Towards a Synthetic Brain Benchmark Suite

***Abstract*—As data collection continuous to become more ubiquitous, the rate of data processing becomes just as crucial as the quantity of data collected and accuracy of results. The only system today with the ability to gather and process vast of amount of data at a fraction of the time and energy of today's best processors is the brain. Each part of the brain is responsible for efficiently executing a unique set of data processing tasks such as visual recognition, natural language processing, and learning. Just as our brain uses data to make informed decisions about our surroundings, an emerging class of algorithms that try to mimic this function have shown success performing tasks that range from movie recommendation to disease detection. With the data warehouse growing at an exponential scale multi-core and many core processors are one solution to improving data processing speeds, but need to be coupled with scalable algorithms to take advantage of what these processor designs have to offer.**

The goal with this benchmark suite is to provide a roadmap toward a comprehensive synthetic brain benchmark suite that uses the major lobes of the cerebral cortex as a model for the organization and classification of data processing algorithms. As these algorithms become increasingly complex, the need to realize the potential for parallelism and exploit the architectural underpinnings through an accessible set benchmarks has motivated the creation this benchmark collection.

We present The San Diego Synthetic Brain Benchmark Suite, a library of varying and popular state-of-the-art algorithm written in C with the goals of accessibility, practicality, and usability in mind. The suite includes a spectrum of datasets for each application that varies in size and goal an an attempt to mimic real world situation. Our approach attempts to ensures depth for each individual benchmark and breadth for the entire benchmark suite

I. Introduction

With the increased availability of data, data science continues to focus on algorithms to find patterns, learn rules, and recognize relationships. This has motivated research into a new class of algorithms inspired by the way our brain executes these workloads when performing learning, pattern recognition, and sensory awareness. Although these tasks are trivial for the human brain to execute they traditionally result in prohibitively long execution times on modern processors. As multi-core and many core processors continue to improve, data-driven applications have come to the forefront of high performance computing. There is a growing need for a comprehensive benchmark suite to help architecture researchers profile the architectural tendencies of these algorithms, find what kind of parallelization can be exploited, and determine if these algorithms are scalable.

In the organization of this benchmark suite we draw parallels between the three major data processing classes and the four majors lobes of the sensory cortex - the temporal, occipital, parietal, and frontal. With this abstraction, we classify the occipital and temporal lobe, which are the visual processing centers of the brain, as a computer vision core. The parietal lobe, which is responsible for language comprehension, as the natural language core. And the frontal lobe as learning core.

To account for the varied nature of data processing tasks, our proposed taxonomy helps maintain benchmark consistency when extending this benchmark suite. For example, vision benchmark suites such as MEVBench[1] and SD-VBS[2] can be integrated into the occipital node of the suite and machine learning benchmark suites such as the ones in MineBench[3] and Sophia-ml[4] can be integrated into the frontal node.

The benchmark suite contains eight unique applications from natural language processing, computer vision, and machine learning domains. Some of our most exciting algorithms include Restrictive Boltzman Machines, Sphinx Speech Recognition, Super Resolution Reconstruction. Restrictive Boltzman Machines and Sphinx are part of the frontal node and utilize neural networks and hidden markov models to solve problems in classification[5], topic modeling[6], dimentionality reduction[7], and speech recognition[8]. Super Resolution Reconstruction takes sets of low resolution images to create one high resolution one, similar to how involuntary movements in the eye, microsaccades, improve the resolution of the eye in the occipital lobe. Each of the eight algorithms have been selected by consulting with researchers in the learning and vision areas in an effort to provide complete coverage of popular and interesting algorithms in their respective areas. The applications have been coded to avoid the use of pointers, large memory accesses, and machine-specific optimizations. In doing this the goal is to facilitate the analysis, transformation, parallelization, and simulation of the benchmarks by compiler and architecture researchers[2].

