Full correlation matrix analysis of fMRI data

Yida Wang* [1], Jonathan D. Cohen[2,3], Kai Li[1], Nicholas B. Turk-Browne[2,3]

[1]Department of Computer Science, Princeton University
[2]Department of Psychology, Princeton University
[3]Princeton Neuroscience Institute, Princeton University

Abstract

Functional brain imaging produces huge amounts of data, of which only a fraction are analyzed. Existing univariate and multivariate analyses of brain activity ignore interactions between regions, and analyses of interactions (functional connectivity) are typically biased toward regions of interest chosen based on their activity profile. This technical report provides a provisional description of an unbiased technique for functional connectivity, full correlation matrix analysis (FCMA). This technique calculates and analyzes all pairwise relationships between voxels over multiple time windows by leveraging advances in parallel computing and machine learning. FCMA enables the identification of neural mechanisms that support cognitive processes but may be invisible to activity-based methods.

I. Introduction

Functional magnetic resonance imaging (fMRI) studies often seek to associate regions of neural activity with specific cognitive processes[1, 2]. However, this univariate, sometimes phrenological approach has been challenged by multivariate pattern analysis (MVPA) methods[3, 4]. Findings from MVPA have been offered as evidence that representations in the brain, rather than being strictly localized, are distributed over multiple regions. However, the neural mechanisms supporting cognitive processing may be even more widely distributed than previously thought. The failure to recognize this reflects biases in existing analysis methods.

First, most applications of MVPA have focused on neural *activity*. Although activity can reveal important information about neural representations, it may fail to reveal interactions between brain regions that support neural processes. That is, even if neurons show identical mean firing rates across behavioral tasks, when and how they interact with each other can distinguish between tasks[5]. For example, consider a brain region that is equally active for two conditions, but interacts differentially with other regions depending upon the condition; such a region would not be identified by analyses that contrast these conditions or classify their patterns of activity. However, insofar as this region's interactions with other regions differ according to condition, it should show condition-specific patterns of *correlations*.

Second, although the analysis of correlations in fMRI data, or functional connectivity[6, 7], has become prevalent, this approach has other limitations. Such analyses typically involve first identifying a small set of "seed" regions that show task-related activity, and then examining correlations between these seeds and the rest of the brain. However, by choosing seeds based on activity, this procedure is saddled with the limitations of activity-based methods, and may fail to identify the region in the example above (since its activity is not selective). Thus, correlations may be needed at the outset to identify regions that exhibit task-related in-

*Please address correspondence to: yidawang@cs.princeton.edu

teractions. Furthermore, the use of seeds itself, regardless of how they are chosen, is subject to the limitation of focusing on localized rather than distributed processes.

Here we describe initial technical details of a new method, full correlation matrix analysis (FCMA), that surmounts these limitations by performing unbiased multivariate analyses of whole-brain functional connectivity. The currency of this technique is the full correlation matrix: the temporal correlation in fMRI activity of every voxel in the brain with every other voxel. A separate matrix is computed for each time epoch of interest in a task, such as trial or block, just as would be done for mean activity or activity patterns. These matrices are then labeled with the task condition for that epoch, and they are submitted to MVPA. Critically, the input is now correlation patterns, rather than activity patterns. This analysis determines, in an exhaustive manner, which patterns of correlations distinguish between conditions.

This analysis may sound straightforward, but it is intractable using a naive approach such as calculating the Pearson correlation pair-by-pair, storing these values, and then analyzing the full set of correlations. Consider a relatively small fMRI dataset with 18 subjects, 12 epochs per subject over two experimental conditions, and 34,470 voxels in the brain. On a machine with 2 4-core 2.6 Ghz Xeon CPUs, running Matlab *corr* function in batch mode, the computation of all 594 million pairwise correlations for all epochs above takes 2.5 hours and requires at least 475.2 GB of disk space at the end (and much more memory at intermediate stages). Reducing the Pearson correlation to matrix multiplication and rewriting the computation in C++ using highly tuned linear algebra packages shortens the total running time to 348 seconds on the same machine. Thus, the correlation computation alone is not an impediment.

The real problem arises when this massive amount of data needs to be analyzed. To make sense of the data, machine learning techniques such as MVPA are needed. The correlations would first need to be preprocessed, including normalizing each coefficient with the Fisher transform and then z-scoring all coefficients across the matrix within subject. To classify two conditions, we would divide these correlation matrices into training and test sets, such as by leaving out one subject at a time for random-effects cross-validation. In the example above, the classifier would be trained on 204 matrices to find a boundary separating the conditions in a 594 million dimensional hyperspace and tested on the remaining 12 matrices to obtain a classification accuracy. Using nested feature selection for data-driven dimensionality reduction, training and testing a basic linear classifier on the correlation matrices in C++ takes 35.91 days on the same machine described above. For all 18 subjects, this process must repeated 17 more times for a total of 646.31 days.

