A DISTRIBUTED CLOUD-BASED PLATFORM FOR FMRI BIG DATA ANALYTICS

by

MILAD S MAKKIE

(Under the Direction of Tianming Liu)

ABSTRACT

The sheer complexity of the brain has forced the neuroscience community and specifically the neuroimaging experts to transit from the smaller brain datasets to much larger hard-to-handle ones. The primary goal of flagship projects such as the BRAIN Initiative and Human Brain Project is to gain a better understanding of the human brain and to treat the neurological and psychiatric disorders through the cutting-edge technologies in the biomedical imaging field. In the context of fMRI, the primary challenge is obtaining meaningful results from the intrinsic complex structure of large fMRI data and lack of clear insight into the underlying neural activities. However, archiving, analyzing, and sharing the fast-growing neuroimaging datasets posed significant challenges. New computational methods and technologies have emerged in the domain of Big Data but have not been fully adapted for use in neuroimaging. In this dissertation, I introduce my efforts toward creating a comprehensive platform to store, to manage and to process such datasets. I further present my GPU-based deep learning solution for distributed data processing that employs TensorFlow, Apache Spark, and Hadoop using cloud computing services. Finally, I demonstrate the significant performance gains of our platform enabling data-driven extraction of hierarchical information from massive fMRI data using a distributed deep convolutional autoencoder model.

INDEX WORDS: Distributed Processing, Bigdata, Deep Learning, Machine Learning, fMRI

A DISTRIBUTED CLOUD-BASED PLATFORM FOR FMRI BIG DATA ANALYTICS

by

MILAD S MAKKIE

M.S. Amirkabir University of Technology, Iran, 2012

B.S. Azad University, Iran, 2010

A Dissertation Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA

2018

A DISTRIBUTED CLOUD-BASED PLATFORM FOR FMRI BIG DATA ANALYTICS

by

MILAD S MAKKIE

Major Professor:	Tianming Liu
Committee:	Thiab Taha
	Hamid Arabnia
	Nicole Lazar

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
May 2018

DEDICATION

To my mother

who taught me to not limit my challenges but challenge my limits.

To my father

whose legacy taught me the great use of life is to spend it for something that will outlast it.

ACKNOWLEDGEMENTS

This dissertation represents not only my work at the department of computer science but the circuitous path it led me to work with a great pool of talent and brilliant minds at UGA, whom challenged, supported and stuck with me through the good times and bad. Although I have been far from home since 2012, my experience at UGA has been nothing but wonderful; it is truly my second home. Therefore, I would like to address my most sincere gratitude to Dr. Tianming Liu, for leading me in many projects and publications. His vision in the advancement of functional neuroimaging analysis is like no other, let alone his enthusiasm in conducting real important (albeit difficult) research. He has always been supportive of me. It has truly been a pleasure to be part of a young, yet highly proficient, and most importantly passionate team of researchers at Cortical Architecture Imaging and Discovery laboratory with Dr. Liu's mentorship over the years. This dissertation represents the lessons I learned in my term under Dr. Liu's guidance and is the result of work by many remarkable individuals whom I also wish to acknowledge and thank.

I would like to express my gratitude to my advisory committee, Dr. Thiab Taha and Dr. Nicole Lazar for accepting and playing a major role in my advisory committee. I must express my special gratitude to Dr. Hamid Arabnia as my committee member and most importantly, my mentor. I have been honored to have an extraordinary and an inspiring mentor whom has helped me find ways to stay committed in more than one way.

Completing this work would have been even more difficult were it not for the support and friendship provided by three of my dear friends whom have been essential supporters throughout

the most difficult times when progressing my research. Dr. Amir Nowroozzadeh, Arad Ghodrati and Ayda Farhadi, they made this dissertation seem doable when it felt daunting.

Moreover, I would like to thank Dr. Quanzheng Li, who kindly hosted me at the Harvard Medical School to learn about the latest advances of High-Performance computing and AI in neuroimaging.

Lastly, I must express my sincere gratitude to my mother and my brother, for their continued support and encouragement. I was amazed by their patience and willingness to help and to provide invaluable advice through these years of my life in all the aspects.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

After the success of the Human Genome Project (HGP) [1], [2], [3] to map 3 billion nucleotides representing human inheritance, the US Brain Research Through Advancing Innovative Neurotechnologies (BRAIN) [4] Initiative, European Union Human Brain Project (HBP) [5] launched in 2013 and China Brain Project were initiated to reflect the aspiration and investment in neuroscience research for understanding the human brain structure and function, especially to treat many brain disorders.

The sheer complexity of the brain has forced the neuroscience community and specifically the neuroimaging experts to transit from the smaller brain datasets to the extent far less manageable. The cutting-edge technologies in the biomedical imaging field, as well as the new techniques in digitizing, all lead to collect further information from the structural organization and functional neuron activities in the brain [6].

Understanding the relationship between functional neural activity, structural organization of brain regions, and subsequent behavior became the main goals of neuroscience. These goals are only achievable by analyzing covariance in large scale studies [6]. Aligned with these goals, discovery-based approaches have been employed to empower the investigation of brain-behavioral relationships. These goals are not reachable but through large-scale datasets. The possible challenges of holding and analyzing this much data have been one of the main topics of the annual meetings of the Organization for Human Brain Mapping (OHBM) since 2012.

Certainly, Human Connectome Project (HCP) with more than 1200 healthy subjects is a perfect example of these large datasets [7], [8]. HCP was awarded about $40 million in 2010 to develop advanced neuroimaging methods and to recruit a large number of individuals to map brain regions and their connectomes [9, 10]. The main goal is to understand the human brain better and eventually to treat the neurological and psychiatric disorders. The other examples can be 1000 functional connectomes [11] and openfMRI project [12]. These efforts clearly draw a portrait clarifying the emphasis of neuroscience community to employ new techniques to deal with neuroimaging bigdata.

As a few studies have shown [13], [3], the arrival of big data in neuroscience demands a cultural shift from isolated single efforts applying limited methods over small dataset to a more horizontal effort to cover a wider range of problems, using larger datasets and more comprehensive techniques. This transition, however, will require the community to address certain challenges [13]. A few of these challenges are as follows.

Handling more comprehensive datasets demands sophisticated techniques and substantial resources that necessitate close collaboration among laboratories. In recent years, numerous articles have stressed the importance of data sharing, particularly neuroscience MRI data [11], [12], [14], [15], [16]. They mostly indicate that adoption of new data sharing tools along with close collaboration among researchers will benefit researchers methodologically, financially, and ethically, fully allowing researchers to exploit the sizeable quantities of information generated across laboratories.

Techniques for studying the neural activities and the brain structure are varied, consisting of strategies to represent a vast range of temporal and spatial resolutions [13]. Each of these methods is limited to a specific resolution and only applicable to a portion of the brain studies. These

techniques can be as fast as 0.0001s for patch clamping and as accurate as electron microscopy with ~0.0001mm accuracy, to electroencephalography and fMRI with lower spatial and temporal resolutions. Each of these techniques carries its own set of vocabulary and metadata, and thus different standardizations are needed. This complexity makes the cross-pipelines harder to automate, as multidimensional problems involving multiple modalities and techniques are required to reach an appropriate level of scientific certainty.

Among various neuroimaging methods, functional magnetic resonance imaging, fMRI, has been widely used to assess functional activity patterns in the brain [17], [18], [19], [20]. Since the early 1990s [21], [22], when fMRI came to dominate the brain mapping research, more than 42,000 papers have been published according to PubMed which indicates the significant interest of scientists to use this modality to understand brain functions. Researchers have vastly used both Task-based (tfMRI) and Resting-state (rfMRI) fMRI techniques for functional brain mapping. [23], [24], [25], [22], [27], [28], [29], [30]. From a total of 12 available shared neuroimaging datasets at 2014, 8 of those contained rfMRI and four of them tfMRI scans [15]. This demonstrates the fundamental role of fMRI as a tool for discovery, shedding light on the unexplored functional brain activities.

Given the popularity and the importance of fMRI to map functional brain networks, tremendous efforts have been devoted to the establishment of fMRI neuroinformatics systems through which users can easily employ comprehensive statistical and computational approaches for fMRI analysis [31], [32], [33], [34], [35], [36]. These systems are expected to host large-scale datasets and to provide a modular independent platform to run wide-ranging complex algorithms and processes in which tasks can be run in a distributed and/or parallel fashion. Storing, analyzing, visualizing,

and sharing large datasets need intensive computational and storage resources that more traditional methods could not deliver.

1.1 Thesis Statement

A series of studies from my collaborators at Cortical Architecture Imaging and Discovery laboratory and I, have been performed towards characterizing the functional organization pattern and the cognitive process of human brain. These tasks were mostly achieved by developing models on the functional Magnetic Resonance Imaging (fMRI) data on various scales and conditions. These models were first designed and tested on a scale of few fMRI subjects and then we created scalable distributed versions to apply over much larger datasets. Analyzing very large amount of data, however needs intensive computational and storage resources and techniques, that traditional platforms could not deliver. Thus, designing and implementing customized frameworks that fulfill the needs of such a large community were critical.

My work, represented in this dissertation, focuses on providing large-scale analytical solutions to better characterize the functional pattern and the cognitive process of human brain. My main concentration since starting my research at 2014, was to design and develop practical tools that can efficiently help the neuroimaging researchers, store, analyze, visualize and share very large datasets. My efforts, inspired by recent machine learning and distributed processing researches, offer a practical approach in resolving the computationally intensive, structurally complex, large neuroimaging data.

I fit the current computational challenges for neuroimaging bigdata analytics in 6 categories of data management systems, processing pipelines, computing platforms, distributed storages, data visualization tools and processing engines. Following these challenges, I will explain a few

examples where experts have addressed each correspondingly. I then discuss my comprehensive solutions, developed at CAID in chapter 2 to 4.

1.2 Computational Challenges for Neuroimaging Bigdata Analytics

1.2.1 Data Management System

Data management is the core requirement to both organize and present data to the researchers. The Extensible Neuroimaging Archive Toolkit, XNAT [37] is one of the best examples, designed particularly to host and manage neuroimaging data in which supports the standard input formats such as DICOM and covers a broad range of meta-data standards. A hierarchical Extensible Markup Language (XML) schema provides a framework in which users can define their own types of inference, depend on the imported data, and easily import the experiments' descriptors through both web interface and command environment. XNAT is an active project, and the modified version of this toolkit serves as the basis of Human Connectome Project Database [38]. The open-source availability and the RESTful application programming interface allow communication between package components via the web, making XNAT a unique solution for neuroimaging data management system.

1.2.2 Data Processing Pipeline

This is another essential element of neuroimaging bigdata analysis where end-to-end processing workflows are specified, and users can manage workflow parameters and execution. There exist a few of neuroimaging pipelining solutions, including LONI [39], [40] with a graphical user interface, Nypype [41] a Python-based pipelining tool, and XNAT, an XML-based solution with grid computing capability.

1.2.3 Computing Platform

Computing platforms are the critical requirement for bigdata analysis. For example, preprocessing fMRI data takes roughly 5 minutes per subject using an 8-core machine with 16 gigabytes memory dedicated to this task. Preprocessing compromises skull removal, motion correction, slice time correction, and spatial smoothing as well as global drift removal [30]. Applying this step over hundreds of subjects will take hours to days using a single machine. Therefore, running computationally-intensive tasks in parallel is essential to reduce the overall computational time from days and months to hours and minutes; high-performance computing (HPC) is a very common solution. With the use of CPU and GPU-based clusters, substantial speedups can be achieved with no need of modifying the existing software tools. Incorporating GPUs and CPUs in parallel processing has recently become a popular topic among researchers to study [42], [43], [44], [45]. Amazon Elastic Compute Cloud (EC2) is one of the most successful instances in providing scalable computing capacity on-demand.

1.2.4 Cloud Storage and Cloud Computing

Using cloud resources are inseparable parts of bigdata analysis. High-speed access to the stored data is essential in cloud computing due to the constant read and write flow among computing nodes. Amazon Simple Storage System, or S3, as an example, is an efficient choice of cloud storage with instant access to the data from EC2 computing nodes. The read and write speed and fault tolerance, as well as pricing, make S3 a competitive choice for researchers.

1.2.5 Data Visualization

Visualization is an imperative entity of bigdata: making complex results understandable and interpretable by a human, and dynamic visualization is to improve the insight gained from data. A well-designed pipeline should generate graphics that represent the rich variety of date in neuroimaging, including time series, regions of interest, networks, and connectomes. There exist several tools and libraries that in combination with statistical and analytical frameworks generate data-related graphics. However, it is hard for general users to implement and to apply and consequently, more efforts are needed to create customized tools for neuroscience experts that can be easily applied in the existent pipelines. As Freeman in [46] suggests, visualizing the results with an interactive environment is far valuable than a static image representing only a portion of information especially when we are interacting with large datasets with rich data.

1.2.6 Processing Engines

Processing engines enable researchers and programmers to load and analyze data in a distributed fashion and to create new methods to handle sophisticated analytics processes faster and with ease of use. As I discussed earlier, dealing only with a portion of datasets is ideal only at the testing stage, but in benchmark analysis, a more substantial portion of datasets is necessary. In 2003 and 2004, the Google file system and MapReduce were introduced, respectively, to the world as a simplified abstraction for parallel manipulation of massive datasets [47]. The main idea of MapReduce is to store data in a distributed file system located in a cluster environment and then use individual nodes to do the computation. This way, data is accessible from all the nodes and only the subsequent aggregation steps of the computation will be transferred to the master node. The whole workflow works in two stages: map and reduce. At first, a function will apply to

partitions of the data in parallel, and then an associative operator will aggregate the results across partitions. Figure 1.1 shows an example of word count problem solved by MapReduce. Although MapReduce is widely used by researchers and programmers to model variety of computationally intensive tasks and machine learning methods [48], due to some data modeling constraints, it is not considered an all-purpose big data tool. MapReduce loads the data into the memory from the hard disk and returns the results at every round of analysis that causes a substantial amount of disk I/O and queries especially for iterative machine learning algorithms in neuroimaging. It is also hard to represent complex series of computations given pipelining in neuroimaging.

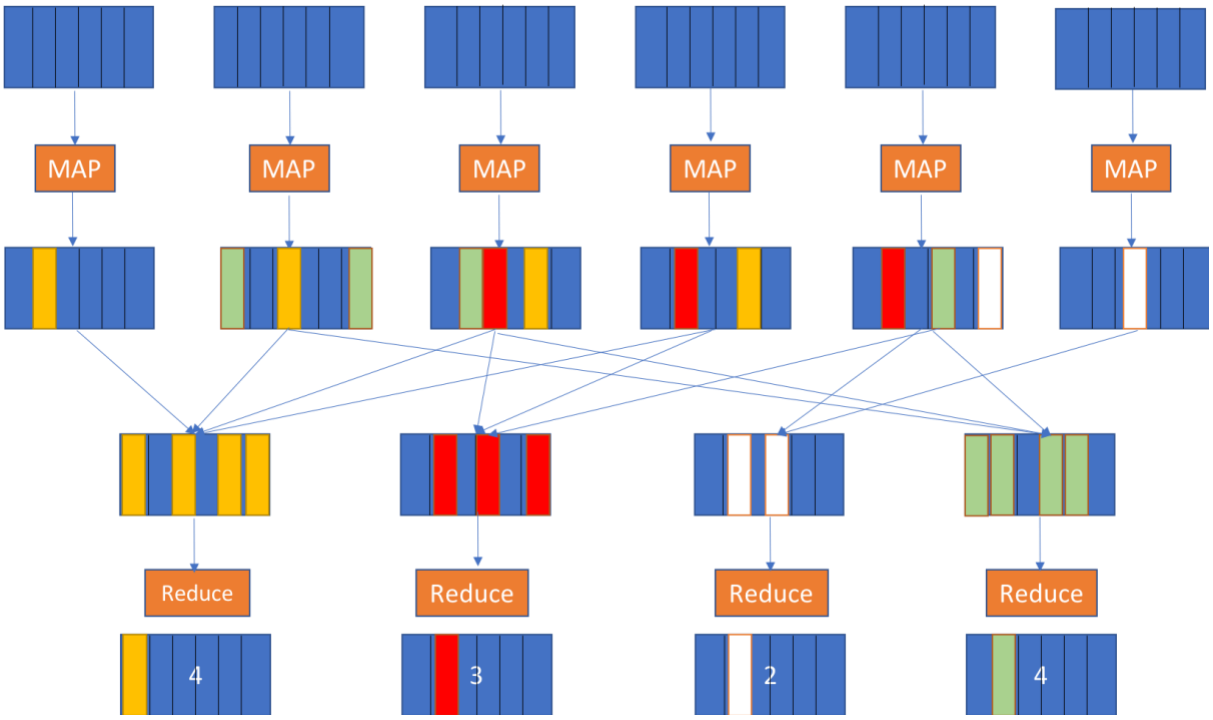

Figure 1.1. Illustration of the mapReduce model applied to counting words problem. A potentially large list of words is processed into key-value pair records of form (word, 1) in parallel during the Map step. During the Reduce step, records with the same key (word) will be combined and an associative operator computes a sum for each word.

In 2009, the Spark framework [49] was developed at the University of Berkeley AMPlab. This framework addresses deficiencies of MapReduce by introducing resilient distributed datasets (RDD) abstract which the operations are performed in the memory. Spark compiles the action lineages of operations into efficient tasks, which are executed on the Spark engine. Spark's scheduler will execute the duties across the whole cluster. Spark minimizes the repetition of data loading by caching data in memory which is crucial in complex processes. Also, Spark supports multiple programming languages, including Java, Python, Scala, and R. figure 1.2 shows the general Spark workflow and how it operates tasks in different stages. Spark uses Hadoop filesystem as a core distributed file system (HDFS) but networking file systems (NFS) can also be used if it runs on an HPC cluster. Apache Spark is the single most active Apache project. The new version 2.0 is promised to repair the performance leaks already found in the earlier version of 1.5 and 1.6. While Spark has considerable traction in industry and academia, Apache Flink [50], developed originally as Stratosphere in 2014, is another new distributed processing engine with similar goals but an entirely new architecture. Flink offers a full compilation of execution plans, optimizing the operations performed and minimizing repeated computations and network accesses. However, this project is still under development, having only reached version 1.0 in recent months.

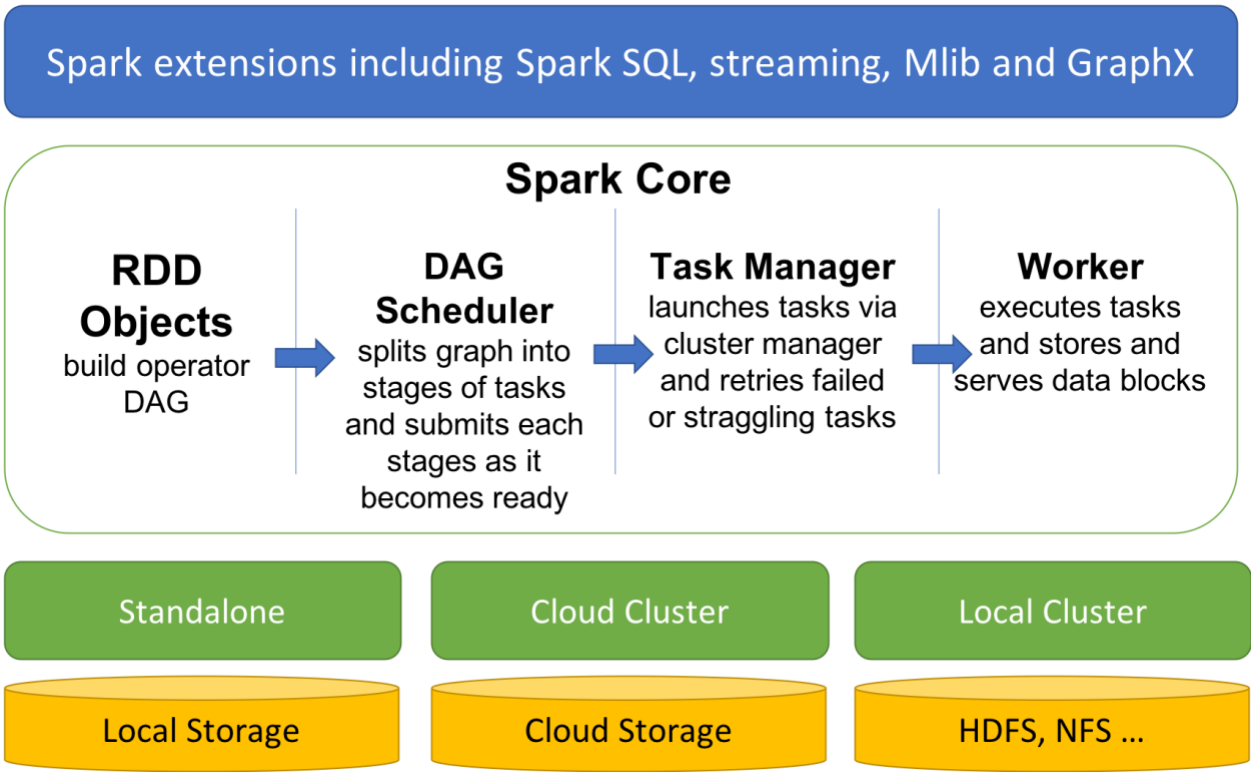

Figure 1.2. Illustration of the spark stack with its components. Spark offers a functional programming API to manipulate Resilient Distributed Datasets (RDDs). RDDs represent a collection of items distributed across many compute nodes that can be manipulated in parallel. Spark Core is a computational engine responsible for scheduling, distribution and monitoring applications which consists of many computational tasks across worker machine(s) on a computation machine/cluster.

1.3 Contributions and Outlines

For the rest of this dissertation, I explain each of my efforts to address the abovementioned challenges using the most recent technologies and advances in computer science. I categorize these efforts as follows.

1.3.1. A Large-Scale Platform for the Neuroimaging Informatics

At first, I developed a large-scale platform for neuroimaging informatics dubbed as HELPNI [51] to store and visualize large-scale multi-modal neuroimages datasets. This platform was first intended to facilitate running and to control complicated neuroimaging multi-stage processes with a smooth, user-friendly web interface and later to give researchers parallel and distribute computing accessibility while they implement their own analytical and visualization tools. We applied our Holistic Atlases of Functional Networks and Interactions framework [52] (HAFNI), for the sparse representation of whole brain fMRI signals over more than 5 thousand publicly available fMRI images. HAFNI is recognized as an efficient method for inferring a comprehensive collection of concurrent functional networks in the human brain. [52]. HELPNI will be discussed at Chapter 2.

1.3.2 A Distributed Platform for fMRI Big Data Analytics

I then concentrated on developing and extending the data storage, data management, and also data processing aspects of HELPNI. The primary goal was to add a distributed file system as well as empowering the computational platform with distributed processing features. Consequently, my collaborator, Xiang Li and I devolved a novel Distributed rank-1 Dictionary Learning [55] (D-r1DL) model, leveraging the distributed computing in handling large scale fMRI big data. This model estimates one rank-1 basis vector with sparsity constraint on its loading coefficient from the input data at each learning step through alternating least squares updates. By iteratively learning the rank-1 basis and deflating the input data at each step, the model is then capable of decomposing the whole set of functional networks. I implemented and parallelized the rank-1 dictionary learning algorithm using Apache Spark engine [49] and deployed the resilient

distributed dataset (RDDs) abstracts on top of HDFS [57] for the data distribution and operations. It is expected that the D-r1DL algorithm and methodology could be widely applicable to many other domains of applications that entail sparse representation of big data. D-r1DL will be discussed at Chapter 3.

1.3.3 Distributed Cloud-Based Platform for fMRI Big Data Analytics

The recent advances of new data-driven computational intensive Neural Network approaches such as deep learning along with fMRI intrinsic complex structure, and the sheer size of fMRI data inspired me to implement a platform to process the very large amount of data using deep learning, employing the powerful GPU-based computational technologies. In result, I implemented a Distributed Cloud-based Deep Learning framework [56] (DCDL) leveraging the Apache Spark and TensorFlow [43], to parallelize millions of fMRI time series and to train our model over a large cluster of GPUs. Furtherly, I proposed a novel fast and scalable distributed Deep Convolutional Autoencoder [56] (dist-DCA) that hierarchically models large-scale task-fMRI time series data while gaining a higher-level abstraction of the tfMRI signals. DCDL and dist-DCA will be discussed at Chapter 4.

CHAPTER 2

HAFNI-ENABLED LARGESCALE PLATFORM FOR NEUROIMAGING INFORMATICS (HELPNI)[1]

[1] Makkie, Milad, Shijie Zhao, Xi Jiang, Jinglei Lv, Yu Zhao, Bao Ge, Xiang Li, Junwei Han, and Tianming Liu. "HAFNI-enabled largescale platform for neuroimaging informatics (HELPNI)." Brain informatics 2, no. 4 (2015): 225-238.

ABSTRACT

Tremendous efforts have thus been devoted on the establishment of functional MRI informatics systems that recruit a comprehensive collection of statistical/computational approaches for fMRI data analysis. However, the state-of-the-art fMRI informatics systems are especially designed for specific fMRI sessions or studies of which the data size is not really big, and thus has difficulty in handling fMRI 'big data'. Given the size of fMRI data is growing explosively recently due to the advancement of neuroimaging technologies, an effective and efficient fMRI informatics system which can process and analyze fMRI big data is much needed. To address this challenge, in this work, we introduce our newly developed informatics platform, namely, 'HAFNI-Enabled Largescale Platform for Neuroimaging Informatics (HELPNI)'. HELPNI implements our recently developed computational framework of sparse representation of whole-brain fMRI signals which is called HAFNI (Holistic Atlases of Functional Networks and Interactions) for fMRI data analysis. HELPNI provides integrated solutions to archive and process large scale fMRI data automatically and structurally, to extract and visualize meaningful results information from raw fMRI data, and to share open-access processed and raw data with other collaborators through web. We tested the proposed HELPNI platform using publicly available 1000 Functional Connectomes dataset including over 1200 subjects. We identified consistent and meaningful functional brain networks across individuals and populations based on resting state fMRI (rsfMRI) big data. Using efficient sampling module, the experimental results demonstrate that our HELPNI system has superior performance than other systems for large scale fMRI data in terms of processing and storing the data and associated results.

2.1 Introduction

Understanding the organization of brain function has received significant interest since the establishment of neuroscience. During the past two decades, functional magnetic resonance imaging (fMRI), which is an in-vivo neuroimaging technique, has revolutionized the functional mapping of the brain [23], [24], [25], [22], [27], [26], [29], [8]. Specifically, task-based fMRI (tfMRI) has been widely used to record functional brain activities during a specific task performance and further to identify brain regions that are functionally involved in the task performance [24], [22], [27]. Meanwhile, resting state fMRI (rsfMRI) has also received intense interest more recently to acquire brain activities while participants are in a task-free state. The coherence in the functional brain organization which is free from the task performance constraint can be reflected based on the spontaneous signal changes during resting state [23], [25], [22], [27], [26], [29], [8].

Given the importance of fMRI (including both tfMRI and rsfMRI) data for functional brain mapping, tremendous efforts have been devoted on the establishment of fMRI informatics systems which recruit a comprehensive collection of statistical/computational approaches for fMRI data analysis [31], [32], [33], [26], [35], [36]. For example, MEDx is one of the earliest tools which was produced to incorporate advances in neuroimaging methods in 1993 [31]. Later on, FSL (FMRIB's Software Library) toolbox was developed to bring more insights to the neuroscience analysis tools and since June 2000 it has helped researchers globally apply FEAT, MELODIC, FABEER, BASIL and VERBENA tools for fMRI data processing and analysis [32], [33]. Moreover, statistical methods and tools have become one of the main tools to study brain networks and connectivity. For example, statistical parametric mapping (SPM) is one of the most influential tools which have been designed for brain imaging data sequence analysis from different cohorts

or time-series [30]. Analysis of Functional NeuroImages (AFNI) package is another tool to visualize and statistically analyze of fMRI data sets [35]. Furthermore, some have dedicated their resources to create a concentrate database to index the context and content of the fMRI literature in a searchable fashion, considering the multidisciplinary nature of fMRI researches and thousands of investigators around the globe. Fox and Lancaster have discussed demands of such a system and proposed BrainMap to address required applications [36], [58]. Although significant successes have been achieved for these fMRI informatics systems [59], [39], a considerable limitation is that all of those state-of-the-art systems are especially designed for specific fMRI sessions or studies of which the data size is not really big. As a consequence, there is difficulty for those systems to preprocess, analyze, and visualize fMRI 'big data' simultaneously.

