

A Microservices Architecture for Processing Large Electroencephalogram Studies

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

Soo-Yeon Ji
Department of Computer Science
Bowie State University
Bowie, MD
sji@bowiestate.edu

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

Abstract—Electrical signatures characteristic of complex neurological activity and neuropsychiatric disease are embedded in electroencephalography (EEG) signal data. To firmly establish new correlations between these brain electrical pulses and cognition, behavior, and disorders, researchers must achieve adequate statistical power to validate and mitigate uncertainties in their findings. This necessitates the usage of extensive studies involving large volumes of raw EEG data files from multiple subjects, data which must be preprocessed before conducting further analysis. While conventional processing and analysis of these raw data have been performed using isolated physical lab environments and stovepiped applications, there is a growing necessity for processing and analysis solutions that enable distributed processing of large data collections.

This study presents a novel microservices approach as an alternative and complementary solution for retrieving and preprocessing EEG signal data. The approach leverages serverless technologies to deliver a highly scalable solution for processing massive amounts of EEG data. Deployed within a public cloud environment, we assess the efficacy of this method when employing various container orchestration configurations. This work demonstrates the capability for substantial enhancements in processing speeds, particularly when dealing with extensive EEG datasets.

Index Terms—Electroencephalography, Serverless, Microservice, SaaS, Big Data

I. INTRODUCTION

Electroencephalography (EEG) has remained a front-running neuroactivity modality for brain-computer interface (BCI) systems for a number of applications and has seen significant growth in recent years with the emergence of consumer-grade EEG signal data recording headsets and other wearable sensors [1], [2]. EEG provides a non-invasive direct and instantaneous measure of electrical brain activity that has proven effective in characterizing brain dynamics both in healthy and pathological conditions [3], [4]. Combined with artificial intelligence (AI), EEG has enabled the translation of neural patterns into communication and control commands, thereby allowing disabled patients to control neuro-prosthetics and utilize synthetic telepathy [5]–[7]. News of such breakthroughs has helped re-energize researchers in the domain and fueled added academic and industrial interests in the field of Neurotechnology. This has particularly expanded EEG research from a primarily neuropsychiatric focus to multidisciplinary investigations of a wider variety of research questions

such as adaptive learning [8], virtual reality [9], immersive user experience analysis and assessments [10], neurofeedback [11], mindfulness, and neurorehabilitation [11].

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, and cardiac activity [12], [13]. Consequently, successful research depends on proper artifact removal while minimizing neural signal loss so that the resulting data is suitable for subsequent feature extraction and analysis. Researchers must also collect enough data to achieve the desired statistical power. Therefore, the fundamental building blocks of EEG data analysis include data acquisition and preprocessing, which should not be minimized [14].

While this makes EEG data preprocessing a necessity and common practice, much variation still exists in current approaches with considerable domain knowledge required to perform this task successfully [15], [16]. Individual labs may rely on domain experts to manually craft custom EEG preprocessing pipelines, while others utilize more standardized software or other well-established tools found in online repositories. More manual approaches may even add a heavy reliance on visual inspections. Yet, even when using many existing desktop software applications, the ability to efficiently process large volumes of EEG signal data is not present. This contributes to inefficiencies in the domain, casting a shadow over EEG’s enormous potential to support more automated healthcare and research solutions. Fortunately, there has been some recognition of the challenges associated with EEG data acquisition and preprocessing, pushing desires to limit or even banish practices that only serve as a hindrance to research breakthroughs.

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 of producing more consistent results within the EEG research domain. Such software solutions include EEGLAB [17], Brain Vision Analyzer [18], and MNE suite [19] 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, which users must download to their local computers, properly configure their environments, and install and/or compile the software before use. Some research in the use of parallel and distributed computing approaches for EEG data preprocessing has also been documented. Apache Hadoop and Spark have become well-known for large-scale data processing and analytics. Their applications in EEG data preprocessing and analysis have been explored as promising solutions to its big data issues [20], [21]. However, with the rapidly increasing cloud computing trends, even big data researchers have started realizing how the cloud and alternatives to their common design patterns can not only solve some of the big data issues but also help unite distributed teams, eliminate hardware maintenance, and even accelerate the development of solutions. One major cloud computing trend over the past few years has been an increase in the use of microservices and serverless technologies. Microservices decompose large systems into smaller, loosely coupled services that can scale to support an increase in network traffic, resource-intensive machine learning tasks, or even other applications in big data [22]. A second is serverless technologies, which serve as abstractions that hide the details of program execution and container orchestration from developers [23].

Past efforts focused on the creation of cloud-based environments to advance EEG research have been minimal in comparison to other domains. This state of affairs is only aggravated by cloud platforms that were not sustained for future work and those that are not made available to others in the EEG research community. Yet, with the rapid evolution of cloud technologies, a more unified research approach to 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, increase collaboration, and expedite solutions. This includes exploration of the most recent trends in cloud computing, such as deep learning and serverless technologies [24]. With all of this in mind, we questioned how can cloud and serverless technologies be applied to reduce the inefficiencies and burden of time-intensive, manual tasks associated with EEG data preprocessing. As a result of our contemplation, we proposed a simple and automated method for EEG signal data preprocessing that leverages the accessibility, scalability, and distributed computing nature of the public cloud [25]. We further enhanced its microservices architectural design that incorporates serverless functions and containers to standardize and expedite EEG signal data preprocessing for large-scale studies. With this, our contributions herein include:

- 1) A new microservices approach to EEG signal data preprocessing that supports large-scale EEG studies.
- 2) The use of function-as-a-service (FaaS) in an EEG processing pipeline to distribute work amongst multiple containers.
- 3) The use of serverless containers and autoscaling to handle increases the amount of data being processed in dynamic environments.

- 4) A programmatic approach to EEG data retrieval from cloud-based data repositories to eliminate unnecessary downloads or other data handling.
- 5) To highlight the benefits of our approach over traditional approaches, we collect and evaluate system performance metrics, including with and without the use of autoscaling.

