ReAl-LiFE: Accelerating the Discovery of Individualized Brain Connectomes on GPUs

Sawan Kumar,[*1] **Varsha Sreenivasan,**[*2] **Partha Talukdar,**[1,3]
Franco Pestilli,[4] **Devarajan Sridharan**[2,3]
[1]Computational and Data Sciences, Indian Institute of Science, Bangalore, India,
[2]Centre for Neuroscience, Indian Institute of Science, Bangalore, India,
[3]Computer Science and Automation, Indian Institute of Science, Bangalore, India,
[4]Psychological and Brain Sciences, Indiana University, Bloomington, USA
[*]contributed equally
{sawankumar,varshas,sridhar}@iisc.ac.in

Abstract

Diffusion imaging and tractography enable mapping structural connections in the human brain, *in-vivo*. Linear Fascicle Evaluation (LiFE) is a state-of-the-art approach for pruning spurious connections in the estimated structural connectome, by optimizing its fit to the measured diffusion data. Yet, LiFE imposes heavy demands on computing time, precluding its use in analyses of large connectome databases. Here, we introduce a GPU-based implementation of LiFE that achieves 50-100x speedups over conventional CPU-based implementations for connectome sizes of up to several million fibers. Briefly, the algorithm accelerates generalized matrix multiplications on a compressed tensor through efficient GPU kernels, while ensuring favorable memory access patterns. Leveraging these speedups, we advance LiFE's algorithm by imposing a regularization constraint on estimated fiber weights during connectome pruning. Our regularized, accelerated, LiFE algorithm ("ReAl-LiFE") estimates sparser connectomes that also provide more accurate fits to the underlying diffusion signal. We demonstrate the utility of our approach by classifying pathological signatures of structural connectivity in patients with Alzheimer's Disease (AD). We estimated million fiber whole-brain connectomes, followed by pruning with ReAl-LiFE, for 90 individuals (45 AD patients and 45 healthy controls). Linear classifiers, based on support vector machines, achieved over 80% accuracy in classifying AD patients from healthy controls based on their ReAl-LiFE pruned structural connectomes alone. Moreover, classification based on the ReAl-LiFE pruned connectome outperformed both the unpruned connectome, as well as the LiFE pruned connectome, in terms of accuracy. We propose our GPU-accelerated approach as a widely relevant tool for non-negative least squares optimization, across many domains.

1 Introduction

Mapping anatomical connections between brain regions is essential for understanding how the brain produces behavior. At present, diffusion-weighted magnetic resonance imaging (dMRI) and tractography are among the only techniques for estimating structural connectivity in the human brain *in vivo*. While dMRI measures the diffusion of water molecules through the brain's white matter, tractography is a computational tool that involves tracing contiguous white matter streamlines, utilizing local information about fiber orientation, to reconstruct white matter connections in three dimensions. The non-invasive nature and ability to acquire dMRI scans at high spatial resolutions, with relatively short scan times, has led to the rising popularity of this approach for measuring individual-specific brain connectomes in large databanks, comprising healthy subjects (Van Essen et al. 2012; Sudlow et al. 2015) and patient populations (Mueller et al. 2005).

A key challenge with connectome generation algorithms is the lack of access to ground truth: structural connectome estimates can differ significantly depending on the experimenter's choice of algorithmic parameters (e.g. minimum radius of curvature or maximum fiber length). Post-hoc evaluation (or pruning) of whole-brain connectomes has, therefore, become an essential post-processing step in measuring structural connectivity in the human brain. Typically, connectome evaluation is achieved by estimating a large number of structural connections, including potentially redundant connections, and pruning these down to a subset of connections that best fit the measured diffusion data (Pestilli et al. 2014; Smith et al. 2013; Smith et al. 2015).

One popular algorithm for evaluating connectomes is Linear Fascicle Evaluation (LiFE) (Caiafa et al. 2017; Pestilli et al. 2014). LiFE models the diffusion signal in each voxel as arising from a weighted contribution of all fibers traversing that voxel. Estimating the streamline weights is achieved by minimizing the error between the actual and the connectome-predicted diffusion signal, subject to a non-negativity constraint on the weights. Streamlines with non-zero weights constitute the pruned connectome.

In its original formulation, the LiFE algorithm suffered from constraints on memory as well as execution time, which limited its use with large-scale connectome data (Pestilli et al. 2014). These memory constraints have been recently addressed by encoding brain connectomes in multidimensional arrays (Caiafa and Pestilli 2017; Caiafa et al. 2017). Despite its more efficient memory management, LiFE converges very slowly on desktop CPUs. Convergence, even for a single subject's connectome (1 million stream-

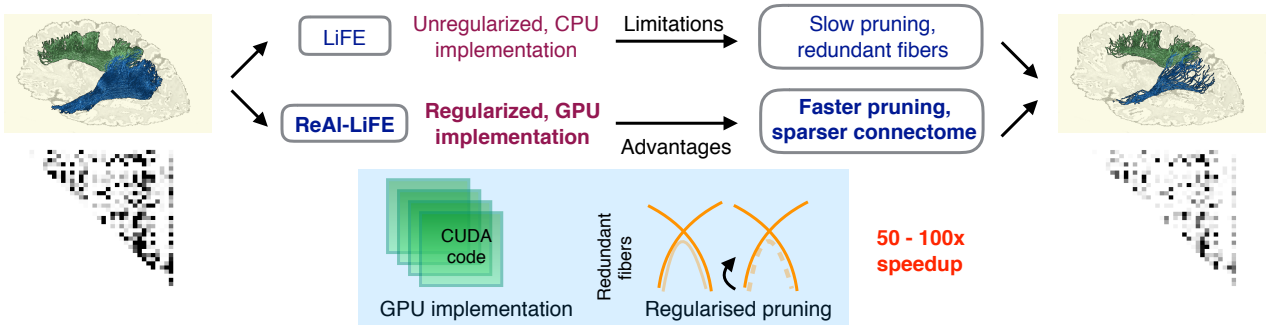

Figure 1: *Schematic illustrating ReAl-LiFE.* (Left) dMRI and tractography enable estimation of white-matter connections in the brain. The LiFE algorithm (top row) prunes this connectome to fit the underlying diffusion data and assigns a weight to each fiber. LiFE suffers from limitations of long CPU execution times and retains redundant fibers. Our GPU-based implementation, ReAl-LiFE (bottom row), generates sparser, more accurate connectomes with 50-100x speedups.

lines), typically requires ~500 iterations of the algorithm, and takes up to ~17 hours. This slow speed precludes LiFE's extensive use for individualized connectome discovery in large databases with thousands of subjects (Van Essen et al. 2012; Sudlow et al. 2015).