With the advancement of neuroimaging technologies, the size of fMRI data is growing explosively. Given the lack of a uniform resource center for fMRI data providers, researchers and developers, Neuroimaging Informatics Tools and Resources Clearinghouse (NITRC) was established in 2006 to facilitate finding and computing neuroimaging resources for functional and structural neuroimaging analyses to be a common place to share required tools and data [60]. Although it was not for the first time that a government-funded project became an international neuroscience resource provider to cover pioneers worldwide, for example Neuroscience Information Framework (NIF) in 2004 [61] as well as Biomedical Informatics Research Network (BIRN) in 2001 [61], but NITRC was successful and popular to host and provide one of the biggest fMRI data-bases named 1000 Functional Connectomes (1000FC) resting state fMRI project. [https://www.nitrc.org/projects/fcon_1000/]. Moreover, there are other fMRI big datasets that are publicly available for researchers such as OpenfMRI [12] and Human Connectomes Project (HCP) [7]. HCP is a recent NIH-funded project devised to map the brain's communication network called

connectome. This project provides a collection of neural data along with an interface to graphically navigate the data. The OpenfMRI is a National Science Foundation funded project established in 2010 to provide resources for researches to upload their owned fMRI data and make them publicly available.

In short, the availability of fMRI big-data has globally attracted increasing attention for researchers in the neuroimaging field to test various methods and algorithms based on a 'big data' strategy. For instance, the velocity of studies as well as the variety and volumes of neuroimages are aggregating exponentially, which are among the biggest challenges nowadays [63]. As Van Horn studied and mentioned [16], the calculated neuroimaging data from listed articles in representative issues of Neuroimage have been increased drastically and it is being expected to grow exponentially. The average size of raw data per study is expected to be 15 GB in 2015 and 20 GB in 2020. Therefore, effective and efficient fMRI informatics systems which can process and analyze fMRI big data are much needed.

To deal with the abovementioned limitation of previous fMRI informatics systems and to address the need of effective fMRI informatics system which can process and analyze fMRI big data for researches, in this paper, we have developed a HAFNI-Enabled Largescale Platform for Neuroimaging Informatics (HELPNI) (http://bd.hafni.cs.uga.edu/helpni). This system is established using the extensible neuroimaging archive toolkit web application and storage solutions [64], a widely used open source system for storing, managing and analyzing medical images and related meta data [37]. RESTful application programming interface makes it especially useful for data sharing since the entire database's contents are reachable programmatically through the web application [37]. Specifically, the proposed HELPNI system in this work, implements our latest computational framework of sparse representation of whole-brain fMRI signals which is

called ‘Holistic Atlases of Functional Networks and Interactions’ (HAFNI) [52]. The main idea of HAFNI is to aggregate all of hundreds of thousands of tfMRI or rsfMRI signals within a whole brain of one subject into a big data matrix, which is subsequently factorized into an over-complete dictionary basis matrix (represented by the panel (I) of figure 2.1) and a reference weight matrix (represented by the panel (II) of figure 2.1) via an effective online dictionary learning algorithm [53], [54]. The time series of each over-completed basis dictionary represents the functional BOLD (blood-oxygen-level dependent) activities of a brain network (the white curves in the panel (II) of figure 2.1) and its corresponding reference weight vector stands for the spatial map of this brain network (the volume images in the panel (II) of figure 2.1). The HAFNI framework has been found to be effective and efficient in inferring a comprehensive collection of concurrent functional networks in the whole brain [52]. HELPNI covers the fMRI big data both from big data matrix and high volume of subjects. This happens first through employing HAFNI framework to handle the big data matrix for each subject and second by utilizing a database to store large scale datasets, and then using an scheduling engine to distribute analyzing tasks to multiple machines and to process multiple subjects simultaneously. HELPNI as an advanced informatics system, provided us with resources to identify large scale (over all 1200+) functional connectomes subjects automatically via automated computational pipeline based on our HAFNI framework function, to store the results in an organized data structure, and to generate detailed reports for data analysis (containing registration, online dictionary learning, and identified functional brain networks results) accessible through our web interface publicly. The HELPNI system significantly expands the previous neuroimaging archive toolkit by adding HAFNI capabilities, that is HAFNI-enabled, while significantly enhancing HAFNI by integrating the advanced informatics system.

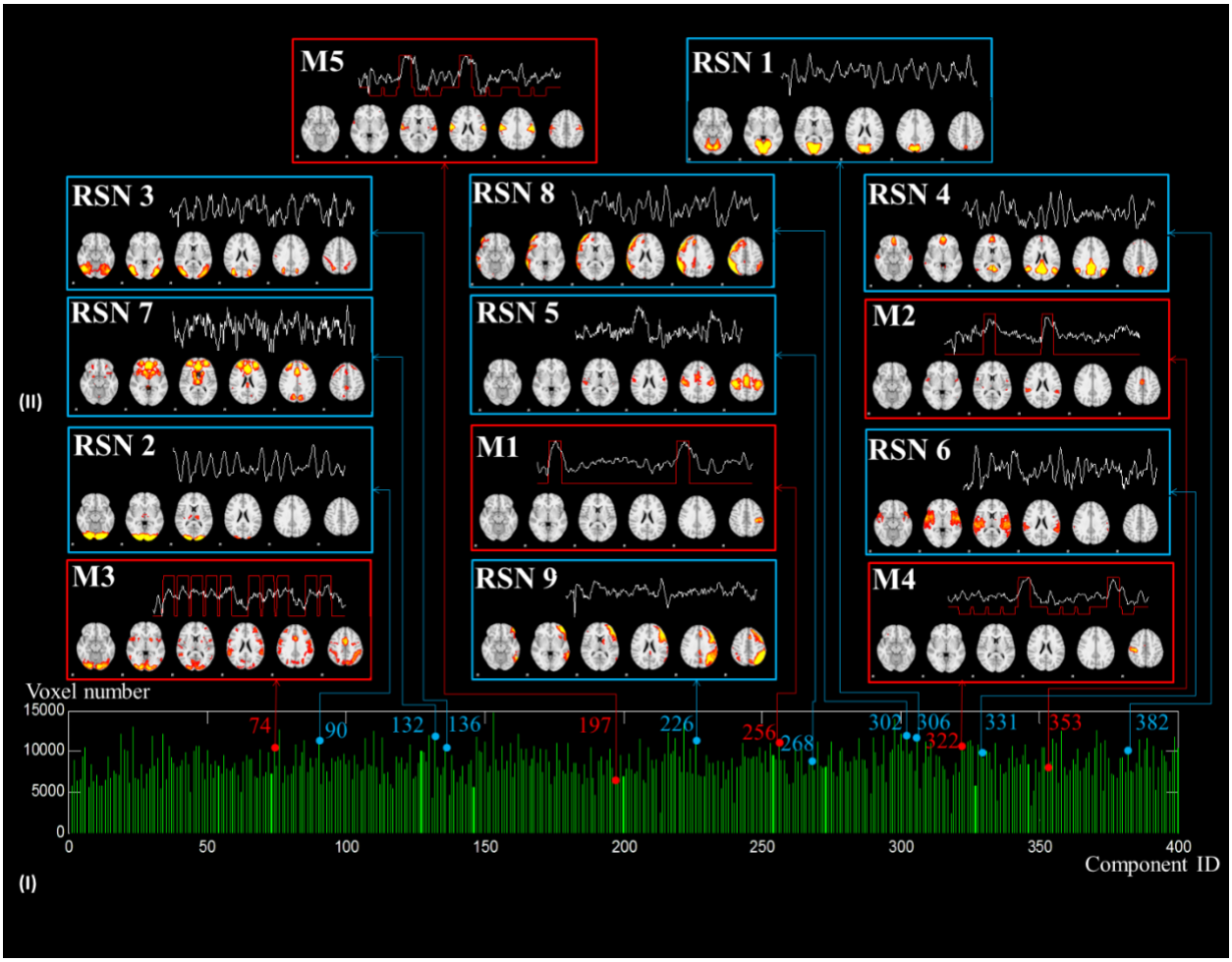

Figure 2.1. (I) The decomposed dictionary components from the fMRI data during one single task and (II) the 14 corresponding reference weight maps by applying the HAFNI method to the whole-brain fMRI signals. This figure visualizes 14 selected dictionary components which are either motor task-evoked networks (M1-M5) or resting state networks (RSN1-RSN9). The green bars in (I) show 400 dictionary network components (indexed along x-axis) and the spatial non-zero voxel numbers that each component's reference weight map contains (represented by the horizontal height of each bar). The panels in (II) visualize the temporal time series (white curve) and spatial distribution map (eight representative volume images) of each network. The red curves represent the task contrast designs of the motor tfMRI data.

The rest of this paper is organized as follows. We will describe the methods of development in addition to obtained results of HAFNI implementation in Section 2.2. We will also discuss the significance of this system in comparison to the previous methods of fMRI analysis studies. Results are provided in Section 2.3 and discussion and conclusion are in Section 2.4.

2.2 Method

In this section we first provide a technical overview of HELPNI system and then we discuss HAFNI implementation details and its workflow in our system. Subsequently, we will discuss the 1000FC database we used as the test bed in this paper.

2.2.1 Overview of HELPNI System

The main purpose of HELPNI is to store and manage large diverse imaging datasets to facilitate neuroimaging researches with complicated processes and large amount of data. The interesting feature of this platform is the extendibility, through which developers can customize their desired analytical and visualization tools. The platform uses XML schema to generate custom components, modules, workflows for different tiers. As the figure 2.2 elaborates, the standardize workflow helps users to a) capture imaging/non-imaging data and meta-data (either from neuroimaging machines or other databases manually); b) inspect data by means of pre-archiving feature; c) analyze data remotely or locally on-demand; d) collaborate easier using the predefined filter (In this way, collaborators can be noticed when a related dataset were added to system); and e) control access and share data where datasets and linked results can be shared publicly through the web interface to facilitate collaboration.

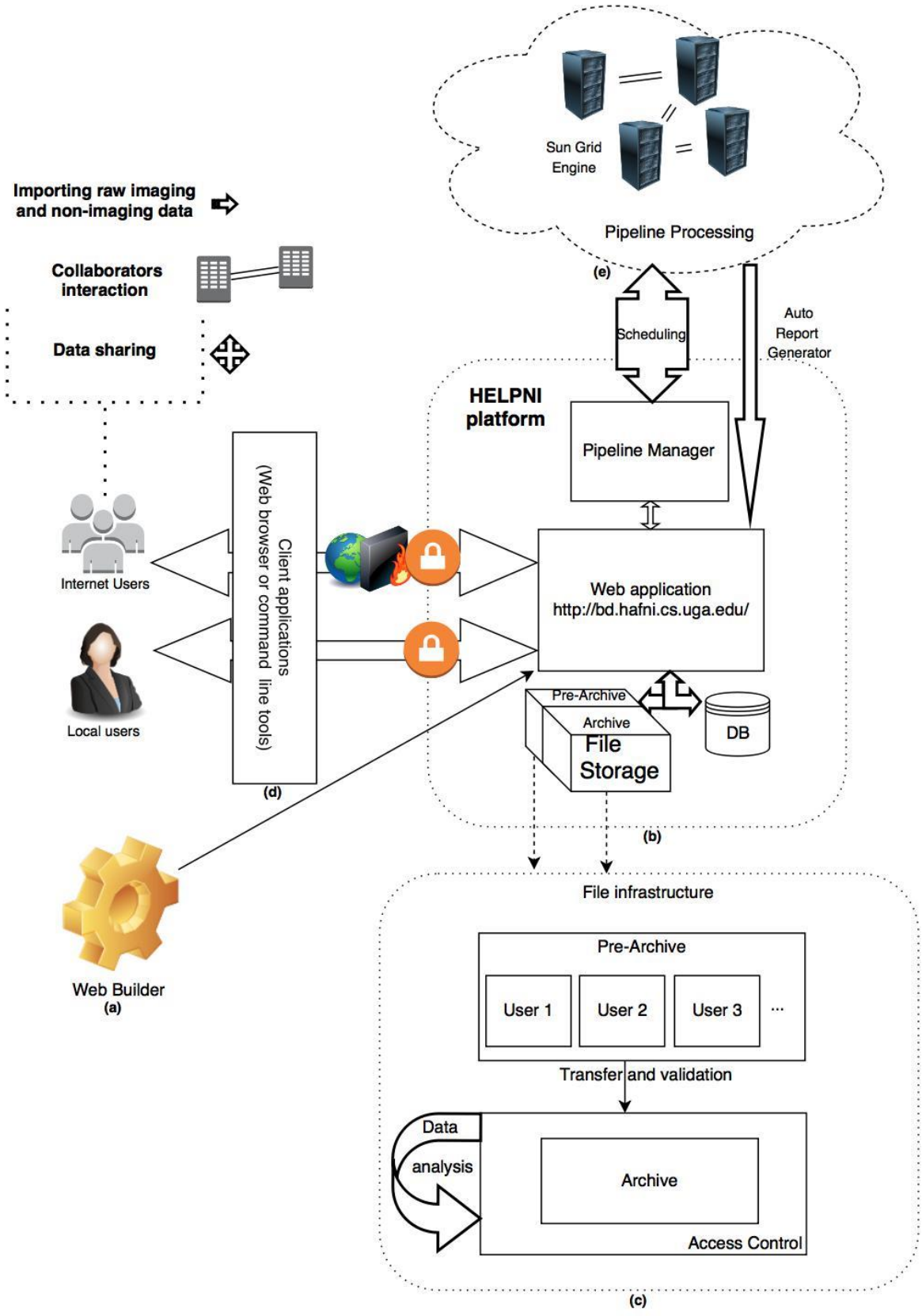
Sun Grid Engine
Pipeline Processing
(e)
Importing raw imaging and non-imaging data
Collaborators interaction
Data sharing
Scheduling
Auto Report Generator
HELPNI platform
Pipeline Manager
Internet Users
Local users
Client applications (Web browser or command line tools)
(d)
Web application http://bd.hafni.cs.uga.edu/
Pre-Archive
Archive
File Storage
DB
(b)
Web Builder
(a)
File infrastructure
Pre-Archive
User 1
User 2
User 3
...
Transfer and validation
Data analysis
Archive
Access Control
(c)

Figure 2.2. HELPNI structure and connected components. a) Web builder through which the web application will be built. b) HELPNI platform big picture. c) File infrastructure workflow consist of pre-archive and archive in which data will be temporary stored and then after user inspection and running required processes, data will be moved to their permanent destination where pipelines processes will be run on. d) Client application and users transactions. Local and global users connect to the web interface after logging into the system and passing firewall, using their preferred client application. Then they will be able to process, share, download and upload data interactively. e) Pipeline processing unit(s) that dynamically receive parameters and executives from pipeline manager and after running pre-defined steps, generate a user-friendly report along with required notifications and then will store the results into file storage.

In the HELPNI system, we implemented our recently developed HAFNI framework for fMRI data analysis using the extendible pipelines. Pipeline is a workflow described in an XML document. Parameters could be specified within the XML document or be sourced as another XML document. So far we have implemented a few pipelines each of which contains different sets of scripts for our HAFNI framework. These pipelines can both extract input parameters from subjects automatically or ask users to provide them manually. Pipeline engine works based on the Java framework which parses parameters out of XML document and it links sequence of activities into a defined process flow and can manage data flow at each step. It can be configured to send notification at desired step(s) for quality control or to modify parameters, then restart pipeline from where it stopped. We have used pipeline to automate the whole processes of fMRI data registration and online dictionary learning (ODL) and to reduce the processing time. It also helped to run the data over a very large set of data in much less amount of time as we implemented it over the

1000FC data. Pipelines can leverage from distributed computing and in this way a huge amount of processes can result in much less computation time.

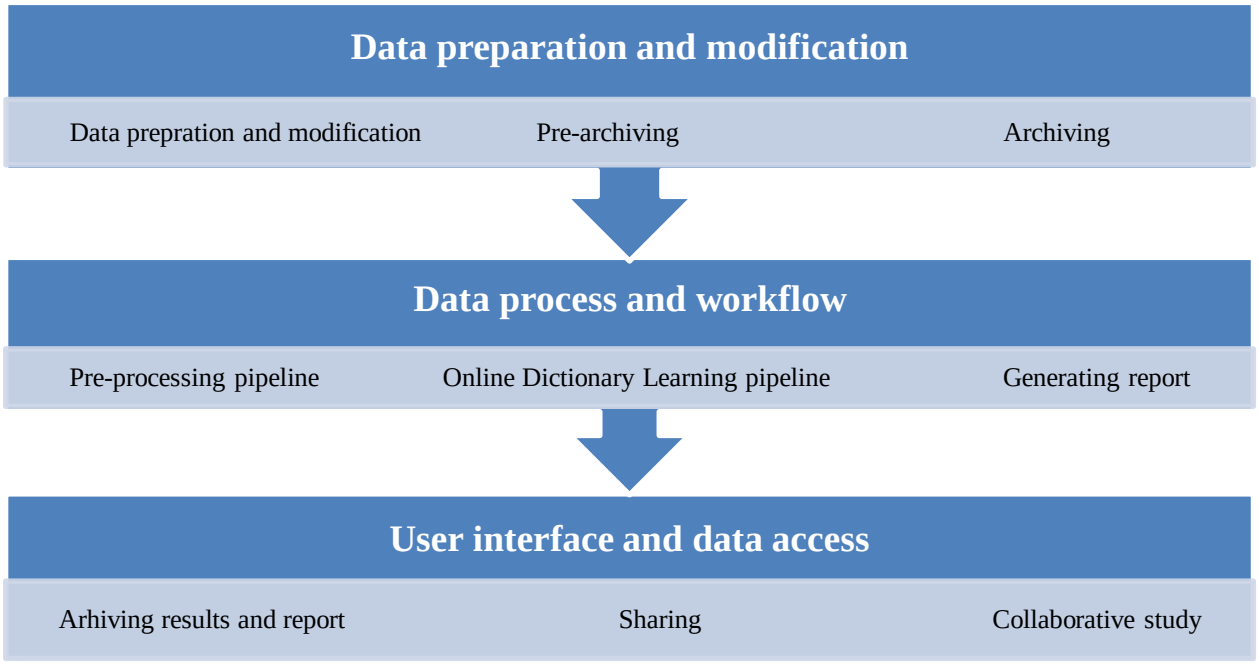

Figure 2.3. An overview of HAFNI implementation through HELPNI and its workflow.

In this work, we used the 1000FC project datasets as test bed for HELPNI system developing and testing. The 1000FC project contains 1200+ resting state functional MRI (rsfMRI) images collected from 33 locations. We defined a workflow to obtain the result as we discuss here. Figure 2.3 shows the implemented pipelines and workflow of our process from the beginning of obtaining fMRI data from NITRC to data process steps and finally result reporting. The main three steps of this workflow are a) data preparation and modification; b) data process and workflow; and c) user interface and data access as detailed in Sections 2.2 and 2.3, respectively.

2.2.2 Data Preparation and Modification

At the very first step, users need to prepare data to import to system. We first obtained data from 1000FC database and modified the data structure as our own predefined structure. After modifying hierarchy and trimming data, images with correspondent meta-data should be uploaded to pre-archive for primary tests and analysis. The required format of data should be created in file system including ID and sequence type as well as any special data type that needs to be defined in system. To do so we prepared required meta-data including TR value, field strength, gender and handedness of each subject and experiment. Then data were transferred to pre-archive as a temporary cache destination for further tests and review of quality (figure 2.2c). Pre-archiving step keeps data integrated and protects them from data loss or corruption. We also tested our workflow to fix any possible flaw in implemented algorithms. When data became ready and analytical methods turn mature to be modeled in XML schema, we imported data into the archive as final destination for viewing purposes and/or running standard processes on prepared data. We used curl to upload fMRI data through REST API [65] from command line.

2.2.3 Data Process and Workflow

The next step in HELPNI platform is data processing. The raw fMRI data need to be pre-processed before data analysis. We implemented the rsfMRI and tfMRI pre-processing pipeline in HELPNI to address this demand. Our preprocessing step includes skull removal, motion correction, slice time correction and special smoothing as well as global drift removal.[8]. We used *Build* and *ArcBuild* [37] predefined XNAT tools for image session scan selection and running processing steps, respectively.

Applying the major processing pipeline is the next step. We integrated our HAFNI (Holistic Atlases of Functional Networks and Interactions) computational framework in HELPNI. The basic

idea of HAFNI framework [52] is to aggregate all of the thousands of fMRI signals within the whole brain from one subject into a big data matrix and then decompose it into an over-completed dictionary matrix and a reference coefficient matrix. Specifically, each column of the dictionary matrix represents a typical brain activity pattern and the corresponding row in coefficient matrix naturally reveals the spatial distribution of the activity pattern. Typically, each subject brain's signals form an $m \times n$ matrix S, with m represents the fMRI time points (observations) and n represents the number of voxels. In order to sparse represent the signal matrix S using D, we aimed to learn a meaningful and over-completed dictionary matrix $D \epsilon \mathbb{R}^{m \times k}$ ($k>m$, $k<<n$), with k being the dictionary atoms (i.e. components). The loss function is defined in Eq. (1) with a ℓ_1 regularization that yields to a sparse resolution of α_i.

$$\ell(s_i, D) \triangleq \min_{\alpha_i \epsilon \mathbb{R}^m} \frac{1}{2} ||s_i - D\alpha_i||_2^2 + \lambda ||\alpha_i||_1 \quad (1)$$

Here α_i is the coefficient matrix and λ is a sparsity regularization parameter. In order to prevent D from arbitrarily large values, the columns $d_1, d_2, \ldots \ldots d_m$ are constrained by Eq. (2).

$$C \triangleq \left\{ D \epsilon \mathbb{R}^{t \times m} \quad s.t. \quad \forall j = 1, \ldots m, \qquad d_j^T d_j \leq 1 \right\} \quad (2)$$

$$\min_{D \epsilon C, \alpha \epsilon \mathbb{R}^{m \times n}} \frac{1}{2} ||S - D\alpha||_F^2 + \lambda ||\alpha||_1 \quad (3)$$

Briefly, the problem can be transferred into a matrix factorization problem in Eq.(3) and we adopted the state-of-the-art online dictionary learning algorithm [54] for the sparse representation of the whole brain fMRI signals.

Once we obtained the learned dictionary matrix D and coefficient matrix α, we mapped each row in the α matrix back to the brain volume and examine their spatial distribution patterns, through which functional network components are characterized on brain volumes [52]. At the conceptual level, the sparse representation framework in figure 2.4 can achieve both compact high-fidelity

representation of the whole-brain fMRI signals (figure 2.4c) and effective extraction of meaningful patterns (figure 2.4d) [53] , [54], [66], [67], [68], [69]. For more details please refer to our recent literature report [52].

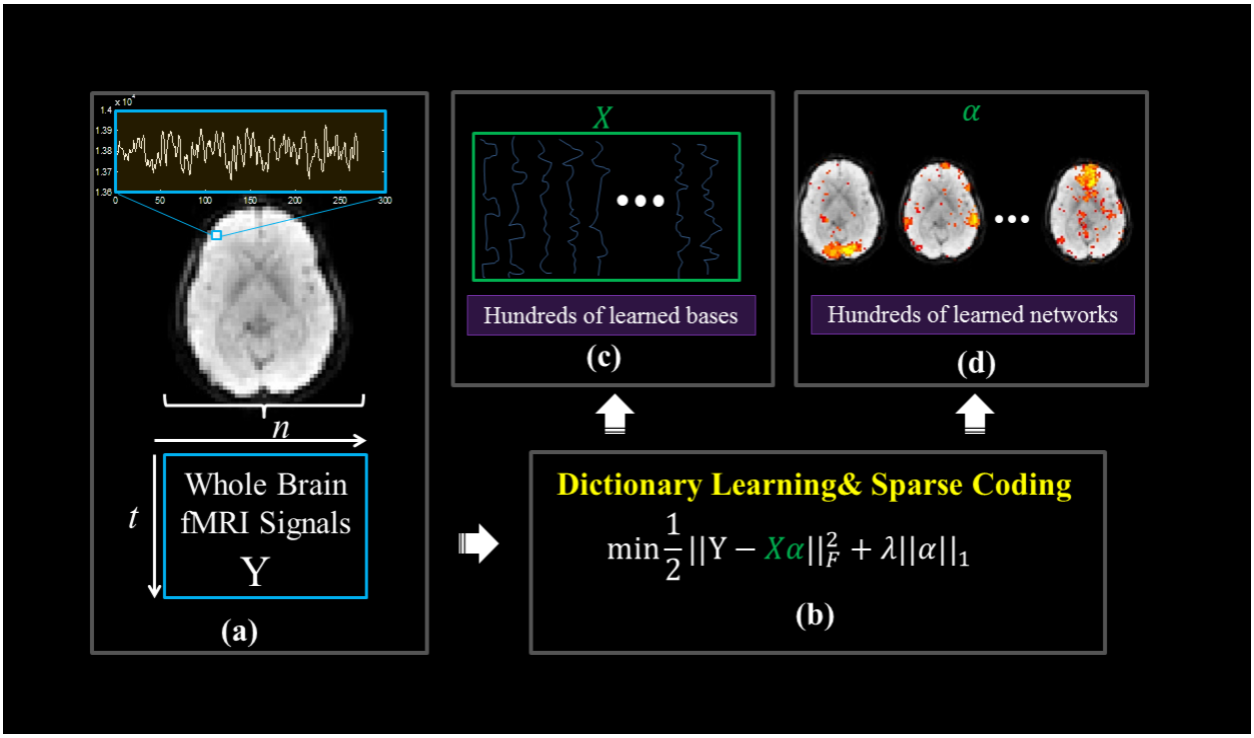

Figure 2.4. The computational pipeline of sparse representation of whole-brain fMRI signals using an online dictionary learning approach. (a) The whole-brain fMRI signals are aggregated into a big data matrix, in which each row represents the whole-brain fMRI BOLD data in one time point and each column contains the time series of one single voxel. (c) Illustration of the learned atomic dictionary, each dictionary represents one functional network component. (d) The coefficient matrix, each row in the matrix measures the weight coefficient of the corresponding dictionary over the whole brain. That is, each row defines the contribution of one dictionary to the composition of all voxel-wise fMRI signals.

The system is designed to feed the preprocessing as the input of online dictionary learning pipeline automatically or manually after filtering the preprocessed data. For visualization purposes and to make the generated results easy to explore, both preprocessing and ODL pipelines will generate a PDF report at the end after which will be automatically uploaded to the web-interface. These reports contain generated results from the executed pipelines identified by experiment ID appended to pipeline name. For example, ODL report contains 400 png files sorted sequentially.
Pipelines can also be set to send notification within different steps of workflow. For example, user can be notified when a specific step is done to evaluate the result and then if it meets the quality, let the pipeline continue. Otherwise user can modify the input variables and restart the pipeline. Also, at the end of workflow, assigned users will be notified of a successful run.

2.2.4 User Interface and Data Access

Large scale fMRI data usually needs group-wise analysis and collaborators need to work together. In HELPNI, users can connect to system remotely and choose their desired subset of archive through bundle feature in the system. Users are also able to email other collaborators a link containing selected subset of archive.

The standard user interface features useful tools including a search box which provides searching through all archived subjects and sessions and menus in which users upon their permissions can access. Users need to login to system to be able to modify or upload new data but viewing and downloading 1000FC data as well as preprocessing and ODL results are publicly available (http://bd.hafni.cs.uga.edu/helpni). User can browse experiments and data via three methods. One is by selecting project and subject subsequently, the other is through searching for a subject name from search box, and the last is through selecting a listing. Where user can input certain

information of project/subject or image modality and then query a list containing correspondent filtered data.

2.3 Results

We tested the proposed HELPNI platform by applying the implemented computational framework of HAFNI on one of the largest open-source resting-state fMRI (rsfMRI) database: 1000 Functional Connectomes project (known as 1000FC). This database has gathered more than 1200 rsfMRI datasets independently collected from all over the world containing over 130 Giga Bytes 0f data. Table 2.1 summarized rsfMRI datasets. Age, sex and imaging center information are provided for each of datasets and all subjects have been uploaded to the HELPI. As detailed in Section 2.2, HELPNI automatically preprocessed the raw rsfMRI data, extracted the rsfMRI signals from each subject, applied the HAFNI computational framework, and returned and stored meaningful experimental results. In this experiment, we used 8-core Intel® Xeon® E5-2650 v2 2.60GHz, 20M Cache CPU and 32GB RDIMM, 1600MT RAM. With the help of HELPNI, we identified consistent and meaningful functional brain networks across individuals and populations based on rsfMRI big data which are detailed in section 2.3.1. Moreover, using HELPNI possess modularity and plug-and-play capability, we developed an efficient sampling module and integrated it with HAFNI framework to speed up the HAFNI overall computational time and to automatically calculate and obtain meaningful functional brain networks in a much faster fashion. The detailed results are demonstrated in section 2.3.2.