To address these challenges, we have been developing a software toolbox for FCMA that exploits advances in parallel computing and machine learning. Using this toolbox on a cluster with 66 of the aforementioned machines, we show that the time required for the analysis outlined above is reduced from two years to about an hour.

II. FCMA Toolbox

Our toolbox exploits hardware and software advances to render FCMA tractable, all in a user environment that is readily accessible to neuroscientists. It does so by incorporating efficient algorithms for correlation computation, massive parallelization to manage the scale of the data, and on-demand analysis without the need for storage space or time. The toolbox runs on a compute cluster with any number of nodes, uses controller-worker architecture, and takes standard NIFTI format images as its input. A dataset consisting of fMRI activity over time in a set of voxels is divided in multiple time epochs and is stored on a disk accessible to each node. The (relatively small) data are copied into the memory of each worker and the controller dynamically allocates a subset of voxels whenever a node becomes available. A vector is computed for every epoch of each voxel using matrix multiplication, reflecting

its correlation with every other voxel in that epoch. To handle the high dimensionality without imposing biases, these epoch vectors are submitted to MVPA and the resulting cross-validation accuracy is assigned to the voxel. This can be used for nested feature selection, leading to a final round of MVPA over the correlation matrices of the automatically selected voxels.

Optimization of correlation computation. The Pearson correlation between several pairs of variables can be reduced to a matrix multiplication by normalizing the data[8]. Specifically, the timeseries for a given voxel and epoch is normalized by subtracting the mean and then dividing each value by the root sum of squares of the mean-centered data. These are O(n) operations and thus computation time scales linearly and poses little burden. The derivation of how this normalization turns the Pearson correlation for a set of variables into matrix multiplication is provided in the Appendix.

Matrix multiplication is significantly faster than other approaches for computing correlations. In addition, it can benefit from generally available technological advances in modern CPUs, such as the single instruction, multiple data (SIMD) set. The toolbox implements advanced linear algebra algorithms from the GotoBlas library[9] to exploit this hardware. At peak performance, using SSE instructions over 128-bit XMM registers, these algorithms allow for eight single-precision floating point operations (four additions plus four multiplications) in one CPU cycle.

Parallelization of correlation computation. The FCMA toolbox was written in C++ to run on a compute cluster with modern commercial machines and X86 architecture, each with reasonable memory size (e.g., 16 GB). Because of this, it can leverage large-scale computing techniques to further accelerate computation and analysis. A controller/worker model is used (Fig. 1), in which a controller process coordinates numerous worker processes running on multiple machines with the Message Passing Interface (MPI). The controller allocates computation and analysis tasks to the workers; typically, one process is assigned to each node in order to fully utilize its resources. Each process consists of multiple threads to compute and analyze multiple voxels simultaneously within one node.

The first step in the toolbox is to read in preprocessed fMRI data (e.g., corrected for head motion and other sources of noise), as well as text files specifying the experimental design. Based on this design, the data are partitioned into epochs. For every voxel and epoch, the data are normalized as described above. The controller then directs all available worker processes to load the full data from the storage device into memory and dynamically assigns each a subset of voxels to analyze. In other words, the full correlation matrix is automatically divided into groups of rows, and they are spread across worker processes. When analysis in a worker finishes, the controller collects the results, stores them in memory, and assigns another group of voxels.

By distributing the full correlation computation in this way over a 66-node cluster, and by using data normalization and optimized matrix multiplication algorithms in GotoBlas, total correlation computation time for our example dataset is reduced from 2.5 hours to 0.73 seconds.