Here, we develop a regularized, accelerated version of LiFE's connectome pruning algorithm (ReAl-LiFE), implemented on GPUs. We demonstrate that ReAl-LiFE provides significant speedups (~100x) compared to the original CPU version, with increased model accuracy for fitting the underlying diffusion signal (Fig. 1). Next, we demonstrate a key real-world application of ReAl-LiFE, by predicting pathological connectivity in a large database of patients with Alzheimer's Disease (AD). We also provide an open source implementation of ReAl-LiFE[1], along with software and hardware details needed for reproducing the results in this paper, to readily enable its application to other datasets.

2 Limitations of the LiFE Algorithm

To understand critical bottlenecks associated with connectome pruning we introduce, briefly, the LiFE algorithm. LiFE's connectome pruning algorithm models a predicted diffusion signal from the estimated connectome, and eliminates fibers, to minimize the discrepancy between the modeled and measured diffusion signal. The diffusion signal, typically measured along multiple (around 20-100) different gradient directions ($\mathbf{N}_\theta$), is encoded in a vector $\mathbf{b} \in \mathbb{R}^{\mathbf{N}_\theta \mathbf{N}_\mathbf{v}}$, $\mathbf{N}_\mathbf{v}$ being the number of voxels. Standard probabilistic tractography algorithms permit generating a whole-brain connectome based on this diffusion signal by tracking individual streamline fibers (Tournier et al. 2012). Each streamline fiber in the connectome traverses multiple voxels and each voxel is traversed by many fibers. The contribution to the modeled diffusion signal of each fiber f, traversing a voxel v, along a measured gradient direction θ is encoded in a matrix $\mathbf{M} \in \mathbb{R}^{\mathbf{N}_\theta \mathbf{N}_\mathbf{v} \times \mathbf{N}_\mathbf{f}}$, where $\mathbf{N}_\mathbf{f}$ is the number of fibers in the generated connectome. Specifically, the diffusion signal in each voxel is modeled as a weighted sum of the contribution from each fiber passing through it: $\mathbf{b} = \mathbf{M}\mathbf{w}$, where $\mathbf{w} \in \mathbb{R}^{\mathbf{N}_\mathbf{f}}$ encodes the contribution (or weight, w_f) of each fiber f to the diffusion signal $\mathbf{b}$. LiFE minimizes the error between the modeled and measured diffusion signal by assigning a non-negative weight to each fiber. This can be posed as a non-negative least squares optimization problem:

$$\min_{\mathbf{w}}(O(\mathbf{w})), \quad O(\mathbf{w}) = \tfrac{1}{2}\left\|(\mathbf{b} - \mathbf{M}\mathbf{w})\right\|^2, \quad \mathbf{w} \geqslant \mathbf{0} \qquad (1)$$

Solving this problem, as is, imposes significant memory demands (Pestilli et al. 2014). A recent study (Caiafa and Pestilli 2017) overcame this limitation by adopting a more efficient, tensorial representation of $\mathbf{M}$). The demeaned diffusion signal $\mathbf{M}_\mathbf{v} \in \mathbb{R}^{\mathbf{N}_\theta \times \mathbf{N}_\mathbf{f}}$ in each voxel $\mathbf{v}$ was represented, using Sparse Tucker Decomposition, as $\mathbf{M}_\mathbf{v} = \mathbf{S_0}(\mathbf{v})\mathbf{D}\mathbf{\Phi}_\mathbf{v}$, where $\mathbf{S_0}(\mathbf{v})$ is the diffusion signal measured in the absence of a diffusion gradient, $\mathbf{D} \in \mathbb{R}^{\mathbf{N}_\theta \times \mathbf{N_a}}$ is a dictionary matrix comprising canonical diffusion "atoms" ($\mathbf{N_a}$: number of "atoms") used to estimate the individual contributions of each fiber and $\mathbf{\Phi}_\mathbf{v} \in \mathbb{R}^{\mathbf{N_a} \times \mathbf{N_f}}$ is a sparse, binary matrix whose columns indicate the contribution of each atom to each fiber, in that voxel. Collating $\mathbf{\Phi}_\mathbf{v}$ for all voxels $\mathbf{v}$ into a sparse 3-D tensor $\mathbf{\Phi}$, the modeled (or predicted) diffusion signal may be written as $\mathbf{Y}$:

$$\mathbf{Y} = \mathbf{\Phi} \times_\mathbf{1} \mathbf{D} \times_\mathbf{2} \mathbf{S_0} \times_\mathbf{3} \mathbf{w^T} \qquad (2)$$

where $\mathbf{w}$ is the vector of all the individual streamline weights. $\mathbf{M}$ in equation (1), is now given by $\mathbf{M} = \mathbf{\Phi} \times_\mathbf{1} \mathbf{D} \times_\mathbf{2} \mathbf{S_0}$

With this representation, the optimization problem is solved using an efficient Subspace Barzilei-Borwein Non-Negative Least Squares (SBB-NNLS) algorithm. Briefly, given $\mathbf{w^0}$ as an initial weight vector, the weight updates occur as follows (Kim, Sra, and Dhillon 2013):

$$\mathbf{w^{(i+1)}} = [\mathbf{w^{(i)}} - \alpha^{(\mathbf{i})} \bigtriangledown \mathbf{g}(\mathbf{w^{(i)}})]_+$$

where the gradient,

$$\bigtriangledown \mathbf{g}(\mathbf{w}) = \mathbf{M^T}(\mathbf{M}\mathbf{w} - \mathbf{b}) \qquad (3)$$

and α^i, the step value at each iteration, is given by

$$\alpha^i = \frac{<\bigtriangledown \tilde{\mathbf{g}}^{(\mathbf{i-1})}, \bigtriangledown \tilde{\mathbf{g}}^{(\mathbf{i-1})}>}{<\mathbf{M}\bigtriangledown \tilde{\mathbf{g}}^{(\mathbf{i-1})}, \mathbf{M}\bigtriangledown \tilde{\mathbf{g}}^{(\mathbf{i-1})}>}$$

[1] https://github.com/SawanKumar28/real-life

for the odd iterations and

$$\alpha^i = \frac{<\mathbf{M}\bigtriangledown\tilde{\mathbf{g}}^{(\mathbf{i-1})}, \mathbf{M}\bigtriangledown\tilde{\mathbf{g}}^{(\mathbf{i-1})}>}{<\mathbf{M^T M}\bigtriangledown\tilde{\mathbf{g}}^{(\mathbf{i-1})}, \mathbf{M^T M}\bigtriangledown\tilde{\mathbf{g}}^{(\mathbf{i-1})}>}$$

for the even iterations. The tilde denotes the projection of the gradient into the positive space at each iteration.

