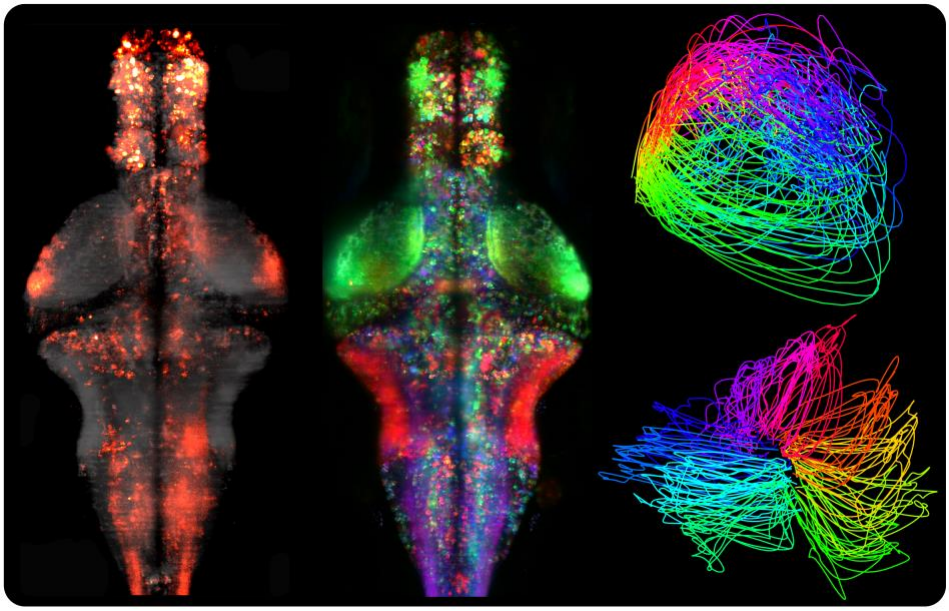
3252 Final Project Report – Designing a Data-Intensive EEG Application on GCP



"We are at the beginning of an exciting moment in large-scale neuroscience. ***We think Spark will be core to our analytics****, but significant challenges lie ahead. Given the scale and complexity of our problems, different research groups must work together to unify analysis efforts, vet alternative approaches, and share data and code. We believe that any such effort must be open-source through-and-through, and we are fully committed to building open-source solutions. We also need to work alongside the broader data science and machine learning community to develop new analytical approaches, which could in turn benefit communities far beyond neuroscience. Understanding the brain will require all of our biological and analytical creativity — and we just might help revolutionize data science in the process."*

~ Jeremy Freeman (databricks)

Providing affordable, low-risk, non-invasive BCI [Brain Computer Interface] devices is dependent on further advancements in interpreting EEG signals.

~ From Kaggle Competition Description for 'Grasp-and-Lift EEG Detection' (link)

Introduction

The following report was prepared to detail the process of creating a neuroscience-based application. The intent was to showcase some of the tools and ideas learned in **3252 - Big Data Management Systems & Tools** and to apply them to an area of personal & professional interest – neuroscience.

Specifically, I attempted to replicate a machine-learning pipeline created by **Dorian Beganovic** (here). The goal was to create a system capable of storing large amounts of distributed EEG data using Apache HDFS, and to analyze it to uncover mental states and other electroencephalographic (EEG) patterns, using Spark's machine learning libraries. Such a tool could be (and indeed, likely is) used by a company such as InteraXon to uncover insights from its customer-generated EEG data (provided they opt-in to sharing their data).

The field of neuroscience is ripe with grand initiatives to map the brain and to open the study of it up to more and more people. For example, the Human Brain and US BRAIN Projects in Europe and the US respectively, are both large-scale brain mapping projects that have started within the last 5 years.

Given the recent research and funding around neuro-science, artificial intelligence, and wearables, there has never been a better time to be a "neuro-hacker." Consumer-grade EEG headsets are available for $300 and easily allow the curious, tech-savvy individual to begin logging and analyzing their brain waves.