Our solution is unlike other cloud-based approaches that rely heavily on virtual computing, storage, and networking infrastructure. This approach provides EEG data preprocessing software-as-a-service (SaaS) to relieve users of software and pipeline complexities and administrative burdens, ultimately enhancing the overall user experience. The serverless deployment of microservices developed in this work provides the scalability required to handle large-scale EEG studies and expedite results.

II. RELATED WORK

A. Localized EEG Data Preprocessing

The need to efficiently process large EEG datasets has been recognized for some time. Still, much of that processing occurs serially via pipelines, desktop software applications, and code shared via online repositories. First released in 1997, a popular option has remained the MATLAB-based EEGLAB toolbox, which provides a significant amount of raw EEG signal data preprocessing capabilities, including high bandpass filtering, line noise removal, bad channel rejection, interpolation of removed channels, and rereferencing [17], [26]. However, 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 to streamline and standardize the usage of EEGLAB and other tools.

Several standardized pipelines based on EEGLAB were developed with the intention for them to be suitable for processing large studies. The standardized preprocessing pipeline (PREP) [27] is one such solution that was designed for large-scale EEG data analysis. The pipeline was created 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. It incorporates the EEGLAB *cleanline* plugin to remove line noise, as shown in Figure 1, as well as the other EEGLAB functions. PREP has been utilized for EEG data preprocessing in a number of research applications, such as EEGNet [28], before any subsequent machine learning or other analysis techniques are applied.

Another pipeline developed with EEGLAB, the Harvard Automated Processing Pipeline for EEG (HAPPE) [29], is capable of reading 64 and 128-channel resting-state EEG data from Electrical Geodesics, Inc. (EGI) file format. Independent component analysis (ICA) with automated component rejection as well as wavelet-enhanced thresholding ICA (W-ICA) are used in this pipeline to achieve data quality improvements. HAPPE also generates quality metrics in a post-processing report, an enhancement not available in pipelines created before

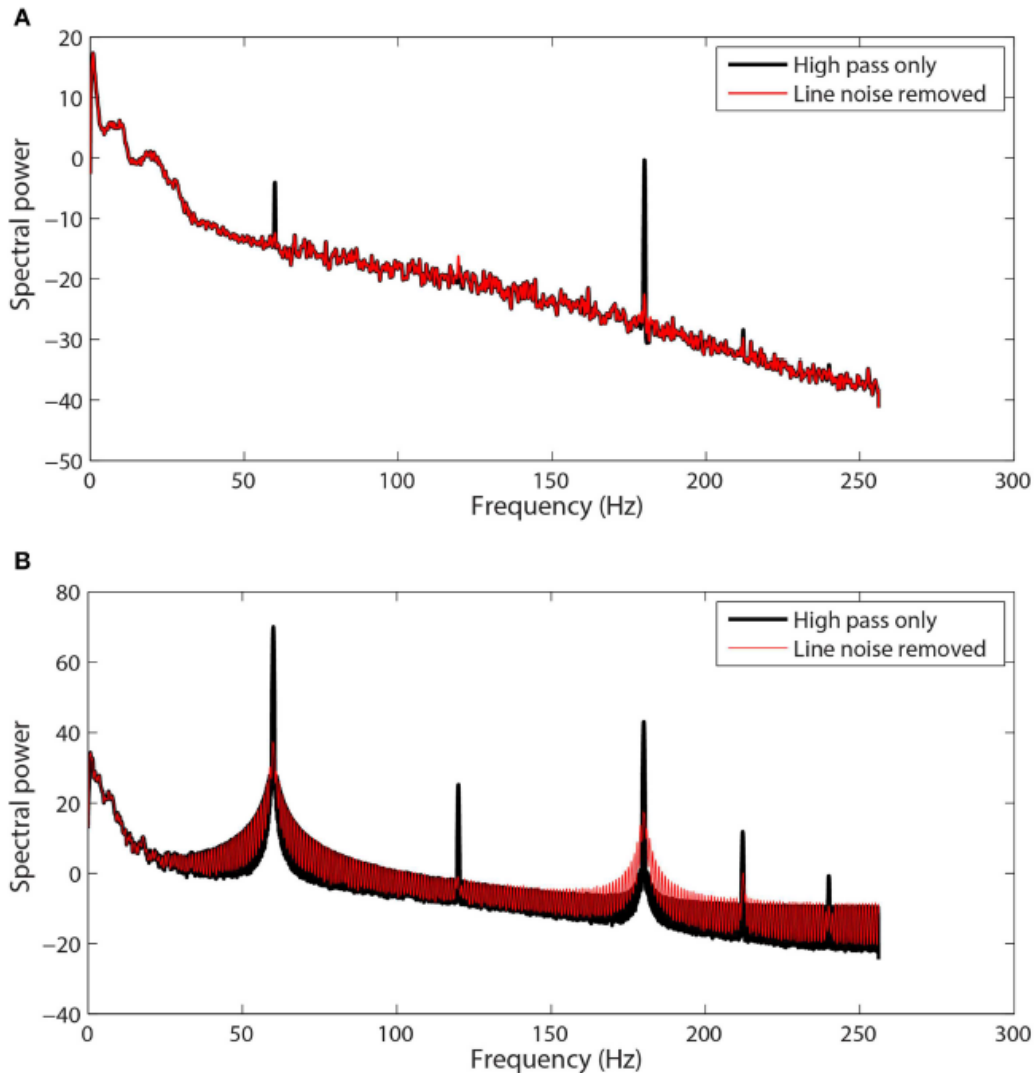


Fig. 1: The PREP pipeline incorporates the EEGLAB *cleanLine* function in its preprocessing pipeline. (A) Selected channel spectra from a 64-channel Biosemi signal before line noise removal. (B) Line noise graph after removing spectral peaks using the PREP Pipeline [27].

it. Other EEG data preprocessing pipelines implemented with EEGLAB functions were published in recent years including Automagic [30], the Maryland analysis of developmental EEG (MADE) [31], and Batch Electroencephalography Automated Processing Platform (BEAPP) [32]. However, these desktop tools were not designed with scalability in mind and their EEGLAB dependence makes MATLAB licensing a requirement.

