# Real-time Full Correlation Matrix Analysis of fMRI Data

Yida Wang*, Bryn Keller*, Mihai Capotă*, Michael J. Anderson*, Narayanan Sundaram*,
Jonathan D. Cohen‡, Kai Li†, Nicholas B. Turk-Browne‡, Theodore L. Willke*
*Parallel Computing Lab, Intel Corporation,
†Department of Computer Science, Princeton University,
‡Princeton Neuroscience Institute, Princeton University

***Abstract*—Real-time functional magnetic resonance imaging (rtfMRI) is an emerging approach for studying the functioning of the human brain. Computational challenges combined with high data velocity have to this point restricted rtfMRI analyses to studying regions of the brain independently. However, given that neural processing is accomplished via functional interactions among brain regions, neuroscience could stand to benefit from rtfMRI analyses of full-brain interactions. In this paper, we extend such an offline analysis method, full correlation matrix analysis (FCMA), to enable its use in rtfMRI studies. Specifically, we introduce algorithms capable of processing real-time data for all stages of the FCMA machine learning workflow: incremental feature selection, model updating, and real-time classification. We also present an actor-model based distributed system designed to support FCMA and other rtfMRI analysis methods. Experiments show that our system successfully analyzes a stream of brain volumes and returns neurofeedback with less than 180 ms of lag. Our real-time FCMA implementation provides the same accuracy as an optimized offline FCMA toolbox while running 3.6–6.2× faster.**

***Keywords*-big data; real-time fMRI; streaming; machine learning; full correlation matrix analysis**

## I. Introduction

Functional magnetic resonance imaging (fMRI) is the dominant technique for investigating human brain activity in neuroscience research. A typical fMRI study generates a 3D brain volume consisting of 25 000–35 000 data points (called *voxels*) every 1–2 seconds. Almost all existing fMRI studies are conducted in an *offline* fashion—statistical analysis occurs only after all data have been acquired and sent to a file server or lab for processing, long after the research subject has been taken out of the scanner. This offline approach allows researchers to design an experiment, perform it on multiple subjects, and then analyze the reliability of the data by examining consistency across subjects. Although sufficient for many purposes, this approach has three major limitations: First, the pace of discovery is very slow, with typical fMRI studies lasting 6-12 months total, and often at least 2-3 months before even tentative results are known. Second, it is assumed that the same exact experiment should be run for each participant, missing opportunities to tailor the study (e.g., the level of difficulty) to an individual's cognitive abilities and to assess whether enough data, or the right kind of data, has been collected in a given session. Third, it misses the opportunity to use information acquired during scanning as feedback to the subject, either to enhance participation and/or training (e.g., on attention [1]).

Recently, it has become possible to conduct online or real-time, closed loop fMRI (rtfMRI) studies, in which data are preprocessed and analyzed as they are collected, keeping pace with the rate of data acquisition [2], [3], [4]. This partially addresses the issues above, at least for limited forms of analysis. For example, a stimulus can be dynamically "triggered" based on the amount or pattern of activity in a brain region, allowing for stronger inferences about the region and behavior [5]; experimental parameters, such as stimuli and tasks, can be altered in order to optimize the experiment to recruit particular brain regions [6]; and some visualization of activity in a brain region can be provided to subjects as a form of "neurofeedback", helping them better engage in a task [7]. Despite these advancements, there still exists a major gulf between online and offline measures.

### A. Challenges

One critical constraint on existing rtfMRI methods concerns the type of analyses that are possible. Most analyses used in rtfMRI treat the activity observed in each voxel independently of one another. However, brain function relies not only on the isolated activity of different areas, but also, if not more critically, on *interactions* between different brain regions, which can be reflected in *correlations* of activity among these. Such correlations, sometimes referred to as functional connectivity, have become an increasingly important focus of brain imaging research as a big-data problem [8]. However, because of computational constraints, such correlational analyses have typically been restricted to predefined regions of interest (ROIs). This is problematic not only because it restricts the extent of analysis, but also because it biases analysis to seed regions identified by their isolated activity. This is blind to effects that reside entirely in

978-1-4673-9005-7/16/$31.00 ©2016 IEEE

patterns of correlations (i.e., without notable changes in mean activity). To address this limitation, we recently developed a method for full correlation matrix analysis (FCMA), allowing unbiased analysis of patterns of correlations over all voxels in the entire brain. We have shown that this can reveal aspects of brain function not revealed by traditional, non-correlational methods [9]. However, because of the computational demands, to date it has not been possible to incorporate this method into rtfMRI studies.

Another problem that has limited the use of rtfMRI is the lack of readily accessible, standardized tools for implementing it. Today, different neuroscience research teams have to set up their own rtfMRI experimental environments, typically using a standalone machine sitting beside the scanner to receive and analyze the incoming data stream. This results in considerable duplication of effort, and often inefficient solutions, or ones limited by local resources.

### B. Solution

To surmount these limitations, we propose algorithms to conduct FCMA in real-time (henceforth referred to as rtFCMA) based on highly optimized batch FCMA code [10] (henceforth referred to as batchFCMA) by leveraging the characteristics of real-time data streams. Specifically, we design a full-stack real-time machine learning solution which includes feature selection, model updating, and classification.

We also present the design and implementation of a distributed system to support rtfMRI analysis and showcase it using rtFCMA as an application. FCMA is an computationally intensive analysis that was not possible even in offline analysis until recently; it represents an upper limit on current computational demands for rtfMRI analyses. The rtFCMA system can be deployed in *Software as a Service (SaaS) mode*, which will allow neuroscientists from MRI centers around the world to conduct their own real-time neurofeedback experiments, leveraging and creating shared computing resources. The distributed system requires minimal resources in the scanner room. The back-end, which resides on a local compute cluster or potentially a cloud, has a simple REST HTTPS API that allows easy integration and fits into existing network administration rules. The REST server is a gateway to the distributed back-end, which provides a flexible set of processes to handle rtfMRI analyses in various topologies, with different experimental configurations, and using a variety of algorithms.