Each algorithm in the Synthetic Brain Benchmark Suite, contains data of varying sizes and applications with several different configuration parameters for each algorithm. The datasets were selected to represent actual commercial and academic scenarios in an effort to mirror real world use cases. The specific contributions of this benchmark suite include:

- We have developed a benchmarking framework for brain like algorithms
- We have created a benchmarking suite that includes algorithms that are the current state-of-the-art algorithms in data processing
- We provide detailed explanations of each benchmark, its implantation and dataset
- We provide explanation of hotspots in each application and discussion parallelization and scalability analysis

Section II of our paper provides explanations of all eight benchmarks and an overview of the implementation methodology. Section III provides hotspot and parallelization analysis,

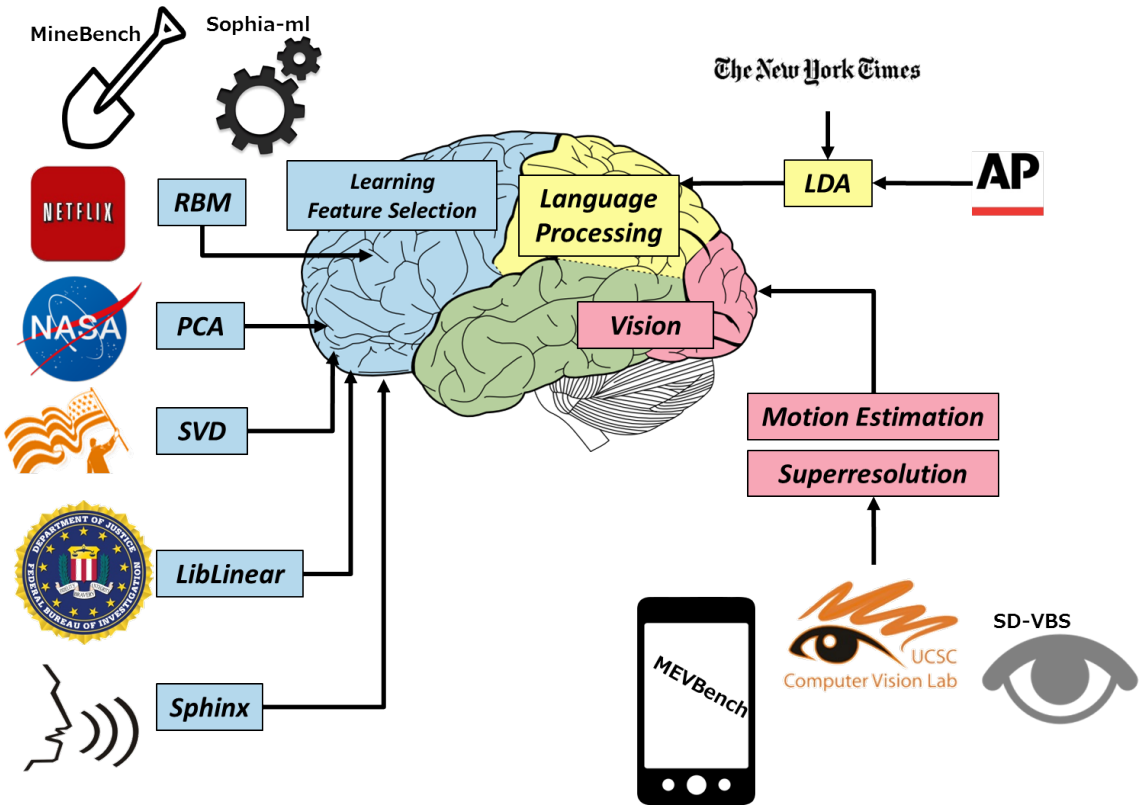

Fig. 1. **The Synthetic Brain:** Applications and datasetes mapped to various parts of the brain's cerebral cortex. Vision is processed primary by the occipital lobe and temporal lobe. Langauge processing is mapped to the parietal lobe. Learning and feature selection are mapped to the frontal lobe. The various benchmarks around each lobe are examples of benchmarking suites that can be easily integrated into our lobe-core framework

section IV looks at Related Works in this area, and our paper Concludes and discusses Future Work in section V.

II. Benchmark Details

The goal of the benchmark suite is to provide a set of algorithms that have the flexibly to cover a broad spectrum of application domains and include data representative of how these algorithms are used commercially and academically. In choosing the algorithms for the suite we consulted with researchers in the computer vision, data mining, and machine learning fields from academia and industry. To maintain usability and readability we limit the use of complex pointer and memory operations and include datasets with real world implications.

A. Applications

The current iteration of the suite contains eight different algorithms with widely varying applications and datasets. The applications are written in C with multiple configuration parameters three datasets per application.

Restricted Boltzmann Machine (RBM) is a stochastic neural network algorithm with applications in collaborative filtering [9], feature learning and topic modeling. SD-VBS utilizes RBM to implement movie suggestions (collaborative filtering) on a reduced Netflix database and provides the benchmark for the training process of RBM. We are interested in the training process because it is very computationally intensive. It can take hours or days to complete depending on the size of the database. For this reason, we reduced the size of Netflix database (without loss of generality) to make the running time of the benchmark feasible.