Parallelization of voxelwise classifier analysis. To avoid the burden of storing full correlation matrices to disk (and associated write/read time), analysis is performed online within the nodes, immediately after correlation computation. Specifically, after a worker process has computed a correlation matrix of the assigned subset of voxels with the rest of the voxels for each epoch, the same row of all matrices is extracted. Each of these rows comprises a vector of the correlations between a given voxel and all other voxels in the brain for one epoch. These correlation vectors are then labeled with the condition of the experimental design to which the epoch was assigned, and submitted to MVPA as training (or test) patterns. Each vector reflects a point in a high-dimensional space, and the goal of MVPA is to determine how accurately the points with

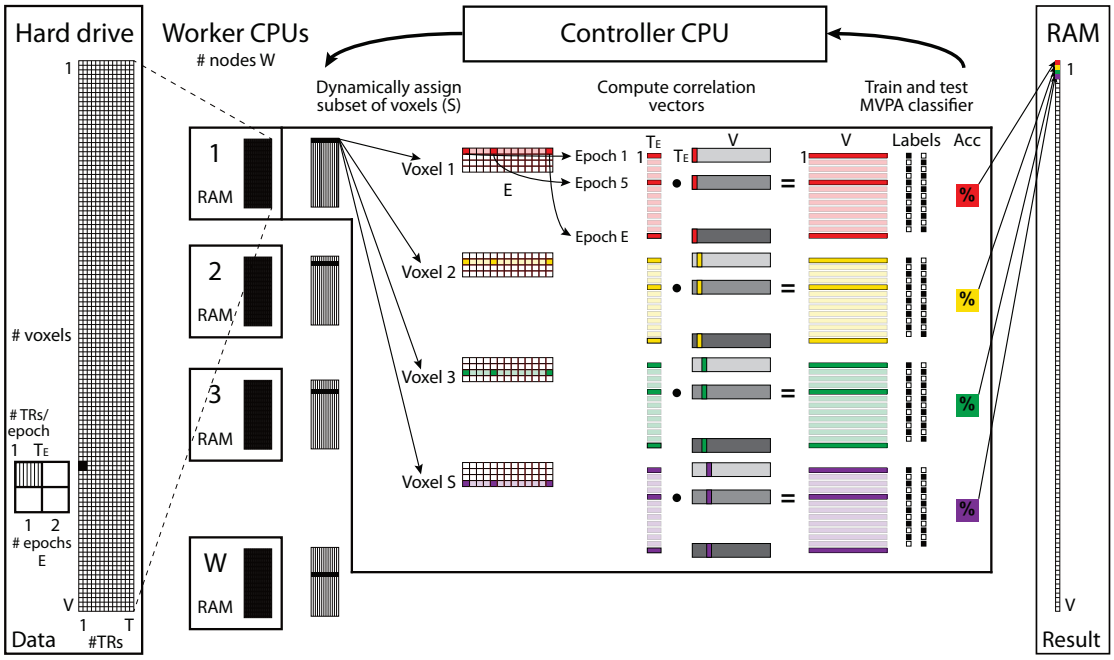

Figure 1: *FCMA correlation computation workflow. The toolbox runs using a controller/worker model, in which each worker process first loads the full data into its memory. The controller process does the following: assigns a subset of voxels to each worker; instructs the worker to compute the correlation between each of these voxels and the rest of the brain in each epoch using matrix multiplication; instructs the worker to analyze the correlation vectors for each voxel across epochs with MVPA and supplied condition labels; collects the analysis result (i.e., cross-validation accuracy) for each voxel and loads it into memory; and returns to the first step to assign another subset of voxels until there are none left. Finally, the controller writes the results to disk.*

different labels can be separated. MVPA is run with multiple threads, with each thread processing the correlation vectors across different epochs for one voxel.

The FCMA toolbox uses a linear Support Vector Machine (SVM) classifier based on LibSVM[10]. On a server with 2 4-core 2.6Ghz Xeon CPUs, the standard single-threaded implementation of LibSVM takes about 90 seconds to perform cross-validation on the 204 correlation vectors from one voxel in our example dataset. To speed up this process, we accelerated the original LibSVM algorithm by pre-computing the linear kernel matrices with GotoBlas. This reduced the running time to 2 seconds per voxel in a single thread.

MVPA of correlation patterns. Classification of the correlation vectors results in a cross-validation accuracy for each voxel, in terms of how informative its patterns of correlations with the rest of the brain are about the task conditions (Fig. 2). FCMA can stop here, providing an unbiased, whole-brain "map" of the extent to which different brain regions exhibit task-related changes in functional connectivity. This map is, for voxelwise correlations, analogous to the map generated by a voxelwise (univariate) GLM analysis of brain activity. By extension, just as MVPA can identify patterns of activity across voxels that differentiate between task conditions, so too can it be applied to identify patterns of correlation that differentiate between conditions.