This optimization is a critical bottleneck in the LiFE algorithm: computing time scales with connectome size, number of voxels and diffusion directions. For dMRI data acquired at millimeter spatial resolutions, with a hundred gradient directions, and several million fiber connectomes, these repeated computations entail significant processing time: single connectomes can take 10-20 hours for LiFE post-processing on standard desktop machines.

In addition to the slow execution time, the LiFE algorithm suffers from a few other, key limitations. First, it retains a significant number of fibers with small weights (Pestilli et al. 2014). While these may represent weak, short fibers in the connectome, these could also indicate fits to noise in the diffusion signal in a local neighborhood of each voxel. Second, the LiFE algorithm does not completely prune away redundant (e.g. physically similar) fibers, but rather, distributes weights evenly across them. To avoid overfitting and redundancy, the LiFE evalution is cross-validated by performing tractography with one dMRI dataset and using a second, independently acquired dataset from the same subject to prune the connectome. Nevertheless, acquiring two datasets doubles the dMRI scan time and renders this validation approach unsuitable, particularly for use with patients.

We sought to build upon the LiFE algorithm to address these limitations. First, we developed a GPU-based approach that significantly speeded up LiFE's optimization procedure. Then, we leveraged this speedup to develop a regularized pruning approach that improved cross-validation accuracy.

3 Regularized, Accelerated LiFE (ReAl-LiFE)

3.1 Accelerating Connectome Pruning with GPUs

To address the first issue of slow execution time of the LiFE algorithm, we developed a GPU-accelerated version of LiFE's SBB-NNLS optimization algorithm, with a CUDA implementation (Nvidia 2018). This optimization is the major time consuming step in the LiFE algorithm, implemented as a series of array multiplications in two key functions (algorithms in Tables 1 & 2) involving the sparse tensor $\mathbf{\Phi}$. Here, we propose a solution (using a single GPU) that scales specifically well with large connectome sizes. We addressed an important bottleneck to ensure favourable memory access patterns into the GPU kernels.

Briefly, the SBB-NNLS optimization algorithm requires several multiplications of the form $\mathbf{Mx}$ or $\mathbf{M^T y}$, where $\mathbf{x}$ and $\mathbf{y}$ are generic notations of matrices that are used in various steps of the optimization (Section 2). A key ingredient of our GPU acceleration approach is splitting the computation among voxels, with each CUDA block handling data associated with one voxel. For storage efficiency, the matrix $\mathbf{M}$ is stored in a sparse tensor (Coordinate list, COO format) with indices into the dictionary matrix $\mathbf{D}$, and it is not feasible to use standard sparse matrix multiplication packages. Following Sparse Tucker Decomposition (STD) of $\mathbf{M}$ (equation (2)), computing $\mathbf{Mx}$ requires computing linear combinations of columns from $\mathbf{D}$ while $\mathbf{M^T y}$ computation requires computation of inner products with columns of $\mathbf{D}$ (Caiafa and Pestilli 2017). The former has high memory write bandwidth requirement while the latter has high memory read bandwidth requirement as well as a reduction operation. To address this issue, we sorted the $\mathbf{\Phi}$ tensor, stored in the COO format, along the voxel dimension; enabling faster per-voxel execution of both $\mathbf{Mx}$ and $\mathbf{M^T y}$, by reducing memory write and read requests, respectively.

We fixed the block size to the warp-size of the GPU, which corresponds to the number of threads processing each voxel. Each thread handled one or more diffusion directions, depending on the total number of diffusion directions. Data along the diffusion direction dimension were padded such that its size was a multiple of the warp-size, to avoid branching in the kernel code that is to be run on the GPUs. This also permitted maximizing the usage of warp shuffle instructions and reducing shared memory usage. We used shared memory only for storing the final results.

The operations and pseudocode within each block for the two matrix operations are detailed in Tables 1 & 2. Briefly, in each block, we read up to warp-size entries from the sparse tensor in parallel, to leverage memory coalescing advantages, and stored them in thread local memory. The threads in a block then computed on different diffusion directions for the read entries sequentially. We used warp broadcast instructions to share data from thread local memories to all threads in a block. In case of $\mathbf{M^T y}$ computation, we used warp shuffle instructions for computing inner products. This freed up resources, potentially allowing more blocks to be scheduled at any given time.

3.2 Regularized Pruning and Evaluation of Fascicle Evidence

To address the second issue of redundant fibers, we developed a regularized pruning algorithm, extending LiFE. We modified LiFE's least-squares error minimization objective function (equation (1)) to incorporate a regularization cost based on the summed weights of all fibers in the connectome. We added an L1 penalty (L1 norm of the weight vector) to the objective function: $O(\mathbf{w}) + \lambda \left\|\mathbf{w}\right\|_1, \mathbf{w} \geqslant \mathbf{0}$. The gradient calculation in equation (3) now changes to $\bigtriangledown\mathbf{g}(\mathbf{w}) = \mathbf{M^T}(\mathbf{Mw} - \mathbf{b}) + \lambda\mathbf{1}$, where $\lambda\mathbf{1}$ is a vector of all 1s scaled by λ. We tested several values of the penalty λ, and chose a value based on the sum of weights in the unregularized connectome or based on cross-validation error (see Section 4, next).

4 Experiments and Validation

4.1 GPU-implementation Accelerates Connectome Pruning by 50-100x

We tested speedups obtainable with the GPU-implementation of the LiFE algorithm (without regularization). For this we generated connectomes of various sizes ranging from 0.5 million to 2 million fibers with MRtrix

Table 1:
GPU Implementation of $\mathbf{y} = \mathbf{M\_times\_x}(\mathbf{\Phi}, \mathbf{D}, \mathbf{x})$

- Block size: 32
- Grid size: Number of voxels
- Each block handles one voxel. $\mathbf{\Phi}$ tensor is stored in COO format - entries $[a, v, f, c]$ where $a(n), v(n), f(n), c(n)$ are the atom index, voxel index, fiber index and the corresponding value at those indices respectively. In the following, $[a_i f_i, c_i]$ denote the corresponding entries for voxel i.
- N_ω, the number of diffusion directions handled by each thread is pre-calculated. It is equal to the (number of diffusion directions)/32.
- $N_v(i)$ refers to the number of $\mathbf{\Phi}$ entries for voxel i.

