurst, A. Lawrence, A. Li, Y. M. Ramos, C. O'Reilly, L. Xiang, and D. Jonte, "PyPREP: A Python implementation of the preprocessing pipeline (PREP) for EEG data," 2018.

[23] O. Ali, A. Shrestha, J. Soar, and S. F. Wamba, "Cloud computing-enabled healthcare opportunities, issues, and applications: A systematic review," *International Journal of Information Management*, vol. 43, pp. 146–158, 2018.

[24] M.-P. Hosseini, H. Soltanian-Zadeh, K. Elisevich, and D. Pompili, "Cloud-based deep learning of big EEG data for epileptic seizure prediction," in *2016 IEEE global conference on signal and information processing (GlobalSIP)*. IEEE, 2016, pp. 1151–1155.

[25] T. Thiagarajan, "Brainbase: a research and data management platform for human EEG," in *2017 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*. IEEE, 2017, pp. 1–5.

[26] M. C. Rushambwa, M. Gezimati, P. Govindaraj, R. Palaniappan, V. Vijean, and F. G. Nabi, "Cloud based analysis and classification of EEG signals to detect epileptic seizures," in *2021 Seventh International conference on Bio Signals, Images, and Instrumentation (ICBSII)*. IEEE, 2021, pp. 1–5.

[27] C.-F. Fan, A. Jindal, and M. Gerndt, "Microservices vs Serverless: A Performance Comparison on a Cloud-native Web Application." in *CLOSER*, 2020, pp. 204–215.

[28] R. Shrestha and B. Nisha, "Microservices vs Serverless Deployment in AWS: A Case Study with an Image Processing Application," in *2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*. IEEE, 2022, pp. 183–184.

[29] "AWS documentation," 2023. [Online]. Available: https://docs.aws.amazon.com/

[30] Y. Jayawardana, M. Jaime, and S. Jayarathna, "Analysis of Temporal Relationships between ASD and Brain Activity through EEG and Machine Learning," in *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE, 2019, pp. 151–158.

[31] C. J. Markiewicz, K. J. Gorgolewski, F. Feingold, R. Blair, Y. O. Halchenko, E. Miller, N. Hardcastle, J. Wexler, O. Esteban, M. Goncavles *et al.*, "The OpenNeuro resource for sharing of neuroscience data," *Elife*, vol. 10, p. e71774, 2021.

[32] A. Miltiadous, K. D. Tzimourta, N. Giannakeas, M. G. Tsipouras, T. Afrantou, P. Ioannidis, and A. T. Tzallas, "Alzheimer's disease and frontotemporal dementia: a robust classification method of EEG signals and a comparison of validation methods," *Diagnostics*, vol. 11, no. 8, p. 1437, 2021.