The results of the voxelwise classifier analysis described above can also be used as feature selection for MVPA, by ranking voxels according to the accuracy with which their individual patterns of correlation differentiate condi-

tions and then selecting a subset that exceeds some threshold (e.g., percentage of voxels, absolute accuracy level, statistical significance over cross-validation folds, etc.) for the final round of MVPA. The FCMA toolbox implements this step as well: A chosen subset of voxels can be submitted to the toolbox again to obtain and analyze with MVPA the correlation matrix for all of these voxels. Critically, this requires that the original data be partitioned into training and testing sets, with the feature selection step only applied to the training set and the overall classification accuracy derived from the held-out testing set. The training and test sets are cycled to obtain overall cross-validation performance. Using the FCMA toolbox on our cluster, the complete procedure was reduced from 646 days to 72.3 mins.

MVPA of activity patterns. For comparison purposes, the toolbox also incorporates standard MVPA on the pattern of averaged activity over time for every voxel. This uses exactly the same process as for correlation patterns (Fig. 2), except taking activity instead of correlation as inputs, and generates an unbiased, whole-brain "map" of the extent to which different brain regions exhibit task-related changes in averaged activity over blocks. This is the same approach as in the standard MVPA toolbox, but we include it in FCMA toolbox to allow easy comparisons.

Analysis benchmarks. In summary, the FCMA toolbox accelerates the analysis of unbiased functional connectivity by optimizing correlation and machine learning algorithms and by leveraging multi-thread parallelism within nodes and multi-node parallelism across nodes. The speedups we achieved with a test dataset are listed in Table 1.

III. Discussion

To the best of our knowledge, FCMA is the first toolbox in the neuroscience community that takes advantage of high-performance computing to efficiently analyze the full correlation matrix of fMRI data. FCMA is extremely powerful because it deals gracefully with large amounts data, optimally splitting and scheduling problems based on the latest techniques in parallel computing and minimizing the need for slow data transfer by managing memory intelligently during online analysis. Beyond flexible parallelization, FCMA accelerates discovery by improving standard algorithms for correlation and classification. FCMA can be run on a compute cluster with any number of nodes, limited only by the computation resources of the hardware.

For performance benchmarking, we used a compute cluster consisting of 66 nodes. Each node was equipped with two Intel Xeon E5430 processors, 16 GB memory, and 4 TB local disk, and could run 8 threads simultaneously at peak. The toolbox has been ported to other systems as well, including a 128-node cluster at the Princeton Institute for Computational Science and Engineering (PICSciE). Construction

Additional analysis component	Elapsed time in minutes (days)	Speedup (x Baseline)
Baseline	930686 (646.31)	1.0
Improved linear SVM	21152 (14.69)	44.0
Multi-core parallelism (8 cores)	3620 (2.51)	257.1
Multi-node parallelism (66 nodes)	72 (0.05)	12855.0

Table 1: *The speedups FCMA achieved using our 66-node cluster by running the example application of 34,470 voxels and 216 epochs. The baseline code was written in C++ using Gotoblas library and LibSVM in single-thread mode. Improved linear SVM code pre-computed the linear kernel matrices used in LibSVM. Multi-core parallelism code used OpenMP to launch 8 shared memory threads in one cluster node to run the program in parallel. Multi-node parallelism code runs MPI to coordinate 66 cluster nodes to work together in controller-worker mode. Long elapsed times were estimated by extrapolating from a portion of the dataset.*

is underway of a dedicated 50-node cluster for FCMA, featuring two Intel E5-2670 processors and two Intel Xeon Phi boards per node (total of 800 processor cores and 6,000 coprocessor cores).

Though applied to voxel pairs in fMRI, this technique could help uncover relationships between variables across a range of applications[11]. Even within cognitive neuroscience, our case study involved the autocorrelation of one dataset, but two or more datasets (e.g., from different phase offsets, regions, or even brains) could be submitted without modification of the toolbox for cross-correlation. By making all code publicly available, we hope that researchers will explore these exciting avenues (`http://princetonuniversity.github.io/fcma-toolbox`).

In summary, our toolbox enables the possibility to examine interactions between brain areas in an unbiased, exhaustive manner. Examining such interactions can reveal the engagement of neural systems transparent to traditional, activity-based analysis methods.