1. Set $index = 1$.
2. In block i, read up to $N_b <= 32$ entries of voxel i from the $\mathbf{\Phi}$ tensor. In thread j, read $a_i(j), x(f_i(j)), c_i(j)$ into thread local memory.
3. Broadcast $a_i(j), x(f_i(j)), c_i(j)$ from thread $j = index\ mod\ 32$ to all threads into local variables a, x, c.
4. In thread j, initialize local memory $y_l(q) = 0$ where $q = 1$ to N_ω.
5. In thread j
for $q = 1$ to N_ω **do**
$y_l(q) = y_l(q) + D(a, j + q * 32) * x * c$
end for
6. $index = index + 1$, if $(index\ mod\ 32) < N_b$, go to Step (3). Else continue.
7. $index = index + 32$; if $index < N_v(i)$, go to Step (2). Else continue.
8. Write the results back to global memory. In block i, thread j
for $q = 1$ to N_ω **do**
$y(i, j + q * 32) = y_l(q)$
end for

Table 2:
GPU Implementation of $\mathbf{x} = \mathbf{M\_transp\_y}(\mathbf{\Phi}, \mathbf{D}, \mathbf{y})$

- Block size: 32
- Grid size: Number of voxels (other notations as in the algorithm in Table 1).

1. In block i, read voxel data from global memory into local memory y_l. In thread j
for $q = 1$ to N_ω **do**
$y_l(q) = y(i, j + q * 32)$
end for
2. Set $index = 1$.
3. In block i, read up to $N_b <= 32$ entries of voxel i from the $\mathbf{\Phi}$ tensor. In thread j, read $a_i(j), f_i(j), c_i(j)$ into thread local memory.
4. Broadcast $a_i(j), c_i(j)$ from thread $j = index\ mod\ 32$ to all threads into local variables a, c.
5. In thread j, initialize local memory $res = 0$.
6. In thread j
for $q = 1$ to N_ω **do**
$res = res + D(a, j + q * 32) * y_l(q) * c$
end for
7. Compute the sum of res local variables using warp shuffle down reduction, the final sum being stored in res of thread 0.
8. In thread 0, store the computed sum in shared variable, $s_x(index\ mod\ 32) = res$.
9. $index = index + 1$, if $(index\ mod\ 32) < N_b$, go to Step (4). Else continue.
10. Write the results back to global memory. In thread $j < N_b$,
$atomic\_add(x(f_i(j)), s_x(j))$
where $atomic\_add(a, b)$ adds b atomically to a.
11. $index = index + 32$, if $index < N_v(i)$, go to Step (3). Else return.

v3.0 (Tournier et al. 2012). Whole brain probabilistic tractography was performed using the grey-matter-white-matter interface as a seed region. For each subject, anatomical (grey and white-matter) segmentations were obtained using Freesurfer v6 (Fischl et al. 2004). We then used the memory-optimised LiFE method (Caiafa and Pestilli 2017; Caiafa et al. 2017) to encode the tractogram and dMRI data.

First, we optimized one dMRI dataset, *I*, with 64 diffusion directions, 2 mm spatial resolution, containing 116,468 white matter voxels. Dataset *I* was preprocessed with a standard pipeline ((Pestilli et al. 2014; Caiafa and Pestilli 2017); implemented in mrDiffusion, Vistasoft package). Key steps included manual alignment of the T1 image based on AC-PC landmarks, eddy-current correction and motion correction with a rigid-body alignment algorithm. We tested five different connectome sizes for 500 iterations of the optimization algorithm (Fig. 2A). We ran the original CPU-based version of LiFE, first, followed by our GPU-accelerated implementation. We observed a clear trend of increase in the speedup with connectome size: the speedup factor reached a maximum of ~80x for a connectome with 3 million fibers, the largest connectome size we tested (Fig. 2B). We also confirmed that the reduction in objective function value at each iteration, and the weights assigned to each streamline, by LiFE (CPU version) and our GPU implementation were identical to within numerical precision.

We further confirmed these speedups on two other independent datasets acquired at other scanners. First, we optimized a dMRI dataset, H[2]. Dataset *H* data was already minimally preprocessed, and no additional proccessing was done. We tested four connectome sizes ranging from 100,000 to 1 million fibers (Fig. 2A, middle). Again, we observed a 63x speedup over the CPU version, for a connectome of 1 million fibers. Finally, we optimized a second dMRI dataset, S[3] with dMRI scans already preprocessed based on a standard pipeline (Pestilli et al. 2014; Caiafa and Pestilli 2017). Here, again, we observed a 93x speedup over the CPU version, for a connectome with 1 mil-

[2]https://www.humanconnectome.org/study/hcp-young-adult/document/1200-subjects-data-release

[3]https://purl.stanford.edu/cs392kv3054

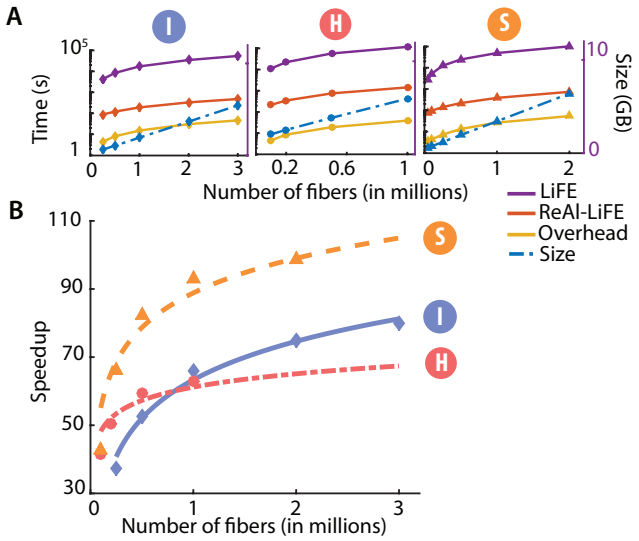

Figure 2: *Speed-up with ReAl-LiFE* tested on three dMRI datasets (*I*, *H* and *S*) with several million fibers each. **A** Execution time as a function of number of fibers in the connectome for dataset *I* (left), *H* (middle) and *S* (right). Purple: LiFE; red: ReAl-LiFE; orange: one-time overhead for ReAl-LiFE. Blue dot-dashed line: Size of each connectome (right axis). **B** Speedup factor with ReAl-LiFE as a function of connectome size for the three datasets: *I* (blue), *H* (red) and *S* (orange). Filled markers: measured speedups. Curves: sigmoidal fits.

lion fibers and a maximum speedup of 98x for a connectome with 2 million fibers (Fig. 2A-B). In each case, we obtained better speedups as the connectome size increased.

Our speedups also exceeded state-of-the-art numbers based on a recently reported MPI-acceleration scheme for LiFE (Gugnani et al. 2017). Our baseline numbers are derived from the same configuration as Gugnani et al's "Cluster A" (single core Intel Xeon E5), enabling us to directly compare our speedup factors with theirs. In that study, the reported maximum multi-node speedups (8.1x) did not exceed the single node speedup value (8.7x), suggesting that MPI may not be an effective parallelization strategy for LiFE. On the other hand, our speedups are an order of magnitude faster (98x), indicating GPU acceleration is an effective strategy.