Figure 1 shows the general concept of the rtFCMA system. An fMRI machine continuously scans the brain of a human subject while they perform certain tasks to generate a data stream of 3D brain volumes. The data stream is sent to a compute cluster where it is processed.

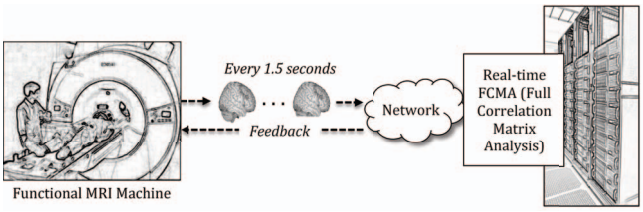


Figure 1. Conceptual diagram of rtFCMA.

The resulting feedback is sent back to the fMRI scanner in real time to form a closed data transfer loop. Our system is the first ever rtfMRI system with low latency closed-loop feedback running on a compute cluster.

The rtFCMA system runs two tasks with different latency requirements. First, it needs to generate neurofeedback with low latency. As an example, it typically takes 1.5 seconds to produce a functional brain volume on an MRI scanner. To keep pace with this data stream from the scanner, rtFCMA needs to process a given volume (e.g., produce a confidence score of subject's mental state that can be used to adjust the stimulus) before the end of the next volume. The latency from the completion of a brain volume to its delivery to rtFCMA by the scanner can be as high as 780 ms in practice (because of image reconstruction, file writing, and network transmission), which leaves 720 ms for rtFCMA neurofeedback. Second, rtFCMA needs to dynamically adjust the classification model to account for incoming data. This process requires a latency of tens of seconds at most, to minimize the time the subject is kept waiting in the scanner. Hence, for real-time operation rtFCMA must provide neurofeedback in 720 ms, and update its model in tens of seconds.

The contribution of this paper can be highlighted as follows: (1) We propose and implement new algorithms for rtFCMA, which are able to provide neurofeedback with $< 180$ ms of lag and to update the classification model in $< 25$ seconds. The rtFCMA algorithms updates its model 3.6–6.2× faster than batchFCMA without sacrificing accuracy, with techniques that can be generalized to many other applications. (2) We design and build a distributed system to enable rtFCMA, as well as potentially other types of rtfMRI.

## II. Full Correlation Matrix Analysis

In this section, we briefly describe FCMA and its applicability to rtfMRI studies.

### A. Background

In fMRI studies, scanned brain volumes consist of voxels. Each voxel has a blood-oxygen level dependent (BOLD) value representing neural activity in the brain area corresponding to the voxel. A typical study is divided in time epochs, with the subject performing a certain cognitive tasks during the fMRI scan in each epoch. Epochs are labeled based on the type of cognitive

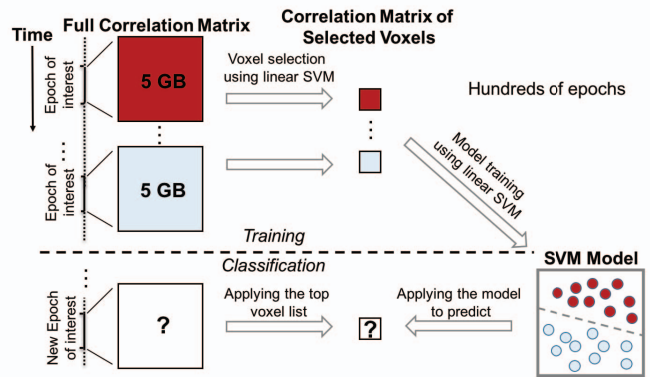


Figure 2. The data processing flow of FCMA.

task, e.g., the label specifies the sort of images shown to the subject during that epoch.

The goal of FCMA is predicting the cognitive state of the subject (i.e., the label of the epoch) by analyzing functional interactions in the brain. Unlike traditional approaches based on the activity of individual voxels, FCMA uses as input the Pearson correlation between the activity of different voxels. Figure 2 shows the general data processing flow of FCMA. The basic unit of operation for FCMA is a full correlation matrix—i.e., the temporal correlation of the activity of every voxel in the brain with every other voxel. A separate full correlation matrix is computed for each epoch in the scanning session. The size of a full correlation matrix in our case is ∼5 GB, and can be even larger when a higher resolution fMRI scanner is used. Based on the task types during the epochs, the matrices are labeled as different categories (depicted in figure 2 as dark red and light blue). For example, in epoch $x$, the subject is attending to a face image, whereas in epoch $y$, the subject is attending to a scene image. In a typical fMRI study, the number of epochs is in hundreds, corresponding to hundreds of full correlation matrices.

In order to extract useful information from the huge and noisy full correlation matrices, FCMA first performs voxel selection (i.e., feature selection), which picks a subset of voxels whose correlation vectors are most predictive of the label classes among all brain voxels. In addition to removing noise, voxel selection also makes results easier to interpret for neuroscientists (by localizing interesting voxels) and speeds up computation. The voxels are selected based on the cross-validated prediction accuracy of their individual correlation vectors: For each voxel, we compute its correlation values with all other voxels in the brain over multiple epochs and then conduct cross validation. This step thus requires a lot of computing and memory resources.

Once the most relevant voxels have been selected, FCMA uses them to build correlation matrices of the top-$K$ selected voxels (the small matrices in figure 2) and trains a machine learning model to classify these correlation matrices. In both voxel selection and classifier training phases, FCMA typically deals with only hundreds of samples (corresponding to the number of epochs of interest) but tens of thousands of dimensions (proportional to the squared number voxels). Therefore, we build models using linear SVM to reduce the risk of overfitting.