Table 2.1. The 1000 Functional Connectomes Project datasets summery.

Baltimore	**Bangor**	**Beijing_Zang**	**Berlin_Margulies**
(n = 23 [8M/15F]; ages: 20-40; TR = 2.5; # slices = 47; # timepoints = 123)	(n = 20 [20M/0F]; ages: 19-38; TR = 2; # slices = 34; # timepoints = 265)	(n = 198 [76M/122F]; ages: 18-26; TR = 2; # slices = 33; # timepoints = 225)	(n = 26 [13M/13F]; ages: 23-44; TR = 2.3; # slices = 34; # timepoints = 195)
Cambridge_Buckner	**Cleveland CCF**	**Dallas**	**Durham_Madden**
(n = 198 [75M/123F]; ages: 18-30; TR = 3; # slices = 47; # timepoints = 119)	(n = 31 [11M/20F]; ages: 24-60; TR = 2.8; # slices = 31; # timepoints = 127)	(n = 24 [12M/12F]; ages: 20-71; TR = 2; # slices = 31; # timepoints = 115)	(n = 42 [n/a]; ages: n/a; TR = n/a; # slices = n/a; X timepoints = n/a)
ICBM	**Leiden_2180**	**Leiden_2200**	**Leipzig**
(n = 86 [41M/45F]; ages: 19-85; TR = 2; # slices = 23; # timepoints = 128)	(n = 12 [12M/0F]; ages: 20-27; TR = 2.18; # slices = 38; # timepoints = 215)	(n = 19 [11M/8F]; ages: 18-28; TR = 2.2; # slices = 38; # timepoints = 215)	(n = 37 [16M/21F]; ages: 20-42; TR = 2.3; # slices = 34; # timepoints = 195)
Milwaukee_a	**Milwaukee_b**	**Munchen**	**Newark**
(n = 18 [n/a]; ages: n/a; TR = 2; # slices = 20; # timepoints = 175)	(n = 46 [15M/31F]; ages: 44-65; TR = 2; # slices = 64; # timepoints = 175)	(n = 16 [10M/6F]; ages: 63-73; TR = 3; # slices = 33; # timepoints = 72)	(n = 19 [9M/10F]; ages: 21-39; TR = 2; # slices = 32; # timepoints = 135)
NewHaven_a	**NewHaven_b**	**NewYork_a_ADHD**	**NewYork_a**
(n = 19 [10M/9F]; ages: 18-48; TR = 1; # slices = 16; # timepoints = 249)	(n = 16 [8M/8F]; ages: 18-42; TR = 1.5; # slices = 22; # timepoints = 181)	(n = 25 [19M/4F]; ages: 20-50; TR = 2; # slices = 39; # timepoints = 192)	(n = 84 [43M/41F]; ages: 7-49; TR = 2; # slices = 39; # timepoints = 192)
NewYork_b	**NewYork_Test-Retest_Reliability**	**Ontario**	**Orangeburg**
(n = 20 [8M/12F]; ages: 18-46; TR = 2; # slices = 33; # timepoints = 175)	(n = 25 [10M/15F]; ages: 22-49; TR = 2; # slices = 39; # timepoints = 197)	(n = 11 [n/a]; ages: n/a; TR = 3; # slices = 29; # timepoints = 105)	(n = 20 [15M/5F]; ages: 20-55; TR = 2; # slices = 22; # timepoints = 165)
Oulu	**Oxford**	**PaloAlto**	**Pittsburgh**
(n = 103 [37M/66F]; ages: 20-23; TR = 1.8; # slices = 28; # timepoints = 245)	(n = 22 [12M/10F]; ages: 20-35; TR = 2; # slices = 34; # timepoints = 175)	(n = 17 [2M/15F]; ages: 22-46; TR = 2; # slices = 29; # timepoints = 235)	(n = 17 [10M/7F]; ages: 25-54; TR = 1.5; # slices = 29; # timepoints = 275)
Queensland	**SaintLouis**	**Taipei_a**	**Taipei_b**

(n = 19 [11M/8F]; ages: 20-34; TR = 2.1; # slices = 36; # timepoints = 190)	(n = 31 [14M/17F]; ages: 21-29; TR = 2.5; # slices = 32; # timepoints = 127)	(n = 14 [n/a]; ages: n/a; TR = 2; # slices = 32; # timepoints = 295)	(n = 8 [n/a]; ages: n/a; TR = 2; # slices = 33; # timepoints = 175)
Atlanta	**AnnArbor_a**	**AnnArbor_b**	
ages: 22-57; TR = 2; # slices = 20; # timepoints = 205)	(n = 25 [22M/3F]; ages: 13-40; TR = 2; # slices = 40; # timepoints = 295)	(n = 36 [17M/19F]; ages: 19-80; TR = 0.75; # slices = 16; # timepoints = 395)	

2.3.1 Group-Wise Consistent Functional Brain Networks Identification Using HELPNI

With the help of HELPNI system and the implemented HAFNI computational framework, we successfully identified 10 meaningful and consistent resting state networks (RSNs) which are in agreement with previous studies across all individuals and datasets in 1000FC database. Figure 2.5 shows the identified 10 group-wise consistent networks in five randomly selected datasets (that are Baltimore, Beijing, Berlin, Cambridge and Cleveland dataset) in 1000FC. Networks #1, #2 and #3 are all located in visual areas and closely related to visual behavior. Network #4 includes ventromedial frontal cortex, bilateral inferior-lateral-parietal and medial parietal areas and are often referred as default mode network (DMN). Network #5 covers the cerebellum and corresponds to action-execution function. Networks #6, #7 and #8 are related to sensorimotor, auditory, and executive control function, respectively. Networks #9 and #10 cover several front parietal areas and are closely related to cognition/language paradigms [70]. Figure 2.6 illustrates the identified 10 consistent networks in 5 randomly selected individual subjects from the same 5 datasets. We can see that the identified 10 functional networks are quite consistent across different datasets and subjects and consistent with the templates in previous studies [70]. Quantitatively, we calculate the spatial overlap between the identified networks and templates which are detailed in Table 2.2 and Table 2.3. The spatial overlap is calculated as the percentage of the overlapping area

between our identified networks and templates (Lv et al., 2015). Based on these results, we can see that our developed HELPNI system is effective and efficient in reconstructing meaningful functional brain networks from rsfMRI data.

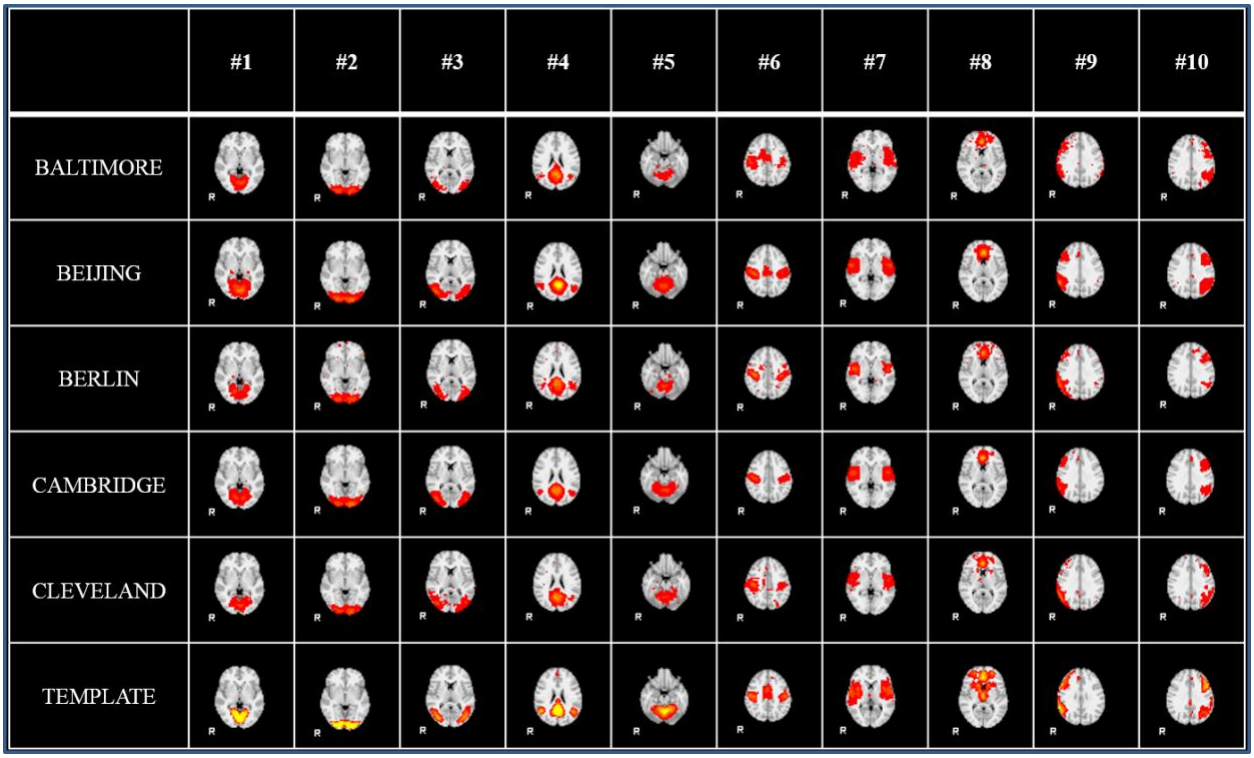

Figure 2.5. The identified group-wise consistent 10 RSN networks from 5 randomly selected datasets (Baltimore, Beijing, Berlin, Cambridge and Cleveland) in 1000 Functional Connectomes Project by HELPNI. Each row represents the networks from one dataset; the last row shows the RSN templates for comparison. Only the most informative slice, which has been overlaid on the MNI152 template, is shown here.

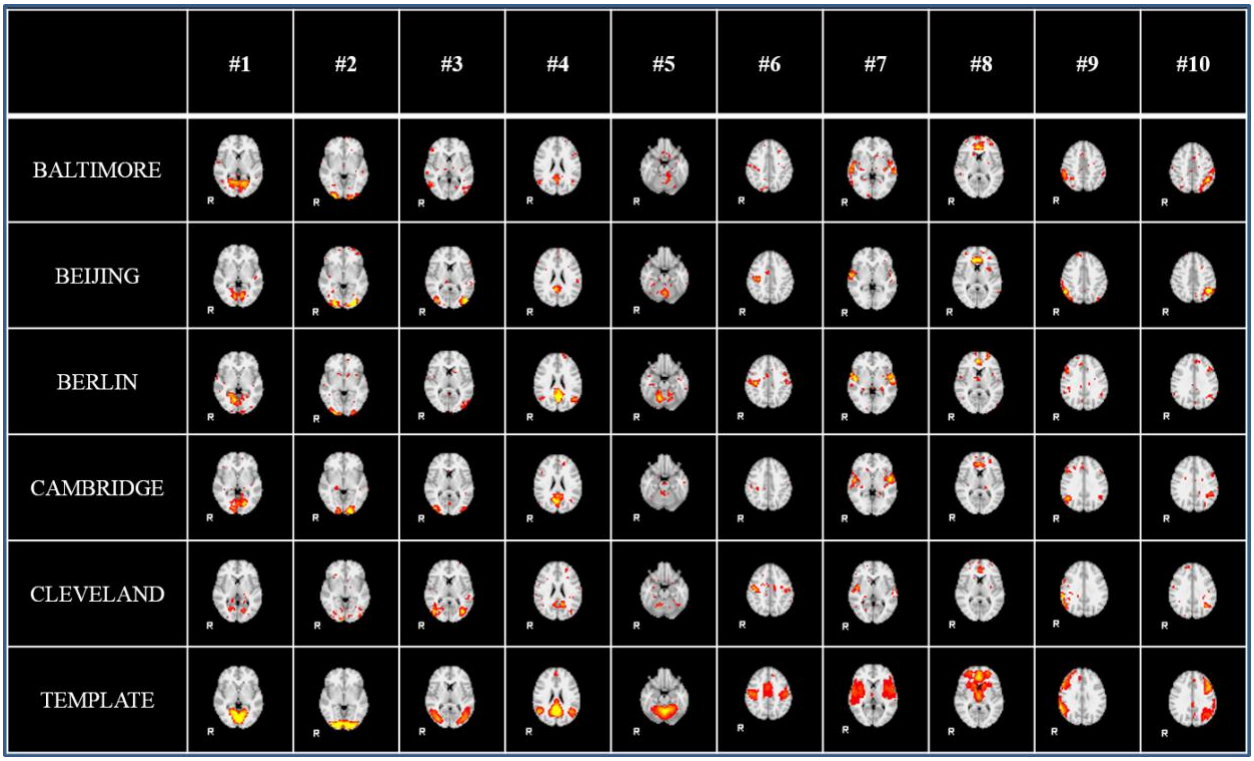

Figure 2.6. The identified 10 RSN networks of individual subject from 5 datasets (Baltimore, Beijing, Berlin, Cambridge and Cleveland) in 1000 Functional Connectomes Project by HELPNI. For each dataset, the 10 RSN networks from one randomly selected subject are shown here.

Table 2.2. Spatial overlap between identified group-wise RSNs and templates in different datasets

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Baltimore	0.88	0.94	0.82	0.74	0.75	0.78	0.65	0.61	0.67	0.71
Beijing	0.95	0.98	0.95	0.82	0.86	0.94	0.85	0.58	0.66	0.82
Berlin	0.81	0.95	0.86	0.80	0.72	0.77	0.71	0.60	0.73	0.82
Cambridge	0.86	0.98	0.92	0.76	0.93	0.79	0.80	0.56	0.69	0.78
Cleveland	0.82	0.89	0.80	0.77	0.72	0.75	0.72	0.58	0.53	0.75

Table 2.3. Spatial overlap between identified individual RSNs and templates in different datasets

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Baltimore	0.34±0.09	0.28±0.09	0.29±0.09	0.33±0.05	0.23±0.05	0.30±0.07	0.21±0.06	0.24±0.05	0.21±0.05	0.23±0.06
Beijing	0.36±0.09	0.29±0.12	0.32±0.12	0.37±0.08	0.28±0.09	0.41±0.10	0.25±0.07	0.27±0.08	0.24±0.06	0.26±0.06
Berlin	0.32±0.06	0.29±0.09	0.24±0.10	0.33±0.06	0.23±0.07	0.36±0.09	0.25±0.06	0.26±0.05	0.27±0.08	0.26±0.05
Cambridge	0.35±0.08	0.32±0.10	0.33±0.12	0.35±0.07	0.41±0.10	0.40±0.09	0.25±0.06	0.29±0.05	0.23±0.05	0.24±0.05
Cleveland	0.32±0.09	0.27±0.13	0.25±0.11	0.35±0.06	0.19±0.08	0.36±0.09	0.22±0.06	0.27±0.06	0.24±0.06	0.22±0.05

2.3.2 Integrating Sampling Module in HELPNI

One important characteristics of our HELPNI system is the plug-and-play capability. Since the implemented pipelines are modularly designed, we could easily develop and test new modules to enhance established computational framework. For example, in order to speed up the current HAFNI framework in the HELPNI system, we developed and integrated and efficient signal sampling module [71] to improve the calculating speed while obtaining comparable results. The average computation time of training a dictionary for one individual brain is about 30 seconds using sampling module, whereas the time cost without sampling is 340 seconds, which speeds up the HAFNI training procedure more than 10 times. At the same time, the returned results could identify the similar consistent and meaningful functional brain networks across datasets and individuals as discussed in Section 3.1. Figure 2.7 shows the same identified 10 group-wise consistent networks with sampling module in the same five datasets (that is Baltimore, Beijing, Berlin, Cambridge and Cleveland dataset) in 1000FC. Figure 2.8 illustrates the identified 10 consistent networks in the same 5 individual subjects in section 3.1. Similar to original HAFNI computational framework with no sampling module, the identified 10 functional networks are quite consistent with each other across different datasets and populations and consistent with the templates in previous studies [70]. Quantitatively, we calculated the spatial overlap between the

identified networks and templates which detailed in Table 2.4 and Table 2.5. From these results, we can see that the integrated sampling module in HAFNI framework via HELPNI system significantly decreased the computing time while achieved comparable results for functional brain network identification at the same time. It also demonstrates the plug-and-play capability of HELPNI system to effectively detect meaningful functional brain networks from raw neuroimaging data.

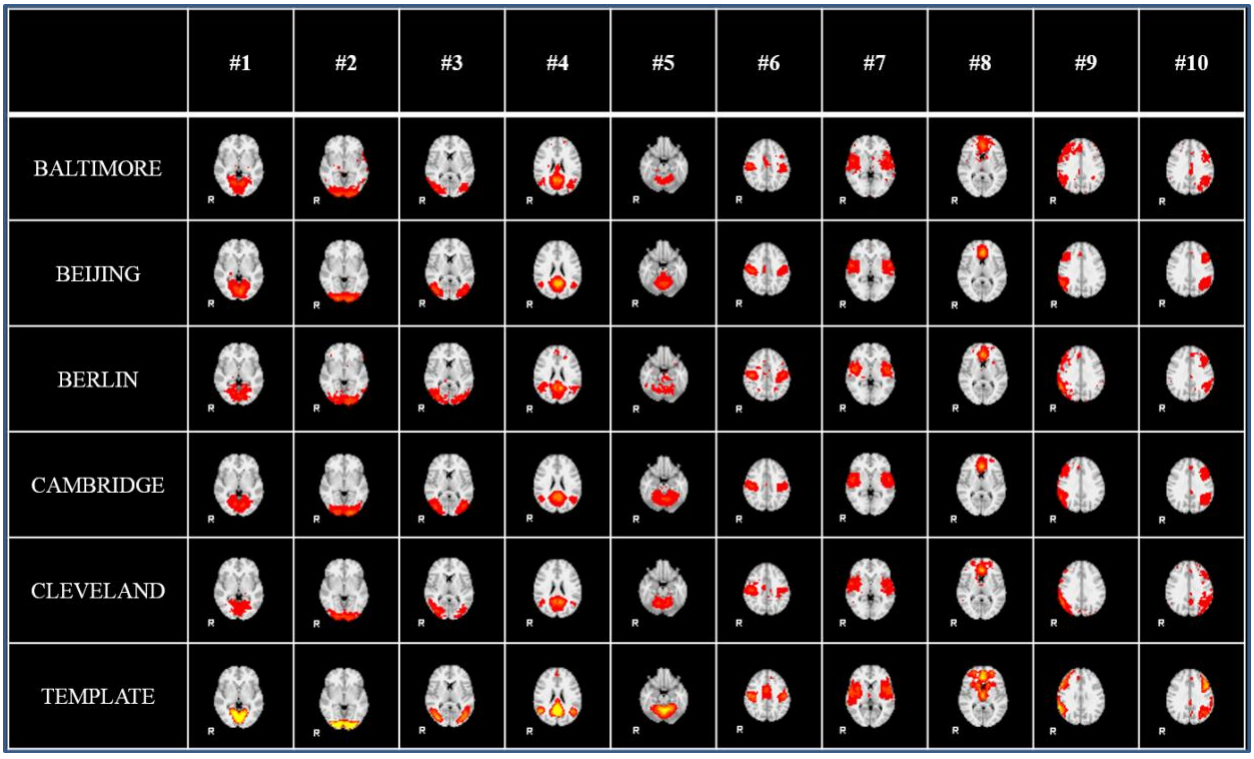

Figure 2.7. The identified group-wise consistent 10 RSN networks from 5 datasets (Baltimore, Beijing, Berlin, Cambridge and Cleveland) in 1000 Functional Connectomes Project by HELPNI with sampling module. Each row shows the networks from one data set and the last row shows the RSN templates for comparison.

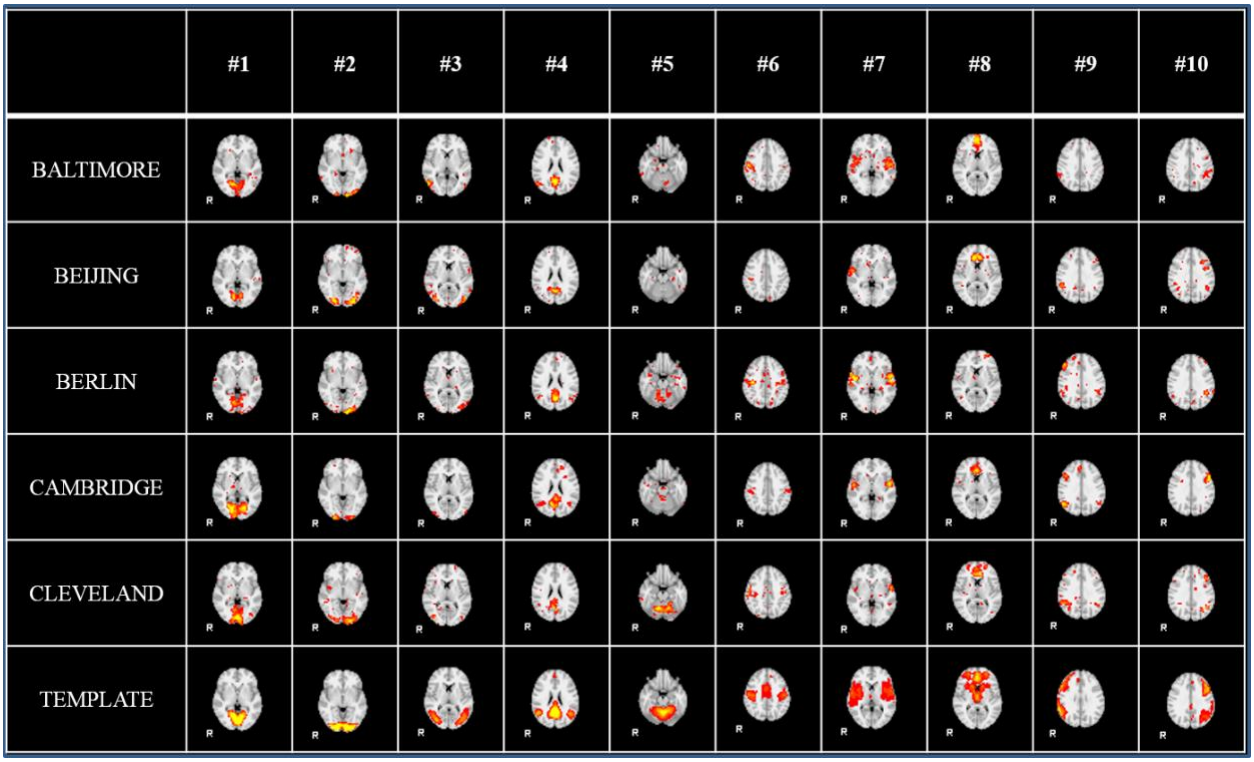

Figure 2.8. The identified 10 RSN networks of individual subject from 5 datasets (Baltimore, Beijing, Berlin, Cambridge and Cleveland) in 1000 Functional Connectomes Project by HELPNI with sampling module. For each dataset, we randomly selected one subject's result as example.

Table 2.4. Spatial overlap between identified group-wise RSNs with sampling module and templates in different datasets

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Baltimore	0.89	0.89	0.82	0.79	0.76	0.92	0.64	0.59	0.68	0.72
Beijing	0.94	1.00	0.95	0.89	0.88	0.97	0.88	0.63	0.74	0.87
Berlin	0.87	0.95	0.90	0.83	0.73	0.87	0.76	0.68	0.88	0.82
Cambridge	0.84	0.98	0.94	0.84	0.95	0.86	0.82	0.57	0.68	0.83
Cleveland	0.80	0.95	0.88	0.82	0.75	0.75	0.77	0.61	0.57	0.74

Table 2.5. Spatial overlap between identified individual RSNs with sampling module and templates in different datasets

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Baltimore	0.38±0.09	0.30±0.10	0.29±0.10	0.35±0.06	0.26±0.06	0.36±0.08	0.21±0.06	0.29±0.07	0.24±0.07	0.25±0.06
Beijing	0.39±0.11	0.32±0.13	0.34±0.13	0.39±0.09	0.31±0.10	0.43±0.11	0.29±0.08	0.31±0.10	0.27±0.07	0.29±0.08
Berlin	0.36±0.06	0.31±0.10	0.28±0.12	0.36±0.08	0.24±0.07	0.39±0.08	0.26±0.06	0.32±0.05	0.28±0.07	0.28±0.06
Cambridge	0.37±0.08	0.34±0.11	0.33±0.12	0.37±0.07	0.44±0.12	0.41±0.09	0.27±0.06	0.32±0.05	0.26±0.06	0.26±0.06
Cleveland	0.34±0.11	0.29±0.13	0.25±0.11	0.36±0.05	0.20±0.08	0.38±0.08	0.24±0.06	0.32±0.08	0.26±0.07	0.24±0.06

2.4 Discussion and Conclusion

In this work, we have designed and developed a neuroimaging informatics platform, HELPNI, to archive large-scale fMRI datasets, to automate sequence of complex processes for fMRI data analysis and finally to use distributed and parallel computing resources to bust up big data analysis time. HELPNI has leverage from extensible neuroimaging archive toolkit to power up the web application and storage part of the system and is composed of three main parts of web application and storage, pipeline analysis framework and the big data analytic tools. This novel platform integrated our recently developed HAFNI computational framework for fMRI data analysis in an accelerated way. As demonstrated in this work, we used the open access 1000 functional connectome datasets as a basic example to import 1200+ rsfMRI data into HELPNI system, to run the HAFNI framework on the rsfMRI data, and to identify consistent and meaningful functional brain networks across individuals and populations. Our experimental results demonstrated that efficient sampling module can be implemented together with HAFNI framework to speed up the dictionary learning and identification of meaningful functional brain networks.

The HELPNI platform is publicly accessible through http://bd.hafni.cs.uga.edu/helpni where users can view all of the archived fMRI data as well as the processed results. Authorized users can also upload new data and run pipelines over their desired fMRI images.

Considering the explained characteristics (Section 3.2) as well as the task scheduling feature of our HELPNI (figure 2.3e) in which tasks can be run in a distributed or parallel fashion, HELPNI with plug and play capability and modularity can significantly speed up the fMRI data processing. Users can easily feed their workflow to the HELPNI and it will schedule, distribute and run all tasks using all available resources and will notify users with the final results. We are also implementing big data analytic tools to empower the processing part through Hadoop and Spark. Parallel optimization procedure has shown significance improvement in sparse dictionary learning computation time [72].

The large-scale datasets can be imported to the HELPNI system and various computational pipelines and analyses can be then run over the big data without corrupting the original archived images. For example, in this paper, we ran the HAFNI pipeline over all subjects in 1000FC project and the users could examine the results in a well-structured report in addition to original image data. We also ran the sampling pipeline on a subset of the dataset and stored them in the same fashion. In this way, users can evaluate and compare the results with sampling and no sampling simultaneously. The HELPNI system saved much computing time since there was no idle time in between of processes using the task scheduling feature. In the future, the distributed scheduling and big data analytics tools are planning to be used to save more time by means of distributed system available at the University of Georgia. This will provide fMRI community to use HELPNI system integrated with other analytical tools on large-scale fMRI datasets and to collaborate with other laboratories and research centers.

Adding a few new features including auto classifying the stored images based on the analysis results, fully implementing the parallel algorithm for HAFNI and improve the current user interface of HELPNI are scheduled as our future improvements to HELPNI. Future applications of HELPNI include testing other big datasets such as HCP and OpenfMRI, implementing new modules such as population clustering of learned dictionary HAFNI spatial maps, and eventually discovering disease specific biomarkers.

Acknowledgements

We thank all investigators contributing data to the 1000 Functional Connectomes project, without whom this analysis could not have been performed. T. Liu was supported by NIH DA033393, AG042599 and NSF IIS-1149260, CBET-1302089, and BCS-1439051. J Lv was supported by the China Government Scholarship and the Doctorate Foundation of NWPU. This work includes XNAT, developed by Randy Buckner at Harvard University and the Neuroinformatics Research Group at Washington University School of Medicine.

CHAPTER 3

A DISTRIBUTED COMPUTING PLATFORM FOR FMRI BIG DATA ANALYTICS [1,2]

[1] Li, Xiang*, Milad Makkie*, Binbin Lin, Mojtaba Sedigh Fazli, Ian Davidson, Jieping Ye, Tianming Liu, and Shannon Quinn. “Scalable fast rank-1 dictionary learning for fMRI big data analysis." In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 511-519. ACM, 2016.

Reprinted here with permission of the publisher.

[2] Makkie, Milad*, Xiang Li*, Shannon Quinn, Binbin Lin, Jieping Ye, Geoffrey Mon, and Tianming Liu. "A Distributed Computing Platform for fMRI Big Data Analytics." Accepted by IEEE Transactions on Big Data (2018).