4.2 Regularized Pruning Provides Higher Cross-validation Accuracy

Next, we compared improvements in cross-validation accuracy by fitting the connectome to the underlying diffusion signal, first, with the original LiFE algorithm and, next, with our novel regularized pruning algorithm (ReAl-LiFE). We adopted the following approach: we performed tractography with data from one dataset and pruned the connectome by optimizing it, either with LiFE or ReAl-LiFE, on the same dataset. In each case, we generated a predicted diffusion signal from the fitted diffusion model, as described above (equation (2)). Next, we compared the root-mean-squared error (RMSE) between the predicted diffusion signal, in each case, and a second, independent, diffusion imaging dataset acquired from the same subject(s).

We performed this cross-validation with data from two datasets: *I* and *S* (see Section 4.1). In each case we compared the root-mean-squared cross-validation error (RMSE) by generating a connectome with 1 million fibers, followed by pruning with LiFE (unregularized), against the accuracy of a connectome generated with 2 million fibers, followed by pruning with ReAl-LiFE. The choice of using a larger connectome for pruning with ReAl-LiFE was motivated by a need to maintain a comparable sum of streamline weights across the two approaches, as the regularization is expected to prune out more fascicles than the unregularized LiFE algorithm. For ReAl-LiFE, we tested several values of the regularization penalty parameter λ, on the fiber weights. This extensive grid search on λ could be completed in a reasonable time frame due to the substantial speedups provided by our accelerated, memory-efficient implementation.

ReAl-LiFE provided significantly higher cross-validation accuracy as compared to LiFE, as quantified by the RMSE across voxels, across a range of regularization parameter values (Fig. 3). Interestingly, we observed that for both datasets the sum of fiber weights estimated with ReAl-LiFE (Fig. 3A, blue curve) closely matched the sum of fiber weights as estimated by the unregularized LiFE algorithm (Fig. 3A, red, dashed horizontal line) in the vicinity of $\lambda = 0.01$. To perform a fair comparison, we compared the RMSE-s of LiFE and ReAl-LiFE optimized connectomes at this value of λ, although, even with larger values of λ (up to 0.2), corresponding to many fewer (up to 80% fewer) connections, the ReAl-LiFE algorithm continued to exhibit better RMSE (Fig. 3B).

Cross-validated RMSE distributions were significantly different across the LiFE and ReAl-LiFE optimized connectomes ($p < 0.001$; Kolmogorov-Smirnov test), with RMSE medians being signficantly lower for the ReAl-LiFE, as compared to the LiFE optimized connectome (dataset *I*: LiFE=0.936 ± 0.0007, ReAl-LiFE=0.922 ± 0.0007; dataset *S*: LiFE=0.854 ± 0.0003, ReAl-LiFE=0.843 ± 0.0003; mean $\pm$ std error, $p < 0.001$, Wilcoxon signed rank test). Next, we identified matched voxels across the two optimized connectomes and computed the distribution of differences in the RMSE-s following each type of optimization. We observed that the difference in RMSE was greater in a significantly higher proportion of voxels following LiFE optimization, as compared to ReAl-LiFE optimization (Fig. 3C, $p < 0.001$, Kolmogorov-Smirnov test).

Finally, we also tested the model's performance with ensemble tractography on dataset *S* (Takemura et al. 2016). Ensemble tractography is a technique that overcomes biases imposed by parameter choices by estimating several connectomes, one for each choice of parameter and consolidating them into a single "ensemble". We created two ensemble connectomes (0.8 and 1.6 million fibers) using smaller connectomes (0.16 and 0.32 million fibers respectively) generated by varying the maximum radius of curvature of fibers to be tracked between one of five values (0.25, 0.5, 1, 2, and 4 mm). This was followed by pruning with LiFE (0.8 million connectome) or ReAl-LiFE (1.6 million connectome).

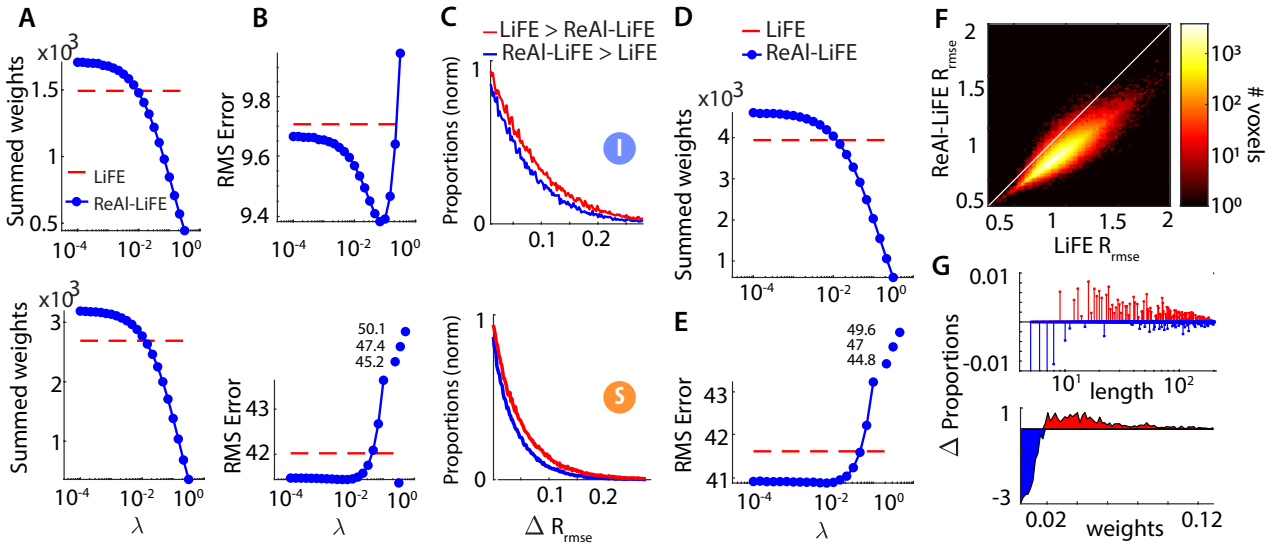

Figure 3: *Improvement in cross-validated root-mean-squared-error (RMSE) with ReAl-LiFE.* **A** Sum of fiber weights for a 1 million fiber connectome pruned with LiFE (red dashed line) and a 2 million fiber connectome pruned with ReAl-LiFE (blue solid line) for various values of the regularization parameter λ for datasets *I* (top) and *S* (bottom). **B** Cross-validated RMSE as a function of λ. Other conventions are as in panel **A**. **C** Histogram showing the proportion of voxels in which the difference of RMSE was greater with LiFE as compared to ReAl-LiFE (red) or vice-versa (blue). **D-E** Summed weights and RMSE as in **A** for an ensemble tractogram (see text). **F** Distribution of the RMSE across voxels of the ensemble tractogram following LiFE and ReAl-LiFE. Hotter colors: Higher proportion of voxels. Solid white diagonal line: Line of equality. **G** Relative proportion of fiber lengths (top) or weights (bottom) in the connectome with LiFE or ReAl-LiFE pruning. Positive (red) and negative (blue) values indicate higher relative proportions of fiber lengths or weights with ReAl-LiFE or LiFE pruning, respectively.