IV. Appendix

The correlations among several voxel timecourses were computed via matrix multiplication. The mean activity was first subtracted from each timecourse, and this mean-centered timecourse was then divided by its root sum of squares. The Pearson correlation between two of the resulting normalized timecourses is their pointwise product, which for an arbitrary number of timecourses becomes a dot product[9]. A derivation of this is provided below:

The formula for the population Pearson product moment correlation is:

$$corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{1}$$

This can be re-written based on estimates of covariance and standard deviations:

$$\begin{aligned} corr(X,Y) &= \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{(n-1)\sqrt{\frac{\sum_{j=1}^{n}x_j^2-n\bar{x}}{n-1}}\sqrt{\frac{\sum_{j=1}^{n}y_j^2-n\bar{y}}{n-1}}} \\ &= \sum_{i=1}^{n}\left(\frac{(x_i-\bar{x})}{\sqrt{\sum_{j=1}^{n}x_j^2-n\bar{x}}}\frac{(y_i-\bar{y})}{\sqrt{\sum_{j=1}^{n}y_j^2-n\bar{y}}}\right) \end{aligned} \tag{2}$$

Applying the mean subtraction and root sum-of-squares division, let:

$$x_i' = \frac{(x_i-\bar{x})}{\sqrt{\sum_{j=1}^{n}x_j^2-n\bar{x}^2}} \tag{3}$$

The formula above then becomes:

$$corr(X,Y) = X' \cdot Y' \tag{4}$$

This can be expanded from vectors to matrices:

$$X' = (x_1', x_2', \cdots, x_n') \tag{5}$$

and

$$A = (X_1', X_2', \cdots, X_n')^T \tag{6}$$

Then the correlation matrix, where every voxel is represented by a row and column, is given by:

$$C = AA^T \tag{7}$$

V. Acknowledgements

This work was made possible by grants from the J. Insley Blair Pyne fund at Princeton University, the John Templeton Foundation, the National Science Foundation MRI BCS1229597, and the National Institutes of Health R01 EY021755. Intel and Fusion-io provided significant in-kind donations of computer hardware. The opinions expressed in this publication are those of the authors and do not necessarily reflect the views of these agencies or corporations. Code available at: `http://princetonuniversity.github.io/fcma-toolbox`.

References

[1] Kanwisher, N.G. Functional specificity in the human brain: A window into the functional architecture of the mind. *Proc. Natl. Acad. Sci. U.S.A.*, **107**, 11163-11170 (2010).

[2] Cabeza, R & Nyberg, L. Imaging cognition II: An empirical review of 275 PET and fMRI studies. *Journal of cognitive neuroscience*, **12**, 1-47 (2000).

[3] Norman, K.A. Polyn, S.M. Detre, G. & Haxby, J.V. Beyond mind-reading: Multi-voxel pattern analysis of fMRI data. *Trends Cogn. Sci.*, **10**, 424-430 (2006).

[4] Kriegeskorte N., Mur M. & Bandettini P. Representational similarity analysis: Connecting the branches of systems neuroscience. *Front Syst. Neurosci.*, **2**, 1-28 (2008).

[5] Vaadia, E. *et al*. Dynamics of neuronal interactions in monkey cortex in relation to behavioural events. *Nature*, **373**, 515-518 (1995).

[6] Fox, M.D. & Raichle, M.E. Spontaneous fluctuations in brain activity observed with functional magnetic resonance imaging. *Nat. Rev. Neurosci.*, **8**, 700-711 (2007).

[7] Smith S.M. The future of FMRI connectivity. *Neuroimage*, **62**, 1257-1266 (2012).

[8] Worsley, K.J. Chen, J-I. Lerch, J. & Evans, A.C. Comparing functional connectivity via thresholding correlations and singular value decomposition. *Philos. Trans. R. Soc. London Ser. B Biol. Sci.*, **360**, 913-920 (2005).

[9] Goto, K. & van de Geijn, R.A. Anatomy of high-performance matrix multiplication. *ACM Trans. Math. Soft.*, **34**, 1-25 (2008). `http://www.tacc.utexas.edu/tacc-software/gotoblas2`

[10] Chang, C.C. & Lin, C.J. LIBSVM: A library for support vector machines. *ACM Trans. Intel. Sys. Tech.*, **2**, 1-27 (2011). `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

[11] Reshef, D.N. *et al*. Detecting novel associations in large data sets. *Science*, **334**, 1518-1524 (2011).