Reprinted here with permission of the publisher.

ABSTRACT

It has been shown from various functional neuroimaging studies that sparsity-regularized dictionary learning could achieve superior performance in decomposing comprehensive and neuroscientifically meaningful functional networks from massive fMRI signals. However, the computational cost for solving the sparse coding problem has been known to be very demanding, especially when dealing with large-scale fMRI data sets. Thus, in this work, we proposed a novel distributed rank-1 dictionary learning (D-r1DL) model and applied it for fMRI big data analysis. The model aims at estimating one rank-1 basis vector with sparsity constraint on its loading coefficient from the input data at each learning step through alternating least squares updates. By iteratively learning the rank-1 basis and deflating the input data by the learning results in each step, the model is then capable of decomposing the whole set of functional networks or connectomes. We implemented and parallelized the rank-1 dictionary learning algorithm using Spark engine and deployed the resilient distributed dataset (RDDs) abstracts for the input data. Experimental results on the Human Connectome Project (HCP) data have shown that the proposed D-r1DL model is fast and scalable towards fMRI big data analytics, thus enabling data-driven neuroscientific discovery from massive fMRI big data in the future.

3.1 Introduction

In recent years, the field of neuroimaging studies based on functional magnetic resonance imaging (fMRI) has featured unprecedented large-scale data availability thanks to the efforts from a series of data collection works including Human Connectome Project [7], 1000 Functional Connectomes [11] and OpenfMRI Project [12]. The rapidly growing data characterized different aspects of brain cognitive processes as well as various disorders, thus providing a great opportunity for decoding and identification of potential functional biomarkers for brain diseases. Consequently, there is an urgent call for more efficient and scalable data analytics and knowledge discovery methods, especially for dealing with fMRI big data. Functional network analysis has become an important and popular approach for discovering the underlying organization structures and meaningful dynamic patterns from the vast and noisy functional brain signals. Focusing on understanding the functional aggregation/co-activation among brain regions through quantitative and data-driven approaches, researchers have employed various types of matrix decomposition methods for the functional network analysis studies, including Independent Component Analysis (ICA) [88], Principal Component Analysis (PCA) [87] and Dictionary Learning [80]. Dictionary learning in particular has been shown to be a powerful tool in image compressed sensing [86], classification [83] and denoising [77], and has shown superior performance in decomposing the meaningful and comprehensive functional networks from various types of fMRI signals [52]. However, the computational cost for solving the sparse coding problem for dictionary learning has been known to be very demanding [81], especially when dealing with large-scale data sets. Furthermore, most of current dictionary learning methods for fMRI data analysis are only implemented for local application without any parallelization scheme. Facing with the rapidly growing fMRI data and the needs for population-level analysis with terabytes or even petabytes of

data size [87], the computation power limit of a single machine will eventually become the bottleneck for efficient and effective knowledge discovery from the fMRI big data.

Following the previous success in using dictionary learning for functional network decomposition [52], in this work we devolved a novel distributed rank-1 dictionary learning (D-r1DL) model, leveraging the power of distributed computing for handling large-scale fMRI big data. Compared to the gradient-based dictionary learning algorithms such as the online dictionary learning [54] (based on stochastic gradient descent) and the K-SVD [73] (based on gradient descent), the proposed rank-1 dictionary learning algorithm has a few critical advantages: 1) The learning process is a fix-point algorithm by alternating least squares updates, thus avoiding the tuning of the learning rate/step size while also avoiding the slow convergence near the solution; 2) The memory cost of the proposed algorithm is very low because it needs not to maintain the potentially large gradient matrix in the memory. The intermediate results will be discarded after each rank-1 basis is learned and stored, which further reduce the memory cost. More importantly, the rank-1 dictionary learning algorithm is very light-weighted regarding to the operational complexities: besides the input data, most of the routines in the algorithm will only take one vector as input and one vector as output. This feature helps the r1DL algorithm to be easily parallelized to its distributed version.

For the algorithm parallelization, in this work we used the Spark engine [50] to implement the D-r1DL algorithm. Spark is a high-performance distributed compute engine for large-scale data processing. It is similar to MapReduce, but has several distinct advantages that make it ideal for the deployment of large-scale analytics frameworks. First, its basic abstraction for distributed data, the resilient distributed dataset (RDD) [90], combines robust fault-tolerance with highly efficient data layout strategies. RDDs track their computation lineage as a directed acyclic graph; therefore,

if a segment is lost, it can be easily recomputed from the lineage. These lineages can be optimized on-the-fly to minimize the overhead of the prescribed computations. Second, all operations in Spark are performed in-memory, thus significantly improving throughput of data pipelines. This is a departure from Hadoop MapReduce, in which data are serialized to disk in between map and reduce steps. Third, the Spark compute engine is much more generalizable than MapReduce, and can efficiently support highly diverse workloads. While Spark supports the map and reduce primitives from Hadoop MapReduce, it also supports graph processing [79] and streaming [91] APIs on the same compute engine, in addition to numerous functional primitives beyond *map* and *reduce*. This structural flexibility is crucial to the efficient implementation of a wide variety of distributed algorithms.

An illustration for the operational and algorithmic pipeline consisting of three layers of model specification is shown in figure 3.1. The first and foremost deliverable of this work is to provide an integrated solution for the large-scale fMRI big data analysis. Therefore, we initially deployed the proposed D-r1DL model on our in-house server (termed "in-house solution") with an integrated neuroinformatics system [51]. The neuroinformatics system provides a web-based user interface for fMRI data uploading, hosting and result post-processing [51], as illustrated in figure 3.1(a). Alternatively, we also tested deploying the D-r1DL model on the cloud computing service provided by Amazon Web Services Elastic Compute Cloud (AWS-EC2), which has been widely applied for biomedical imaging researches due to its resource flexibility and ease of use. For the "AWS-EC2 solution", the data preprocessing was performed before running the D-r1DL model on it. Subroutines of the r1DL algorithm and its logic flow are illustrated in figure 3.1(b). The parallelization subroutines and its relationship with the r1DL algorithm are illustrated in figure 3.1(c).

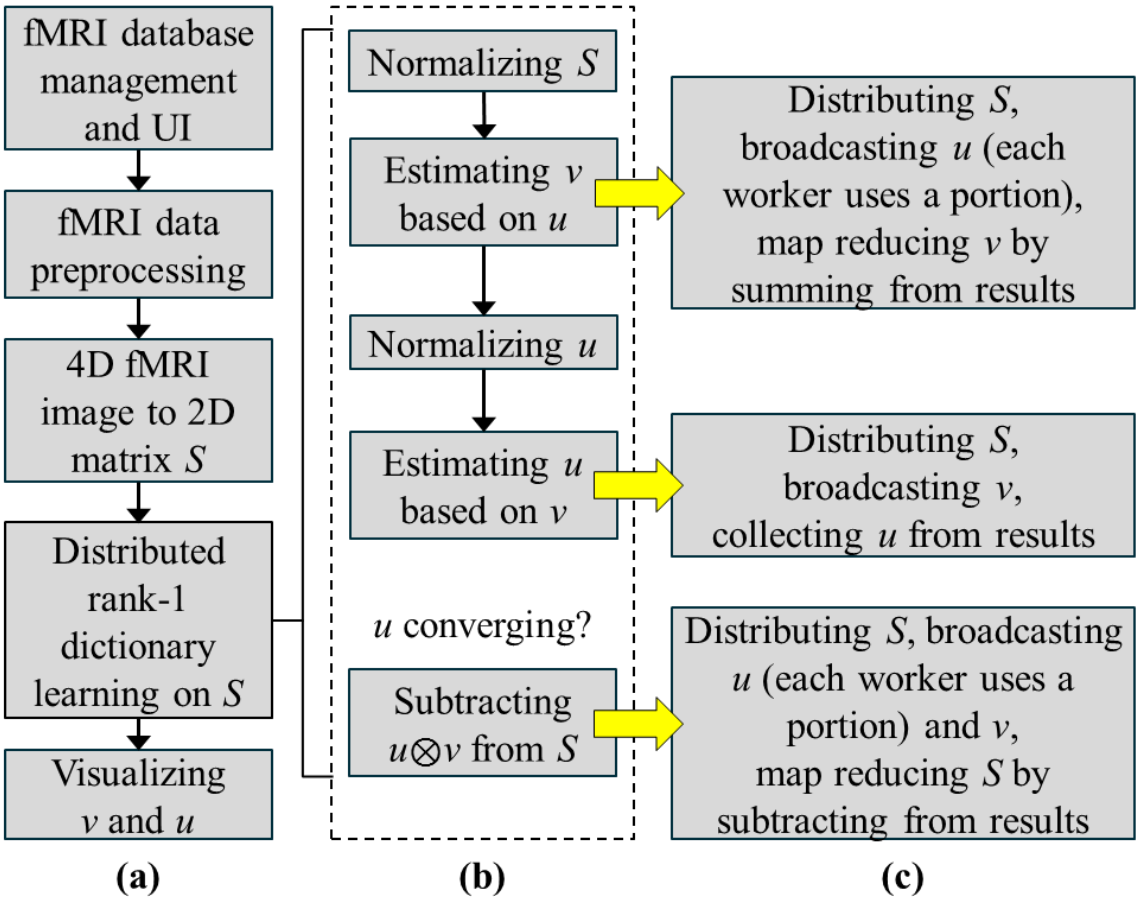

Figure 3.1. (a): Operations on the neuroinformatics system for preparing the application of D-r1DL and post-analysis. (b): Algorithm pipeline of the rank-1 dictionary learning. (c): Parallelization subroutines of the D-r1DL model derived from the corresponding subroutines of the r1DL using Spark. The distribution of input data S is based on RDDs.

3.2 Background and Related Work

Functional network analysis based on matrix decomposition methods has the basic premise: the observed functional signals are the result of the linear combination from the signals of many latent source (i.e. functional networks), plus noises and/or artifacts signals [74]. The methods then aim to identify the latent source signals as well as the loading matrix based on various learning priors including spatial/temporal independence [74], [88] (for ICA), or the sparsity in the

loading coefficients [80], [52] (for dictionary learning). The decomposition results consist of two parts: the signals of the latent sources (i.e. temporal pattern of the functional networks) that are regarded as basis activation patterns, and the loading matrix characterizing how each source contributes to the formation of each observed signal across voxels/ brain regions (i.e. spatial pattern of the functional networks). The spatial and temporal pattern of a sample network decomposed result is shown in figure 3.2.

Several sets of consistent and meaningful functional networks have been identified in the previous literature: including the 10 well-established resting-state networks (RSNs) [88] obtained using ICA, the task-related patterns obtained using PCA [89], and the HAFNI (holistic atlases of functional networks and interactions) atlases [52] featuring 32 group-wise consistent patterns during both task and resting-state using dictionary learning. Visualization of one functional network from the matrix decomposition results using the proposed r1DL method is shown below, to illustrate the temporal (the time series curve) and spatial (on cortical volumetric space) patterns of the brain network.

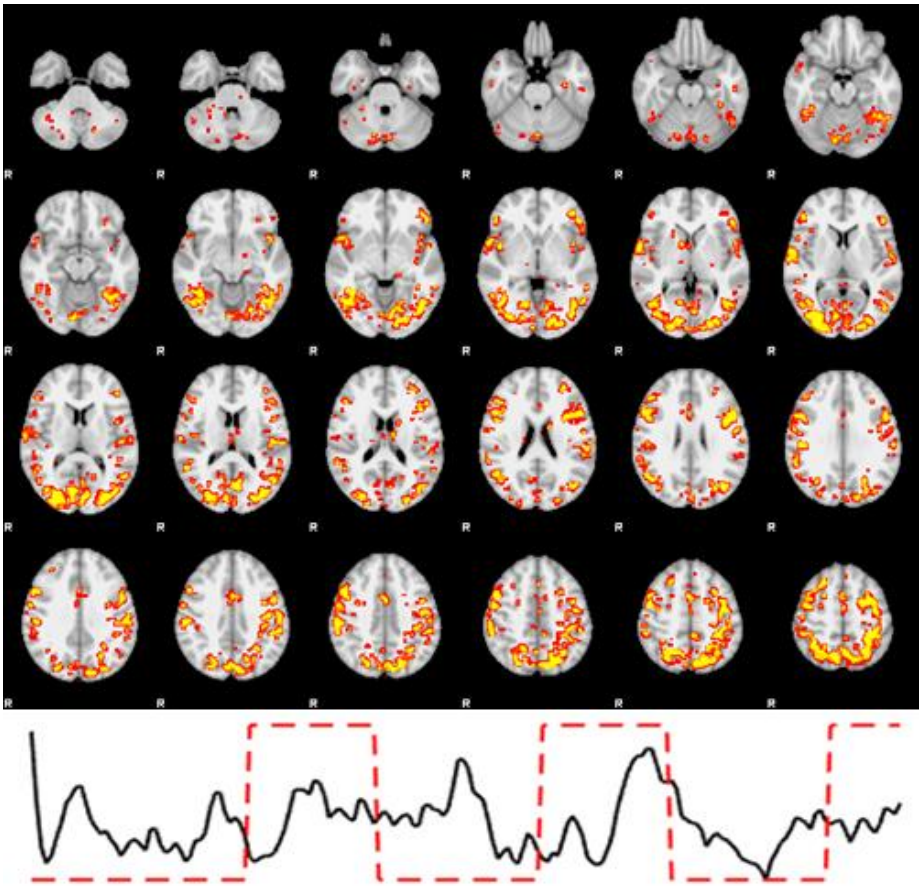

Figure 3.2. Top: spatial pattern visualized on cortical volumetric space of one decomposed network. Bottom: visualization of its temporal pattern.

While dictionary learning in general is an active area of research, there has been significantly less effort in scaling the algorithm. One of the few methods proposed was [72], in which the authors designed a sparse coding framework on Hadoop MapReduce. The method was parallelized by splitting the core operations in two main phases: the sparse coding phase, in which the loading weights were learned in parallel; and the dictionary learning phase, in which the dictionary atoms were updated. By taking advantage of hard sparsity constraints, the authors avoided materializing the entire data matrix in memory at once, instead operating on blocks of the matrix in parallel and constructing the loading matrix row by row. In this work we undertook the similar dataflow

optimization techniques: the sparse coding and dictionary atoms are assumed to fit easily in memory for fast and efficient computation. However, using the Spark framework instead of Hadoop MapReduce provides us intrinsic speedups. Where Hadoop MapReduce excels in batch processing, Spark is optimized for iterative computation: intermediate results are cached in-memory on the workers and re-used in subsequent iterations, and the updates are efficiently broadcasted to the workers. Most importantly, the RDDs abstraction pipelines the requested operations and lazily executes them after determining the optimal computational path using the least amount of resources. We leverage these advantages to provide a substantial performance gain in our D-r1DL dictionary learning implementation.

3.3 Rank-1 Dictionary Learning For fMRI Data Analysis

The rank-1 dictionary learning algorithm aims to iteratively estimate multiple rank-1 basis vector u (T×1 vector with unit length) and its loading coefficient vector v (P×1 vector) to decompose the input signal matrix S of dimension T×P, by minimizing the following energy function $L(u, v)$:

$$L(u, v) = \|S - uv^T\|_F, \text{s.t.} \|u\| = 1, \|v\|_0 \leq r. \quad (1)$$

Eq. 1 indicates that the product of u and v is supposed to well-fit the input S while the total number of non-zero elements in v should be smaller than or equal to the given sparsity constraint parameter r. The minimization problem in Eq. 1 can be solved by alternatively updating u (randomly initialized before the first iteration) and v until convergence:

$$v = \underset{v}{\operatorname{argmin}} \|S - uv^T\|_F, s.t. \|v\|_0 \leq r,$$

$$u = \underset{u}{\operatorname{argmin}} \|S - uv^T\|_F = \frac{Sv}{\|Sv\|}, \qquad (2)$$

Converging at step j if: $\left\|u^{j+1} - u^j\right\| < \varepsilon, \varepsilon = 0.01.$

Eq. 2 involves multiplication between input matrix *S* and vector *u*, followed by setting all elements in the resulting vector smaller than its *r*-th largest value to zero, essentially performing the vector partition operation. One rank-1 basis [*u*, *v*] can be estimated in each step; afterwards the input matrix *S* will be deflated to its residual *R*:

$$R^n = R^{n-1} - v^T R^{n-1}, R^0 = S, 1 < n \leq K, \qquad (3)$$

where *K* is the total number of expected basis (i.e. dictionary atoms) to be discovered from the input data. It could be seen that the formulation of the proposed rank-1 dictionary learning algorithm is similar to the sparse PCA problem [75]. However, the goal of PCA and sparse PCA is to derive a low-dimensional basis (i.e. learning a smaller set of high representative basis); in contrast, the goal in dictionary learning is to learn an over-complete dictionary set [81]. Regarding the algorithm convergence, it is easy to show that the value of the energy function as in Eq. 1 decreases at each iteration (until convergence), thus the objective is guaranteed to converge. The convergence of the learning in Eq. 2 was also empirically tested in this work. The deflation operation in Eq. 3 is based on Hotelling's deflation method for estimating the eigenvectors, where each step of deflation leads to the corresponding eigenvalue replaced by zero. The validity of using Hotelling's deflation for sparse PCA was provided in [75], while better deflation methods were also discussed in [84]. figure 3.3 shows a running example illustrating the data preparation and networks decomposed by r1DL.

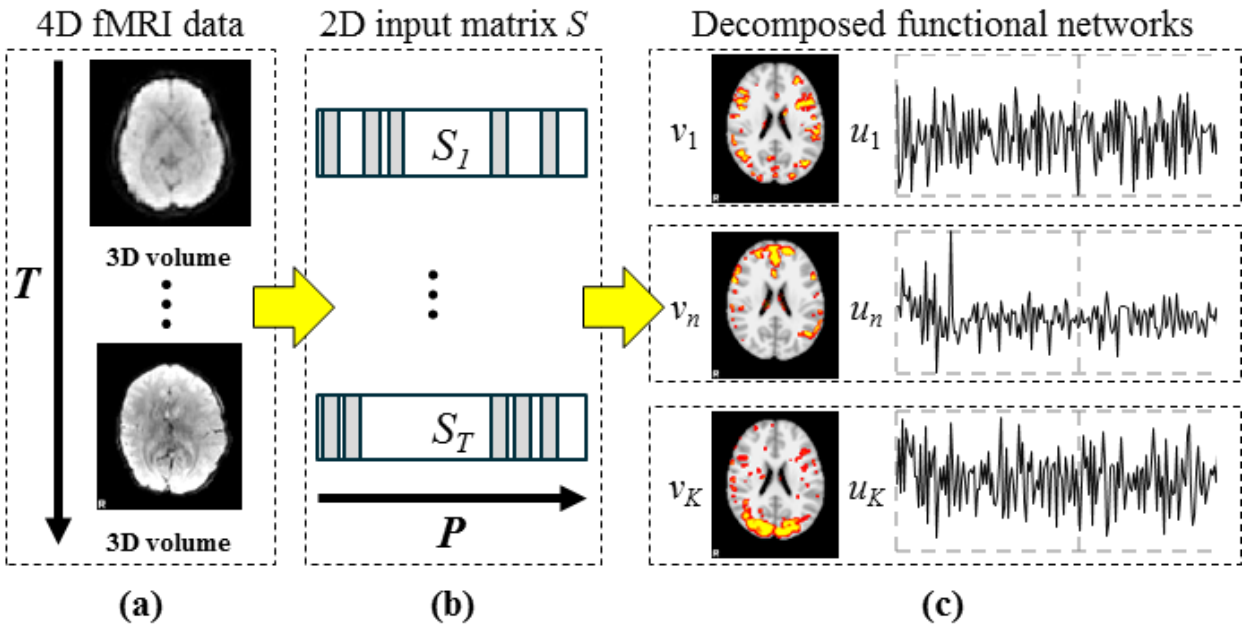

Figure 3.3. Illustration of the r1DL model applied on fMRI data as a running example. (a) 4D fMRI data represented as a series of 3D volume images. (b) Converting 4D data into 2D input matrix *S*. (c) Spatial ($v_1 \ldots v_K$) and temporal ($u_1 \ldots u_K$) patterns of the decomposed functional networks from *S* using r1DL dictionary learning.

3.4 Algorithm Parallelization and Deployment on Spark

In this work, the rank-1 dictionary learning algorithm introduced above was implemented and parallelized on the Spark engine. Specifically, the vector-matrix multiplication and the matrix-vector multiplication steps in Eq. 2 were implemented by their corresponding distributed primitives in Spark. Reading and partitioning the input data *S* is supported by the RDD abstraction; therefore, the distribution of *S* to each node as a series of key-value pairs is inherently straight forward: data formation of the current work is based on row-vectors. In other words, each column in *S* contains the *T* number of observations for one specific feature, to the total of *P* features. While *S* was maintained as an RDD, the vectors *u* and *v* were broadcast to all nodes. Thus, during the

vector-matrix multiplication, each node will use its portion of the updated *u* vector, and then estimate the *v* vector based on the multiplication of portions of *S* and *u*. The resulting *v* vectors from all the nodes will be then map-reduced by the summation operation. The matrix-vector multiplication is relatively easier, where each node will use all the updated *v* vector then estimate its corresponding portion of the *u* vector. The resulting *u* vector is just the collection of the results from each node. In addition, the matrix deflation operation in Eq. 3 was also parallelized by broadcasting both the *u* and *v* vectors learned from Eq. 2, and then estimating the outer produce between portion of *u* vector and the whole *v* vector at each node. The *S* matrix is then subtracted by the results of each node through mapping over each row and deflating it in parallel. This implementation of the r1DL algorithm after parallelization is termed as the "Distributed rank-1 dictionary learning" (D-r1DL) model.

3.4.1 Complexity of the Distributed Primitives

The computational complexity of the original (un-parallelized) rank-1 dictionary learning algorithm is quite obvious: all the major subroutines including matrix-vector multiplication, vector-matrix multiplications and deflation have complexity of *T***P*, essentially traversing through input matrix *S*. However, the distributed primitives added for the parallelization in D-r1DL will potentially cause large extra computations and/or data transfers across nodes. The problem will be magnified when such transfer occurs over a network (e.g. worker nodes are distributed across Internet). Thus, we analyzed the extra complexity induced by the parallelization of the three subroutines, assuming that there are *M* number of nodes. It should be noted that the estimations are upper bounds for the complexity; the empirical performance will be largely dependent on how

Spark optimizes the transformation lineage for the RDD, and how the RDD is distributed across the cluster.

- For the vector-matrix multiplication for estimating v, the total complexity is ($T*M + P*M+T\log(T) + P$): $T*M$ caused by the broadcasting, $P*M+T\log(T)$ caused by the map reduce and network shuffle, and P caused by the updating of v.
- For the matrix-vector multiplication, the total complexity is ($P*M + T$): $P*M$ caused by the broadcasting, and T caused by the updating of u.
- For the matrix deflation, the total complexity is ($P*M+T*M$): both u and v will be broadcasted to all M nodes.

3.4.2 Deployment and Configuration

The D-r1DL model was deployed on two different sets of server clusters, leading to two solutions for the data analysis. The first set is based on the in-house server, called the "in-house solution". Spark version 1.5.2 pre-built for Hadoop 1.0 and python version 2.7.11 with all the required dependencies were installed on the in-house server. We setup one standalone spark cluster on the server with a master node consisting of 16 cores and 16GB RAM. The in-house solution also featured an integrated neuroinformatics system named HELPNI (HAFNI-enabled largescale platform for neuroimaging informatics) as introduced in [51]. Authorized users of the system can upload, manage, and perform the preparations of the fMRI data for the model analysis. For running the D-r1DL model, the preparation steps include fMRI signal preprocessing (gradient distortion correction, motion correction, bias field reduction, and high pass filtering) [78], converting the 4D fMRI images to 2D data matrix, as well as the generation of shell scripts according to user

specifications. When the computations by D-r1DL are finished, the reports of the results (consisting of the statistics and visualizations of the decomposed functional networks) will be automatically generated by the HELPNI neuroinformatics system [51] and uploaded to the server, accessible via web interface for viewing and sharing. The in-house solution is mainly used for testing the single-server (on the local threads), standalone mode performance of the D-r1DL algorithm.

The second set was based on the computational resources provided by AWS-EC2 service. For the AWS-EC2 implementations, we used the provided EC2 deployment scripts with the Spark distribution. These created a Spark cluster on EC2 with one master node and 16 workers. Each worker consisted of 2 cores and 7.5GB memory. EC2 clusters are highly scalable, as the number of workers recruited could be adjusted within the cluster. The preprocessed and converted fMRI data was stored at the cloud through Amazon S3 and accessible by the EC2 cluster. The nodes featured Hadoop distributed file system (HDFS), which ensured data consistency and improved I/O speed. Hadoop version 1.0, Spark version 1.5.2, and Python version 2.7.11 with the Anaconda scientific programming stack (e.g. NumPy, SciPy) were installed on EC2 cluster. An illustrative diagram showing the organization and execution architecture of the two solutions are shown below.

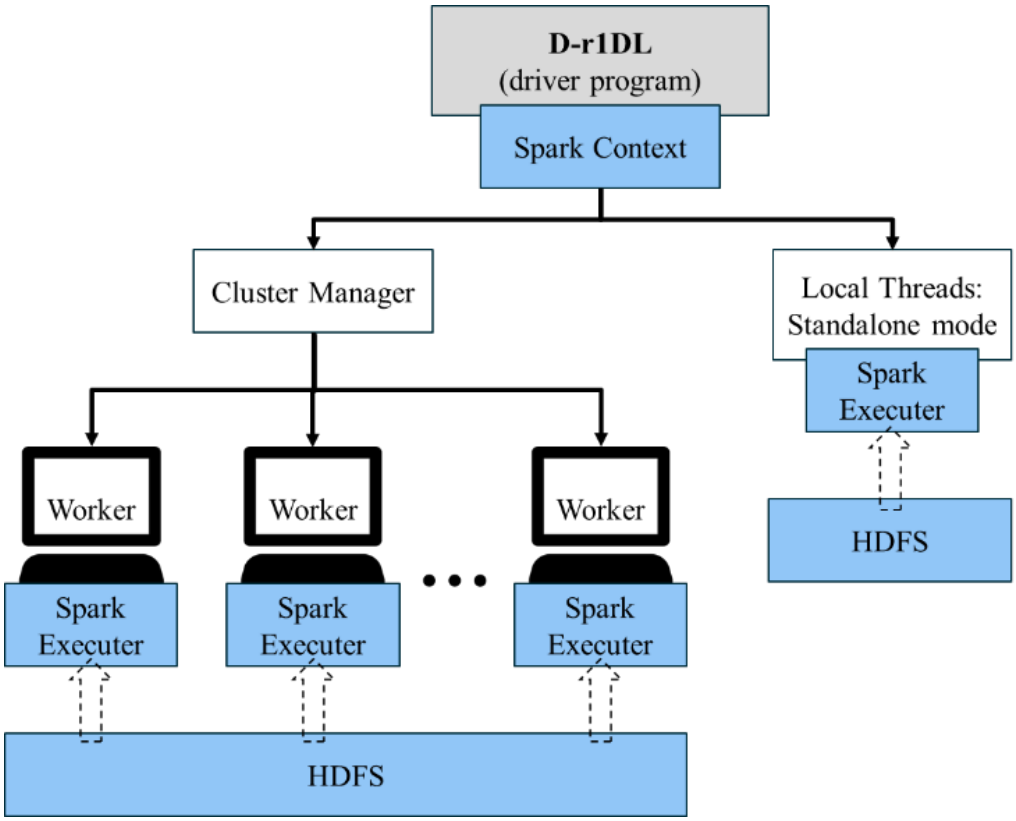

Figure 3.4. Illustrative diagram showing the organization and execution architectures for the standalone local mode and the multi-worker cluster mode.

3.5 Experimental Application on Brain Functional Imaging Data

3.5.1 Validation of the D- r1DL Model

To validate the effectiveness of the proposed D-r1DL model in terms of its capability of decomposing fMRI data into meaningful functional networks, we applied the framework on the fMRI data acquired during multiple tasks from the Human Connectome Project (HCP) Q1 release dataset [7]. The HCP dataset is advantageous in its high temporal and spatial resolution (TR=0.72s, varied temporal length from 176 to 1200 volumes; 2mm isotropic voxels, to the total of over 200,000 voxels), which enables more detailed characterization of the brain's functional behavior. Additionally, the HCP dataset includes acquisitions of fMRI data during 7 tasks and resting-state

from 68 subjects, to the total of over 500 individual data, which matches the aim of the proposed framework for population-level fMRI big data analysis.