While there are proprietary commercial solutions available for use, the chosen alternative to EEGLAB is often intensive programming to create custom solutions. The MNE software libraries for Python and C can remove some of the complexities of software development by providing functions for many of the common EEG data preprocessing and analysis tasks [19]. A Python-based version of the earlier PREP pipeline, PyPrep, was built using the MNE Python library [33]. Unlike the version built on EEGLAB, PyPrep better enables reuse

and automation while eliminating MATLAB licensing requirements. However, custom code solutions are often shared online as source code, not as executables. Therefore, code, compilers, and other dependencies must be downloaded, installed, and configured before usage. These approaches require users to not only have a sufficient understanding of EEG signal data processing but also a certain level of technical proficiency that may dissuade some researchers from this option.

B. EEG Big Data Preprocessing

Preprocessing EEG signal data for large studies has somewhat evolved to include the experimental use of conventional big data technologies. Solutions that incorporate Apache Hadoop as the defacto solution for parallel processing have been published in the literature. One such example is the Cloudwave platform for epilepsy research, which enabled distributed processing of EEG data using Hadoop in a private

cloud environment [34]. Adhering to a Model-View-Controller (MVC) architecture, the core system consisted of a Ruby on Rails-based web application coupled with a MySQL database to retain information on user interactions and metadata. The core system was designed to interface with a Hadoop-based module for parallelized processing of EEG signal data processing. Initially, published test results showed that Cloudwave required only a fraction of the time required for processing multiple studies in comparison to their standalone baseline system.

Berrada *et al.* [35] also used core components of Apache Hadoop for medical EEG data storage and processing platform. The group published results from benchmark experiments against their prototype system as feasibility measures. The implementation used the Hadoop Distributed File System (HDFS) to allocate data storage while preprocessing was accomplished through the Hadoop MapReduce framework. The reduction in computation times through the use of Hadoop, as observed by the research group, showed significant improvements over Python parallel processing. However, Hadoop does have its share of limitations including the lack of real-time processing and degraded processing speeds from disk reads and writes [36].

As a more recent technology, Apache Spark tries to overcome Hadoop's drawbacks through the use of memory-based storage and micro-batch processing [36]. It was used in a Spark-based machine-learning approach for epileptic seizure detection implemented via MATLAB, which included the Spark Streaming and Spark Machine Learning library (Mlib) extensions [20]. The Apache Kafka streaming platform was incorporated into the solution to publish batches of patient data to a pipeline that performs preprocessing, feature extraction using an autoregressive (AR) model, and classification via a support vector machine (SVM). Meanwhile, training data is fed to a separate offline batch processing component to further train the classifier. The implementation was tested in MATLAB using data obtained from Physionet, producing an average detection accuracy, sensitivity, and specificity of 99.32%, 99.41%, and 95.25%, respectively. As seen with this system, Spark is often used in conjunction with other technologies and libraries as it does not provide storage or streaming on its own [36]. However, many cloud service providers are beginning to offer Apache Spark as a service.

C. Cloud-Based EEG Data Preprocessing

Cloud environments can provide a wide variety of IT services as well as powerful computing resources that are scalable on demand without upfront investments in infrastructure [37]. Cloud technology can also unite teams spread worldwide, transforming isolated research facilities into a network of virtual labs that can build upon group outcomes. The potential of the cloud has increased interest in its application for EEG research as well as other applications [37]. Public cloud environments are especially advantageous for big data analytics [34], which requires large amounts of data storage along with sufficient processing power to produce timely results.

Despite the advertised benefits, research investments in cloud technology for EEG signal data processing and analysis have not been significant.

Even with the slow adoption of the cloud, there are still researchers who have gotten beyond barriers to cloud-based EEG research. Hosseini *et al.* [38] used Amazon Web Services (AWS) infrastructure to establish a cloud-based solution for epileptic seizure prediction. RESTful web services were developed to transfer clinical intracranial EEG (iEEG) data from a local computing environment to a cloud-based data processing environment. A cluster of Amazon Elastic Compute Cloud (EC2) virtual servers served as the hosts for the unsupervised feature extraction to support seizure prediction with promising results [38].

More recently, the BrainBase cloud-based research platform was established to support EEG data management and facilitate collaboration [39]. The environment enables the uploading and sharing of data, code, and research protocols with other users. It is built in the AWS cloud and incorporates several AWS cloud services including EC2 virtual servers, Amazon Simple Storage Solution (S3), and Amazon Simple Queue Service (SQS) to handle file processing. The initial version also incorporates a PostgreSQL database and provides clinical file parsing capabilities as well as a protocol library. However, further details of how these tools were used in the implementation and processing capabilities were not included in its publication.

Like Hosseini, Rushambwa *et al.* [40] developed a system to perform EEG signal data processing and analysis for epileptic seizure detection. In this group's solution, EEG data acquisition and processing is performed in a local environment before being transferred to a private channel in ThingSpeak.com, a cloud-based Internet of Things (IoT) analytics platform service for aggregating, visualizing, and analyzing live data streams. Once the preprocessed data is in the cloud, a feature extraction and analysis process is used to classify data sets as epileptic or non-epileptic.

While these solutions have incorporated cloud technologies, their focus has been on the use of infrastructure as a service (IaaS). However, in just the last few years, the use of serverless technologies has significantly increased in industry [24]. Initially, it was 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 [41], [42]. However, most FaaS had execution time limitations to prevent run-away functions, making them inappropriate for longer-running services and more complex applications. This likely contributed to the introduction of serverless container options like Amazon Fargate, which enabled the auto-scaling of microservices and expanded the idea of serverless technologies [23].

There remains a heavy dependence on tried-and-true solutions like EEGLAB and EEGLAB-based pipelines for EEG processing and analysis. Nevertheless, a shift in the EEG

research paradigm has begun with the incorporation of potentially revolutionary technologies enabling big data analytics, artificial intelligence, and cloud computing. Yet, with constant technological advancements in the industry, some of the very latest trends have yet to be fully examined in the EEG research arena. A focus is needed on using these capabilities to provide scalable and extensible EEG processing and analysis tools for a variety of applications to meet the demands of current and future research.