The offline FCMA implementation, batchFCMA, runs as a distributed program on a compute cluster. It uses different nodes to deal with different portions of brain voxels in parallel, which greatly reduces the memory and computation burden of a single node and accelerates the processing time. It takes seconds or minutes to analyze a typical fMRI dataset in terms of correlation in batchFCMA [10].

### B. Real-time FCMA

It is beneficial to incorporate FCMA into rtfMRI studies to understand subtle changes in functional interactions in the human brain during different cognitive tasks as we discussed in Section I. A closed-loop rtFCMA system can use a "pre-trained" model generated by the "pre-selected" voxels to classify the incoming data in terms of correlation on the fly. In this case, only the classification component of FCMA is needed in the pipeline, which is supposed to return the classification result as the neurofeedback in a short time. However, in order to better characterize functional interactions in the human brain, it is advantageous to leverage the data collected from the current scanning subject to asynchronously update the model and the corresponding "pre-selected" voxels. For example, the important voxels might change as the subject learns or vary across subjects. As a result, the computationally intensive voxel selection component of FCMA also needs to be applicable to the real-time experiments.

The rtFCMA system is a second-order machine learning system. Unlike most distributed machine learning systems, it does not directly take the incoming data to form samples for training and classification. Instead of using voxel activity directly, it uses the Pearson correlation between voxel activity, so it must compute the correlation based on the incoming data before applying any machine learning techniques. Using the correlation data as input for machine learning expands the data size by roughly three orders of magnitude. In order to meet the real-time requirement, high-performance computing techniques are critical in this scenario.

## III. rtFCMA algorithms

FCMA contains three components: voxel selection, model training and classification. In order to perform FCMA in the real-time environment efficiently, we modify the batchFCMA voxel selection algorithm processing

the fMRI data in batch to work in an incremental fashion. In the real-time setting, when a new brain volume comes in, we can leverage what we already have from previous processing to avoid unnecessary re-computation. The model training using the top-$K$ selected voxels, however, is not able to be processed incrementally because the top-$K$ voxels can change after a new round of voxel selection. Finally, the classification must be done quickly for providing the neurofeedback in time. From the algorithmic point of view, rtFCMA improves the voxel selection by implementing a full-stack incremental algorithm, and inherits the high-performance model training and classification algorithms from batchFCMA.

### A. Voxel Selection

Voxel selection represents the feature selection step in FCMA. It is a voxel-wise screening to pick the most category-selective voxels in terms of correlation. As the real-time stream of brain volumes comes in, we use the master-worker model to do voxel selection in a distribution fashion. Algorithm 1 describes the processing procedure.

**Algorithm 1** Voxel selection procedure

```
1:  The master allocates voxels to workers
2:  for each worker in parallel do
3:    for each assigned voxel do
4:      for each epoch do
5:        Compute the correlation vector (i.e., correlation with every other voxel in the brain)
6:      end for
7:      Normalize the correlation vectors within subject
8:      Compute the kernel matrix
9:      Compute cross-validated linear SVM accuracy
10:   end for
11: end for
12: The master chooses the top-K voxels by accuracy
```

In practice, each worker is assigned $V$ disjoint voxels by the master, so the correlation computation at Line 5 can be processed as a matrix multiplication. The normalization at Line 7 is to bring the activation values from different subjects to the same scale. The cross validation at Line 9 uses subjects as folds and trains a linear SVM classifier to test within fold for each voxel. That is, for $N$ subjects, we build $N$ linear SVM classifiers for each voxel using normalized correlation vectors of $N-1$ subjects as training data and 1 subject as test data. Moreover, in our SVM problems the number of samples (hundreds) is much smaller than the number of dimensions of a sample (tens of thousands), it saves a lot of computation and memory to precompute the kernel matrices before training (Line 8). Finally, the master collects all voxels' accuracy numbers, which are the indicators of their category selectivity, to form the voxel list sorted by cross-validated accuracy numbers in descending order (Line 12).

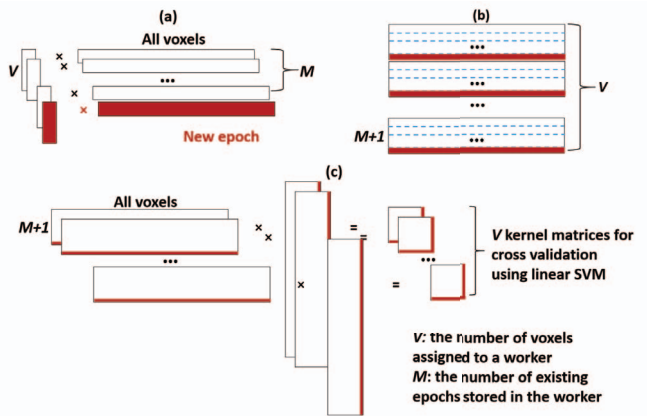


Figure 3. Incremental computation of voxel selection in one worker node. The red blocks indicate the incrementally computed data.

Voxel selection is the most computationally intensive component in FCMA. We have already optimized it in batchFCMA by merging processing pipeline stages and organizing the data layout to be cache and vectorization friendly [10]. rtFCMA inherits the master-worker model from the batchFCMA design. However, batchFCMA reads all data in at once to process together, which is not feasible in the real-time context, in which data come in volume by volume as a stream.

Figure 3 shows the processing that takes place in a worker node when the data of a new epoch of interest (red) is completely received, after $M$ epochs (white) have already been processed. The worker is assigned $V$ voxels by the master, with $V$ chosen so data can reside in memory. When a new voxel selection takes place after receiving epoch $M+1$, data from the first $M$ epochs is reused; the necessary operations are correlation computation over epoch $M+1$ (Figure 3a), normalization using the existing data of the same subject (Figure 3b), and computing the similarities with the existing correlation samples to obtain SVM kernel matrices (Figure 3c). This eliminates re-computation overhead and overlaps computation with data acquisition.