As before, ReAl-LiFE provided significantly lower RMSE, across a range of regularization parameter values (Fig. 3D-E). Again, for $\lambda = 0.01$, RMSE was significantly lower for the ReAl-LiFE optimized connectome as compared to the LiFE connectome (Fig. 3F; $p<0.001$; Wilcoxon signed rank test). We also observed that although the number of fibers with non-zero weights was higher (by ~12%) in the ReAl-LiFE optimized connectome (0.232 million) as compared to the LiFE optimized connectome (0.207 million), ReAl-LiFE retained a relatively higher proportion of fibers with larger weights and greater lengths and pruned out more fibers with small weights and shorter lengths as compared to the LiFE connectome (Fig. 3G). While weak, short fibers cannot be distinguished from noisy data based on weights alone, cross validation demonstrates that L1 regularization significantly reduces RMS error (Fig. 3F), thereby removing noisy (unreliable) fibers in the connectome. Further, the regularization enabled pruning the connectome to a significantly smaller size (22% fewer connections than the LiFE connectome) without a significant increase in cross-validation RMSE (Fig. 3E).

5 ReAl-LiFE Effectively Classifies Brain Pathology in Alzheimer's Disease

As a real-world application, we tested whether regularized connectivity, as estimated with ReAl-LiFE, would effectively distinguish normal from pathological brains. Data from 90 individuals were drawn from the ADNI database (Mueller et al. 2005) comprising dMRI scans of patients with Alzheimer's Disease (AD; n=45, age: mean = 74.9; std = 8.5) and healthy, age-matched controls (NC; n=45, age: mean = 72.6; std = 5.6). Briefly, we first quantified connectivity strength between the hippocampus and cortical regions using either the number of fibers, LiFE or ReAl-LiFE weights. We then employed these connectivity measures as features in a support vector machine (SVM) classifier to test whether AD patients could be accurately distinguished from normal controls. In addition, we also estimated connectivity strengths with SIFT2 (Smith et al. 2015), another popular approach for connectome evaluation. SIFT2 filters tractograms with a constrained spherical deconvolution (CSD) algorithm, and estimates individual streamline weights, representative of the cross-sectional area of each streamline.

Because declarative memory is known to be impaired in AD, we examined structural connections between the cortex and hippocampus, a sub-cortical brain region important for declarative memory (Tulving and Markowitsch 1998; Eichenbaum 2001; Eichenbaum 2004). We first obtained an automated segmentation of the hippocampus and 68 different cortical regions (34 per hemisphere) using FreeSurfer (Fischl et al. 2004; Desikan et al. 2006) for every subject. Next, we generated whole-brain cortical connectomes with 1 million fibers for each subject. This was followed by targeted tracking of 1 million fibers between the hippocampus and each of the 34 different cortical regions, in each hemisphere, using the hippocampus as a seed. We com-

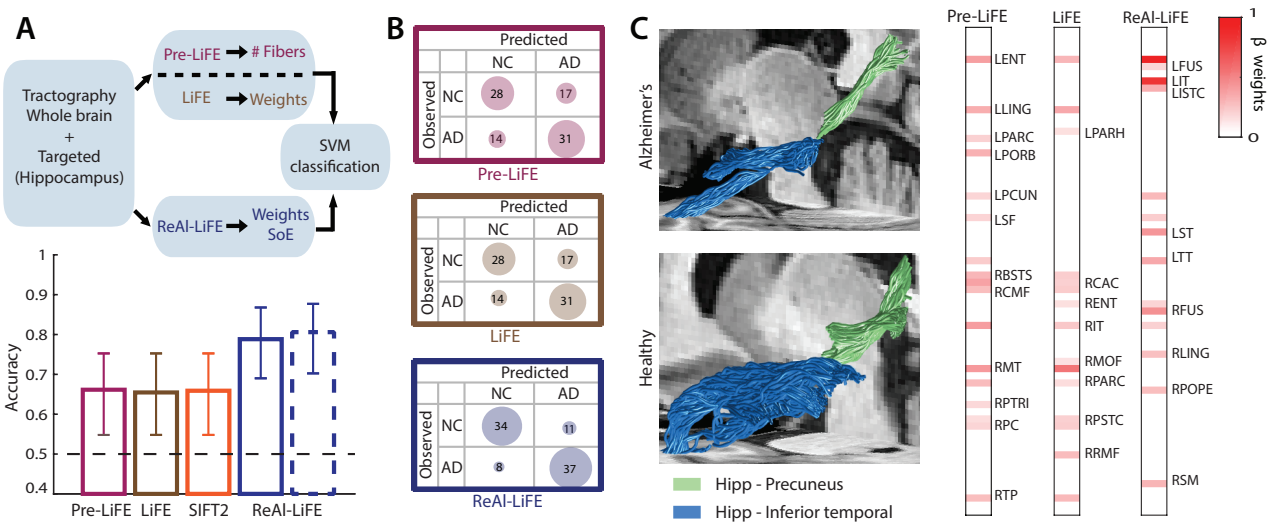

Figure 4: *Classifying Alzheimer's Disease (AD) patients from normal controls (NC) using ReAl-LiFE connectivity*. **A** (Top) Schematic summarizing the classification analysis. (Bottom) Classification accuracies based on number of fibers for the unpruned connectome (Pre-LiFE; magenta bar), connection weights with LiFE pruning (brown bar), with SIFT2 pruning (orange bar) and connection weights or strength of evidence with ReAl-LiFE pruning (blue bars with solid or dashed outlines, respectively). Error-bars indicate 95% binomial confidence intervals. Dashed horizontal line: Chance (50%). **B** Confusion matrices for the classification based on Pre-LiFE (Top), LiFE (Center) and ReAl-LiFE (Bottom) features. Size of circles reflect proportions of subjects for each contingency; number of subjects (rounded) indicated within each circle. **C** (Left) White-matter fibers connecting the hippocampus to the precuneus (pale green) and inferior temporal (pale blue) cortex for an exemplar AD patient (top) and control subject (bottom). (Right) Connections corresponding to highest SVM β weights for classifying AD from NC based on number of fibers (left), LiFE weights (middle) or ReAl-LiFE weights (right). Deeper shades: higher weights. L/R: Left/Right hemisphere, BSTS: Bank of the superior temporal sulcus, CAC: Caudal anterior cingulate cortex, CMF: Caudal middle frontal cortex, ENT: Entorhinal cortex , FUS: Fusiform gyrus, ISTC: Isthmus of the cingulate cortex, IT: Inferior temporal cortex, LING: Lingual gyrus, MOF: Medial orbitofrontal cortex, MT: Middle temporal cortex, PARC: Paracentral lobule, PARH: Parahippocampal cortex, PC: Posterior cingulate cortex; PCUN: Precuneus, POPE: Pars opercularis, PORB: Pars orbitalis, PSTC: Postcentral gyrus, PTRI: Pars triangularis, RMF: Rostral middle frontal cortex, SF: Superior frontal cortex, ST: Superior temporal cortex, TP: Temporal pole, TT: Transverse temporal cortex.