III. METHODOLOGY

We apply and evaluate the use of microservices and serverless technologies into a scalable solution capable of efficiently processing data for large EEG studies [25]. We break down the larger system into smaller services that can be scaled independently. The application’s front-end developed in this work is small and focused. Therefore, it could be incorporated as a micro front-end into a larger application. A containerized application that handles the data file processing requests is able to respond to increases in workload through the Amazon Elastic Container Service (ECS) with the Fargate option. AWS Lambda functions can scale their execution environments as

needed to handle increases in requests. As later described, this serverless capability is used to split and distribute file processing requests as they are consumed by the system.

As seen in Figure 2, we also use several serverless services, including Amazon Fargate with ECS, AWS Lambda, and DynamoDB, that eliminate infrastructure management tasks like capacity provisioning and patching. This allows us to easily establish and scale our system to handle larger data sets with complex configurations. We chose to deploy our EEG data retrieval and preprocessing software capabilities in the AWS public cloud to take advantage of these services without upfront investments in hardware infrastructure. AWS also provides a number of other features, configurations, and services we intend to use, including several services in its free tier [43].

Rather than requiring users to download data and subsequently upload it into another application as documented in other EEG research publications we have reviewed, our application retrieves selected data files for EEG studies directly from cloud-based data storage repositories [25]. The Python Flask micro web framework was used to create the web applications deployed to AWS. It is a lightweight, scalable

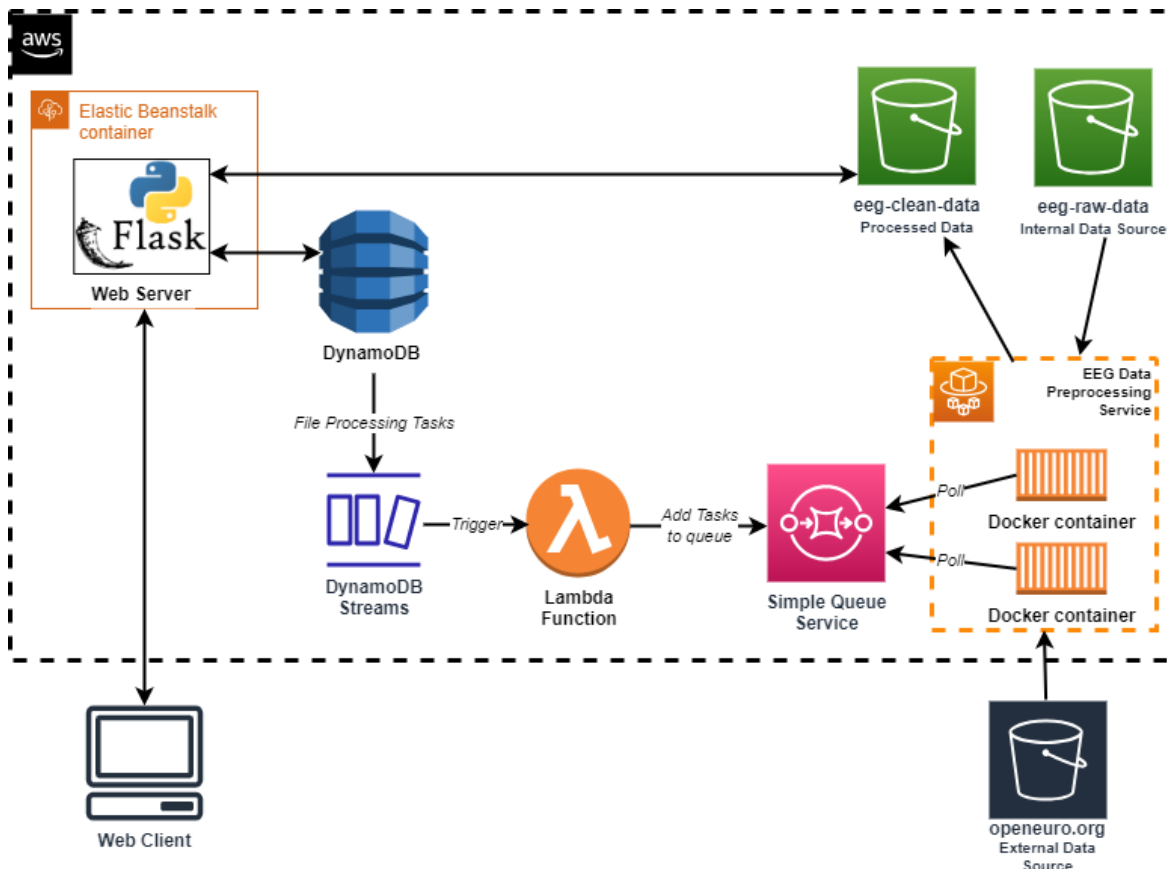


Fig. 2: Cloud-Based EEG Data Preprocessor High-Level Architectural View (© 2023 IEEE). The AWS Lambda function retrieves user requests from the DynamoDB event stream and sends individual file-processing tasks to SQS. ECS will scale the number of containers running based on the number of pending tasks held by SQS. Containers running will long poll SQS for tasks and then retrieve raw files from internal and internal S3 buckets for processing. Processed data is saved to a separate S3 bucket accessible by the UI, which is deployed via Elastic Beanstalk.

option that is suitable for smaller applications and 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 before being deployed to the AWS environment.

A. Data Sources and Management

A benefit of our framework is that it eliminates manual data transfers by users before preprocessing data, unlike other studies that we have come across. Instead, EEG signal data is programmatically retrieved from an internal and external Amazon Simple Storage Solution (S3) bucket [25]. Raw data obtained from an Autism Spectrum Disorder (ASD) study [44] was preloaded in a private S3 bucket to make it available for application users. Containers responsible for processing data have access to the S3 bucket via roles established in the Identity and Access Management (IAM) service. In addition

to the internally maintained data, OpenNeuro provides public access to anonymized raw EEG data files via its website, its Python library, and an open-access Amazon S3 bucket [45]. Our application directly retrieves the OpenNeuro S3 bucket datasets, which are stored in accordance with the Brain Imaging Data Structure (BIDS) standard for file storage and sharing [45]. While our implementation does not currently enforce strict adherence to the BIDS format for data consumption, 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 in our AWS environment to store files resulting from completed EEG data preprocessing tasks.