Using the subject as the random effect, fMRI analysis methods typically do leave-one-subject-out cross validation. In rtFCMA, we do not need to retrain SVM models for every fold from scratch; instead, we are able to leverage the historical models. Suppose the system has performed $N$-fold cross validation on $n_{old}$ samples. Now the data of the $(N+1)$th subject comes in to make the total number of samples increase to $n_{new}$. When performing the $(N+1)$-fold cross validation, in the first $N$ folds, which use the 1st to $N$th subjects' data as test data, the $(N+1)$th subject is added to expand the training set. In the last fold, which uses the $(N+1)$th subject's data as test data, the training set

can be expanded from any of the first $N$ training sets by adding in the missing subject. We apply the sequential minimal optimization (SMO) algorithm [11] to train an SVM model. The original SMO starts from a random set of states and runs until convergence. If the training set expands with new samples, SMO can start from the last converged states, which is likely to converge sooner than starting from a random set of states. Algorithm 2 shows how our incremental SMO works. The iteration starts from the last converged states (specifically, the weight vector $\boldsymbol{\alpha}$, the tracking vector $\boldsymbol{f}$, and the boundary values $b_{high}$ and $b_{low}$) and converges when the tolerance $\tau$ is reached again. Experiments in Section V show that the number of iterations (i.e., lines 4-8 in Algorithm 2) is significantly reduced compared to starting from a random set of states.

**Algorithm 2** Incremental SMO algorithm

**Input:** training data $x_i$, labels $y_i$ $\forall i \leq n_{new}$, convergence tolerance $\tau$, old state $f_i, \alpha_i$ $\forall i \leq n_{old} < n_{new}, b_{high}, b_{low}$

1: Initialize $\alpha_i = 0$ $\forall i \in \{n_{old}+1 \dots n_{new}\}$
2: Initialize $f_i = \Sigma_{j=1}^{n_{old}} \alpha_j y_j \phi(x_i, x_j) - y_i$ $\forall i \in \{n_{old}+1 \dots n_{new}\}$
3: Update $b_{high}$, $b_{low}$ based on new $\boldsymbol{f}$
4: **repeat**
5: Update $f_i$ $\forall i \leq n_{new}$ (eqn 4 in [12])
6: Compute: $b_{high}, b_{low}, i_{high}, i_{low}$ (eqn 5,6 in [12])
7: Update $\alpha_{i_{high}}$ and $\alpha_{i_{low}}$ (eqn 2,3 in [12])
8: **until** $b_{low} \leq b_{high} + 2\tau$

### B. Training with the Latest Data

Traditionally, researchers have used a fixed machine learning model trained from previously collected data [1] in rtfMRI studies (e.g., from the same subject earlier in the session) because of the difficulty of dynamically updating a model using the latest data in real-time. A fixed model does not reflect the neural status of the subject, which therefore limits neuroscientific studies.

In rtFCMA, we implement a training pipeline that incorporates the most recent data into the model. We first compute the correlation matrices of selected voxels consisting of correlations between the top-$K$ selected voxels produced by the latest voxel selection over all training epochs. Then we serialize the matrices into high dimensional samples, normalize them within subject, and train a linear SVM model over them. Note that since the top-K voxel list changes after a new round of voxel selection, we have to reconstruct the correlation matrices accordingly, which prevents reusing old training samples as in the cross validation stage of voxel selection.

In the real-time context, it is not feasible to enlarge the training set infinitely. Instead, we apply a moving window to the training data, i.e., when the number of training samples reaches a configurable threshold, we start to discard the oldest training sample while incorporating the newly coming ones to maintain the training set scale as illustrated in Figure 4.

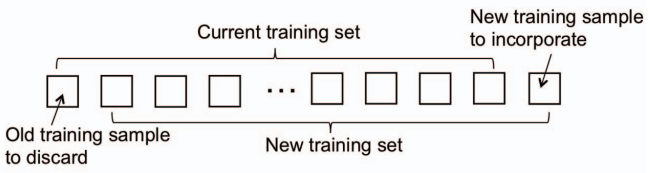


Figure 4. The sliding window of training samples.

### C. Real-time Classification

The classification produces a prediction for incoming data and returns it to the fMRI scanner. This must be done in real-time so that the experimental task can be updated depending on the prediction of the classifier. Normally, an rtfMRI study requires the neurofeedback from brain volume $B$ to be produced while volume $B+1$ is being generated so that it can guide the cognitive task the subject performs when volume $B+2$ is scanned.

Because classification is based on the temporal correlation, it is only performed after several brain volumes within the same epoch of interest are collected. We define a configurable window size $L$ to compute the correlation. After accumulating $L$ volumes within the same epoch of interest, we take the latest top-$K$ voxels, compute a correlation matrix, and apply it to the latest model as a test sample. In practice, $L$ is defined by the neuroscientists for producing reliable correlation values. If $L$ is less than the length of an epoch of interest, the window can be slid within epoch, i.e., each time a new brain volume is received, we drop the volume that was collected $L$ time points before and incorporate the new one to compute the correlation. When $L$ is equal to the length of an epoch, we only conduct one correlation-based classification per epoch.

## IV. System architecture

In this section, we present the design and implementation of a distributed system that supports rtFCMA and other rtfMRI analysis methods. The system is configurable so that various processes involved can be assigned to nodes at launch time, and the communication and system organization are independent of the machine learning algorithms used.

### A. Design

On receiving a brain volume, the system dispatches it to different processes for different purposes. Specifically for rtFCMA, the brain volumes go both to the classification process to get the prediction results for neurofeedback, and to the voxel selection processes to assign an accuracy to each voxel, as discussed in Section III. This behavior naturally fits the master-worker model, where

the master is in charge of receiving the brain volumes and dispatching them to different workers conducting different analyses. Specifically in rtFCMA, there are a classification worker and several voxel selection workers.

The classification worker is in the critical path of the closed-loop system, which must provide neurofeedback within a few hundred milliseconds. On the other hand, the actions of the voxel selection workers must take at most tens of seconds.