The need for data-applications and particularly machine learning is crucial for a better understanding of EEG data patterns, as the signal-to-noise ratio of EEG data is considerable, and identifying relevant trends in a noisy data stream is something machine-learning algorithms do particularly well.

EEG Background & Recent Developments in Neurotechnology

Neuroscience and the study of human intelligence are being transformed by data science, machine learning, and increasingly high-resolution mappings & models of the brain. Designing data-intensive applications that utilize advances in wearable technologies, big data, and machine learning is an area of considerable research & investment.

If Elon Musk is correct, in order to 'keep pace with the machines' we will need to be able to increase our cognitive bandwidth by literally linking our computers and our neurology. His startup, Neuralink has raised $27 million in 2017 and is in the early stages of developing what it calls the neural lace – a brain-computer interface (BCI) that would "allow people to communicate directly with machines without going through a physical interface. Neural lace involves implanting electrodes in the brain so people can upload or download their thoughts to or from a computer, potentially allowing humans to achieve higher levels of cognitive function."

Closer to home, Toronto-based InteraXon is a leading neuroscience technology company with its roots at U of T and the Toronto maker/hacker community. Two of the company's founders – Ariel Garten and Chris Aimone – are U of T graduates. The company has raised almost $30 million since its inception, and has recently announced exciting collaborations with leading eyewear company Smith Optics. Their leadership position has brought EEG from a relatively fringe technology to a niche, but still sizeable market.

As with all good technology platforms & projects, it is important that there is a community of researchers, prototypers, engineers, and designers building, making, and creating new applications with these technologies.

InterAxon is an excellent example of this ecosystem approach to development, and is why I wanted to try to build an app that took data from one of its headbands, and to use it to prototype a scalable data-application using Hadoop, Spark, and various tools & technologies through Google's Cloud Platform (GCP).

EEG (**E**lectro**e**ncephalo**g**raphy)
EEG is a common medical technology in which data are generated by patterns of electrical activity in the brain that are captured and visualized and used to understand brain function on an electrical level. With billions of neurons and trillions of connections, the brain generates flurries of electrical activity which can be detected by sensitive electrodes placed on the scalp. The processing of these signals and patterns can be used to identified recognized brain states – characterized by the relative occurrence of different types of frequency bands. These frequency bands are known as delta, theta, alpha, beta, gamma (and mu) – and range from a slow ~4Hz for delta waves to a fast 32+ Hz for gamma waves.

The following table from Wikipedia provides a handy comparison of the bands and their associated locations, and normal/pathological function.

Comparison of EEG bands

Band	Frequency (Hz)	Location	Normally	Pathologically
Delta	< 4	frontally in adults, posteriorly in children; high-amplitude waves	• adult slow-wave sleep • in babies • Has been found during some continuous-attention tasks[51]	• subcortical lesions • diffuse lesions • metabolic encephalopathy hydrocephalus • deep midline lesions
Theta	4–7	Found in locations not related to task at hand	• higher in young children • drowsiness in adults and teens • idling • Associated with inhibition of elicited responses (has been found to spike in situations where a person is actively trying to repress a response or action).[51]	• focal subcortical lesions • metabolic encephalopathy • deep midline disorders • some instances of hydrocephalus
Alpha	8–15	posterior regions of head, both sides, higher in amplitude on dominant side. Central sites (c3-c4) at rest	• relaxed/reflecting • closing the eyes • Also associated with inhibition control, seemingly with the purpose of timing inhibitory activity in different locations across the brain.	• coma
Beta	16–31	both sides, symmetrical distribution, most evident frontally; low-amplitude waves	• range span: active calm → intense → stressed → mild obsessive • active thinking, focus, high alert, anxious	• benzodiazepines • Dup15q syndrome [52]
Gamma	> 32	Somatosensory cortex	• Displays during cross-modal sensory processing (perception that combines two different senses, such as sound and sight)[53][54] • Also is shown during short-term memory matching of recognized objects, sounds, or tactile sensations	• A decrease in gamma-band activity may be associated with cognitive decline, especially when related to the theta band; however, this has not been proven for use as a clinical diagnostic measurement
Mu	8–12	Sensorimotor cortex	• Shows rest-state motor neurons.[55]	• Mu suppression could indicate that motor mirror neurons are working. Deficits in Mu suppression, and thus in mirror neurons, might play a role in autism.[56]

The complexity of the brain is significantly beyond our current level of comprehension. But with that said, applied innovations in medical sciences, machine learning, and data engineering techniques are rapidly advancing our understanding of neuroscience, cognition, and intelligence.