The learned collection of functional networks represented by rank-1 dictionary basis was then compared with the HAFNI atlases [52]. The HAFNI atlas was obtained by applying online dictionary learning method [54] on all the individual fMRI data in the same HCP Q1 dataset. Subsequently, 32 group-wise consistent networks were identified through manual inspection over more than 200,000 decomposed networks [52]. Thus our aim in this validation was to identify the presence (or absence) of those atlas networks from the results of D-r1DL. The main rationale of performing comparison with network templates and atlas network for the model validation in this work was due to the lack of ground truth in fMRI data. At this stage, the capability and accuracy of the proposed model could only be validated by comparing with the previously-reported and well-established results from the same dataset.

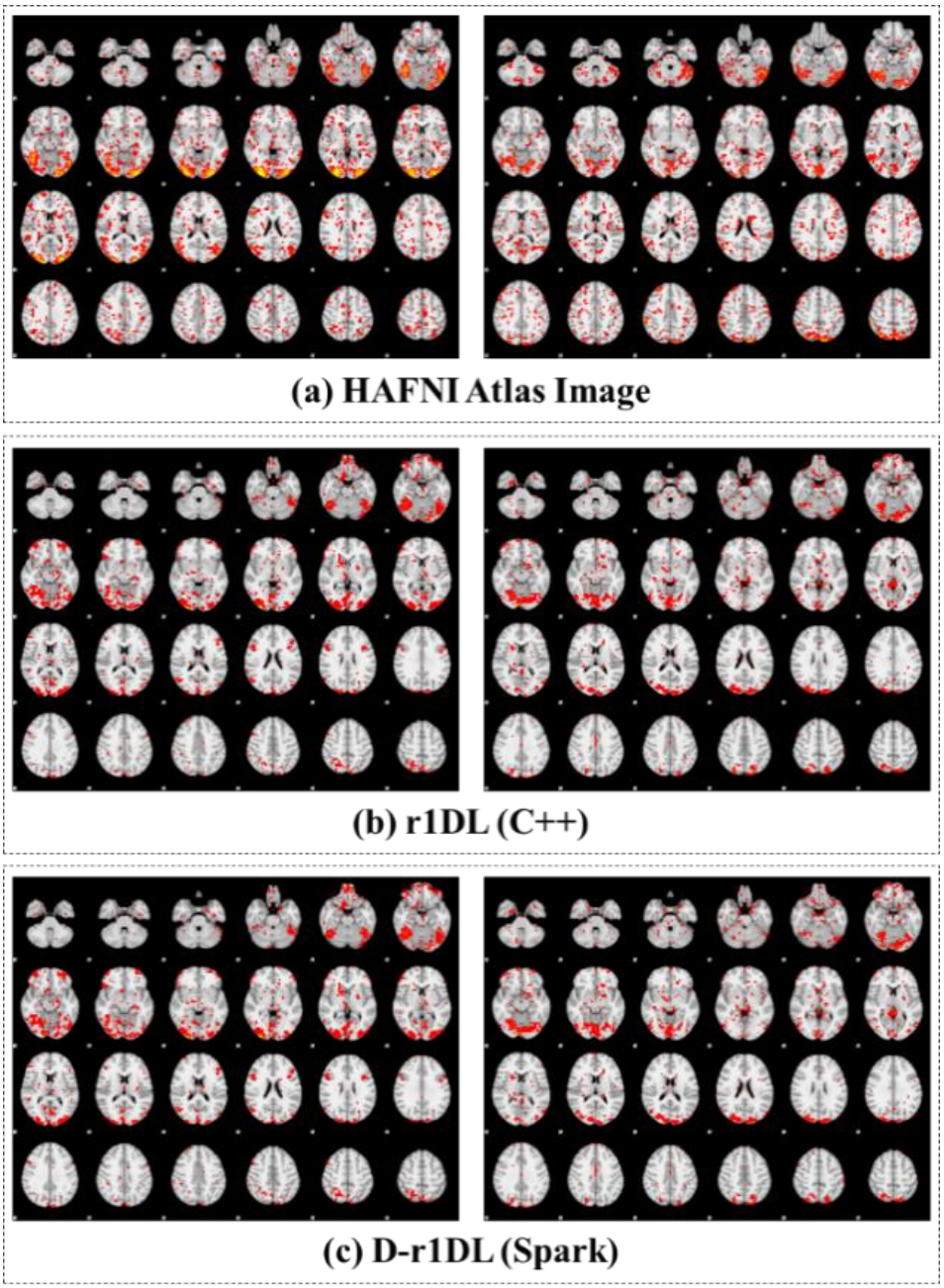

Figure 3.5. Spatial maps obtained from applying the dictionary learning method on the same fMRI dataset implemented by HAFNI, r1DL in C++, and D-r1DL in Spark.

In addition, the D-r1DL model was compared with the rank-1 dictionary learning algorithm implemented in C++ without any parallelization, in order to investigate whether the parallelization would affect the model performance. Both of the implementations were deployed on the same in-

house server and applied on the HCP Q1 individual resting-state fMRI data as well as two task fMRI datasets ("Emotion" and "Working Memory") using the same parameter setting (r=0.07, K=80). An illustration for the spatial maps of two sample networks decomposed by r1DL implemented in C++ and D-r1DL, as well as the corresponding individual-level atlas network in HAFNI, is shown in the three panels in figure 3.5 (a)-(c), respectively.

The obtained functional networks from D-r1DL were then matched to the individual-level atlas networks on the same subjects of the same tasks (or resting-state) by maximizing the spatial similarity between them:

$$R(P_1, P_2) = |P_1 \cap P_2|/|P_2|, \qquad (4)$$

where P_1 and P_2 are the spatial map vectors of the two networks. In this work P_1 is the network decomposed by D-r1DL and P_2 is the atlas network. Operator $|\bullet|$ counts the total number of voxels with non-zero values in the given spatial pattern. R ranges from 0 (no voxels overlapping) to 1 (exactly the same maps). The spatial similarity results show that all of the atlas networks defined in HAFNI could be found from the results of D-r1DL. Specifically, the 3 atlas networks from Emotion dataset were identified from D-r1DL results with average spatial similarity of 0.82. The 6 atlas networks from WM dataset were identified with average spatial similarity of 0.79. The 10 resting-state networks, originally reported in [76] and later included in the HAFNI atlas were identified from the results of applying D-r1DL on resting state fMRI data with average spatial similarity of 0.70. The spatial similarity results also show that the networks decomposed by r1DL implemented in C++ and D-r1DL implemented in Spark are almost identical, with average spatial similarity of 0.99.

3.5.2 Experiment on Algorithm Convergence

As discussed in section 3.3 for the algorithm description, we have performed extensive experiments for testing the convergence of alternating least squares with *l*-0 constraint problem in Eq. 2. By running the r1DL algorithm on each individual fMRI dataset in the HCP Q1 release across 7 tasks with dictionary size parameter K=400 (i.e. decomposing 400 functional networks from each fMRI data), we found that the learning of u and v converged for all the 190,400 networks decomposed. The average number of alternative updates needed for learning one network for different datasets of 5 randomly selected sample subjects is listed in Table 3.1 below. It can be seen that regardless of the input data size, the majority of the learning would be finished within only a few iterations.

Table 3.1. Average number of iterations needed for convergence across 7 tasks of 5 subjects

	sbj1	**sbj2**	**sbj3**	**sbj4**	**sbj5**
Emotion	6.1	6.1	6.2	6.1	6.2
Gambling	6.3	6.3	6.5	6.3	6.2
Language	6.4	6.3	6.4	6.3	6.3
Motor	6.4	6.2	6.3	6.5	6.3
Relational	6.2	6.2	6.2	6.2	6.2
Social	6.3	6.3	6.2	6.3	6.3
WM	6.4	6.3	6.4	6.4	6.4

3.5.3 Performance Boost by r1DL Comparing with Other Dictionary Learning Algorithms

One of a major premise of the proposed r1DL algorithm is that because of its smaller memory cost and robust learning mechanism (no need to set learning rate), the algorithm should have similar or faster running speed, compared with other dictionary learning methods, even without the parallelization. Based on the performance statistics from running r1DL over the whole HCP task fMRI (tfMRI) dataset as introduced above, we compare the r1DL with the other two

dictionary learning algorithms: online dictionary learning implemented in SPAMS package [54] and the stochastic coordinate coding (SCC) algorithm introduced in [82], by applying these two methods on the same HCP Q1 dataset using the same in-house server. The performance comparison is shown in figure 3.6 (averaged across all 68 subjects). From the comparison, it can be seen that r1DL has exhibited improved computational speed over the other two methods in all the 7 tfMRI datasets. It should be noted that we used the r1DL implemented in C++ for the testing in this experiment, and in the same way the other two methods were implemented to ensure consistency in the comparison.

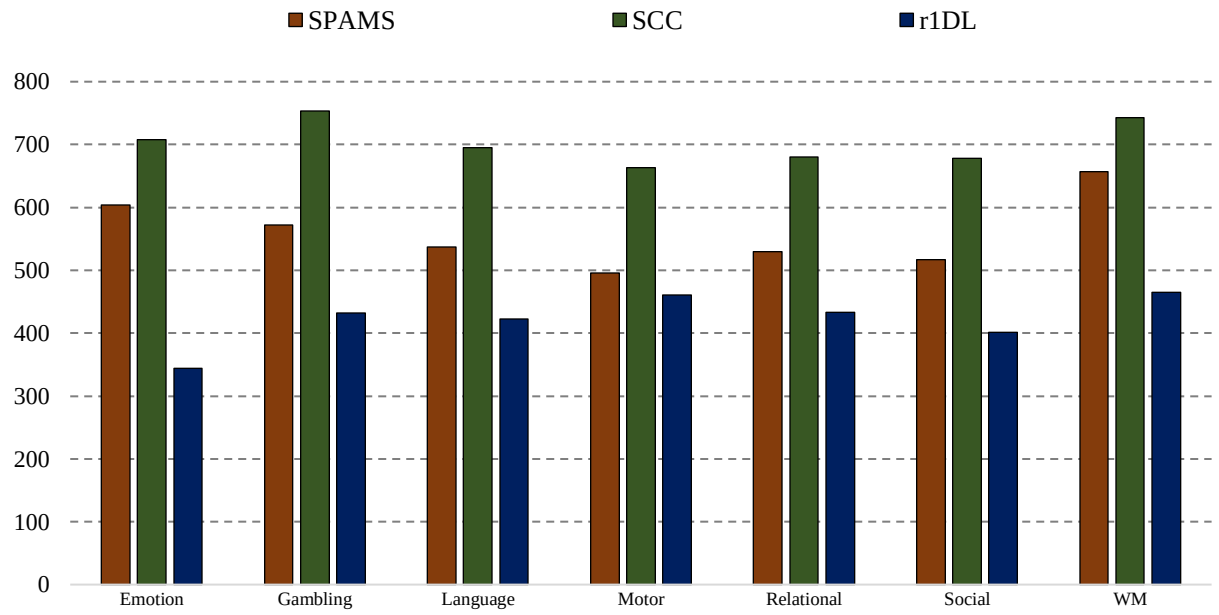

Figure 3.6. Average time cost (measured in seconds) for functional network decomposition from individual tfMRI data during 7 tasks across 68 subjects, using the three dictionary learning methods.

Furthermore, to benchmark the D-r1DL efficiency on the running time, we designed an experiment using two popular parallel processing platforms of Spark and Flink. We set up a virtual cluster of three nodes, each with four virtual CPUs, 8192 MB RAM, and 30 GB disk storage. As we examined both platforms using varying of input matrixes, the preliminary testing shows that Flink Dr1DL could offer performance gains over Spark Dr1DL for large data. Figure 3.11 illustrates the performance gain of Flink as the input data growth.

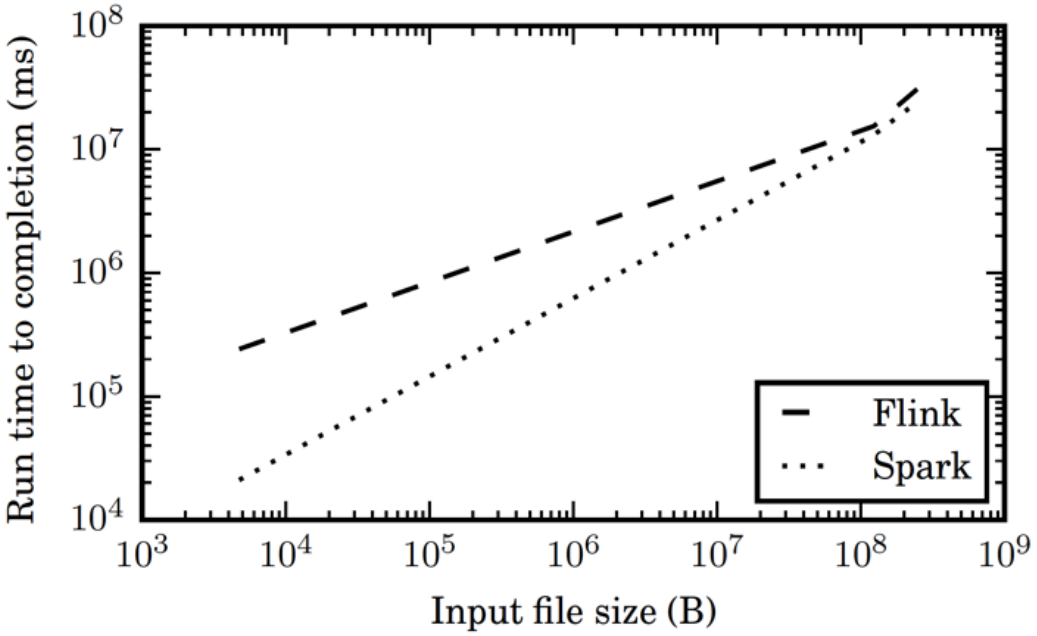

Figure 3.11. Run time comparison of D-r1DL using Flink and Spark with varying input data sizes.

3.5.4 Performance and Scalability Analysis for D-r1DL Using the In-House Solution

As introduced in 3.5.1, in this work we applied the D-r1DL model on the three types of datasets for functional network decompositions: tfMRI data of Emotion task with dimension of 176*2M, tfMRI data of Working Memory (WM) task with dimension of 405*2M, and resting state fMRI (rsfMRI) data with dimension of 1200*2M. The testing input files sizes of the three types of dataset were 300MB, 700MB and 2GB, respectively. Using the in-house solution as specified in

4.2, we firstly analyzed the performance of the D-r1DL model using different numbers of cores on a single machine. The performance statistics measured in time cost are shown in figure 3.7.

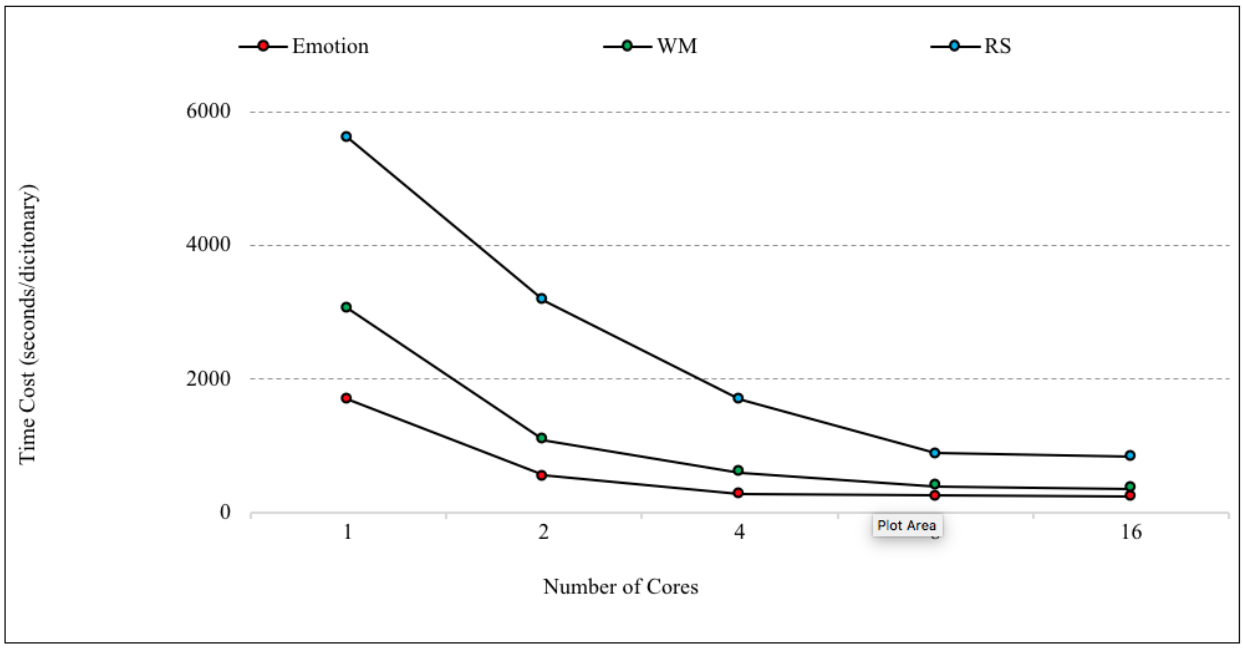

Figure 3.7. Time cost for decomposing one functional network from three different fMRI datasets by recruiting varying number of cores, using the in-house solution.

For all the three datasets, there exists clear logarithmic relationship (R^2=0.84, 0.89 and 0.92) between the number of cores recruited and the total time cost for the decomposition. The speed boosts by recruiting more cores for the computation comparing with the baseline (1-core) configuration for the three datasets using the in-house solution are listed in Table 3.2, showing the ratio between the time cost using 1 core and the time cost using multiple cores.

Table 3.2. Ratios of time cost changes by recruiting various number of workers comparing with standalone mode

	Emotion	**WM**	**RS**
2 workers	3.1	2.8	1.8
4 workers	6.0	5.1	3.3
8 workers	6.6	7.7	6.3
16 workers	6.8	8.6	6.7

As the configuration for using only one core for D-r1DL is equivalent to the non-parallel algorithm, the performance statists indicate that the parallelization based on Spark could greatly improve the performance of the rank-1 dictionary learning algorithm. Specifically, the better performance gain on larger dataset indicates that the parallelization of the rank-1 dictionary learning could potentially overcome the computational bottleneck for analyzing big neuroimaging data, potentially enabling high-throughput analysis on a locally-deployed high-performance computation cluster in the future.

Another analysis of the performance of D-r1DL on the in-house server was aiming at investigating the relationship between dictionary size K (i.e. number of functional networks to be decomposed) and the time/memory cost. The rationale is that, as discussed in 3.3.1, the rank-1 dictionary learning algorithm has advantages in the iterative estimation of the basis vectors [u, v]. Thus the program does not need to maintain the learning results in the memory. As shown in figure 3.8, the average time cost for estimating one dictionary only marginally increased when using larger dictionary sizes K. Further, the total memory cost of running the D-r1DL was independent with K. This feature is especially useful when the spatial dimension of the input data is large, either because of the higher spatial resolution or because of the aggregated dataset from multiple subjects in a

population. As in such cases, the size of all decomposed networks, which equals to *P**K*, could be very large.

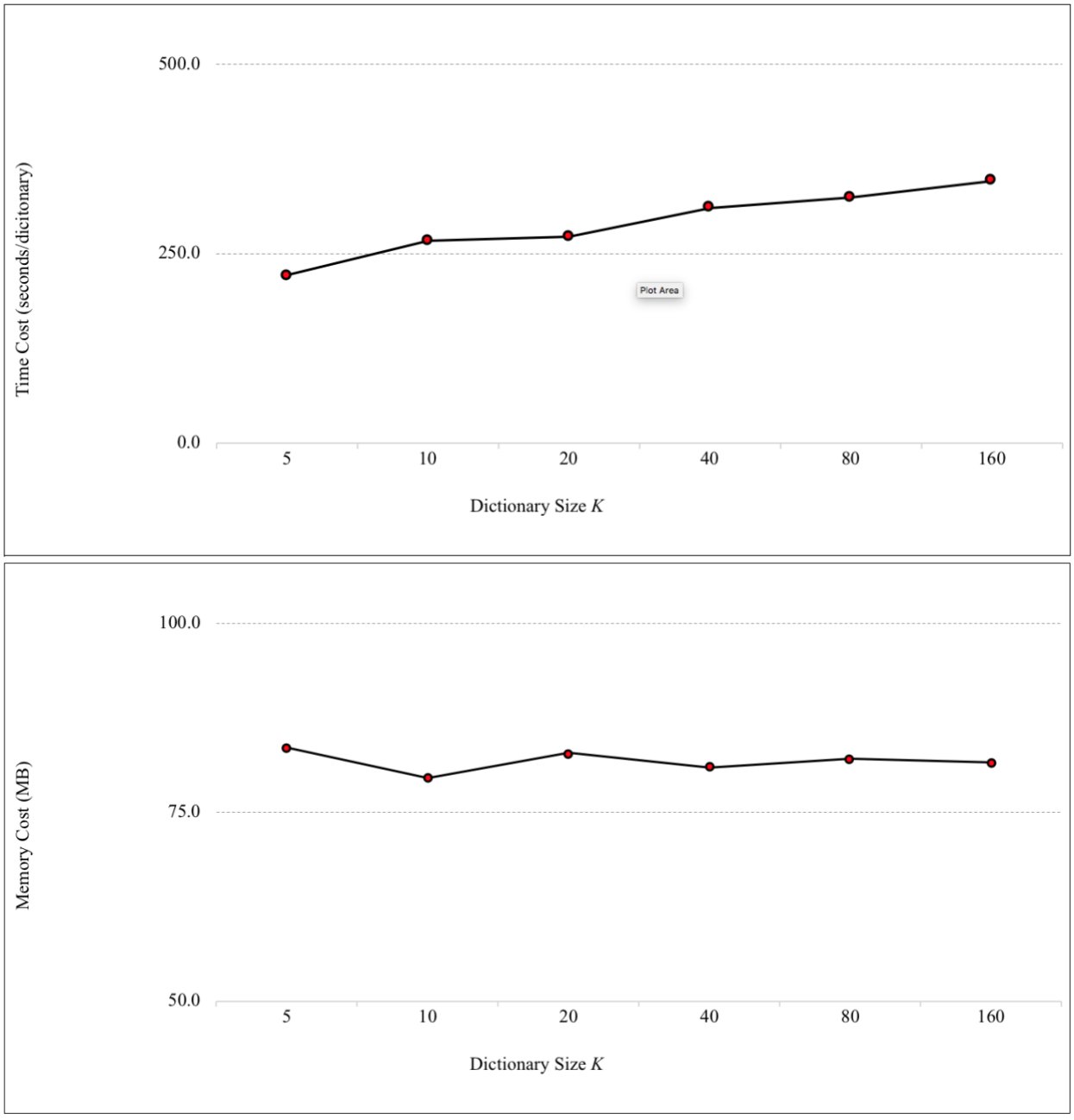

Figure 3.8. Time and memory costs for decomposing Emotion tfMRI datasets with varying dictionary size *K*, recruiting 16 cores, using the in-house solution.

3.5.5 Performance Of D- r1DL in Multi-Worker Mode Using AWS-EC2 Solution

In addition to the experiments of the single-machine multi-core configurations conducted using the in-house solution, we have also applied the D-r1DL on the same datasets using the cloud computing service provided by AWS-EC2 as introduced in 3.4.2. We aimed to investigate the performance of D-r1DL when applied over multiple machines through a network interface. Specifically, as the Spark Python architecture and the resilient distributed dataset abstracts have been designed for supporting large-scale, high efficient analytic framework, we are interested to test its capability of utilizing the distributed computational resources provided by AWS-EC2. In this work, we tested the performance in terms of time and memory cost of the D-r1DL model using 1, 2, 4, 8 and 16 workers on three datasets, while each worker has two cores for the computation. The D-r1DL would be running in stand-alone mode under single-worker configuration, similar to the configuration used in the in-house solution. As discussed in 3.4.1, the communications through network interfaces caused by the parallelization of computation (e.g. the broadcasting of u and v) will potentially increase the time cost mainly due to latencies. Thus, the single-worker configuration serves as the baseline for testing whether recruiting more workers will be beneficial from the performance perspective. The performance results of the time and memory cost are summarized in figure 3.9, with the baseline results from single-worker configuration highlighted.

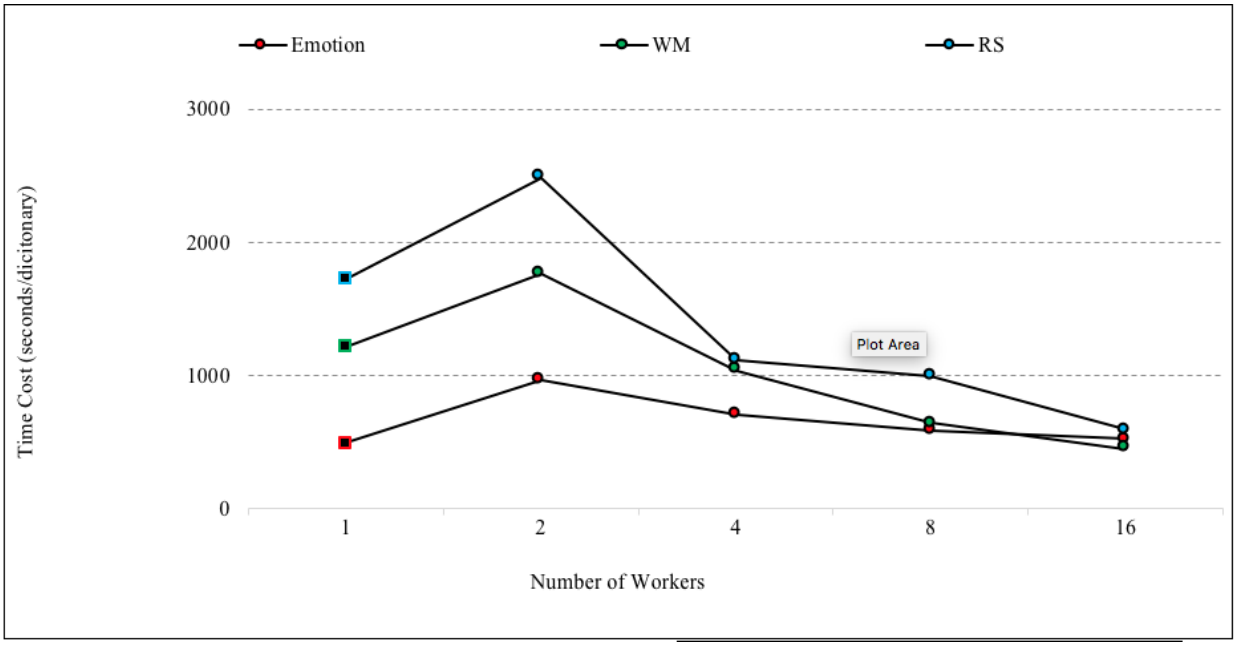

Figure 3.9. Time cost for decomposing one functional network from three different fMRI datasets by recruiting varying numbers of workers, using the AWS-EC2 solution. The results from single-worker (standalone mode) configuration are highlighted as blocked markers.

First of all, from the results it can be seen that the AWS-EC2 solution recorded faster computation speed (10%~80% faster) comparing with the in-house solution, especially on larger dataset, when both of them use two cores. Considering the fact that the hardware configuration of AWS-EC2 features larger memory capacity better optimized for computation purposes, such difference in performance is within our expectation.

On the other hand, it is interesting to observe that for AWS-EC2 solution, there exists the break-even point at which the multiple-worker mode outperformed the stand-alone mode, but only for the two larger datasets. The ratio between the time cost using standalone mode and the time cost using multiple workers are for the three datasets summarized in Table 3.3. It can be observed that, for the 700MB WM and the 2GB RS dataset, using 4 or more workers could lead to faster speed

comparing with the standalone mode using 1 worker. While for the smaller 300MB Emotion data, the standalone mode is the fastest among all experiments. Thus, it can be concluded that the multi-worker configuration will be more suitable for analyzing larger datasets, while standalone mode or the simpler in-house server solution might be preferred for datasets with typically smaller sizes.

Table 3.3. Ratios of time cost changes by recruiting various number of workers comparing with standalone mode

	Emotion	**WM**	**RS**
2 workers	3.1	2.8	1.8
4 workers	6.0	5.1	3.3
8 workers	6.6	7.7	6.3
16 workers	6.8	8.6	6.7

The memory cost on each worker as summarized in figure 3.10 indicates that the multi-worker mode under AWS-EC2 solution scales good with the increasing input file size, as it maintains reasonable small (~100MB) memory cost for all configurations including the single-worker standalone mode. That is the major advantage of using Spark Python model and its resilient distributed dataset for the parallelization: one or multiple workers need not to load the whole dataset at once, but only its corresponding portion of the data according to the data partitioning strategy implemented in the RDDs abstract.