We depict the design of the rtFCMA system in Figure 5. The master node communicates with the front-end fMRI scanner and coordinates the workers. In a closed-loop rtFCMA experiment, the master hands the brain volumes from the fMRI scanner to the classification worker for classification. At the same time, the master also sends the brain volumes to a number of voxel selection workers for getting the latest voxel list so that it can update the model and send it to the classification worker asynchronously.

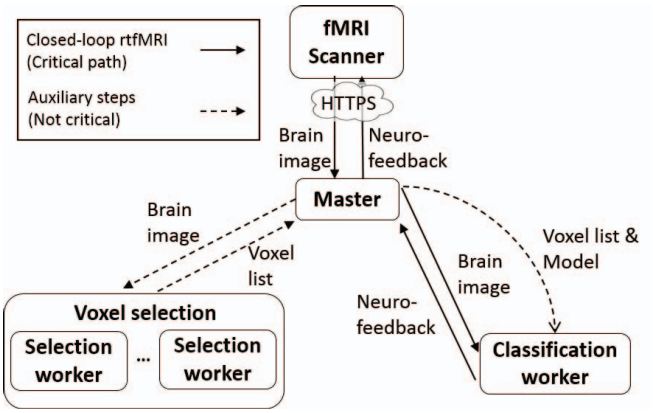


Figure 5. The rtFCMA system architecture.

### B. Implementation

The services rtFCMA provides are accessed using a RESTful web API via HTTPS. The REST server is implemented using the Spray framework [1]. It does relatively little processing itself, instead it relies on the other distributed processes in the system. There is a *reaper* that runs on the fMRI scanner's computer. Its job is to send brain volumes to the REST server (using HTTP POST requests) as they are generated by the scanner. The reaper is written in Python, and is borrowed from the SciTran project [13] with minimal modification.

We built our rtFCMA system using the actor model [14], as implemented in the Akka [2] toolkit, to distribute computation across a cluster. The master, voxel selection workers, and classification worker are all abstracted as *actors* in Akka. These actors are written in Scala and are lightweight and focused only on communication and system management. As a message that requires time to process arrives at one of the system's actors, it is validated and queued for processing by a different thread dedicated for that purpose, so that the actor is free to respond to the next message. Furthermore, the heavy analytical computation in the system is delegated via Java Native Interface wrappers to C++ routines (in this case, rtFCMA algorithms) that analyze the brain volumes. As a matter of design, the system is built to be extended to use many different algorithms, not only FCMA. Therefore, the actors communicate with the algorithm-specific code via algorithm-agnostic interfaces.

[1]http://spray.io
[2]http://akka.io

## V. Evaluation

To evaluate the performance of rtFCMA, we aim to answers to the following questions:

1) Can the classification produce neurofeedback in real-time?
2) Can the incremental voxel selection and training update the model efficiently?
3) What is the performance gain of each of the incremental algorithms?
4) Is rtFCMA classification accuracy comparable to batchFCMA?

### A. Experimental Setup

All testing was done on *Metacortex*, a 50-node cluster located at the Princeton Neuroscience Institute. All nodes are interconnected by an Arista 10 Gbit/s Ethernet switch. Each node has two Intel®Xeon®E5-2670 processors (2.6 GHz) and 256 GiB RAM. All our C++ code is multithreaded using OpenMP. We use 32 threads per process and one process per compute node. We used 40 nodes to do voxel selection, 1 node to do model updating and 1 node to do classification.

We use two datasets to simulate rtfMRI studies. The *face-scene* dataset was used as a proof of concept for FCMA [9]. It consists of 244 brain volumes per subject from 18 subjects. The volumes are grouped in 12 epochs per subject, each corresponding to the subject in the fMRI scanner being shown a series of either face or scene images. The interval between the starting points of two epochs is 30 s. The *attention* dataset was collected in [15]. It consists of 360 brain volumes per subject from 30 subjects. The volumes are grouped in 18 epochs per subject, each corresponding to the subject being asked to attend to images on either the left or right side of the screen. The interval between the starting points of two epochs in *attention* is 30 seconds. Note that in both datasets there is only one experimental run per subject. In most rtfMRI studies, each subject goes through multiple experimental runs, so in the simulation we treated the interval between subjects as the interval between experimental runs within the same subject.

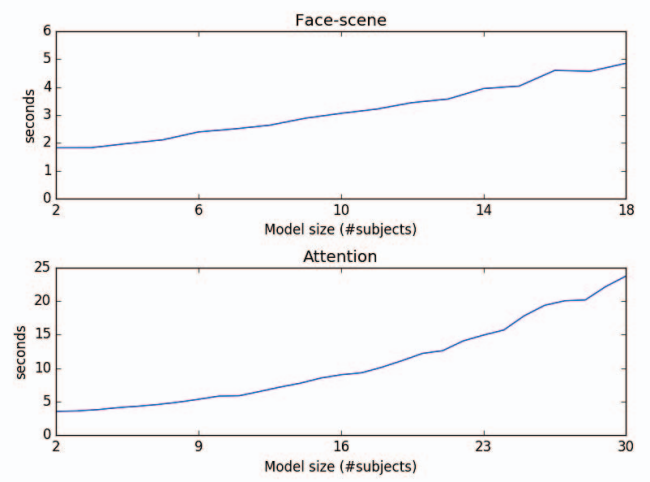


Figure 6. Duration of voxel selection and model update as a function of the model size using 40 voxel selection workers.

In order to fully simulate an rtfMRI experiment, we streamed the data to our system volume by volume with a 1.5 seconds pause, equivalent to one fMRI scanner TR. We also introduced a 1-minute pause between subjects, which is a reasonable duration in practice between experimental runs.

Finally, we used top-$K = 500$ voxels for neurofeedback since the number is enough to depict reasonable patterns of correlations [9], [15].