Technology Objectives & Data Architecture Overview

The goal of this project was to see if I could replicate a data pipeline that did the following as described by a blogger, and summer-intern at Google Dorian Bagenovic:

1. **Collect and store EEG data** from a series of recorded EEG sessions on a remote cluster using the Hadoop Distributed File System
2. **Create a distributed processing engine for analyzing and applying machine learning to the EEG data** using Apache Spark and dl4j
3. **Use a GUI to manage the data and building data analysis workflows** - choose from different signal processing methods and machine learning techniques.

As described by Begenovic, the data pipeline includes 3 key sub-components:

1. **Client GUI** - a portable Desktop Java application allowing the user to browse/manage the data on remote Hadoop Distributed File System as well as visually build data analysis workflows using feature extraction and classification methods.
2. **Data Analysis Package** - a Java application made using Apache Spark, Apache Hadoop and Deep Learning for Java (dl4j) frameworks whose purpose is to provide a modular way of specifying data pipelines consisting of input sources, data processing, feature extraction and classification methods
3. **Remote Server -** a Spring Boot server and the main communication point for the Client GUI with the Data Analysis Package on the Hadoop server. It listens for requests such as job submittals, fetching the results of a job, listing of trained classifiers and etc.

Although I first tried to clone Bagenovic's github repository running the .jar files he has linked on his github page, I was unsuccessful in setting up the required environment & configurations to make it run.

So, I decided to try a different approach based upon a Google Cloud Platform guide called "Real-time Stream Processing IoT." While I was not able to entirely set up that system either, the following sections describe the blueprint for such a system including the major processes, packages, libraries, hardware, and services required. See Appendix B for Google's Architecture overview diagram.

The goal of building this application was to gain real-world experience doing data architecture and engineering tasks, and to familiarize myself with an industry-leading IaaS platform (I went with GCP).

As will be described below, I used GCP's free trial to explore many of their cloud services and to create an architecture (and some engineering) that would, in principle, take EEG data from a variety of sources, store it in Google's HDFS, analyze it using Spark, and build models with

EEG Toolkit

- **Muse Bluetooth EEG Headband** by InteraXon
- **Muse-IO** – basic input/output script to establish connection with EEG headset over TCP port 5000
- **Muse-Player** – script to replay and convert .muse recordings into a variety of other formats including .txt and .csv.
- **MuseLab** – a software studio for visualizing raw EEG data and other sensor data (i.e. accelerometer).

Big Data Toolkit

- Bash/Terminal/Google Cloud **Shell** – Local & cloud-based shell/command line interface
- **Google Cloud Platform** (IaaS provider of choice)
    - **Google Cloud Storage** (Storage Buckets)
    - **Google Compute Engine** (Virtual Machines)
    - **Google Cloud Dataproc** (Managed Spark/Hadoop service to handle cluster creation, management, monitoring, and job orchestration automatically. Cloud Dataproc is integrated with the YARN application manager to make managing and using your clusters easier. Cloud Dataproc provides out-of-the box and end-to-end support for many of the most popular job types, including Spark, Spark SQL, PySpark, MapReduce, Hive, and Pig jobs.
    - **Google IoT Core** (Internet of Things Protocol)
    - **Google Cloud Pub/Sub** (Cloud Pub/Sub natively connects to other Cloud Platform services, gluing together data import, data pipelines, and storage system.)
    - **Google Cloud SDK** (to run gcloud commands on local machine)
    - **Google Stackdriver Monitoring & Logging** (provides time-series metrics on key health indicators and can alert as soon as problems occur)
    - **Google Cloud Dataflow** (Fully managed multi-tool for data processing)
    - **Google Cloud APIs** (the
    - **Google BigQuery Connector for Apache Spark** - blend the power of BigQuery's seamlessly scalable SQL engine with Apache Spark's Machine Learning capabilities.