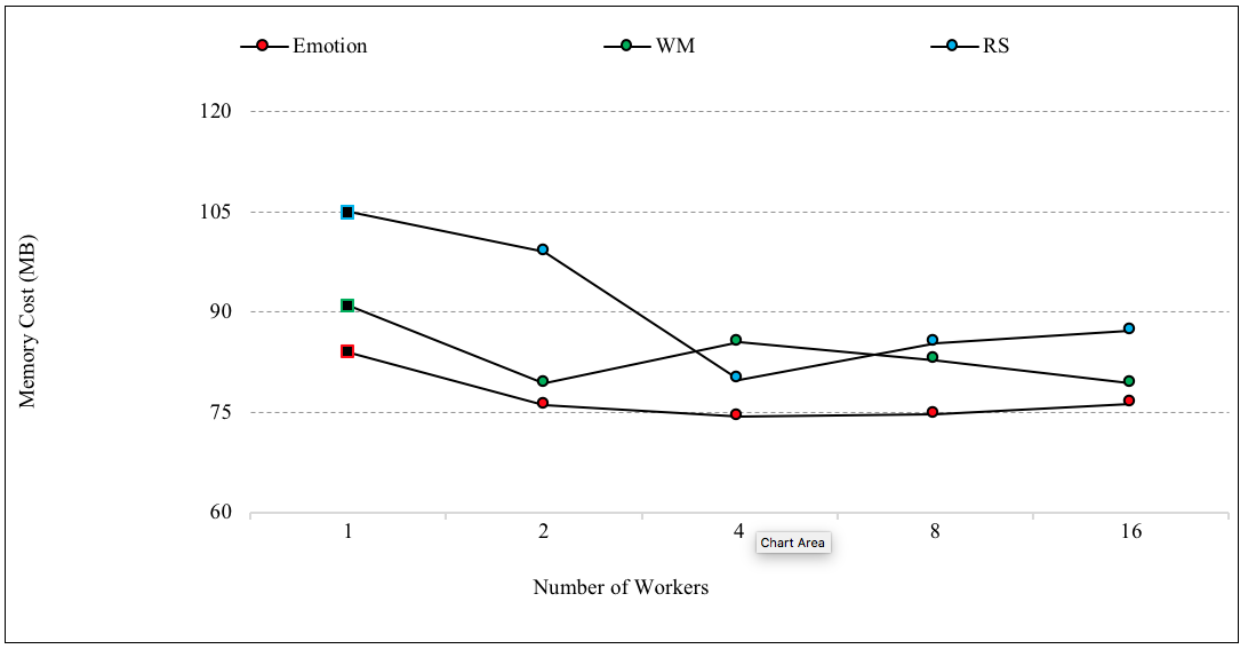

Figure 3.10. Memory cost for decomposing three different fMRI datasets by recruiting varying number of workers, using the AWS-EC2 solution. The results from single-worker (standalone mode) configuration are highlighted as blocked markers.

3.6 Conclusions

In this paper, we proposed a novel and effective distributed dictionary learning model based on iterative rank-1 basis estimation. The model was implemented and parallelized in Spark, and then deployed using the in-house solution as well as the AWS-EC2 solution. The aim of this work is to meet the challenges posed by fMRI big data for more efficient and scalable data analytics methods. The testing results from running both solutions on the HCP Q1 dataset show that functional network decomposition using rank-1 dictionary learning could benefit from parallelization for both single-worker multi-core configuration and the multi-worker cluster configuration, with significant performance improvement especially on larger datasets. In the current work, we have only analyzed individual-level fMRI data, with the largest data size of 2GB.

As it has been shown from the performance statistics that the Spark engine supported by RDDs abstract could effectively perform the data partition and reduce the memory cost for large-scale input data, we will test the model performance on larger, population-level datasets (e.g., the HCP full dataset) with the size of dozens or hundreds of terabytes in the near future. The ultimate goal of the proposed D-r1DL model with the HELPNI neuroinformatics system is to provide an integrated solution for functional neuroimaging big data management and analysis, enabling high-throughput neuroscientific knowledge discovery. In addition, the similar parallelization scheme used in this work for D-r1DL could be implemented on other algorithms as well. Thus, the experience of this work also offers a practical perspective for improving the efficiency and scalability of general machine learning and data mining algorithm developments.

CHAPTER 4

A CLOUD-BASED DISTRIBUTED DEEP LEARNING PLATFORM FOR fMRI BIGDATA ANALYTICS [1,2]

[1] Makkie, Milad., Huang, Heng., Zhao, Yu., Vasilakos, A.V. and Liu, Tianming., 2018. Fast and Scalable Distributed Deep Convolutional Autoencoder for fMRI Big Data Analytics, Submitted to SIGKDD, International Conference on Knowledge Discovery and Data Mining.

[2] Makkie, Milad., Huang, Heng., Zhao, Yu., Vasilakos, A.V. and Liu, Tianming. A cloud-based distributed deep learning platform for fmri bigdata analytics. To be submitted to IEEE Transactions on Cloud Computing.

ABSTRACT

In recent years, analyzing task-based fMRI (tfMRI) data has become an essential tool for understanding brain function and networks. However, due to the sheer size of tfMRI data, its intrinsic complex structure, and lack of ground truth of underlying neural activities, modeling tfMRI data is hard and challenging. Previously proposed data modelling methods including Independent Component Analysis (ICA) and Sparse Dictionary Learning only provided shallow models based on blind source separation under the strong assumption that original fMRI signals could be linearly decomposed into time series components with corresponding spatial maps. Given the Convolutional Neural Network (CNN) successes in learning hierarchical abstractions from low-level data such as tfMRI time series, in this work we propose a novel scalable distributed deep CNN autoencoder model and apply it for fMRI big data analysis. This model aims to both learn the complex hierarchical structures of the tfMRI big data and to leverage the processing power of multiple GPUs in a distributed fashion. To deploy such a model, we have created an enhanced processing pipeline on the top of Apache Spark and Tensorflow, leveraging from a large cluster of GPU nodes over cloud. Experimental results from applying the model on the Human Connectome Project (HCP) data show that the proposed model is efficient and scalable toward tfMRI big data modeling and analytics, thus enabling data-driven extraction of hierarchical neuroscientific information from massive fMRI big data.

4.1 Introduction

The sheer complexity of the brain has forced the neuroscience community and particularly the neuroimaging experts to transit from the smaller brain datasets to much larger hard-to-handle ones. The cutting-edge technologies in the biomedical imaging field, as well as the new techniques in digitizing, all lead to collect further information from the structural organization and functional neuron activities in the brain through rich imaging modalities like fMRI [6]. Projects such as Human Connectome Project (HCP) [7], 1000 Functional Connectomes [11] and OpenfMRI [12] are the perfect examples of such large neuroimaging datasets. The primary goal of these efforts is to gain a better understanding of the human brain and to diagnose the neurological and psychiatric disorders. Among various neuroimaging methods, task-based functional magnetic resonance imaging, tfMRI, has been widely used to assess functional activity patterns and cognitive behavior of human brain [17], [18], [19], [20]. However, the main challenges are to obtain meaningful patterns from the intrinsic complex structure of tfMRI and also lack of clear insight into the underlying neural activities. Given the hierarchical structure of functional networks in human brain, the previously data-driven methods such as Independent component analysis (ICA) [85] and sparse coding for Dictionary Learning [52] as well as model-driven approaches such as General Linear Model (GLM) [92] have been demonstrated to disregard some of the information contained in the rich tfMRI data [93], [94]. Thus, these shallow machine learning models are not capable of fully understanding the deep hierarchical structures of functional networks in human brain [93], [94]. Consequently, there is an urgent call for more efficient and scalable data analytics and knowledge discovery methods to crack the underlying brain activities.

Recently, new data-driven computational intensive neural network approaches such as deep learning have gained increasing interest among researchers, due to their efficiency of extracting

meaningful hierarchical features from the low-level raw data. Particularly, Convolutional Neural Network (CNN) is among the top deep learning methods in the scientific community [95], [96], [97], [98], [99], [100], [101], especially in classifying and learning image data [102].

In the context of high dimensional data such as fMRI, however, the large size of training examples (dozens of millions of time series each with hundreds of time points) and the sheer size of model parameters can drastically impact the computational cost and accuracy of learning the fMRI signals. Furthermore, most of the current neural network methods for fMRI analysis are only implemented for local application without any parallelization scheme [107], [108], [109], [110], [112]. As indicated by an extensive battery of literature [103], [104], [105], [106], many of scaling deep learning applications by using large-scale clusters of GPUs can solve the computational bottleneck for efficient and effective knowledge discovery from fMRI big data.

Following the previous successes in using distributed GPU processing for scaling neural network model, in this work we aim to design a fast and scalable distributed framework and to implement a deep convolutional model, dubbed distributed Deep Convolutional Autoencoder (dist-DCA) to leverage the power of distributed optimization, distributed data partitioning, and multiple GPU processing. The distributed optimizer is based on an asynchronized Stochastic Gradient Descent (SGD) method [103]. In this model, we have used multiple replicas of a single model to optimize parameters, which lead to reducing the training time significantly. For data parallelization, we utilized Apache Spark [90] and Hadoop Distributed File System (HDFS). Considering the computationally intensive operations in tuning the parameter, Spark acts as a fast extract transfer load layer to optimize the data partitioning for the underlying Hadoop ecosystem. This is being accomplished via constructing the Resilient Distributed Dataset, RDD, which provides a functional

interface to partitioned data across the cluster. Our major contributions of this work can be summarized as follows.

1) We implement a distributed deep learning framework using TensorFlow on Spark to take advantage of the power of distributed GPUs cluster.

2) We propose a distributed deep convolutional autoencoder model to gain meaningful neuroscience insight from the massive amount of tfMRI big data.

3) We validate our proposed dist-DCA model using a novel high-level sparse dictionary learning method.

Compared to the existing distributed deep learning frameworks such as dist-keras [117], elephas [123] and dl4j [124], our proposed framework has a few critical advantages: 1) The migration from a standalone code to a distributed version can be done with only a few lines of change. 2) Despite of the previous framework, our framework works efficiently with HDFS, allowing Spark to push datasets. 3) Integrating the model with the current pipeline is easy as Spark is in charge of parallelizing the data. 4) The framework is easy to deploy and scale over the cloud or in-house clusters. We have created an Amazon Machine Image (AMI), which in combination with a spark-ec2 script, can easily scale up the cluster.

The rest of this paper describes our dist-DCA model and architecture in detail. In section 4.2, we briefly introduce the primary components in which dist-DCA is implemented. We also review related works in this domain. We then thoroughly describe our deep convolutional model in section 4.3. Section 4.4 is dedicated to data parallelism and distributed optimization. Section 4.5 describes our scalable experiments in large GPU clusters where we explain how our model can be easily distributed among dozens of GPU nodes to reduce computational time efficiently.

4. 2 Preliminary and Related Works

Recent advances in building affordable high-performance GPUs with thousands of cores were one of the critical factors in advancing large-scale deep learning models. This breakthrough has also encouraged the scientific community to utilize GPUs more often, as CPU's capacity does not seem to grow in proportion to the rate of increasing demand. However, the limited memory capacity of typical GPUs on the market (usually 8 gigabytes) has become a bottleneck in feeding extensive datasets as far as the training speed is concerned. Therefore, two common approaches of data parallelism and model parallelism are of the researchers' interest.

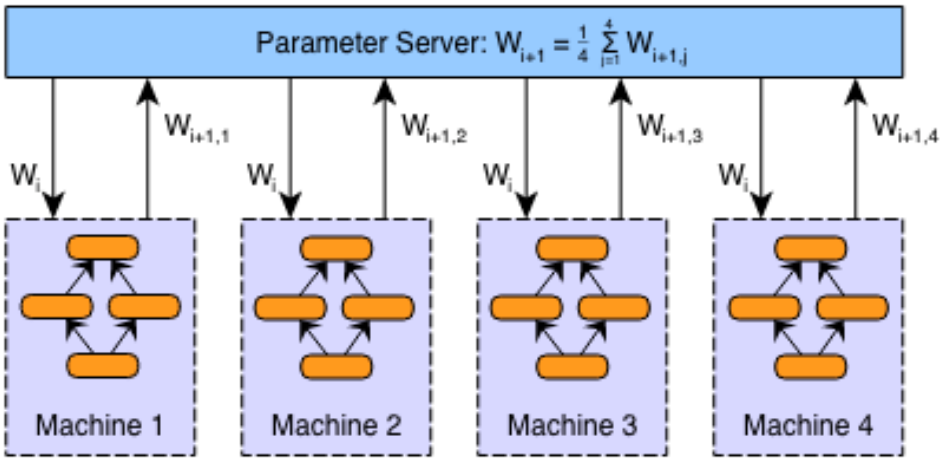

Figure 4.1. An asynchronous data parallelism model using Asynchronous SGD

In model parallelism, different portions of a model computation are done on different computational devices simultaneously for the same batch of examples while sharing the parameters among devices as a single model. This approach, however, is efficient for very large models as splitting a neural network model needs to be done in a case-by-case manner and is very time-consuming. Data parallelism, on the other hand, seems more straightforward for general implementation and can be easily scaled to larger cluster sizes. Figure 4.1 demonstrates a data

parallelism paradigm. We will discuss our dist-DCA data parallelism scheme in more depths in section 4.4.

Our main motivation behind this work is to implement a scalable, asynchronous data parallelism model leveraging TensorFlow on Spark [113] to efficiently learn meaningful hierarchical abstraction of massive size of fMRI data.

4.2.1 TensorFlow

TensorFlow [122] is a mathematical software and an open-source software library for Machine Intelligence, developed since 2011, by Google Brain Team and initially aimed to machine learning research and deep neural networks. TensorFlow is a numerical computation library using data flow graphs that enables machine learning experts to do more data-intensive computing, e.g., it contains some robust implementations of conventional deep learning algorithms. It offers a very flexible architecture that enables deploying computation to one or more CPUs or GPUs in a standalone, parallel or distributed fashion. We selected TensorFlow in our work as it efficiently supports distributed and parallel GPU processing and it supports Keras. However, having an easy to scale framework is required for running TensorFlow applications when the model and data become large. So, a queuing framework to both seamlessly feed data into the cluster nodes and to schedule and manage tasks efficiently is needed. Pipelining pre-processing, training and inferences steps is a known challenge yet to be addressed by the TensorFlow ecosystem.

4.2.2 Spark

Since 2009, the Spark framework [90] was developed at the University of Berkeley AMPlab and currently is being maintained by Databricks. This framework addresses deficiencies

of MapReduce by introducing resilient distributed datasets (RDD) abstract where the operations are performed in the memory. Spark compiles the action lineages of operations into efficient tasks, which are executed on the Spark engine. Spark offers a functional programming API to manipulate Resilient Distributed Datasets (RDDs). RDDs represent a collection of items distributed across many computing nodes that can be manipulated in parallel. Spark Core is a computational engine responsible for scheduling, distributing and monitoring applications. It consists of many computational tasks across executor node(s) on a computation node/cluster. Spark's scheduler will execute the duties across the whole cluster. Spark minimizes the repetition of data loading by caching data in memory, which is crucial in complex processes. Spark uses Hadoop filesystem as a core distributed file system (HDFS). Apache Spark is one of the most active Apache projects on GitHub.

In this work, we used a combination of TensorFlow and Spark [122] to leverage the data parallelism and scheduling of Spark, thus enabling direct tensor communication among TensorFlow executors and parameter server(s). Process-to-process direct communication enables TensorFlow program to scale effortlessly. In section 4.4, we will describe such communication in more details.

4.2.3 Previous Works

In the past few years, there have been multiple studies in adopting neural network methods to model fMRI data and its associated applications. For instance, Chen et al. [107] used convolutional autoencoder in fMRI data aggregation; Plis et al. [108] used deep belief network (DBN) to learn physiologically important representations from fMRI data; Suk et al. [109] combined the Deep Auto-Encoder with Hidden Markov Model to investigate the functional

connectivity in resting-state fMRI; Huang et al. [110] used the restricted Boltzmann machine to mine the latent sources in task fMRI data; Ren et al. [112] used convolutional neural networks to classify fMRI-derived functional brain networks, and Wen et al. [33] have used AlexNet to reconstruct the visual and semantic experiences using fMRI data. In the context of applying deep learning applications to fMRI data, however, most works have focused on the classification problem by using a single computation node. Our focus in this paper is the provision of an unsupervised distributed CNN encoder that effectively models the tfMRI big data. This enables us to learn hierarchical feature abstraction while lowering the spatial and temporal noises contained in fMRI data and ensures us to efficiently reduce model training and inferences time by easily scaling cluster of GPUs.

4.3 Deep Convolutional Autoencoder

Figure 4.2 illustrates both the structure of our proposed dist-DCA model and a validation pipeline based on the online dictionary learning (ODL) algorithm. We will describe this pipeline later in section 4.5. A neuroinformatics platform [51] is used to preprocess the tfMRI signals. Then Keras and Tensorflow APIs are used to construct the DCA model. In section 4.4, we will explain how asynchronous gradient computation reduces the model's training time by communicating and updating parameters values.

A non-distributed version of DCA model is elaborated in [120]. To facilitate the understanding of the model, we recapitulate the model in the following paragraphs. The purpose of autoencoder in DCA is to first encode the input fMRI time series by mapping them into higher level feature maps and then to decode the signals by reversing the process. Throughout this process, we obtain a hierarchical abstraction of fMRI signals while denoising them. As mentioned below, we assume

that the model consists of only one convolutional layer both at the encoder and decoder and later we extend it to the real model. A summary of the key model parameters is shown in Table 4.1.

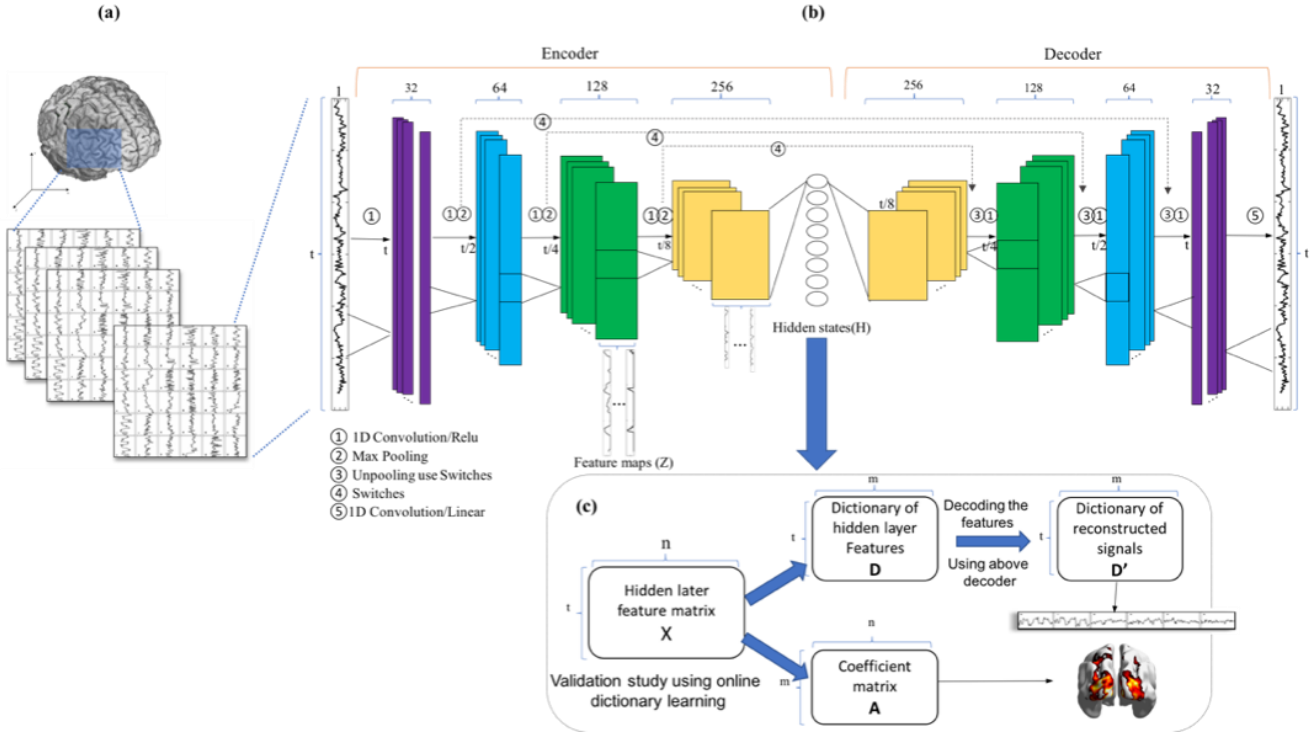

Figure 4.2. An illustration of the dist-DCA model and the online dictionary learning validation study. (a) demonstrates the preprocessing step of the tfMRI data including signal extraction and normalization. (b) demonstrates the structure of the dist-DCA model and its components including all hidden layers and feature maps. (c) shows the validation study through which we obtain the brain activity pattern.

Table 4.1 dist-DCA model summary

Feature map/filter	Layer1	Layer2	Layer3	Layer4
Encoder	32/21	64/9	128/9	256/9
Decoder	128/9	64/9	32/9	1/21
Total Parameters	6,023,549			

4.3.1 Encoder

The Encoder takes one 1D tfMRI signal x as shown in figure 4.2.b and then by convolving the filters throughout the entire signal generates the feature map in the next layer using the equation 1.

$$z_i = f(p_i * x + b_i) \qquad (1)$$

where x is the signal input, and p_i and b_i are the corresponding filter and bias for the i-th feature map. f is the activation function. In this paper, except for the output convolutional layer in the decoder layer where we use linear activation function, we use the Rectified Linear Unit (ReLU) as activation functions. The advantages of choosing ReLU in our study is first to reduce the possibility of vanishing gradient and second to represent the signal sparsely as we later use the sparse representation of the hidden layer for data validation. A fully connected layer is used at the end of the encoder to match the encoder final hidden layer feature size with the input signal and to ensure that the hidden states are learned with a full receptive field of input as we use it as the final desired output of the model as mentioned in [120].

$$H = Z \times W + C \qquad (2)$$

In the equation 2, the hidden layer states are represented by H, whereas Z, W and C are the feature maps, weight and bias of the fully connected layers, respectively.

4.3.2 Decoder

The decoder is following a symmetric pattern and attached to the previous encoder. To reconstruct the input signal, first, the hidden states are mapped and reshaped to a reconstructed version of feature maps Z' via fully connected layer in the decoder. In equation 3, W' and C' denote the weights and bias of the fully connected layer in the decoder, respectively.

$$Z' = H \times W' + C' \qquad (3)$$

In the end, input signal will be reconstructed by linearly combining these feature maps, where $\hat{x}$ denotes the reconstructed signal, and p_i' and b_i' are the filters and biases in the decoder as shown in equation 4.

$$\hat{x} = \sum_i p_i' * z_i' + b_i' \quad (4)$$

The same concept is extended to a model with more layers (4 layers in encoder and 4 in the decoder) by transforming the input layer into different feature map in each convolutional layer by a chain rule. To minimize the mean square error between fMRI signals and their reconstructions, we also used an L2 regularization term between feature maps in the top layer of the encoder and the bottom layer of the decoder. Doing so ensures us that the fully connected layer does not randomly shuffle the timing order when reconstructing features maps in the decoder. λ in equation 5 controls the significance of the L2 regularization term and we experimentally set it to 0.006.

$$\min \frac{1}{2}||X - \hat{X}||_2^2 + \frac{1}{2}\lambda||Z - Z'||_2^2 \qquad (5)$$

4.3.3 Max-Pooling and Unpooling

The max pooling is applied on each layer after the convolutional layer. This helps first by substantially reducing the computational cost for the upper layer and second, by gaining translation-invariance. The translation-invariance is particularly important in tfMRI due to possible time-shift phenomena while acquiring the raw signal [115], [113].

Given the invertible property of max-pooling, we utilized switches [116] in the encoder to memorize the location of the local max in each pooling regions and then we applied the location of the corresponding local max value to its original position. In validation studies (section 4.5) when "switches" are not available, we simply use traditional up-sampling.

In the next section, we explain how such a model is replicated among spark executor nodes.

4.4 Data Parallelism and Model Deployment

In data parallel approaches, a copy of the entire model is sent to each executor, parallelizing processing of gradient descent by partitioning data into smaller subsets. A parameter server then combines the results of each subset and synchronize the model parameters between each executor after receiving gradient delta from each executor. This can be done synchronously or asynchronously. However, in homogeneous environments where nodes share the same hardware specifications and communicate via a reliable network of communication, asynchronous [103] methods outperform for two reasons [103], [106]. First, executors do not wait for others to commit before start processing the next batch of data. Second, asynchronous method is more robust to failure of nodes as if one node fails the others will still train their own data partitions and fetch new updates from parameter server.

For example, given a batch size of 100 elements, 5 replicas of the model compute the gradient for 20 elements, and then combine the gradients in a separate node, known as parameter server, and apply parameters updates synchronously, in order to behave exactly as if we were running the sequential SGD algorithm with a batch size of 100 elements.

We have implemented downpour SGD [103] in our distributed framework and have fixed the ηfetch and ηpush of weights and gradients to one, for speeding up convergence and for ease of comparison to simple SGD. Our experiment shows that relaxing consistency requirements are remarkably effective. Downpour SGD comes from the intuition that if we view the gradient descent as a water droplet toward minimizing the error rate, then individual executors can be considered as several droplets near each other, all separately flowing down into the same valley.

Moreover, we practiced a warm-up phase, wherein a single executor node starts training on its own data partition before starting other executors. This has significantly decreased the probability of diverging of each executor being trapped in its own local optima.

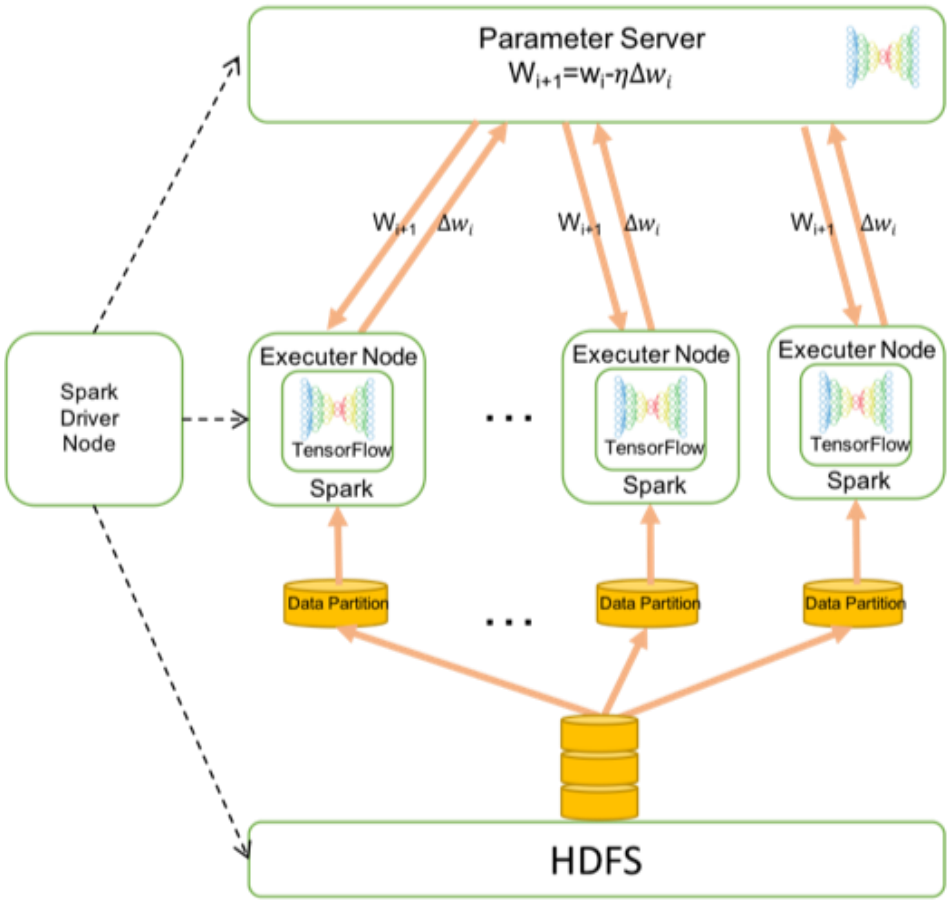

Figure 4.3. Dist-DCA. Executor nodes asynchronously fetch parameters w and push gradients to the parameter server. Spark driver is also in charge of data penalization and task scheduling.