### B. Full System Performance

Our system created the SVM model for neurofeedback from scratch and updated it using the incoming data stream while the real-time experiment simulation was taking place. The system invoked the voxel selection workers at the end of each epoch to incrementally compute the correlation. At the end of each subject, the voxel selection workers updated the voxel list based on the voxels' accuracy numbers obtained from the leave-one-subject-out cross validation. Then a new model with the latest top-$K$ voxels was constructed.

Figure 6 shows the incremental voxel selection and model updating performance for both *face-scene* and *attention* datasets. As more subjects' volumes were collected, the model size grew larger and correspondingly the time it took to select voxels and update the model got longer. The voxel list and model were updated after every subject but the correlation data were computed after each epoch to overlap the computation with the data acquisition. From Figure 6 we can tell that the model updating was able to complete before the next subject came in. That is, the classification worker always used an up-to-date model that was updated by the data from the latest subject.

The classifier took the latest model along with the top-500 voxels in its corresponding voxel list to classify the correlation sample computed over a sliding window which included the latest brain volume. In our experiment, we set the sliding window size to be 8, i.e., after getting 8 TRs within the same epoch, the system started to compute correlation to classify, and when the 9th TR came in, the 1st TR was dropped to maintain the length of the time course over which the correlation was computed to be 8, so on and so forth until the end of the epoch.

Figure 7 shows the classification performance of our system. We used a dedicated node to run the classification worker to prevent any unexpected interference introduced by other components of the system. As we mentioned in Section I, although the time interval of generating a brain volume in our experiment is 1.5 s, the time budget for classification is less than that considering the latency of other necessary steps in the rtfMRI pipeline, e.g., reconstruction and preprocessing, which in practice took up to 780 ms. Therefore, we set a deadline of 720 ms to the classification. From the figure we can see that the classification in our system always finished faster than 180 ms, which is well bellow the deadline. This indicates that our system is able to satisfy the latency requirement of closed-loop rtfMRI studies, resulting from our high-performance classification algorithm, as well as our low latency distributed system.

Note that the times in Figure 7 do not include network communication overhead between the scanner and the REST server. Even if we consider cross-continental links, if we use a conservative RTT estimate of 200 ms [16] and assume the feedback system polls the REST server for results every 100 ms, the overhead should be upper bounded by 500 ms. Since the classification worker completed in less than 180 ms, the system meets the 720 ms deadline, let alone the fact that additional optimizations, such as keepalives and server side events could improve these numbers. That said, even if we occasionally miss the deadline because of noise in the network, this is likely acceptable in the context of real-time fMRI experiments because correlations are calculated over longer time windows and because classifier output is often smoothed or averaged over time to provide more continuous feedback to the subject.

### C. Overall Speedup of Incremental Voxel Selection

We compared the overall speedup we achieved using the incremental voxel selection algorithms. The voxel selection component of rtFCMA was implemented in a full-stack incremental fashion. It took advantage of the characteristics of the data stream by accumulating and processing the volumes when they were received and storing the historical results for further use. On the other hand, the offline algorithm in batchFCMA, although fully optimized, had to read in and process all the data and started from scratch every time, which introduced

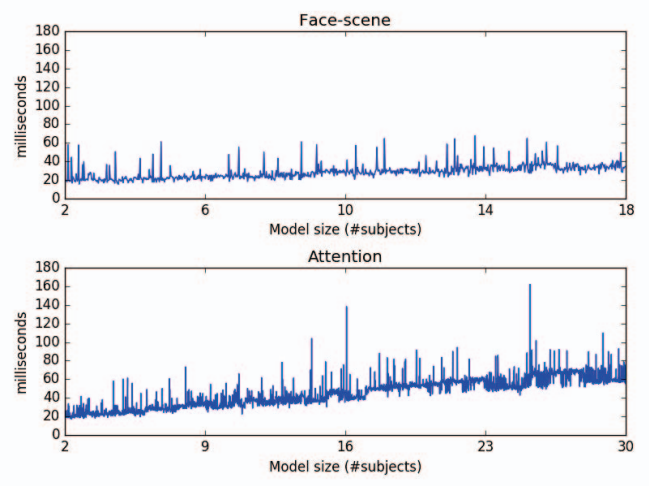


Figure 7. Duration of classification as a function of model size using 1 classification worker.

overhead in the real-time context. Table I summarizes the performance comparison between the voxel selection of rtFCMA and batchFCMA at the last subject of each dataset. Note that in the batch version, we excluded the time it took to read data from storage and broadcast them to all worker nodes. In total, the incremental voxel selection works 3.6–6.2× faster than the batch version at the end of these two datasets.

| Dataset | batchFCMA (s) | rtFCMA (s) | Speedup |
|---|---|---|---|
| **face-scene** | 14.8 | 2.4 | 6.2 |
| **attention** | 56.5 | 15.7 | 3.6 |

Table I
Overall speedup of incremental voxel selection.

### D. Performance Gain of Incremental Algorithms

We compared incremental algorithms and batch algorithms at the last subject of each dataset. We broke the voxel selection process into three stages: correlation computation and normalization, kernel matrix computation, and cross validation. We combined the correlation computation and normalization together because they took very little time individually in the incremental implementation. Table II shows the results attained on one out of 40 selection worker nodes.

| Dataset | Processing steps | Batch (s) | Incremental (s) |
|---|---|---|---|
| **face-scene** | corr comp & norm | 5.58 | 0.29 |
| | kernel matrix comp | 6.84 | 0.93 |
| | cross validation | 1.41 | 0.47 |
| **attention** | corr comp & norm | 8.74 | 0.21 |
| | kernel matrix comp | 21.04 | 2.68 |
| | cross validation | 25.77 | 10.02 |

Table II
Voxel selection time for the last subject of each dataset in one selection worker node.