Data Pipeline

1. Device Registration
    a. Using Google IoT Core & Pub/Sub create a device registry for Muse EEG devices and register IoT devices.

2. Connect to Muse using the following guide: http://developer.choosemuse.com/research-tools-example/grabbing-data-from-museio-a-few-simple-examples-of-muse-osc-servers
3. Data Ingestion using Google's Pub/Sub messaging service.
    a. Acquiring EEG Data from the Muse EEG Headband (in OSC Format) Over TCP/UDP. Write to a *.muse or *.txt file.
    b. Converting to Required Format for Hadoop & Spark
4. Data Pipelines – Cloud Dataflow to manage data once it's arrived through Pub/Sub.
5. Analytics – Cloud Dataproc (Managed Hadoop, Spark, Yarn ecosystem) to run ML jobs on.
6. Storage – BigTable, Cloud Storage or Cloud Datastore
7. Application & Presentation – to package & run the application.

Findings

** The findings apply to my learnings for the project – not the findings of the data analysis, as I was unable to properly set-up the data architecture as described in the reference documentation I was following **

1. Distributed environments present unique challenges. Running a simple function, on a local machine is one thing. But, in order to build scalable applications such as the one described in this report, parallelization, and usually virtualization is required. This presents a unique set of challenges from choosing an Infrastructure-as-a-Service (IaaS) providers such as Amazon Web Services (AWS) or Google Cloud Platform (GCP). Learning how to create instances, install images of OS's, and add SSH keys was all new to me. Although I ultimately ran into configuration errors, I learned an incredible amount about Google's Cloud Platform, using Google's suite of API's, connecting through SSH, and various other tools.
2. GiYF "Google is Your Friend"
3. Google Cloud Platform provides manages versions of many of the tools and concepts we learned about in 3252, including Hadoop, Spark, Yarn, Clusters, Dockers, Message Brokers, Databases/Stores/Warehouses, etc. Building some of these systems using GCP was a valuable experience.