To train RBM, we initialize the model with random parameters. Then, we repeat the training iteration until convergence. In every iteration, we let the current RBM try to reconstruct the input data, and then we adjust its parameters based on the error of the data reconstruction. With more iterations, the reconstruction error diminishes, and we stop the training when the error converges (or until some predetermined number of iterations is reached). These iterations are very computationally intensive.

Sphinx Speech Recognition can be used for the translation of spoken words to text [10]. The common way to recognize speech is the following: we take waveform, split it on utterances by silences, and then try to recognize the word in each utterance [11]. We take all possible combinations of words and try to match them with the audio. We choose the best matching combination.

The algorithm utilizes Viterbi algorithm [12], Hidden

TABLE I
OVERVIEW OF ALGORITHMS AND APPLICATION SPACE

Benchmark	**Category**	Dataset
Sphinx Speech Recognition	Deep Learning	CMU
Restricted Boltzman Machines	Deep Learning	Netflix
Principal Component Analysis	Rank Reduction	NASA
Singular Value Decomposition	Rank Reduction	New York Times
Superresolution	Computer Vision	UCSC
Motion Estimation	Computer Vision	Varying
Latent Dirichlet Allocation	Natural Language Processing	Associated Press
LibLinear	Classification	Varying

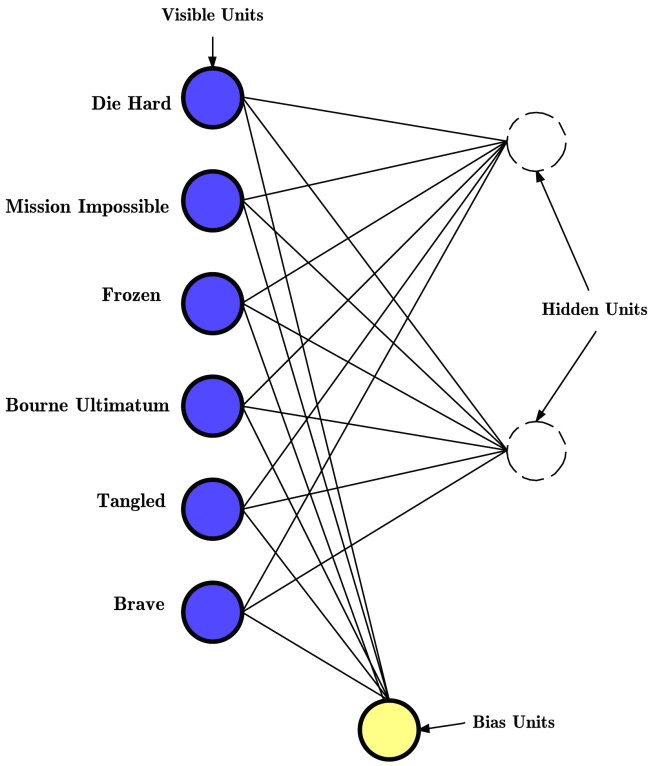

Fig. 2. **RBM Hidden and Visible Nodes:** Visible nodes represent movies while hidden nodes represent genres for each movie. Each movie will have an edge to each genre, where the edge represents the weight of the connection between movie and genre. The value for each weight determines how correlated a movie is to a specific genre. While they are not explicitly named, the two hidden nodes here represent Action Movies and Disney Movies.

Markov Model (HMM) [13], and n-gram language model [14]. The algorithm first translates utterances into possible words using Viterbi algorithm on a predetermined HMM. Then, the possible words are evaluated using a predetermined n-gram language model to find the most likely words combination. Viterbi algorithm is a computationally intensive dynamic programming algorithm, and evaluating the probabilities of words combinations using the n-gram model also requires a lot of computations.

SD-VBS implements speech recognition using CMU Sphinx, an open source speech recognition system developed at Carnegie Mellon University. Specifically, SD-VBS implements the Pocketsphinx version of CMU Sphinx, which is a lightweight speech recognizer written in C .