The performance gain of using the incremental algorithms in the computation and normalization stages is straightforward since only the computation involving the newly coming data would be performed instead of doing everything (see the red parts indicated in Figure 3). On the other hand, incremental cross validation brings a smaller benefit, running 2.6–3× than batch cross validation. We explore this further in Table III, which shows the averaged iteration numbers to reach the converged state across voxels and folds of rtFCMA and batchFCMA at the last subject of both datasets. The incremental cross validation generally saves 80% and 64% of iterations in the two datasets to converge compared to the batch version.

| Dataset | batchFCMA | rtFCMA |
|---|---|---|
| **face-scene** | 1153 | 229 |
| **attention** | 7549 | 2697 |

Table III
Averaged iterations in cross validation for all SVMs trained. $n = 620\,460$ (face-scene), $n = 757\,800$ (attention).

### E. Effectiveness of RtFCMA System

Finally, to verify the effectiveness of our system, we compared the accuracy results it generated on both datasets with the results batchFCMA generated using leave-one-subject-out cross validation. In the rtFCMA pipeline, we sent the data brain volume by brain volume to the system, selected voxels and trained a model using $N-1$ subjects (where $N$ is the number of subjects in a dataset) and used this model to test on the last subject (making one prediction per epoch). We did this for $N$ times to allow all subjects to be tested. In order to match what batchFCMA did, in rtFCMA we set the correlation window size of the classification to be the length of the epoch, that is, only one classification per epoch, which was based on the correlation over the entire epoch. In batchFCMA, we replicated the published pipeline [9]. The top-500 voxels were used in both cases.

Figure 8 shows the averaged accuracy results of both datasets produced by rtFCMA and batchFCMA with standard errors across all predictions. 40 selection nodes were used. For both datasets, rtFCMA has similar results to batchFCMA, verifying the correctness of the rtFCMA system. The results were slightly different because in the correlation normalization steps, rtFCMA could only normalize the data based on what it has got so far, while batchFCMA normalized all data within the same subject.

## VI. Related work

In this section, we discuss the important related work on real-time fMRI, incremental feature selection, and distributed stream processing for machine learning.

Though modern rtfMRI systems are capable of producing full brain scans every 1-2 seconds, analysis of

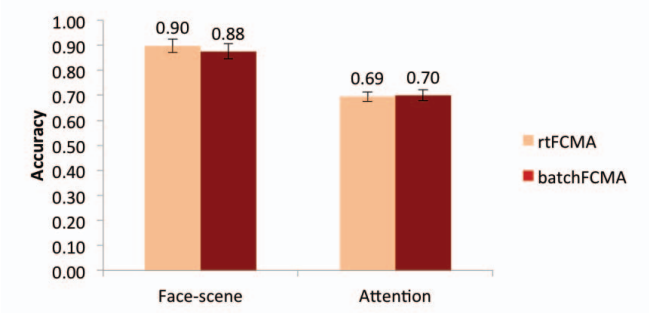


Figure 8. Accuracy for rtFCMA and batchFCMA. Error bars show standard errors. $n = 216$ (face-scene), $n = 540$ (attention).

these scans is typically performed offline. Incremental analysis of real-time functional magnetic resonance imaging is a very recent advancement, motivated in part by a surge in neurofeedback research [17]. A recent study trained subjects to better sustain their attention, by externalizing their internal attentional states using a classifier such that they could be better monitored and controlled [1]. The classifier was based on single-subject voxel activity patterns processed in batch on a single machine between experimental runs while the subject was in the scanner. rtFCMA extends the features considered by the classifier to full-brain correlation patterns ($O(n^2)$ features instead of $O(n)$), incrementally selects the voxels that are most predictive with less time, and uses the selected voxels to train classifiers to predict the incoming fMRI data from the correlation perspective.

The incremental voxel selection algorithm presented in this paper is, generally speaking, a form of feature selection. Feature selection is a common technique in machine learning, often used to increase model accuracy, improve generalization of the model, and to speed up model training. When the data come in as a stream like in the rtfMRI context, the feature selection is usually done in an incremental fashion [18]. rtFCMA uses data-driven feature selection based on pairwise voxel correlations over the entire full brain, which is more computationally challenging to implement incrementally than the classic techniques due to the large amount of memory and number of SVM models involved in the cross validation. The idea of leveraging shared training sets to do incremental learning in cross validation is not new [19], but our implementation shows the advantage of doing incremental learning in a massive parallelized environment, which can be generalized to other applications as well. As far as we know, our cluster instantiation of the pipeline is the most parallelized feature selection model ever applied to fMRI data and the first use of incremental learning on this data.

rtFCMA would not be possible without a parallel and distributed framework. Many frameworks for stream-based parallel and distributed machine learning exist. Apache Spark supports Spark Streaming as a means of joining stream data with historical data or performing window-based operations on the stream. It chops a data stream up into mini-batches to process. Recently, the Spark machine learning library, MLlib, was extended with streaming algorithms [20]. Spark's Resilient Distributed Dataset (RDD), an immutable collection that provides fault tolerance, is inefficient at dealing with small changes to large data structures (e.g., updates to a small number of vectors in the correlation matrix). Distributed stream processing engines (DSPEs), like Apache Storm [21], S4 [22], and Samza [23], have emerged that are designed from the ground up for stream processing data in a reactive manner, one update at time, but may not be feasible to our specific application. The Apache incubator project SAMOA (Scalable Advanced Massive Online Analysis) provides a machine learning framework and library that utilizes DSPEs but abstracts away the execution engine. Like some modern DSPEs and the original Apache Spark, rtFCMA used the open-ended, highly concurrent and resilient Akka runtime (Akka is largely based on and inspired by Erlang [24]). And like SAMOA we represented our algorithms with a directed graph of nodes that communicate via messages. To the best of our knowledge, rtFCMA is the first to implement any form of stream-based machine learning on fMRI data and the first streaming implementation of full correlation matrix-based classification for any application.