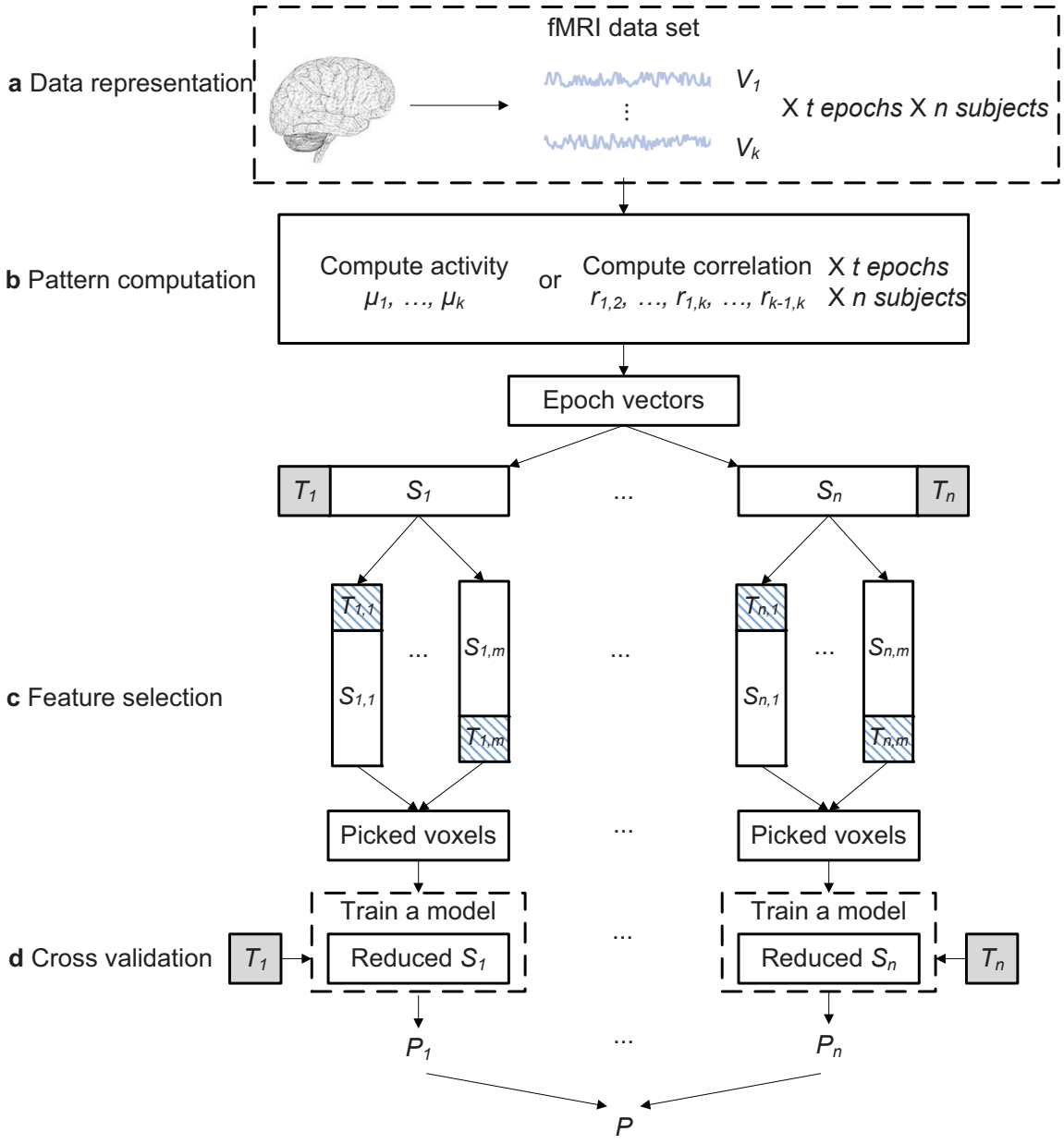

Figure 2: *Classification procedure.* ***(a)*** *The preprocessed fMRI data set contains* n *subjects, each represented with a* k *voxels by* t *epochs matrix.* ***(b)*** *For standard MVPA of activity patterns, vectors are defined for each subject and epoch as the average fMRI signal over time in every voxel (μ_i). For MVPA of correlation patterns, vectors are defined for each subject and epoch as the pairwise correlation of the fMRI signal over time between every voxel and every other voxel (r_{ij}).* ***(c)*** *The same nested cross-validation pipeline can be applied to activity and correlation patterns. The inner loop serves to select features (voxels) for classification: A training set (S_i) is divided into* m *pieces to do an* m*-fold cross-validation that identifies the* l *(e.g., 2000) voxels with highest performance.* ***(d)*** *The outer loop is* n*-fold, with each fold leveraging the selected voxels to train a model on S_i and test it on the left-out test set (T_i). This results in a classification accuracy (P_i), which is then averaged across folds (*P*) to quantify overall performance. In addition, a map of the proportion of the folds in which a voxel was feature selected can be output as an image.*