We also chose the Adagrad optimizer [118] to keep the learning rate update for each parameter as the model is training and to ease extending the number of executing nodes. Adagrad uses a separate

adaptive learning rate for each parameter. Let ηi,K be the learning rate of the i-th parameter at iteration K and Δwi,K its gradient, then in equation 6 we obtain ηi,K.

$$\eta i,K = \gamma / \sqrt{\sum_{j=1}^{K} (\Delta wi, j)^2} \qquad (6)$$

Because these learning rates are computed only from the summed squared gradients of each parameter, Adagrad is easily implemented locally within the parameter server. γ is the constant scaling factor for all learning rates, which is larger than the best fixed learning rate used without Adagrad. The use of Adagrad extends the maximum number of model replicas that can productively work simultaneously.

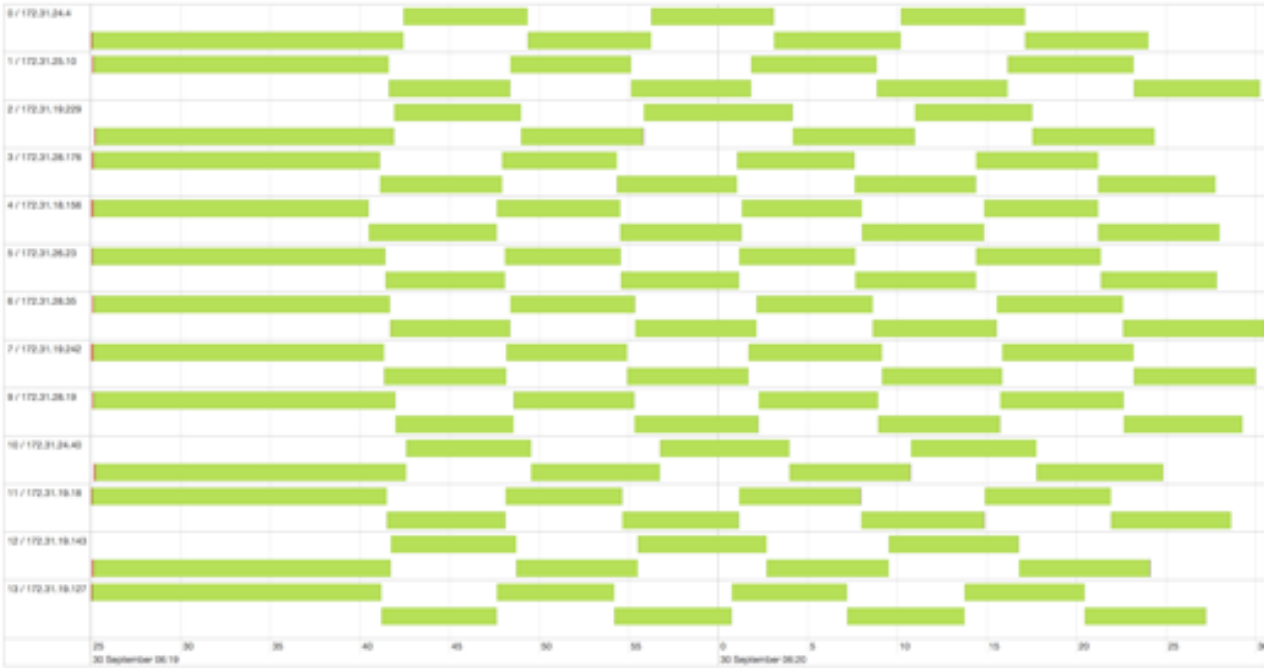

Figure 4.4. Dist-DCA data partitions. Spark driver keeps track of data partitions and executors' computational times. Here you can see the computational time of all active nodes and how the training tasks are scheduled.

The abovementioned optimization procedures ideally address our problem in two ways. One is by empowering us to process massive fMRI data (nearly 10 million tfMRI time series each with 284 time points as in this work). And, the other is by allowing us to train our relatively large model, consisting of more than 6 million trainable parameters, faster. As a result, our proposed dist-DCA benefits from asynchronous data parallelism through two main components of distributed data partitioning and distributed parameter optimization as it is shown in the figure 4.3. We use Hadoop as our main distributed file system and Spark for tasks scheduling and data partitioning. Each Spark executor acts as a wrapper of TensorFlow application where one node handles the parameter synchronization and the rest run the TensorFlow application independently just as one single node setup. Each executor commits its gradient delta to parameter server after each processing batch elements and receives the latest parameter from the server. Meanwhile, Spark core efficiently feeds each of the executors through HDFS by partitioning the data based on the number of epochs and dataset size. The figure 4.4. shows Spark data partitioning among a cluster of 16 nodes consisting of one driver, two parameter servers, and 13 executors. Spark driver is responsible for handling tasks and for replicating TensorFlow model across a cluster. For each stage and each partition, tasks are created and sent to the executors. If the stage ends with a shuffle, the tasks created will be shuffle-map tasks. After all tasks of a particular stage are completed, the driver creates tasks for the next stage and sends them to the executors, and so on. This repeats until the last stage, where the results return to the driver. With the asynchronized implementation, we ensure that both the model replicas and data partitions are run independently, thus reducing the delays induced by the loaded executors.

4.5 Experiments

We evaluated the TensorFlow on Spark performance and scalability by our novel dist-DCA model using Amazon Elastic Cloud Computing (EC2). We trained this model on 9,658,464 fMRI time series of 48 human subjects and evaluated on 4,024,360 time series of 20 separate subjects.

4.5.1 Experiments Setup

4.5.1.1 Dataset

We use the Human Connectome Project (HCP) Q1 release dataset [7] containing 68 healthy subjects' tfMRI data. The HCP dataset is advantageous in its high temporal and spatial resolution (TR=0.72s, varied temporal length from 176 to 1200 volumes; 2mm isotropic voxels, to the total of over 201,218 voxels' signals per subject each with the length of 284 time points), which enables more detailed characterization of the brain's functional behaviour. We use motor task fMRI data in this study, composed of six most basic motor tasks including visual cues (event 1), tapping left (right) fingers (event 2, 3), squeezing left toes (event 4, 5) and moving tongue (event 6). We divided the Motor task Q1 subjects into two separate subsets of 48 training and 20 validating subjects. For running the dist-DCA model, the preparation steps include fMRI signal pre-processing (gradient distortion correction, motion correction, bias field reduction, and high pass filtering) [114], all implemented using FSL FEAT. Furthermore, we recruited our integrated neuroinformatic platform, HELPNI [51], to facilitate the pre-processing and to integrate different steps of data acquisition using its powerful pipelining ability.

4.5.1.2 Cloud Platform

The dist-DCA model is deployed on Amazon Web Service Elastic Cloud Computing, AWS EC2. EC2 clusters are highly scalable, as the number of executor nodes could be adjusted effortlessly within the cluster. The pre-processed and converted fMRI data was stored in the cloud through Amazon S3 and accessible by the EC2 clusters. This enables us to pull data to newly initialized instances easily. We used customized scripts along with an AMI containing a preconfigured instance to scale our cluster according to desire. Each cluster's node contains Apache Spark version 2.2.0, Hadoop version 2.6.0, TensorFlow 1.3, Keras 2.08 and python 2.7. To benchmark the scalability and robustness of our proposed framework, we used a variety of node hardware configurations with a different number of node per experiment as summarized in Table 4.2. The configuration of nodes are as follows. G3 nodes are equipped with High-Frequency Intel Xeon E5-2686 v4 (Broadwell) processors, NVIDIA Tesla M60 GPU, with 2048 parallel processing cores and 8 gigabytes of video memory per GPU with 25 Gbps of aggregate network bandwidth within the cluster. G2 nodes come with Intel Sandy Bridge processors, NVIDIA Kg20 Grid GPU with 1536 CUDA cores and 4 gigabytes of memory per GPU.

Table 4.2. Cloud clusters' configuration, each line represents a separate experiment setup.

No of Spark/TF Executors	vCPU Cores/node	Used GPU Memory per node (GB)	Memory per node (GB)	EC2 Node Type
1	16	8	122	G3-4x
2	16	8	122	G3-4x
4	4	12	61	P2-x
4	8	4	15	G2-2x
4	16	8	122	G3-4x
4	32	4	60	G2-8x
8	16	8	122	G3-4x
13	16	8	122	G3-4x

4.5.2 Performance

We aimed to investigate the performance of our framework with respect to the mean processing time of a single mini-batch (1 fMRI signal) for Downpour SGD with Adagard training as a function of the number of nodes used in a single model instance. To do so, we deployed four clusters of G3 instances with 4, 6, 8 and 16 nodes (correspond to 2, 4, 8 and 13 TensorFlow executors respectively as shown above). Given the broadband network communications, except for the 16-node cluster with two parameter servers, we only dedicated one parameter server along with one spark driver. Moreover, to evaluate the effect of network traffic on training speed, we ran a non-distributed version (called DCA) of the model on a single node with the same configuration. In all the experiments, we trained our models (dist-DCA and standalone DCA) on 9,658,464 time series of HCP Q1 data for 1600 batches and 6036 steps per epoch.

Figure 4.5. demonstrates the speed of various implementations including our standalone and distributed ones on GPU nodes. Since the standalone DCA has no data-parallelism and no network overhead, it obviously outperforms the two-node cluster. However, clusters with the higher number of executor nodes easily exceed regarding computation time. For example, the cluster with 13 executors outperforms the standalone model with almost seven times. It can be observed that training speed linearly grows as the number of executor nodes increase. However, we expect that the performance drops if we increase the number of executor nodes to more than 13. This happens as network overhead starts to rule over our dist-DCA model performance and as executor nodes will have fewer tasks to do while waiting to fetch new parameters.

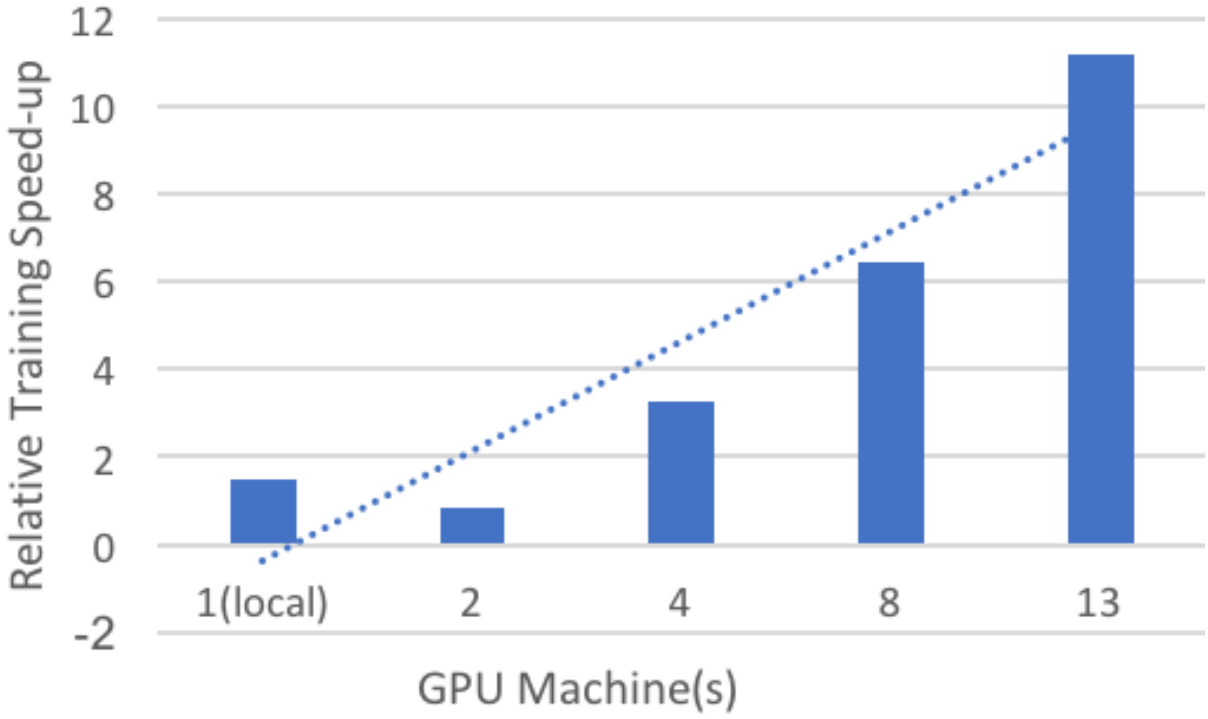

Figure 4.5. Training speed-up versus the numbers of local/spark executors.

4.5.3 Scalability

To further demonstrates the scalability of our implemented distributed framework, we measured the training time of dist-DCA on the previously discussed dataset. We trained our model in 4 different cluster settings with 2, 4, 8 and 13 G3 executor nodes with a total of 2, 4, 8 and 13 GPUs respectively. Please note that for the sake of comparison, in all experiments, only one GPU per node was used. Our goal is to obtain minimum loss in the minimum amount of training time. Figure 4.6. illustrates that the training time is reduced significantly by almost 51 hours in a four-executor with 64 CPU cores (4 GPUs) compared to a two-executor cluster with 32 CPU cores (2 GPUs). However, this increased rate does not hold from the four-node to the eight-node cluster with 128 cores of CPU (8 GPUS). We believe that this is due to network communication overhead and previously discussed warm-up phase. As explained in section 4.4, the TensorFlow application

(here dist-DCA model) is wrapped inside a spark executor at each node. Executors independently start to train the model by pushing gradients and fetching the new parameters from the parameter server at each stage. These recurring network communications can cause the larger clusters to not linearly scale-up as opposed to the ones with fewer nodes. We can conclude that network can always be a bottleneck in larger clusters.

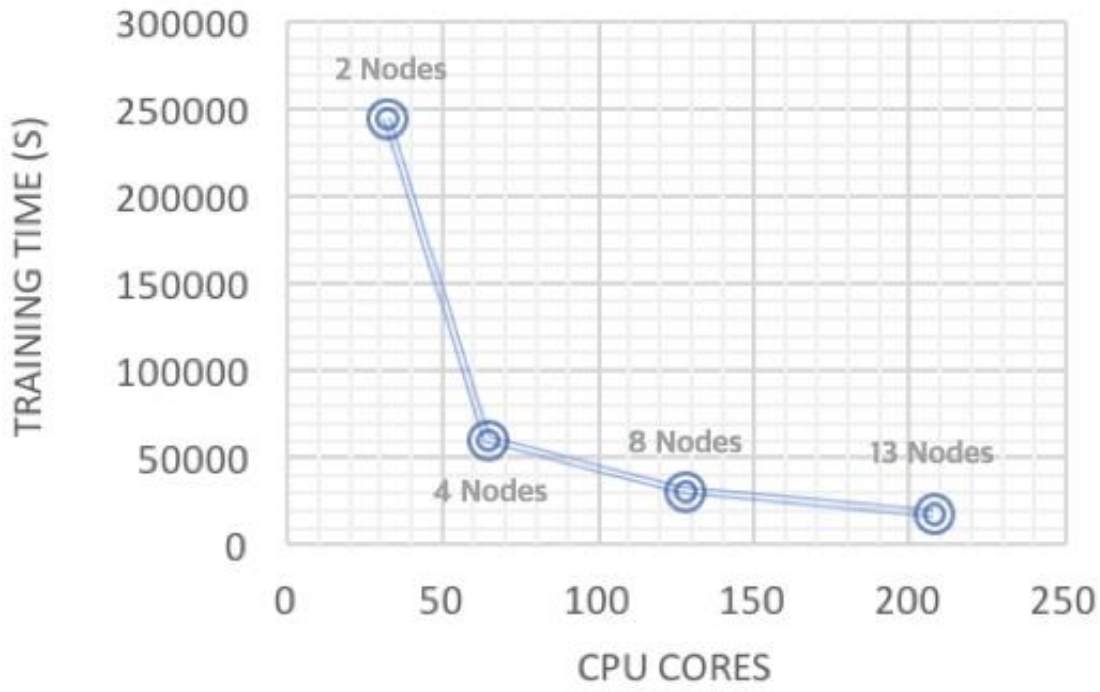

Figure 4.6: Training time of dist-DCA based on the number of CPU cores on different cluster setups.

We also performed another experiment solely to evaluate the effect of CPU cores on our proposed framework performance. To do so, we launched 4 clusters each with 4 nodes to train dit-DCA model over our HCP data. In each cluster setup, we used the same environmental setup and one GPU card per node (total of 4 GPUs). Clusters utilized 4, 8, 16 and 32 CPU cores per node respectively. Demonstrated results in figure 4.7 suggests that increasing only the number of CPUs

would not benefit the training speed significantly. A simple comparison of the results in Fig. 5 with the figure 4.7 shows that while increasing the number of GPUs in a distributed setup reduces training time significantly; such a conclusion cannot be drawn as opposed to increasing CPU cores.

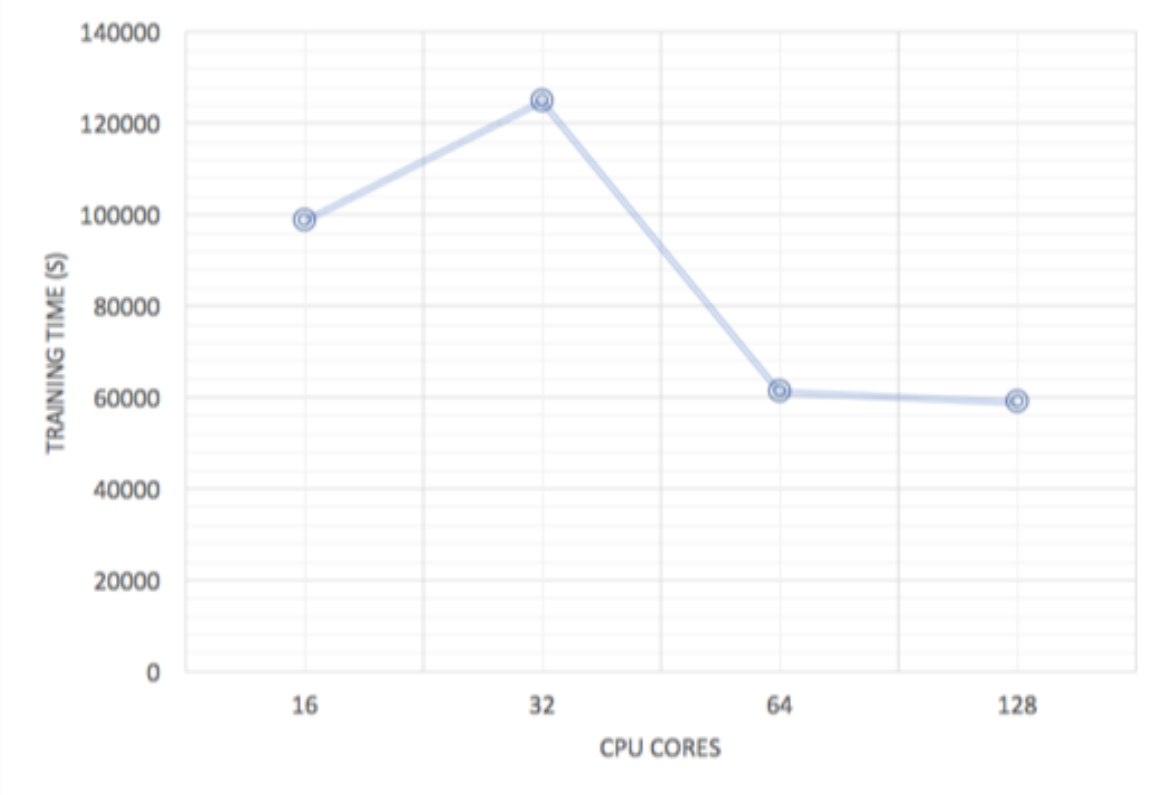

Figure 4.7. Training time of dist-DCA based on the number of CPU cores in clusters with the same number of nodes.

4.5.4 Learned Features Validation and Visualization

To validate the learned features of our proposed model, we have performed a validation study on the hidden layer features of the encoder. An illustration of the computational procedure of this validation study is shown in the figure 4.2c. The rationale behind this is to compare the

detected task-related patterns of brain activity through a sparse dictionary learning method in two setups. One by feeding the high-level features of the hidden layer (setup 1) contained by dist-DCA and the other with the raw tfMRI signals (setup 2). Sparse dictionary learning as an unsupervised learning algorithm aims at finding a sparse representation of input data in the form of a linear combination of basic elements, known as dictionaries along with their corresponding coefficients. [52], [121], [55]. This goal is achieved by aggregating fMRI signals into an over-complete dictionary matrix and a corresponding coefficient matrix through an effective online dictionary learning algorithm [119]. The time series of each over completed dictionary represents the temporal activity of a brain network, and its corresponding reference weight vector stands for the spatial map of every network. This method is recognized as an efficient method for inferring a comprehensive collection of concurrent functional networks in the human brain [52]. The spatial and temporal pattern of a sample network decomposed results is demonstrated in figure 4.8.

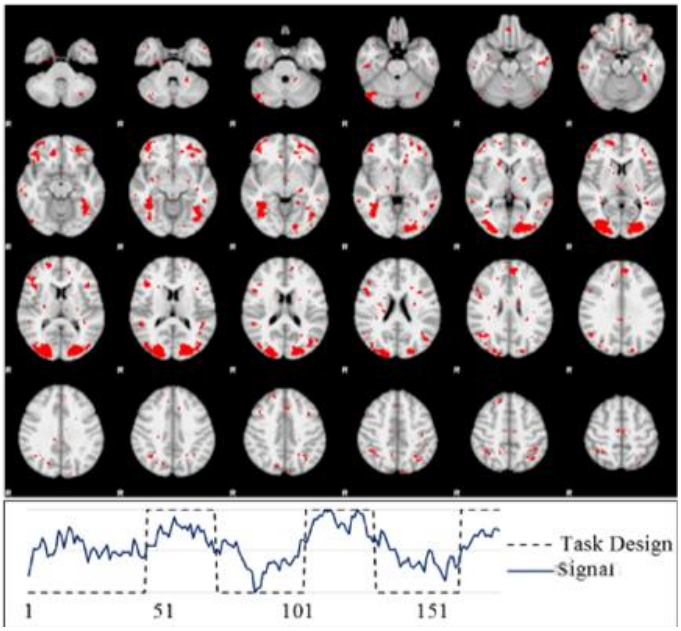

Figure 4.8. Top: spatial pattern visualized on cortical volumetric space of one decomposed network. Bottom: visualization of its temporal pattern.

To draw a fair comparison, we have used the same parameters in both runs. We adopted the parameter-tuning approach that Lv et al. suggested [52]. Both setups learned 400 dictionaries with 0.7 sparsity regularizer [119] to achieve the best performance of the brain network inference. After training, the high-level features of setup one are decomposed as high-level dictionaries and corresponding spatial distributions. Then we use the decoder to project these high-level dictionaries (time series patterns) back to the signal space. The detected patterns are visualized in figure 4.9. As shown on the right side of the figure, although the dictionary learning analysis in both setups has detected all the six motor task patterns, these patterns are mixed with a large number of noises in setup 2, and as a result, the correlation values with task design pattern are quite small. On the other hand, the setup 1 contained much fewer noises in both of the time series patterns and spatial maps. Consequently, we can conclude that our proposed model filters noises in each layer and preserves the useful information of the brain activities. For the sake of simplicity and page limitation, we do not explain the theoretical brain model analysis and reconstruction error analysis. Further details of this comparison can be found at Huang et al. work [120].

Furthermore, we visualized the filters in each layer. Figure 4.10 shows all 32 filters in the first layer of the encoder. The first layer filters summarized the most common sub-shapes of tfMRI time. For example, sinuous and bowl patterns of fMRI are shown with arrows at figure 4.10.

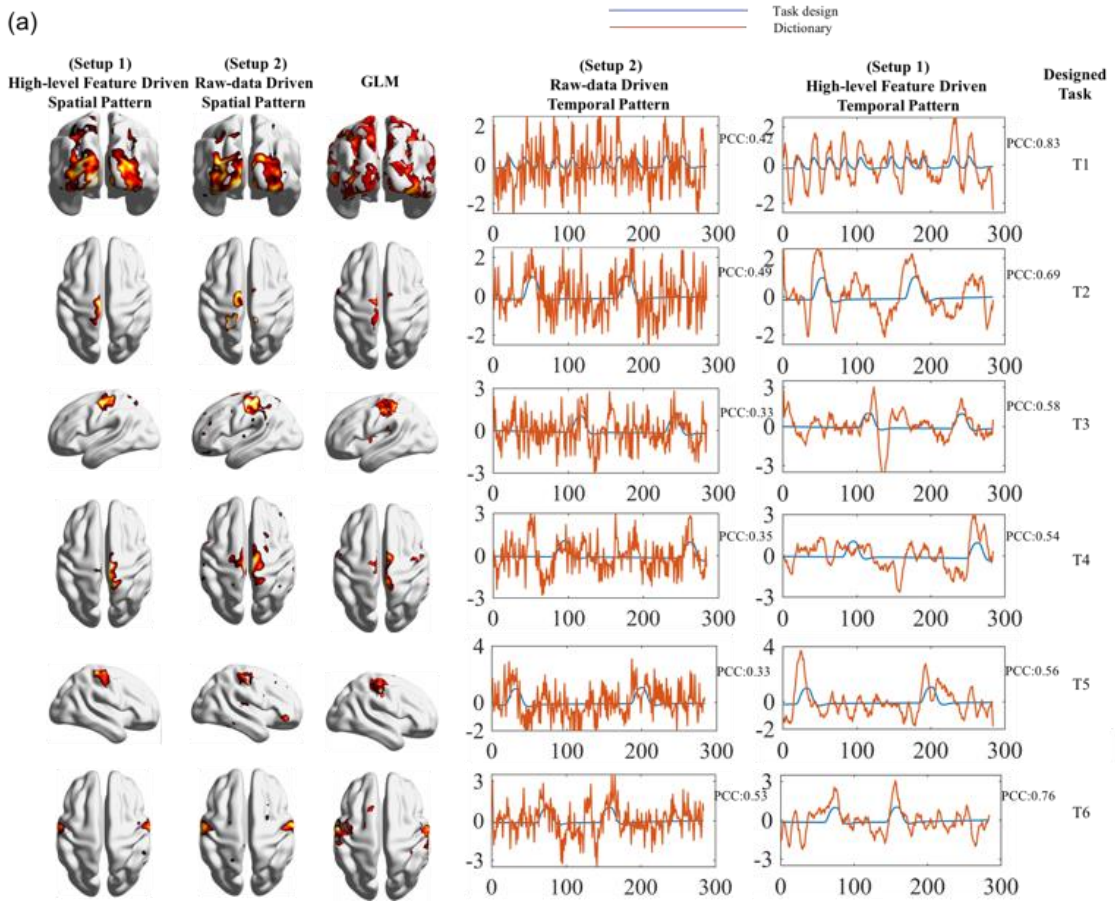

Figure 4.9 Validation study of the dist-DCA. Comparing the temporal and spatial patterns of 6 motor tasks driven from high-level features and raw-data. GLM is for reference. Pearson correlation of the designed tasks with learned dictionary atoms is shown as PCC value

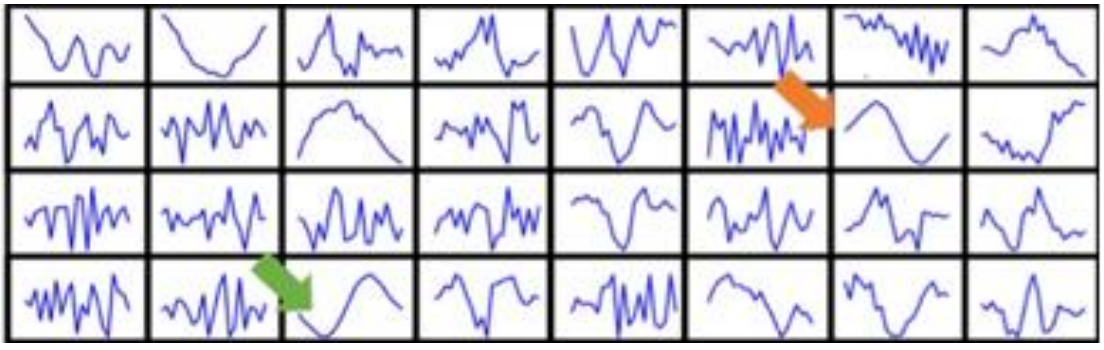

Figure 4.10 All 32 filters in the first layer of encoder. Arrows show the most common pattern of tfMRI time series.