An Amazon DynamoDB NoSQL (key-value) database is used as a part of the overall solution. In the database, a table, FileProcessingTask, was created for storing and tracking the status of all file preprocessing tasks submitted. The IAM service was used to create a user with the AmazonDynamoDB-

Home Process Files View Tasks

User name

S3 Bucket

Open Neuro

Study

ds004067: Moral conviction and metacognitive ability shape

Subjects

- sub-01
- sub-02
- sub-03
- sub-04

Montage

standard_1020

[Additional Parameters](#)

Process

Fig. 3: EEG Data Preprocessor User Interface - Process Files Page. The page allows users to submit requests to preprocess raw EEG data. Users select the data source (S3 Bucket) then the study and one or more subjects from selection menus dynamically populated based on the S3 bucket selection. The montage is modifiable with additional parameter options displayed using the link.

FullAccess AWS managed permission policy in addition to a custom policy to permit the creation of database access keys. The access keys were assigned to the user to support programmatic interactions while testing outside of AWS. The DynamoDB stream service was enabled on the 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 it:

- 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 UI was created primarily using the Flask micro web framework, WTForms, and Bootstrap web design front-end framework [25]. The UI is composed of three main pages: a Home page, a Process File Request page, and a View File Status page. The Home page provides general information regarding the site. The Process File Request page allows users to select and submit data for processing. It provides options for users to select the data source (internal or OpenNeuro S3 bucket), a study, and one or more subjects from the selected study to for processing (refer to Figure 3. Additionally, the user may modify several parameterization options including the montage, down-sampling rate, and signal filtering settings. The View File Status page presents the list of file processing requests that have been submitted with their status and allows users to download the artifacts produced from processing. All three pages are accessible from the navigation menu at the top of each page. The Amazon Elastic Beanstalk service was used to deploy the front-end application to the cloud without creating a container image.

When a file processing request is submitted through the Process File Request page, a record is immediately added to the FileProcessingTask DynamoDB table with a unique identifier created for the request. Boto3, the Amazon SDK for Python, is used to perform basic create, retrieve, update, and delete (CRUD) operations on DynamoDB table [25]. The user is immediately notified once a the request is successfully captured in the database. Since all of the processing is performed in the cloud, none of the user’s local resources are impacted by data processing. Therefore, the user can submit additional requests, navigate to other pages, or perform other work while waiting for the file processing to complete.

C. EEG Data Preprocessing Capability

An event-driven microservice was created to provide a scalable EEG signal data preprocessing capability [25]. In the current design, the service continuously long polls an SQS queue for file processing tasks. Tasks are received in JSON format containing the task identifier, selected S3 bucket, study identifier, subject identifier, and other input parameters as specified by the user. Based on the inputs, the service retrieves

the appropriate raw data file(s) from its source S3 bucket. The MNE library is then used to read data from any of its support EEG data file formats [19]. The PyPrep library is used to clean the EEG signal. It simplifies the process of filtering, removing line noise, bad channel rejection, and interpolation. The MNE will also be used to perform downsampling as needed and produce visualizations before some feature data is extracted using MNE, NumPy, SciPy, and YASA libraries. The resulting files could be fed into other applications.

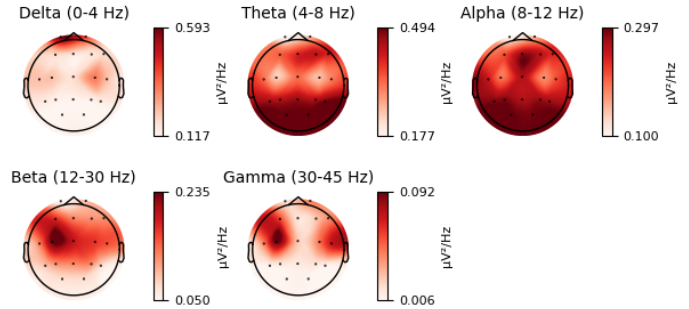


Fig. 4: A Power Spectral Density (PSD) Plot Generated by the EEG Data Preprocessing Service (© 2023 IEEE). The `plot_psd_topomap` method within the `mne.io.Raw` class, as a part of the MNE library, is used to generate plots from preprocessed EEG signal data.

The software dependencies for this EEG data preprocessing capability were saved to a `requirements.txt` file. A `Dockerfile` document was created, and Docker was used to generate a container image based on the Python official Docker image. This Docker container image created for the service was pushed to a private Amazon Elastic Container Registry (ECR) repository by executing the ECR-provided commands via the AWS CLI. Once in ECR, the container image is available for use by other services like the Amazon ECS, which we use with the Fargate cluster option as a serverless container solution to manage our container orchestration [25]. As a container orchestrator, ECS handles container provisioning, deployment, scaling, health status checks, and deletions.

In the previous version of our system, we used an Amazon Lambda function to create file processing tasks for each new record event it encounters on the DynamoDB steam [25]. The function would send these tasks to an application load balancer (ALB), which would distribute these tasks in a round-robin fashion amongst containers running in ECS. As a result of the preliminary observations, modifications to the initial framework were made so that the Lambda function now sends new file-processing tasks to an SQS queue instead of to the ALB. The EEG data preprocessing capability long polls this queue for new file processing tasks. These modifications eliminated the data loss previously observed with our original implementation [25]. To handle larger studies, additional changes were made to the Lambda function so that a separate file processing task for each subject in a request is added to

TABLE I: EEG Data File Processing Times by Configuration. Completion times were captured for each configuration when processing different amounts of data. Configuration 1, which only had a single container running at any time, had the worst performance even though it was allocated the same total amount of resources as Configuration 2.

Total Subjects	Total Data Processed (MB)	Configuration 1	Configuration 2	Configuration 3
		1 Container (8 vCPU & 16 GB RAM) Processing Time	4 Containers (2 vCPU & 4 GB RAM each) Processing Time	2 Containers w/ Autoscaling (2 vCPU & 4 GB RAM each) Processing Time
1	22.90	00:50	00:53	00:52
2	53.21	01:59	01:05	01:08
3	64.97	02:21	01:06	01:20
4	91.99	03:21	01:15	01:51
5	122.71	04:23	01:34	02:17
10	279.72	07:36	02:59	02:43
20	617.16	16:42	05:33	03:16
30	920.80	33:08	08:16	06:43
40	1,258.67	44:33	11:36	09:46

the SQS queue instead of adding a single job per request. This allows the files for one request to be processed concurrently when running multiple containers.