bined this "targeted" connectome of hippocampal fibers with the whole-brain connectome to generate a 2 million fiber connectome for each subject. This combined connectome was then pruned with LiFE, SIFT2, and ReAl-LiFE each. We then constructed a structural connectivity vector corresponding to 68 intra-hemispheric hippocampus-cortex connections, using number of connections (Pre-LiFE), connection weights (LiFE, SIFT2, and ReAl-LiFE), or strength of evidence (ReAl-LiFE; see below), for each subject.

These feature vectors were used to train SVM classifiers. For SVM classification, we used Matlab's "fitclinear" function, with soft margin, employing a linear kernel with C=1 (default values in Matlab). We applied recursive feature elimination (RFE), to find a minimal set of features that provided the highest cross-validation accuracy (De Martino et al. 2008). RFE is an iterative technique that eliminates a subset of features with the lowest SVM weights, following which the SVM is retrained and classified with the retained features. The process is repeated until all features are exhausted, following which the minimal set of features that provide the maximum generalization accuracy are identified. This procedure was repeated 150 times for random classifications of training and testing data, and classification accuracy was averaged across these runs. All numbers reported in the paper indicate mean $\pm$ 95% binomial confidence intervals, as calculated by the Clopper-Pearson method.

Two-way classification accuracy (AD versus NC), based on the number of fibers in the unpruned connectome (Pre-LiFE), was 66.1% (95% binomial confidence interval: [54.8, 75.3]; Fig. 4A, bottom, magenta bar). Classification accuracy, based on the connection weights following LiFE pruning was 65.5% ([54.8, 75.3], Fig. 4A, bottom, brown bar) and the same based on the connection weights following SIFT2 pruning was 65.9% ([54.8,75.3], Fig 4A, bottom, orange bar). On the other hand, classification accuracy based on the connection weights following ReAl-LiFE pruning was 78.8% ([69, 86.8], Fig 4A, bottom, solid blue bar), substantially higher than that of any of the other approaches. Next, we computed the strength of evidence (SoE) for every fascicle (set of fibers) connecting the hippocampus to each of the 68 cortical regions. Briefly, the SoE for a fascicle represents the increase in prediction error of the diffusion signal after removing that fascicle from the connectome (Pestilli et al. 2014). Classification accuracy with ReAl-LiFE SoE was

even higher, at 80.6% ([70.2, 87.7], Fig. 4A, bottom, dashed blue bar).

Similar improvements were observed in precision and recall. Precision, defined as the proportion of correctly classified AD patients among the total number of subjects classified as AD, was 64.6% (31/48) whereas recall, defined as the proportion of AD patients correctly classified as AD, was 68.9% (31/45), for classification based on the number of fibers in the unpruned connectome (Fig. 4B, top). After pruning with LiFE, precision and recall were at 64.6% (31/48), and 68.9% (31/45), respectively (Fig. 4B, middle). However, after pruning with ReAl-LiFE, precision increased to 77.1% (37/48) and recall increased substantially to 82.2% (37/45) (Fig, 4B, bottom).

Classification based on ReAl-LiFE weights relied on a largely non-overlapping set of connections, as compared to the classification based on the number of fibers or LiFE weights. For example, connections of the hippocampus with the left inferior-temporal cortex (LIT; Fig. 4C, right) or with the left superior temporal cortex (LST) were important for classification based on ReAl-LiFE weights, but not based on the other approaches.

Taken together, these results indicate that pruning with ReAl-LiFE could provide key benefits for sensitive diagnostic classification of neuropathologies, such as Alzheimer's Disease. The higher classification accuracies with ReAl-LiFE occurred, perhaps, due to more accurate estimates of connection strengths following ReAl-LiFE connectome pruning.

6 Discussion and Conclusions

We present a GPU implementation of a linear connectome pruning algorithm that provides significant speedups (up to 100x) compared to the original CPU implementation. Our GPU-accelerated, L1-regularized implementation enables fast and more reliable connectome discovery in large-scale datasets and represents a timely, and widely-relevant tool for big data neuroscience applications. Our approach specifically optimizes memory access for voxels containing multiple fibers, thereby improving speedups for larger connectome sizes.

These speedups have important real-world implications. First, our approach permitted robustly classifying AD patients from healthy controls based on their structural connectivity (Section 5). These discoveries were possible because GPU-based acceleration permitted rapid pruning of individual connectomes in this large dataset (n=90 subjects, with 1 million fiber connectomes each), within a few hours, when the original algorithm would have taken several days. Second, the algorithm facilitated developing a regularized objective function for connectome pruning. This function provided a more reliable fit to the data as compared to unregularized pruning, as evidenced by lower cross-validation error. Third, the GPU-based acceleration permitted rapid identification of an appropriate regularization parameter. This parameter search allowed us to show that, much sparser connectomes could be generated with ReAl-LiFE, that matched (or even exceeded) the cross-validation accuracy of LiFE pruned connectomes. Further improvements in speedup may be possible with the use of multiple GPUs or by combining recently proposed MPI-based schemes (Gugnani et al. 2017) with our GPU-parallelization approach.

In addition to the speedups, the increased cross-validation accuracy with ReAl-LiFE has important implications for estimating and evaluating connectomes in patient populations. Estimating the whole-brain connectome on one dMRI dataset and optimizing ReAl-LiFE on the same dataset produced significantly reduced cross-validation error, when validated with an independently acquired, second dataset. In cases where acquiring two independent dMRI datasets may not be feasible due to the extended scan duration, such as in patients with neurological disorders (e.g. AD patients), our results show that ReAl-LiFE can efficiently optimize the connectome with a single dataset, as evidenced by its greater cross-validation accuracy across multiple scans.