## VII. Conclusion

In this paper, we proposed a set of algorithms to exhaustively study functional interactions in the human brain in real-time. We also designed and implemented a distributed system to support rtFCMA, and potentially, other rtfMRI analysis applications. rtFCMA is able to update a full-correlation model in 25 seconds using incremental algorithms and classify brain volumes in less than 180 ms to keep pace with MRI scanners, without sacrificing the classification accuracy of offline approaches. rtFCMA can be deployed as a cloud service, which will allow neuroscientists around the world to conduct their own real-time experiments.

As ours is the first known effort to build such a system, there is a large amount of future work needed. Constructing additional real-time pipelines for other offline fMRI analysis methods is of great interest. Although those methods often seek to exploit different components of the signal (e.g., spatial patterns), it remains an open question how they compare to each other in terms of performance and effectiveness for feedback purposes. From a systems perspective, we will focus on making the system more robust by introducing fault tolerance and fine-grained resource allocation. We believe our effort will help accelerate the pace of discovery in fMRI-based neuroscience research.

## Acknowledgment

This publication was made possible through the support of grants from Intel Corporation, the J. Insley Blair Pyne fund at Princeton University, the John Templeton Foundation, and the National Science Foundation (MRI BCS1229597). The opinions expressed in this publication are those of the authors and do not necessarily reflect the views of the funding sources.

## References

[1] M. T. deBettencourt, J. D. Cohen, R. F. Lee, K. A. Norman, and N. B. Turk-Browne, "Closed-loop Training of Attention with Real-time Brain Imaging," *Nature Neuroscience*, vol. 18, no. 3, pp. 470–475, 2015.

[2] N. Weiskopf, K. Mathiak, S. W. Bock, F. Scharnowski, R. Veit, W. Grodd, R. Goebel, and N. Birbaumer, "Principles of a brain-computer interface (BCI) based on real-time functional magnetic resonance imaging (fMRI)," *Biomedical Engineering, IEEE Transactions on*, vol. 51, no. 6, pp. 966–970, 2004.

[3] R. Christopher deCharms, "Applications of real-time fMRI," *Nature Reviews Neuroscience*, vol. 9, no. 9, pp. 720–729, 2008.

[4] S. M. LaConte, "Decoding fMRI brain states in real-time," *Neuroimage*, vol. 56, no. 2, pp. 440–454, 2011.

[5] J. J. Yoo, O. Hinds, N. Ofen, T. W. Thompson, S. Whitfield-Gabrieli, C. Triantafyllou, and J. D. Gabrieli, "When the brain is prepared to learn: enhancing human learning using real-time fMRI," *Neuroimage*, vol. 59, no. 1, pp. 846–852, 2012.

[6] D. D. Leeds, J. A. Pyles, and M. J. Tarr, "Exploration of complex visual feature spaces for object perception," *Frontiers in computational neuroscience*, vol. 8, 2014.

[7] K. Shibata, T. Watanabe, Y. Sasaki, and M. Kawato, "Perceptual learning incepted by decoded fMRI neurofeedback without stimulus presentation," *science*, vol. 334, no. 6061, pp. 1413–1415, 2011.

[8] N. B. Turk-Browne, "Functional interactions as big data in the human brain," *Science*, vol. 342, no. 6158, pp. 580–584, 2013.

[9] Y. Wang, J. D. Cohen, K. Li, and N. B. Turk-Browne, "Full correlation matrix analysis (FCMA): An unbiased method for task-related functional connectivity," *Journal of neuroscience methods*, vol. 251, pp. 108–119, 2015.

[10] Y. Wang, M. J. Anderson, J. D. Cohen, A. Heinecke, K. Li, N. Satish, N. Sundaram, N. B. Turk-Browne, and T. L. Willke, "Full Correlation Matrix Analsis of fMRI Data on Intel® Xeon Phi™ Coprocessors," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2015.



[11] J. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," Microsoft Research, Tech. Rep. MSR-TR-98-14, Apr 1998.

[12] B. Catanzaro, N. Sundaram, and K. Keutzer, "Fast support vector machine training and classification on graphics processors," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 104–111.

[13] G. Schaefer *et al.*, "Scientific Transparency," https://github.com/scitran.

[14] C. Hewitt, P. Bishop, and R. Steiger, "A Universal Modular ACTOR Formalism for Artificial Intelligence," in *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, ser. IJCAI'73, 1973, pp. 235–245.

[15] J. Hutchinson, Y. Wang, and N. Turk-Browne, "Decoding the locus of attention from the full correlation matrix of the human brain," in *Society for Neuroscience*, ser. SfN '14, Nov 2014.

[16] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *Network and Systems Support for Games (NetGames), 2012 11th Annual Workshop on*, Nov 2012, pp. 1–6.

[17] J. Sulzer, S. Haller, F. Scharnowski, N. Weiskopf, N. Birbaumer, M. L. Blefari, A. Bruehl, L. Cohen, R. Gassert, R. Goebel *et al.*, "Real-time fMRI neurofeedback: progress and challenges," *NeuroImage*, vol. 76, pp. 386–399, 2013.

[18] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Dynamic feature space and incremental feature selection for the classification of textual data streams," in *in ECML/PKDD-2006 International Workshop on Knowledge Discovery from Data Streams. 2006.* Springer Verlag, 2006, p. 107.

[19] P. Joulani, A. György, and C. Szepesvári, "Fast cross-validation for incremental learning," in *Proceedings of the 24th International Conference on Artificial Intelligence.* AAAI Press, 2015, pp. 3597–3604.

[20] X. M. et al., "Mllib: Machine learning in apache spark," *Journal of Machine Learning Research*, vol. 17, no. 34, pp. 1–7, 2016.

[21] "Apache storm," http://storm.apache.org.

[22] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, "S4: Distributed stream computing platform," in *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on.* IEEE, 2010, pp. 170–177.

[23] "Apache Samza," http://samza.apache.org.

[24] J. Armstrong, R. Virding, C. Wikström, and M. Williams, *Concurrent Programming in ERLANG.* Prentice Hall, 1993.