The service saves processed data and files for each subject to a shared directory for a specific user’s request within the `eeeg-clean-data` S3 bucket. This includes time and frequency-based feature data as well as time-domain and power spectral density (PSD) plots. Object creations in the S3 bucket automatically trigger another secondary Lambda function that is responsible for updating the status of the file processing task to *Completed* in DynamoDB once all files have been processed.

D. Framework Evaluation

As a result of previous findings [25], we made modifications to our original framework that we expect to improve performance, especially when processing larger studies. To further assess this implementation, we processed EEG data for different numbers of subjects from an OpenNeuro Dataset ds004504 [46] when using different task configurations for the service. The study has a total of 88 EEG .set data files that range between 11 and 45 MB in size. Initially, the EEG data preprocessing service within the application was configured to have only one container running with 8 virtual CPUs (vCPUs) and 16 GB of memory. We recorded the total processing times for processing different numbers of subject data files as shown in Table I.

We then adjusted the service settings to manually scale the total number of containers to four. The task definition for the containers was also modified to allocate 2 vCPUs and 4 GB of memory. This reduction made the total vCPUs and memory allocated between the four containers the same as the total given to the one container in the first configuration. After running the same set of tests, the service was modified a third time to run eight containers simultaneously. 1 vCPU and 2 GB of memory as allocated to the containers in the task definition for the third configuration. No autoscaling was utilized in any of these cases.

We also assessed the performance of our framework while utilizing autoscaling. The task definition for the service was set

at 2 vCPUs and 4 GB of memory per container. The setting for the minimum number of containers running for the service was set to 2. The service was also configured with custom metrics to respond to the number of tasks waiting in the Amazon SQS queue. If 2 tasks have been waiting in the queue for more than 60 seconds, ECS begins launching 6 additional containers to perform processing for a total of 8. When the number of tasks waiting in the queue reduces to 2, the number of containers running reduces back to only 2. The autoscaling is triggered by alarms we created in Amazon CloudWatch and referenced in the service’s settings in ECS.

IV. DISCUSSION

When processing a single file, configuration 1, which received the highest allocation of resources per container, did not show significant improvements in processing time compared to the other configurations with fewer vCPUs and memory assigned to each container, as observed in Table I. When processing more than one file, the configurations running multiple containers always outperformed the configurations that only used one container. The one exception is when the task definition was modified to run eight containers simultaneously with only 1 vCPU and 1 GB of memory allocated to the containers. The containers could not successfully process the larger files from the study with that configuration, which made that configuration not a viable option so those results were excluded from Table I. However, with sufficient resources, as the number of subjects processed increased, the amount of time saved by processing data using multiple containers increased sharply. This clearly demonstrates the advantages of using microservices in a scalable, event-driven architecture to perform EEG data preprocessing. Although we restricted the maximum number of running containers to 8, the service can be configured to launch many more containers as needed.

When utilizing autoscaling within configuration 3, we did not observe any significant impact on the processing times until at least 10 files were being processed simultaneously. This is due to the fact that the service does not begin launching new containers until multiple tasks have remained in the queue

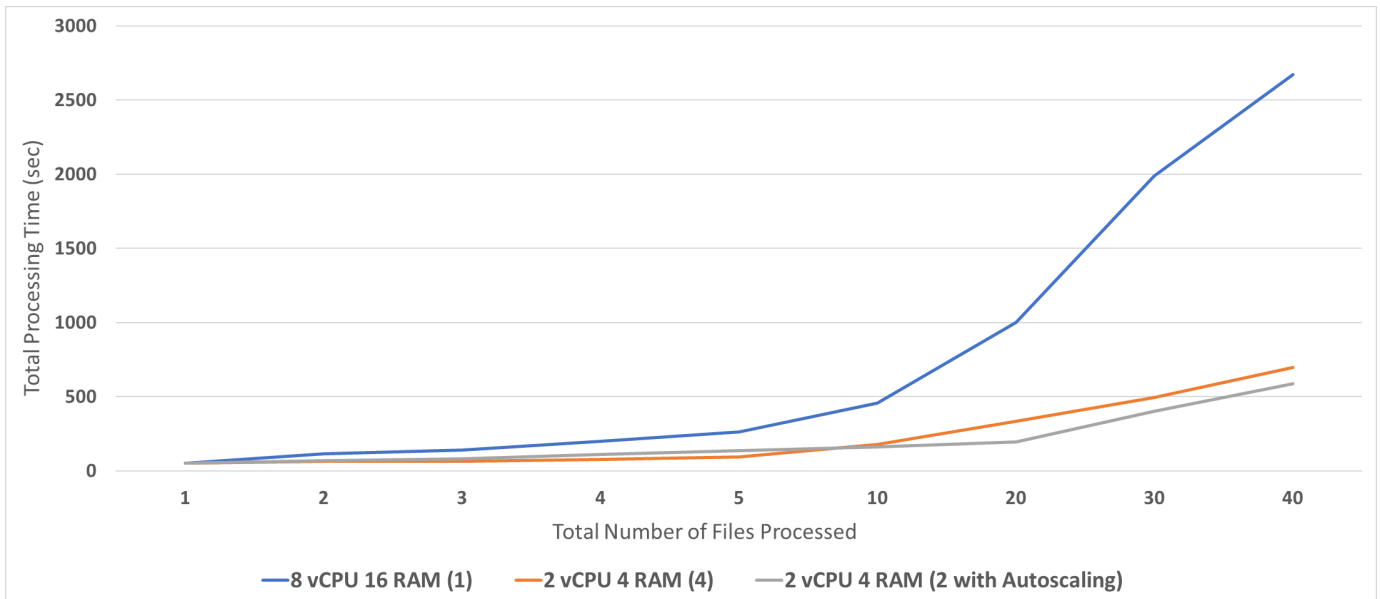


Fig. 5: Total Processing Times (in seconds) by Service Configuration. Several task configurations were used for the service in the ECS and tested with different numbers of files. As seen, processing times for the first configuration (8 vCPU and 16 RAM) drastically increased over the other configurations as the number of files processed rose. The second configuration performed better than configuration 3 (with autoscaling) with the lower number of files where total processing times were too brief for autoscaling to be effective.