4.6 Conclusions

Providing an effective model to represent the large scale tfMRI data to break down the intrinsic complex structure of tfMRI signals has been highly demanded yet challenging. A novel deep learning model along with distributed computing are the keys to transforming our understanding of some of the most complicated brain signals [120]. In this work we presented a novel scalable distributed deep convolutional autoencoder that hierarchically models large-scale tfMRI time series data while gaining a higher level abstraction of the tfMRI signal. We used Apache Spark and TensorFlow as the computational engines to parallelize millions of fMRI time series and to train our model over large cluster of GPUs. Our experiment results showed that such a model can effectively scale-up to dozens of computation nodes, processing extensive dataset over hundreds of computational cores. The significance of network overhead, however, can severely impact the training time. Furthermore, our results showed that the high-level features are superior in task-related regions detection. The proposed autoencoder was also able to denoise the tfMRI signal as the learned dictionary atoms by our novel high-level sparse dictionary learning suggests. In general, our work contributes a novel deep convolution autoencoder framework for fMRI data modelling with significant application potentials in cognitive and clinical neuroscience in the future.

In our future work, we plan to perform further tests to implement a parallel version of our model to use the computational power of multi-GPU on a multi-node distributed setting to maximize the performance. We also plan to use the 1200+ available subjects of all HCP releases including acquisitions of different types of tasks to identify brain areas in a wide range of neural systems (such as Relation, Working Memory, Language, Social Interaction, Motor, etc.). This will benefit

from our proposed distributed model, enabling data-driven hierarchical neuroscientific discovery from massive fMRI big data in the future.

Acknowledgment

This work was supported by National Institutes of Health (DA033393, AG042599) and National Science Foundation (IIS-1149260, CBET-1302089, and BCS-1439051). Tianming Liu is the corresponding author of this work; phone: (706) 542-3478; E-mail: tianming.liu@gmail.com.

CHAPTER 5

CONCLUSION

The neuroscience has entered into the bigdata era just as other leading sciences. This arrival though requires a cultural shift among the community from enormous isolated efforts applying a single technique to the smaller problems in laboratories toward more horizontal approaches researchers integrate data collected using a variety of techniques to solve bigger problems addressing the central questions of how the brain functionally and structurally connected. I have categorized the current computational efforts of neuroscience experts for in dealing with the bigdata challenges in 6 groups of data management, data visualization, Cloud storage, computing platforms, processing pipelines and processing engines. In this dissertation, I introduced my endeavors to address each of the above categories, notably for fMRI data types.

In chapter 2, I introduced HELPNI as an efficient neuroinformatics platform for data storage, processing pipeline, and data visualization. This platform was first intended to facilitate running and to control complicated neuroimaging multi-stage processes with a smooth, user-friendly web interface and later to give our collaborators parallel and distribute computing accessibility while they implement their own analytical and visualization tools. We applied our Holistic Atlases of Functional Networks and Interactions framework [52] (HAFNI), for the sparse representation of whole brain fMRI signals over more than 5 thousand publicly available fMRI images. HAFNI uses an online dictionary learning algorithm.

As explained in chapter 3, I then concentrated on developing and extending the data storage, data management, and also data processing aspects of HELPNI. The primary goal was to add a distributed file system as well as empowering the computational platform with distributed processing features. Consequently, we devolved a novel Distributed rank-1 Dictionary Learning [55] (D-r1DL) model, leveraging the distributed computing in handling large scale fMRI big data. We implemented the D-r1DL framework on Apache Spark and Apache Flink for distributed functional network analysis on large-scale neuroimaging data. I tested its performance on both the individual and group-wise fMRI data from HCP Q1 release dataset and demonstrated the results through an online visualization tool. The results show that the framework can meet the desired scalability and reproducibility requirements for fMRI bigdata analysis and serve as a useful tool for the community. The framework and the neuroinformatics system are both online as a web service for public usage and testing.

In chapter 4, I further present our novel GPU-based deep learning platform for the distributed data processing that employs TensorFlow, Apache Spark, and Hadoop using cloud computing services. Finally, I demonstrate the significant performance gain of this platform enabling datadriven extraction of hierarchical neuroscientific information from massive fMRI big data using a distributed deep convolutional autoencoder model.

REFERENCES

1 Kaye, J., Heeney, C., Hawkins, N., De Vries, J., & Boddington, P. (2009). Data sharing in genomics—re-shaping scientific practice. Nature Reviews Genetics, 10(5), 331-335.

2 Milham, M. P. (2012). Open neuroscience solutions for the connectome-wide association era. Neuron, 73(2), 214-218.

3 Leonelli, S. (2014). Data interpretation in the digital age. Perspectives on Science, 22(3), 397-417.

4 "Brain Initiative," 2014. [Online]. Available: https://www.braininitiative.nih.gov.

5 "Human Brian Project," 2013. [Online]. Available: https://www.humanbrainproject.eu.

6 Choudhury, S., Fishman, J. R., McGowan, M. L., & Juengst, E. T. (2014). Big data, open science and the brain: lessons learned from genomics. Frontiers in human neuroscience, 8, 239.

7 Van Essen, D. C., Smith, S. M., Barch, D. M., Behrens, T. E., Yacoub, E., Ugurbil, K., & WU-Minn HCP Consortium. (2013). The WU-Minn human connectome project: an overview. Neuroimage, 80, 62-79.

8 Smith, Stephen M., Christian F. Beckmann, Jesper Andersson, Edward J. Auerbach, Janine Bijsterbosch, Gwenaëlle Douaud, Eugene Duff et al. "Resting-state fMRI in the human connectome project." Neuroimage 80 (2013): 144-168.

9 Behrens, T. E., & Sporns, O. (2012). Human connectomics. Current opinion in neurobiology, 22(1), 144-153.

10 Jbabdi, S., Sotiropoulos, S. N., Haber, S. N., Van Essen, D. C., & Behrens, T. E. (2015). Measuring macroscopic brain connections in vivo. Nature neuroscience, 18(11), 1546-1555.

11 Mennes, M., Biswal, B. B., Castellanos, F. X., & Milham, M. P. (2013). Making data sharing work: the FCP/INDI experience. Neuroimage, 82, 683-691.

12 Poldrack, R. A., Barch, D. M., Mitchell, J., Wager, T., Wagner, A. D., Devlin, J. T., ... & Milham, M. (2013). Toward open sharing of task-based fMRI data: the OpenfMRI project. Frontiers in neuroinformatics, 7, 12.

13 Sejnowski, T. J., Churchland, P. S., & Movshon, J. A. (2014). Putting big data to good use in neuroscience. Nature neuroscience, 17(11), 1440-1441.

14 Ferguson, A. R., Nielson, J. L., Cragin, M. H., Bandrowski, A. E., & Martone, M. E. (2014). Big data from small data: data-sharing in the'long tail'of neuroscience. Nature neuroscience, 17(11), 1442-1447.

15 Poldrack, R. A., & Gorgolewski, K. J. (2014). Making big data open: data sharing in neuroimaging. Nature neuroscience, 17(11), 1510-1517.

16 Van Horn, J. D., & Toga, A. W. (2014). Human neuroimaging as a "Big Data" science. Brain imaging and behavior, 8(2), 323-331.

17 Abolghasemi, V., Ferdowsi, S., & Sanei, S. (2015). Fast and incoherent dictionary learning algorithms with application to fMRI. Signal, Image and Video Processing, 9(1), 147-158.

18 Ardekani, B. A., & Kanno, I. (1998). Statistical methods for detecting activated regions in functional MRI of the brain. Magnetic Resonance Imaging, 16(10), 1217-1225.

19 Andersen, A. H., Gash, D. M., & Avison, M. J. (1999). Principal component analysis of the dynamic response measured by fMRI: a generalized linear systems framework. Magnetic Resonance Imaging, 17(6), 795-815.

20 Bandettini, P. A., Jesmanowicz, A., Wong, E. C., & Hyde, J. S. (1993). Processing strategies for time-course data sets in functional MRI of the human brain. Magnetic resonance in medicine, 30(2), 161-173.

21 Ogawa, S., Tank, D. W., Menon, R., Ellermann, J. M., Kim, S. G., Merkle, H., & Ugurbil, K. (1992). Intrinsic signal changes accompanying sensory stimulation: functional brain mapping with magnetic resonance imaging. Proceedings of the National Academy of Sciences, 89(13), 5951-5955.

22 Logothetis, N. K. (2008). What we can do and what we cannot do with fMRI. Nature, 453(7197), 869-878.

23 Biswal, B. B., Kylen, J. V., & Hyde, J. S. (1997). Simultaneous assessment of flow and BOLD signals in resting-state functional connectivity maps. NMR in Biomedicine, 10(45), 165-170.

24 Heeger, D. J., & Ress, D. (2002). What does fMRI tell us about neuronal activity?. Nature Reviews Neuroscience, 3(2), 142-151.

25 Fox, M. D., & Raichle, M. E. (2007). Spontaneous fluctuations in brain activity observed with functional magnetic resonance imaging. Nature Reviews Neuroscience, 8(9), 700-711.

26 Biswal, B.B., Mennes, M., Zuo, X.N., Gohel, S., Kelly, C., Smith, S.M., Beckmann, C.F., Adelstein, J.S., Buckner, R.L., Colcombe, S. and Dogonowski, A.M., 2010. Toward discovery science of human brain function. Proceedings of the National Academy of Sciences, 107(10), pp.4734-4739.

27 Friston, K. (2009). Causal modelling and brain connectivity in functional magnetic resonance imaging. PLoS biol, 7(2), e1000033.

28 Biswal, B. B., Mennes, M., Zuo, X. N., Gohel, S., Kelly, C., Smith, S. M., ... & Dogonowski, A. M. (2010). Toward discovery science of human brain function. Proceedings of the National Academy of Sciences, 107(10), 4734-4739.

29 Biswal, B. B. (2012). Resting state fMRI: a personal history. Neuroimage, 62(2), 938-944.

30 Friston KJ, Ashburner J, Heather J (2003) Statistical parametric mapping. In: Neuroscience databases: a practical guide, p 237

31 Aguirre, G. K. (2012). FIASCO, VoxBo, and MEDx: behind the code. Neuroimage, 62(2), 765-767.

32 Smith, S. M., Jenkinson, M., Woolrich, M. W., Beckmann, C. F., Behrens, T. E., Johansen-Berg, H., ... & Niazy, R. K. (2004). Advances in functional and structural MR image analysis and implementation as FSL. Neuroimage, 23, S208-S219.

33 Woolrich, M. W., Jbabdi, S., Patenaude, B., Chappell, M., Makni, S., Behrens, T., ... & Smith, S. M. (2009). Bayesian analysis of neuroimaging data in FSL. Neuroimage, 45(1), S173-S186.

34 Friston, K. J. (2003). Statistical parametric mapping. In Neuroscience Databases (pp. 237-250). Springer US.

35 Cox, R. W. (1996). AFNI: software for analysis and visualization of functional magnetic resonance neuroimages. Computers and Biomedical research, 29(3), 162-173.

36 Fox, P. T., & Lancaster, J. L. (2002). Mapping context and content: the BrainMap model. Nature Reviews Neuroscience, 3(4), 319-321.

37 Marcus, D. S., Olsen, T. R., Ramaratnam, M., & Buckner, R. L. (2007). The extensible neuroimaging archive toolkit. Neuroinformatics, 5(1), 11-33.

38 Marcus, D. S., Harms, M. P., Snyder, A. Z., Jenkinson, M., Wilson, J. A., Glasser, M. F., ... & Hodge, M. (2013). Human Connectome Project informatics: quality control, database services, and data visualization. Neuroimage, 80, 202-219.

39 Rex, D. E., Ma, J. Q., & Toga, A. W. (2003). The LONI pipeline processing environment. Neuroimage, 19(3), 1033-1048.

40 Dinov, I., Lozev, K., Petrosyan, P., Liu, Z., Eggert, P., Pierce, J., ... & Magsipoc, R. (2010). Neuroimaging study designs, computational analyses and data provenance using the LONI pipeline. PloS one, 5(9), e13070.

41 Gorgolewski, K., Burns, C. D., Madison, C., Clark, D., Halchenko, Y. O., Waskom, M. L., & Ghosh, S. S. (2011). Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in python. Frontiers in neuroinformatics, 5, 13.

42 Moritz, P., Nishihara, R., Stoica, I., & Jordan, M. I. (2015). SparkNet: Training Deep Networks in Spark. arXiv preprint arXiv:1511.06051.

43 Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). TensorFlow: A system for large-scale machine learning. arXiv preprint arXiv:1605.08695.

44 Xing, E. P., Ho, Q., Xie, P., & Dai, W. (2015). Strategies and Principles of Distributed Machine Learning on Big Data. arXiv preprint arXiv:1512.09295.

45 Kim, H., Park, J., Jang, J., & Yoon, S. (2016). DeepSpark: Spark-Based Deep Learning Supporting Asynchronous Updates and Caffe Compatibility. arXiv preprint arXiv:1602.08191.

46 Freeman, J. (2015). Open source tools for large-scale neuroscience. Current opinion in neurobiology, 32, 156-163.

47 Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.

48 Chu, C., Kim, S. K., Lin, Y. A., Yu, Y., Bradski, G., Ng, A. Y., & Olukotun, K. (2007). Map-reduce for machine learning on multicore. Advances in neural information processing systems, 19, 281.

49 Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: cluster computing with working sets. HotCloud, 10, 10-10.

50 "Apache Spark," 2014. [Online]. Available: http://flink.apache.org

51 Makkie, M., Zhao, S., Jiang, X., Lv, J., Zhao, Y., Ge, B., ... & Liu, T. (2015). HAFNI-enabled largescale platform for neuroimaging informatics (HELPNI). Brain informatics, 2(4), 225-238.

52 Lv, J., Jiang, X., Li, X., Zhu, D., Zhang, S., Zhao, S., ... & Ye, J. (2015). Holistic atlases of functional networks and interactions reveal reciprocal organizational architecture of cortical function. IEEE Transactions on Biomedical Engineering, 62(4), 1120-1131.

53 Wright, J., Yang, A. Y., Ganesh, A., Sastry, S. S., & Ma, Y. (2009). Robust face recognition via sparse representation. IEEE transactions on pattern analysis and machine intelligence, 31(2), 210-227.

54 Mairal, J., Bach, F., Ponce, J., & Sapiro, G. (2010). Online learning for matrix factorization and sparse coding. Journal of Machine Learning Research, 11(Jan), 19-60.

55 Li, X., Makkie, M., Lin, B., Fazli, M. S., Davidson, I., Ye, J., ...& Quinn, S. (2016). Scalable Fast Rank-1 Dictionary Learning for fMRI Big Data Analysis. ACM KDD.

56 Makkie, M., Huang, H, Zhao, Y., Vassilakos, A., Liu, T., 2017. Fast and Scalable Distributed Deep Convolutional Autoencoder for fMRI Big Data Analytics. arXiv preprint arXiv: 1710.08961 [cs.DC].

57 Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010, May). The hadoop distributed file system. In Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on (pp. 1-10). IEEE.

58 Laird, A.R., Eickhoff, S.B., Kurth, F., Fox, P.M., Uecker, A.M., Turner, J.A., Robinson, J.L., Lancaster, J.L. and Fox, P.T., 2009. ALE meta-analysis workflows via the brainmap database: progress towards a probabilistic functional brain atlas. Frontiers in neuroinformatics, 3, p.23.

59 Goebel R (2012) BrainVoyager—past, present, future. Neuroimage 62(2):748–756

60 Luo XJ, Kennedy DN, Cohen Z (2009) Neuroimaging informatics tools and resources clearinghouse (NITRC) resource announcement. Neuroinformatics 7(1):55–56

61 Gardner, Daniel, Huda Akil, Giorgio A. Ascoli, Douglas M. Bowden, William Bug, Duncan E. Donohue, David H. Goldberg et al. "The neuroscience information framework: a data and knowledge environment for neuroscience." Neuroinformatics 6, no. 3 (2008): 149-160.

62 Keator, David B., Jeffrey S. Grethe, D. Marcus, B. Ozyurt, Syam Gadde, Sean Murphy, Steve Pieper et al. "A national human neuroimaging collaboratory enabled by the Biomedical Informatics Research Network (BIRN)." IEEE Transactions on Information Technology in Biomedicine 12, no. 2 (2008): 162-172.

63 Fan J, Han F, Liu H (2014) Challenges of big data analysis. Natl Sci Rev 1(2):293–314

64 Marcus, D. S., T. Olsen, M. Ramaratnam, and R. L. Buckner. "XNAT: a software framework for managing neuroimaging laboratory data." In Proceedings of the 12th annual meeting of the organization for human brain mapping, Florence. 2006.

65 Masse M (2011) REST API design rulebook. O'Reilly Media Inc, Sebastopol

66 Donoho, David L. "Compressed sensing." IEEE Transactions on information theory 52, no. 4 (2006): 1289-1306.

67 Huang, Ke, and Selin Aviyente. "Sparse representation for signal classification." In Advances in neural information processing systems, pp. 609-616. 2007.

68 Wright, John, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S. Huang, and Shuicheng Yan. "Sparse representation for computer vision and pattern recognition." Proceedings of the IEEE 98, no. 6 (2010): 1031-1044.

69 Yang, Meng, Lei Zhang, Xiangchu Feng, and David Zhang. "Fisher discrimination dictionary learning for sparse representation." In Computer Vision (ICCV), 2011 IEEE International Conference on, pp. 543-550. IEEE, 2011.

70 Smith, Stephen M., Peter T. Fox, Karla L. Miller, David C. Glahn, P. Mickle Fox, Clare E. Mackay, Nicola Filippini et al. "Correspondence of the brain's functional architecture during activation and rest." Proceedings of the National Academy of Sciences 106, no. 31 (2009): 13040-13045.

71 Ge, Bao, Milad Makkie, Jin Wang, Shijie Zhao, Xi Jiang, Xiang Li, Jinglei Lv et al. "Signal sampling for efficient sparse representation of resting state FMRI data." Brain imaging and behavior 10, no. 4 (2016): 1206-1222.

72 Sindhwani, Vikas, and Amol Ghoting. "Large-scale distributed non-negative sparse coding and sparse dictionary learning." In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 489-497. ACM, 2012.

73 Aharon, Michal, Michael Elad, and Alfred Bruckstein. "$ rm k $-SVD: An algorithm for designing overcomplete dictionaries for sparse representation." IEEE Transactions on signal processing 54, no. 11 (2006): 4311-4322.

74 Biswal, Bharat B., and John L. Ulmer. "Blind source separation of multiple signal sources of fMRI data sets using independent component analysis." Journal of computer assisted tomography 23, no. 2 (1999): 265-271.

75 D'aspremont, A., Ghaoui, L.E., Jordan, M.I., and Lanckreit, G.R., 2004. A Direct Formulation for Sparse PCA Using Semidefinite Programming. In Advances in Neural Information Processing Systems.

76 Damoiseaux, J.S., Rombouts, S.A.R.B., Barkhof, F., Scheltens, P., Stam, C.J., Smith, S.M., and Beckmann, C.F., 2006. Consistent resting-state networks across healthy subjects. Proceedings of the National Academy of Sciences of the United States of America 103, 37, 02/20/received), 13848-13853.

77 Elad, M. and Aharon, M., 2006. Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. Image Processing, IEEE Transactions on 15, 12, 3736-3745.

78 Glasser, M.F., Sotiropoulos, S.N., Wilson, J.A., Coalson, T.S., Fischl, B., Anderson, J.L., Xu, J., Jbabdi, S., Webster, M., Polimeni, J.R., Van Essen, D.C., and Jenkinson, M., 2013. The minimal preprocessing pipelines for the Human Connectome Project. NeuroImage 80, 105-124.

79 Gonzalez, J.E., Xin, R.S., Dave A., Crankshaw, D., Franklin, M.J., and Stoica, I., 2014. GraphX: graph processing in a distributed dataflow framework. In Proceedings of the Proceedings of the 11th USENIX conference on Operating Systems Design and Implementation, USENIX Association, 2685096, 599-613.

80 Kangjoo, L., Sungho, T., and Jong Chul, Y., 2011. A Data-Driven Sparse GLM for fMRI Analysis Using Sparse Dictionary Learning With MDL Criterion. Medical Imaging, IEEE Transactions on 30, 5, 1076-1089.

81 Lee, H., Battle, A., Raina, R., and NG, A.Y., 2006. Efficient sparse coding algorithms. In Advances in Neural Information Processing Systems.

82 Lin, B., Li, Q., Sun, Q., Lai, M.-J., Davidson, I., Fan, W., and Ye, J., 2014. Stochastic Coordinate Coding and Its Application for Drosophila Gene Expression Pattern Annotation. arXiv:1407.8147.

83 Liu, B.-D., Wang, Y.-X., Zhang, Y.-J., and Shen, B., 2013. Learning dictionary on manifolds for image classification. Pattern Recognition 46, 7, 1879-1890.

84 Mackey, L.W., 2008. Deflation Methods for Sparse PCA. In Advances in Neural Information Processing Systems.

85 McKeown, M.J., Sejnowski, T.J.: Independent component analysis of fMRI data: examining the assumptions. Hum. Brain Mapp. 6, 368–72 (1998).

86 Ravishankar, S. and Bresler, Y., 2011. MR Image Reconstruction From Highly Undersampled k-Space Data by Dictionary Learning. Medical Imaging, IEEE Transactions on 30, 5, 1028-1041.

87 Smith, S.M., Hyvarinen, A., Varoquaux, G., Miller, K.L., and Beckmann, C.F., 2014. Group-PCA for very large fMRI datasets. NeuroImage 101, 0, 738-749.

88 Smith, S.M., Miller, K.L., Moeller, S., XU, J., Auerbach, E.J., Woolrich, M.W., Beckmann, C.F., Jenkinson, M., Andersson, J., Glasser, M.F., Van Essen, D.C., Feinberg, D.A., Yacoub, E.S., and Ugurbil, K., 2012. Temporally-independent functional modes of spontaneous brain activity. Proceedings of the National Academy of Sciences 109, 8, 3131-3136.

89 Thirion, B. and Faugeras, O., 2003. Dynamical components analysis of fMRI data through kernel PCA. NeuroImage 20, 1, 34-49.

90 Zharia, M., Chowdhury, M., Das, T., Dave, A., MA, J., Mccauley, M., Franklin, M.J., Shenker, S., and Stoica, I., 2012. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, 2228301, 2-2.

91 Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., and Stoica, I., 2013. Discretized streams: fault-tolerant streaming computation at scale. In Proceedings of the Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, 2522737, 423-438.

92 Friston, K.J., Holmes, a. P., Worsley, K.J., Poline, J.-P., Frith, C.D., Frackowiak, R.S.J.: Statistical parametric maps in functional imaging: A general linear approach. Hum. Brain Mapp. 2, 189–210 (1995).

93 Ferrarini, L., Veer, I.M., Baerends, E., Van Tol, M.J., Renken, R.J., Van Der Wee, N.J.A., Veltman, D.J., Aleman, A., Zitman, F.G., Penninx, B.W.J.H., Van Buchem, M.A., Reiber, J.H.C., Rombouts, S.A.R.B., Milles, J.: Hierarchical functional modularity in the resting-state human brain. Hum. Brain Mapp. 30, 2220–2231 (2009).

94 Meunier, D., Lambiotte, R., Fornito, A., Ersche, K.D., Bullmore, E.T.: Hierarchical modularity in human brain functional networks. Front. Hum. Neurosci. 3, 1–12 (2009).

95 McCulloch, Warren S., and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." The bulletin of mathematical biophysics 5.4 (1943): 115-133.

96 Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012).

97 Sun, Jian, Huibin Li, and Zongben Xu. "Deep ADMM-Net for compressive sensing MRI." In Advances in Neural Information Processing Systems, pp. 10-18. 2016.

98 Cireşan, Dan C., Alessandro Giusti, Luca M. Gambardella, and Jürgen Schmidhuber. "Mitosis detection in breast cancer histology images with deep neural networks." In International Conference on Medical Image Computing and Computer-assisted Intervention, pp. 411-418. Springer, Berlin, Heidelberg, 2013.

99 Turaga, Srinivas C., Joseph F. Murray, Viren Jain, Fabian Roth, Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and H. Sebastian Seung. "Convolutional networks can learn to generate affinity graphs for image segmentation." Neural computation 22, no. 2 (2010): 511-538.

100 Taigman, Yaniv, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. "Deepface: Closing the gap to human-level performance in face verification." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1701-1708. 2014.

101 Pu, Yunchen, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. "Variational autoencoder for deep learning of images, labels and captions." In Advances in neural information processing systems, pp. 2352-2360. 2016.

102 Masci, J., Meier, U., Cireşan, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. In: International Conference on Artificial Neural Networks. pp. 52–59. Springer (2011).

103 Dean, Jeffrey, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior et al. "Large scale distributed deep networks." In Advances in neural information processing systems, pp. 1223-1231. 2012.

104 D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Deep big simple neural nets excel on handwritten digit recognition. CoRR, 2010.

105 Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In AISTATS 14, 2011.

106 Q.V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A.Y. Ng. On optimization methods for deep learning. In ICML, 2011.

107 Chen, Po-Hsuan, Xia Zhu, Hejia Zhang, Javier S. Turek, Janice Chen, Theodore L. Willke, Uri Hasson, and Peter J. Ramadge. "A convolutional autoencoder for multi-subject fmri data aggregation." arXiv preprint arXiv:1608.04846 (2016)

108 Plis, S.M., Hjelm, D.R., Slakhutdinov, R., Allen, E.A., Bockholt, H.J., Long, J.D., Johnson, H., Paulsen, J., Turner, J., Calhoun, V.D.: Deep learning for neuroimaging: A validation study. Front. Neurosci. 1–11 (2014).

109 Suk, H.-I., Wee, C.-Y., Lee, S.-W., Shen, D.: State-space model with deep learning for functional dynamics estimation in resting-state fMRI. Neuroimage. 129, 292–307 (2016).

110 Huang, H., Hu, X., Han, J., Lv, J., Liu, N., Guo, L., Liu, T.: Latent source mining in FMRI data via deep neural network, (2016).

111 Dehua Ren, Yu Zhao, Hanbo Chen, Qinglin Dong, Jinglei Lv, Tianming Liu, 3D functional brain network classification using convolutional neural networks, accepted, ISBI 2017.

112 Wen, Haiguang, Junxing Shi, Yizhen Zhang, Kun-Han Lu, Jiayue Cao, and Zhongming Liu. "Neural encoding and decoding with deep learning for dynamic natural vision." Cerebral Cortex (2017): 1-25.

113 Andy Feng, TensorFlowOnSpark, (2016), GitHub repository, https://github.com/yahoo/TensorFlowOnSpark

114 Barch, D.M., Burgess, G.C., Harms, M.P., Petersen, S.E., Schlaggar, B.L., Corbetta, M., Glasser, M.F., Curtiss, S., Dixit, S., Feldt, C.: Function in the human connectome: task-fMRI and individual differences itn behavior. Neuroimage. 80, 169–189 (2013).

115 Friston, K.J., Harrison, L., Penny, W.: Dynamic causal modelling. Neuroimage. 19, 1273–1302 (2003).

116 Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European Conference on Computer Vision. pp. 818–833. Springer (2014).

117 Joeri Hermans and CERN IT-DB, Distributed Keras: Distributed Deep Learning with Apache Spark and Keras, (2016), GitHub repository, https://github.com/JoeriHermans/dist-keras

118 J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12:2121–2159, 2011.

119 Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online dictionary learning for sparse coding. In: Proceedings of the 26th annual international conference on machine learning. pp. 689–696. ACM (2009).

120 Huang, Heng, Xintao Hu, Yu Zhao, Milad Makkie, Qinglin Dong, Shijie Zhao, Lei Guo, and Tianming Liu. "Modeling Task fMRI Data via Deep Convolutional Autoencoder." IEEE transactions on medical imaging (2017).

121 Weiner, Michael W., Dallas P. Veitch, Paul S. Aisen, Laurel A. Beckett, Nigel J. Cairns, Robert C. Green, Danielle Harvey et al. "The Alzheimer's Disease Neuroimaging Initiative: a review of papers published since its inception." Alzheimer's & Dementia 9, no. 5 (2013): e111-e194.

122 Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin et al. "TensorFlow: A System for Large-Scale Machine Learning." In OSDI, vol. 16, pp. 265-283. 2016.

123 Max Pumperla, elephas, (2015), GitHub repository, https://github.com/maxpumperla/elephas

124 Deeplearning4j Development Team. Deeplearning4j: Open-source distributed deep learning for the JVM, Apache Software Foundation License 2.0. https://deeplearning4j.org

125 Zhang, Sixin, Anna E. Choromanska, and Yann LeCun. "Deep learning with elastic averaging SGD." Advances in Neural Information Processing Systems. 2015.