Our GPU acceleration is applicable to a general class of non-negative least-squares (NNLS) optimization problems: regression ($\mathbf{b} = \mathbf{M}\mathbf{w}$) based on sparse Tucker Decomposition of $\mathbf{M}$. The need for such optimization is frequently encountered in neuroscience, healthcare (Ho, Ghosh, and Sun 2014), behavior modeling (Jiang et al. 2014), NLP(Kang et al. 2012) and other domains. Moreover, our CUDA implementation can be readily applied to various "big data" applications in neuroimaging; for example, sparse tensorial decompositions of large scale fMRI datasets (Beckmann and Smith 2005). Recently, there is increasing interest with identifying brain-behavior relationships from neuroimaging data, in large databanks of tens of thousands of participants, with a secondary goal of identifying sensitive "imaging markers" that permit early detection of the onset of neurodegenerative disorders (Smith and Nichols 2018). Our ReAl-LiFE algorithm enables utilizing the structural connectome as a candidate imaging "biomarker" for diagnosis and predicting disease onset in such big data applications.

Acknowledgments The authors would like to thank Madhav Gumma for contributions to an early version of the algorithm, and Cesar Caiafa for sharing LiFE (version 2.0) code. This research was funded by MHRD, Govt. of India (to SK, VS); NSF IIS-1636893, NSF BCS-1734853 (to FP); a Wellcome Trust-Department of Biotechnology India Alliance Intermediate fellowship, a Science and Engineering Research Board Early Career award, a Pratiksha Trust Young Investigator award, a Department of Biotechnology-Indian Institute of Science Partnership Program grant, a Sonata Software foundation grant and a Tata Trusts grant (to DS). We would also like to acknowledge the Alzheimers Disease Neuroimaging Initiative (ADNI) for access to dMRI data from AD patients and healthy controls; details regarding ADNI funding sources are available at *https://adni.loni.usc.edu*.

References

Beckmann, C. F., and Smith, S. M. 2005. Tensorial extensions of independent component analysis for multisubject fMRI analysis. *Neuroimage* 25(1):294–311.

Caiafa, C. F., and Pestilli, F. 2017. Multidimensional encoding of brain connectomes. *Scientific Reports* 7(1):11491.

Caiafa, C. F.; Sporns, O.; Saykin, A.; and Pestilli, F. 2017. Unified representation of tractography and diffusion-weighted MRI data using sparse multidimensional arrays. In *Advances in Neural Information Processing Systems*, 4343–4354.

De Martino, F.; Valente, G.; Staeren, N.; Ashburner, J.; Goebel, R.; and Formisano, E. 2008. Combining multivariate voxel selection and support vector machines for mapping and classification of fMRI spatial patterns. *Neuroimage* 43(1):44–58.

Desikan, R. S.; Ségonne, F.; Fischl, B.; Quinn, B. T.; Dickerson, B. C.; Blacker, D.; Buckner, R. L.; Dale, A. M.; Maguire, R. P.; Hyman, B. T.; et al. 2006. An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest. *Neuroimage* 31(3):968–980.

Eichenbaum, H. 2001. The hippocampus and declarative memory: cognitive mechanisms and neural codes. *Behavioural Brain Research* 127(1-2):199–207.

Eichenbaum, H. 2004. Hippocampus: cognitive processes and neural representations that underlie declarative memory. *Neuron* 44(1):109–120.

Fischl, B.; Van Der Kouwe, A.; Destrieux, C.; Halgren, E.; Ségonne, F.; Salat, D. H.; Busa, E.; Seidman, L. J.; Goldstein, J.; Kennedy, D.; et al. 2004. Automatically parcellating the human cerebral cortex. *Cerebral Cortex* 14(1):11–22.

Gugnani, S.; Lu, X.; Pestilli, F.; Caiafa, C.; and Panda, D. K. 2017. MPI-LiFE: Designing high-performance linear fascicle evaluation of brain connectome with MPI. In *High Performance Computing (HiPC), 2017 IEEE 24th International Conference on*, 213–222. IEEE.

Ho, J. C.; Ghosh, J.; and Sun, J. 2014. Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 115–124. ACM.

Jiang, M.; Cui, P.; Wang, F.; Xu, X.; Zhu, W.; and Yang, S. 2014. Fema: flexible evolutionary multi-faceted analysis for dynamic behavioral pattern discovery. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1186–1195. ACM.

Kang, U.; Papalexakis, E.; Harpale, A.; and Faloutsos, C. 2012. Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 316–324. ACM.

Kim, D.; Sra, S.; and Dhillon, I. S. 2013. A non-monotonic method for large-scale non-negative least squares. *Optimization Methods and Software* 28(5):1012–1039.

Mueller, S. G.; Weiner, M. W.; Thal, L. J.; Petersen, R. C.; Jack, C. R.; Jagust, W.; Trojanowski, J. Q.; Toga, A. W.; and Beckett, L. 2005. Ways toward an early diagnosis in Alzheimer's disease: the Alzheimer's Disease Neuroimaging Initiative (ADNI). *Alzheimer's & Dementia: The Journal of the Alzheimer's Association* 1(1):55–66.

Nvidia, C. 2018. Nvidia CUDA C programming guide. *Nvidia Corporation*.

Pestilli, F.; Yeatman, J. D.; Rokem, A.; Kay, K. N.; and Wandell, B. A. 2014. Evaluation and statistical inference for human connectomes. *Nature Methods* 11(10):1058.

Smith, S. M., and Nichols, T. E. 2018. Statistical challenges in "Big Data" Human Neuroimaging. *Neuron* 97(2):263–268.

Smith, R. E.; Tournier, J.-D.; Calamante, F.; and Connelly, A. 2013. SIFT: spherical-deconvolution informed filtering of tractograms. *Neuroimage* 67:298–312.

Smith, R. E.; Tournier, J.-D.; Calamante, F.; and Connelly, A. 2015. SIFT2: Enabling dense quantitative assessment of brain white matter connectivity using streamlines tractography. *Neuroimage* 119:338–351.

Sudlow, C.; Gallacher, J.; Allen, N.; Beral, V.; Burton, P.; Danesh, J.; Downey, P.; Elliott, P.; Green, J.; Landray, M.; et al. 2015. UK biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *PLoS Medicine* 12(3):e1001779.

Takemura, H.; Caiafa, C. F.; Wandell, B. A.; and Pestilli, F. 2016. Ensemble tractography. *PLoS Computational Biology* 12(2):e1004692.

Tournier, J.; Calamante, F.; Connelly, A.; et al. 2012. MRtrix: diffusion tractography in crossing fiber regions. *International Journal of Imaging Systems and Technology* 22(1):53–66.

Tulving, E., and Markowitsch, H. J. 1998. Episodic and declarative memory: role of the hippocampus. *Hippocampus* 8(3):198–204.

Van Essen, D. C.; Ugurbil, K.; Auerbach, E.; Barch, D.; Behrens, T.; Bucholz, R.; Chang, A.; Chen, L.; Corbetta, M.; Curtiss, S. W.; et al. 2012. The Human Connectome Project: a data acquisition perspective. *Neuroimage* 62(4):2222–2231.