for at least 60 seconds. There is also additional time required to provision resources for the new containers and activate them. Therefore, it is at least 2 minutes before the system starts to see any benefit from the use of autoscaling in this implementation. Up until that point, the manually scaled configuration 2, with more containers running at the start, outperformed configuration 3 with autoscaling enabled. Configuration 3 eventually outperforms Configuration 2 once it has provisioned more workers and catches up with processing as depicted in Figure 5. Therefore, the decision to enable autoscaling in a particular environment is dependent on several factors, including the amount and consistency of incoming requests as well as the average processing time. For an environment in the cloud that may see occasional bursts in the number of requests but normally experiences much less usage, autoscaling could lower costs while providing additional processing power when needed most. However, the system must be able to afford the delay in response time expected when provisioning resources in response to increases in traffic or demand.

While utilizing AWS 12-month free-tier services, the only costs incurred when running this system were from ECS, which varied based on configuration. Running 2 containers with 2 vCPU and 4 GB RAM each for 24 hours a day, as in configuration 3 without autoscaling, costs about \$4.62 per day. Running 1 container with 8 vCPU with 16 GB RAM for configuration 2 resulted in costs that were about twice as much. Therefore, reducing the number of containers that are constantly running and using autoscaling could reduce costs. Yet, underestimating the workload could result in the service scaling up the number of containers and running them for

an extended period of time, which could drive costs back up closer to that of configuration 1. A local server running an orchestration tool like Docker Swarm could totally eliminate public cloud costs. However, any discussions on costs should take into consideration those expenses eliminated via the use of the public cloud infrastructure including hardware, software, and other operational costs, especially for shorter-term projects.

The initial design of our system, which used the ALB for distributing work, performed well with a few files. However, we found that EEG study processing times for greater numbers of files could be improved by distributing the files amongst multiple running containers to process. We also used SQS with polling to better handle higher numbers of long-running tasks. Polling the queue instead of constantly polling the database helps to improve the database availability. Additionally, queues are more fault-tolerant and scalable than databases. We configured autoscaling to handle workload increases by creating more worker containers to process files in response to changes in the queue size. Still, as shown, the choice to use autoscaling is dependent upon specifics regarding the application’s usage and performance requirements.

V. CONCLUSION AND FUTURE OUTLOOK

In this work, we developed and demonstrated an EEG data retrieval and preprocessing framework that dramatically reduces the processing time for larger studies. Through our implementation, we also removed data download requirements via the application’s direct interfaces with private and public cloud data repositories. Microservices were developed to help

break the system into more manageable, loosely coupled, scalable units. Serverless technologies sped deployment by simplifying or eliminating many provisioning and configuration requirements. With this system, we demonstrated how manually and autoscaling the back-end service's containers dramatically improved processing times over a single instance of the container, even when both approaches utilized the same amount of resources overall. Our results also indicate that autoscaling may not be best suited for short bursts in traffic due to the time it takes to provision new containers.

While the container image developed in this work could be deployed in a private cloud or even a local computing environment to eliminate public cloud costs, sufficient hardware resources and a container orchestration tool would still be needed to take advantage of autoscaling. With that, the total costs of maintaining any environment for a given use case must be considered to determine the best approach, which can vary significantly. Nevertheless, the framework presented here shows promise for supporting data preprocessing and analysis for large EEG studies and other big data tasks. Yet, the technologies used here do not have to replace other big data analytic solutions but can, rather, complement those approaches. Many cloud service providers now provide big data solutions, which should also be explored. We do plan, however, to continue to refine and validate our own cloud-based approach to EEG signal data processing. We intend to collect additional performance metrics with even larger data sets and look for opportunities for direct comparisons with other EEG-related big data solutions. Further evaluation of the system in terms of system performance as well as usability through an approved user study will be completed in the near future.

VI. DATA AVAILABILITY

The data that support the findings of this study are openly available in OpenNeuro at <https://doi.org/10.18112/openneuro.ds004504.v1.0.6>, reference number ds004504.