Appendices

Google Cloud Shell Output:

```
tests-Air:manager MillersFiles$ gcloud dataproc clusters describe cluster-06a7
clusterName: cluster-06a7
```

```
clusterUuid: d0118217-66cf-4be9-90e4-39dbaa38d658
config:
  configBucket: dataproc-8027c474-ab75-40da-bc43-c56be7c36310-us-east4
  gceClusterConfig:
    serviceAccountScopes:
    - https://www.googleapis.com/auth/bigquery
    - https://www.googleapis.com/auth/bigtable.admin.table
    - https://www.googleapis.com/auth/bigtable.data
    - https://www.googleapis.com/auth/cloud.useraccounts.readonly
    - https://www.googleapis.com/auth/devstorage.full_control
    - https://www.googleapis.com/auth/devstorage.read_write
    - https://www.googleapis.com/auth/logging.write
    subnetworkUri: https://www.googleapis.com/compute/v1/projects/cloudera-
188117/regions/us-east4/subnetworks/default
    zoneUri: https://www.googleapis.com/compute/v1/projects/cloudera-188117/zones/us-
east4-a
  masterConfig:
    diskConfig:
      bootDiskSizeGb: 500
    imageUri: https://www.googleapis.com/compute/v1/projects/cloud-
dataproc/global/images/dataproc-1-2-20171113-174546
    instanceNames:
    - cluster-06a7-m
    machineTypeUri: https://www.googleapis.com/compute/v1/projects/cloudera-
188117/zones/us-east4-a/machineTypes/n1-standard-2
    numInstances: 1
  softwareConfig:
    imageVersion: 1.2.13
    properties:
      capacity-scheduler:yarn.scheduler.capacity.root.default.ordering-policy: fair
      core:fs.gs.block.size: '134217728'
      core:fs.gs.metadata.cache.enable: 'false'
      distcp:mapreduce.map.java.opts: -Xmx1638m
      distcp:mapreduce.map.memory.mb: '2048'
      distcp:mapreduce.reduce.java.opts: -Xmx1638m
      distcp:mapreduce.reduce.memory.mb: '2048'
      hdfs:dfs.datanode.address: 0.0.0.0:9866
      hdfs:dfs.datanode.http.address: 0.0.0.0:9864
      hdfs:dfs.datanode.https.address: 0.0.0.0:9865
      hdfs:dfs.datanode.ipc.address: 0.0.0.0:9867
      hdfs:dfs.namenode.http-address: 0.0.0.0:9870
      hdfs:dfs.namenode.https-address: 0.0.0.0:9871
      hdfs:dfs.namenode.secondary.http-address: 0.0.0.0:9868
      hdfs:dfs.namenode.secondary.https-address: 0.0.0.0:9869
      mapred:mapreduce.job.maps: '15'
      mapred:mapreduce.job.reduce.slowstart.completedmaps: '0.95'
      mapred:mapreduce.job.reduces: '5'
      mapred:mapreduce.map.cpu.vcores: '1'
      mapred:mapreduce.map.java.opts: -Xmx1638m
      mapred:mapreduce.map.memory.mb: '2048'
      mapred:mapreduce.reduce.cpu.vcores: '1'
      mapred:mapreduce.reduce.java.opts: -Xmx1638m
      mapred:mapreduce.reduce.memory.mb: '2048'
      mapred:mapreduce.task.io.sort.mb: '256'
      mapred:yarn.app.mapreduce.am.command-opts: -Xmx1638m
      mapred:yarn.app.mapreduce.am.resource.cpu-vcores: '1'
      mapred:yarn.app.mapreduce.am.resource.mb: '2048'
      spark:spark.driver.maxResultSize: 960m
      spark:spark.driver.memory: 1920m
      spark:spark.executor.cores: '1'
      spark:spark.executor.memory: 2688m
      spark:spark.yarn.am.memory: 640m
      yarn:yarn.nodemanager.resource.memory-mb: '6144'
```

```
      yarn:yarn.scheduler.maximum-allocation-mb: '6144'
      yarn:yarn.scheduler.minimum-allocation-mb: '512'
  workerConfig:
    diskConfig:
      bootDiskSizeGb: 500
    imageUri: https://www.googleapis.com/compute/v1/projects/cloud-
dataproc/global/images/dataproc-1-2-20171113-174546
    instanceNames:
    - cluster-06a7-w-0
    - cluster-06a7-w-1
    machineTypeUri: https://www.googleapis.com/compute/v1/projects/cloudera-
188117/zones/us-east4-a/machineTypes/n1-standard-2
    numInstances: 2
labels:
  goog-dataproc-cluster-name: cluster-06a7
  goog-dataproc-cluster-uuid: d0118217-66cf-4be9-90e4-39dbaa38d658
  goog-dataproc-location: global
metrics:
  hdfsMetrics:
    dfs-capacity: '1056621535232'
    dfs-nodes-decommissioned: '0'
    dfs-nodes-decommissioning: '0'
    dfs-nodes-running: '2'
    dfs-present-capacity: '1004488208670'
    dfs-remaining: '1004483796992'
    dfs-used: '4411678'
  yarnMetrics:
    yarn-containers: '0'
    yarn-memory-mb-configured: '12288'
    yarn-memory-mb-used: '0'
    yarn-nodes: '2'
    yarn-virtual-cores-configured: '4'
    yarn-virtual-cores-used: '0'
projectId: cloudera-188117
status:
  detail: Last reported status is over 435694 seconds old
  state: RUNNING
  stateStartTime: '2017-12-06T01:40:39.231Z'
  substate: STALE_STATUS
statusHistory:
- state: CREATING
  stateStartTime: '2017-12-06T01:39:02.904Z'
```

Appendix B: Real Time Stream Processing Architecture Diagram (link)