ACKNOWLEDGMENT

This work was supported in part by NSF awards 2219634 and 2045523. 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] P. Sawangjai, S. Hompoonsup, P. Leelaarporn, S. Kongwudhikunakorn, and T. Wilaiprasitporn, "Consumer grade EEG measuring sensors as research tools: A review," *IEEE Sensors Journal*, vol. 20, no. 8, pp. 3996–4024, 2019.
- [2] R. Riedl, R. K. Minas, A. R. Dennis, and G. R. Müller-Putz, "Consumer-grade EEG instruments: insights on the measurement quality based on a literature review and implications for NeuroIS research," *Information Systems and Neuroscience: NeuroIS Retreat 2020*, pp. 350–361, 2020.
- [3] J. J. Newson and T. C. Thiagarajan, "EEG frequency bands in psychiatric disorders: a review of resting state studies," *Frontiers in human neuroscience*, vol. 12, p. 521, 2019.
- [4] M. McVoy, S. Lytle, E. Fulchiero, M. E. Aebi, O. Adeleye, and M. Sajatovic, "A systematic review of quantitative EEG as a possible biomarker in child psychiatric disorders," *Psychiatry Research*, vol. 279, pp. 331–344, 2019.
- [5] 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.
- [6] D. Bansal and R. Mahajan, *EEG-based brain-computer interfacing (BCI)*. Netherlands: Elsevier Science, Jan. 2019.
- [7] J. Tang, A. LeBel, S. Jain, and A. G. Huth, "Semantic reconstruction of continuous language from non-invasive brain recordings," *Nature Neuroscience*, pp. 1–9, 2023.
- [8] P.-C. Hu and P.-C. Kuo, "Adaptive learning system for E-learning based on EEG brain signals," in *2017 IEEE 6th Global Conference on Consumer Electronics (GCCE)*. IEEE, 2017, pp. 1–2.
- [9] T. D. Parsons, T. McMahan, and I. Parberry, "Classification of video game player experience using consumer-grade electroencephalography," *IEEE Transactions on Affective Computing*, 2020.
- [10] C. M. Perchtold-Stefan, I. Papousek, C. Rominger, M. Schertler, E. M. Weiss, and A. Fink, "Humor comprehension and creative cognition: Shared and distinct neurocognitive mechanisms as indicated by EEG alpha activity," *NeuroImage*, vol. 213, p. 116695, 2020.
- [11] N. Du Bois, A. D. Bigirimana, A. Korik, L. G. Kéthina, E. Rutembesa, J. Mutabaruka, L. Mutesa, G. Prasad, S. Jansen, and D. Coyle, "Neurofeedback with low-cost, wearable electroencephalography (EEG) reduces symptoms in chronic post-traumatic stress disorder," *Journal of Affective Disorders*, vol. 295, pp. 1319–1334, 2021.
- [12] 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.
- [13] 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.
- [14] S. De Silva, S. Dayarathna, G. Ariyaratne, D. Meedeniya, and S. Jayarathna, "A survey of attention deficit hyperactivity disorder identification using psychophysiological data," *International Journal of Online and Biomedical Engineering*, no. 13, 2019.
- [15] 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.
- [16] 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 (MERCCon)*. IEEE, 2019, pp. 123–128.
- [17] 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, 2004.
- [18] BrainVision. (2021) Brainvision analyzer (version 2.2.2). Brain Products GmbH. Accessed Oct, 11, 2022. <https://www.brainproducts.com/new-brainvision-analyzer-222/>.
- [19] 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.
- [20] M. S. Sendi, M. Heydarzadeh, and B. Mahmoudi, "A Spark-based analytic pipeline for seizure detection in EEG big data streams," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2018, pp. 4003–4006.
- [21] J. Zheng, J. Zhao, J. Li, C. Zhan, and T. Wang, "Spark-based platform for neurophysiological data storage and processing: a proof of concept," in *2021 6th International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, vol. 6. IEEE, 2021, pp. 16–19.
- [22] S. Zhelev and A. Rozeva, "Using microservices and event driven architecture for big data stream processing," in *AIP Conference Proceedings*, vol. 2172. AIP Publishing, 2019.
- [23] M. Roberts and J. Chapin, *What is Serverless?* O'Reilly Media, Inc., Jun. 2017.
- [24] B. Marr. (2022, Nov.) The 5 biggest cloud computing trends in 2022. Accessed Feb 15, 2023. <https://www.forbes.com/sites/bernardmarr/>

2021/10/25/the-5-biggest-cloud-computing-trends-in-2022/?sh=2f2f955e2267.

- [25] B. Farrow and S. Jayarathna, "A serverless electroencephalogram data retrieval and preprocessing framework," in *2023 IEEE 24th International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE, 2023, pp. 221–226.
- [26] Swartz Center for Computational Neuroscience. (2020) What is EEGLAB? San Diego, CA. Accessed April 4, 2022. <https://scn.ucsd.edu/eeqlab/index.php>.
- [27] 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, Jun. 2015.
- [28] 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.
- [29] 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.
- [30] A. Pedroni, A. Bahreini, and N. Langer, "Automagic: Standardized preprocessing of big EEG data," *Neuroimage*, vol. 200, pp. 460–473, 2019.
- [31] 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.
- [32] A. R. Levin, A. S. Méndez Leal, L. J. Gabard-Durnam, and H. M. O’Leary, "BEAPP: the batch electroencephalography automated processing platform," *Frontiers in Neuroscience*, vol. 12, p. 513, 2018.
- [33] S. Appelhoff, A. J. Hurst, A. Lawrence, A. Li, Y. J. Mantilla Ramos, C. O’Reilly, L. Xiang, and J. Dancker, "PyPREP: A Python implementation of the preprocessing pipeline (PREP) for EEG data," Oct. 2022.
- [34] C. P. Jayapandian, C.-H. Chen, A. Bozorgi, S. D. Lhatoo, G.-Q. Zhang, and S. S. Sahoo, "Cloudwave: distributed processing of "big data" from electrophysiological recordings for epilepsy clinical research using Hadoop," in *AMIA Annual Symposium Proceedings*, vol. 2013. American Medical Informatics Association, 2013, p. 691.
- [35] G. Berrada, M. van Keulen, and M. B. Habib, "Hadoop for EEG storage and processing: a feasibility study," in *Brain Informatics and Health: International Conference (BIH) 2014*. Warsaw, Poland: Springer, 2014, pp. 218–230.
- [36] K. Aziz, D. Zaidouni, and M. Bellafkih, "Real-time data analysis using Spark and Hadoop," in *2018 4th International Conference on Optimization and Applications (ICOA)*. IEEE, 2018, pp. 1–6.
- [37] 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, Dec. 2018.
- [38] 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.
- [39] 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.
- [40] M. C. Rushambwa, M. Gezimati, P. Govindaraj, R. Palaniappan, V. Vijejan, 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.
- [41] C.-F. Fan, A. Jindal, and M. Gerndt, "Microservices vs serverless: A performance comparison on a cloud-native web application," in *10th International Conference on Cloud Computing and Services Science (CLOSER)*, 2020, pp. 204–215.
- [42] 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.
- [43] Amazon Web Services. (2023) AWS documentation. Accessed Aug, 14, 2023. <https://docs.aws.amazon.com/>.
- [44] 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.
- [45] C. J. Markiewicz, K. J. Gorgolewski, F. Feingold, R. Blair, Y. O. Halchenko, E. Miller, N. Hardcastle, J. Wexler, O. Esteban, M. Goncavles, A. Jwa, and R. Poldrack, "The OpenNeuro resource for sharing of neuroscience data," *Elife*, vol. 10, p. e71774, 2021.
- [46] A. Miltiadous, K. D. Tzimourta, T. Afrantou, P. Ioannidis, N. Grigoriadis, D. G. Tsalikakis, P. Angelidis, M. G. Tsipouras, E. Glavas, N. Giannakeas, and A. T. Tzallas, "A dataset of EEG recordings from: Alzheimer’s disease, frontotemporal dementia and healthy subjects," 2023.