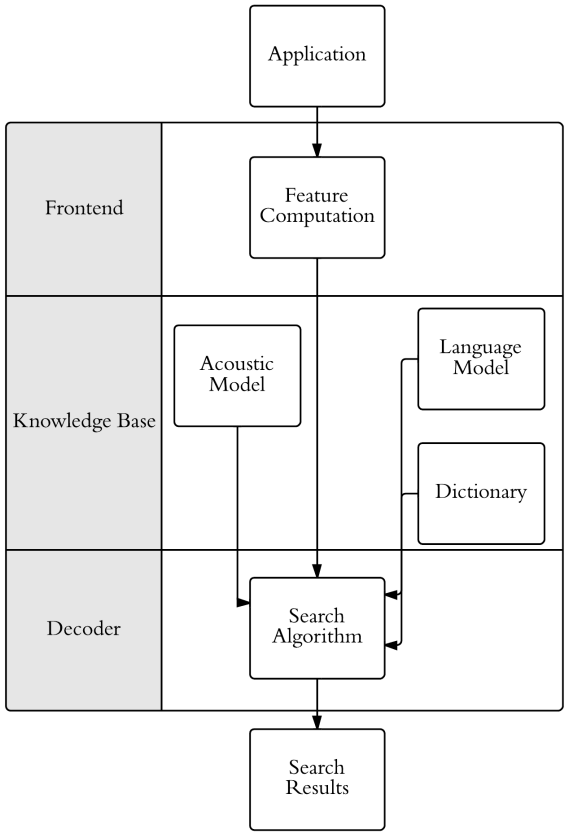

Fig. 3. **Sphinx Speech Recognition:**Sphinx first extracts features from speech waveform into feature vectors. The search algorithm will use the features to determine possible words using the acoustic model (HMM), and then combine the possible words into sentences using the language model (n-gram) and dictionary.

Super-resolution Reconstruction (SRR) is based on the idea that slight variations in the information encoded in a series of low resolution (LR) images can be used to recover one high resolution (HR) image [15], [16]. Computational resolution enhancement has many applications in the fields of healthcare, security, astronomy, military etc. In nature the manifestation of super-resolution can be seen in the compound eye structures of the insects and flies. The dynamic theory of vision states that eye movements improve the resolution of the eye. During fixation eyes sample a scene through miniature eye movements called microsaccades. These sampled images provide the human brain with enough information to resolve hyperacuity tasks - superresolution. [17]

We implement the parallel algorithm for super resolution proposed by Qiang Zhang $et.al.$ [16]. The basic superresolution problem can be posed as an inverse and ill-posed problem [18] as the information is lost in the process of image formation in a camera.

$$\min_f \|DH_iS_if - g_i\|_2^2, i = 1, ...l^2 \tag{1}$$

where f is the vectorized true high resolution image, g_i is a vectorized low resolution image, D is the decimation matrix, H_i is a blurring matrix, S_i is a shift matrix and l is the up-sampling factor. The decimation matrix D is a local averaging matrix that aggregates values of non-intersecting small neighborhoods of HR pixels to produce LR pixel values. The shift matrix S_i; assigns weights according to a bilinear interpolation of HR pixel values to perform a rigid translation of the original image. The blurring matrix H_i is generated from a point spread function (PSF) and represents distortion from atmospheric and other sources.

The l^2, DH_iS_i matrices are stacked to create one large least squares problem.

$$\min_f \|Af - g\|_2^2 + \alpha \|f\|_2^2 \tag{2}$$

where α is the Tikonov regularization factor.

As can be seen from the above equation super-resolution is generally regarded as a memory and computationally intensive process due to the large matrix-vector calculations involved. The dimensionality of the above least squares problem is $O(m^2 \times n^2)$ where $(m \times n)$ is the size of the desired high resolution image. However because of the high locality of reference, this problem can be easily solved in parallel. It can be seen that while solving for a given low resolution pixel (which is super-resolved into $(l \times l)$ high resolution pixels), only the neighboring pixels participate into computation. Hence we can solve all the odd pixels at a time and then solve for all the even pixels and repeat the process.

Latent Dirichlet Allocation (LDA) is used for topic modeling large numbers of documents. The input to this algorithm is a bag-of-words representation of documents, which is a sparse matrix of words and associated word counts. The underlying algorithm uses the assumption that each document was generated using a Dirichlet Distribution, which serves as a prior distribution to the Multinomial Distribution[20]. The goal of the algorithm is to find values for the multinomial parameter vectors and topics for each document. The underlying data sets used in this algorithm are obtained from David Bleis AP data [20] for the large dataset, UCIs NIPS data[21] for the medium dataset, and UCIs KOS data [21] for the small dataset.

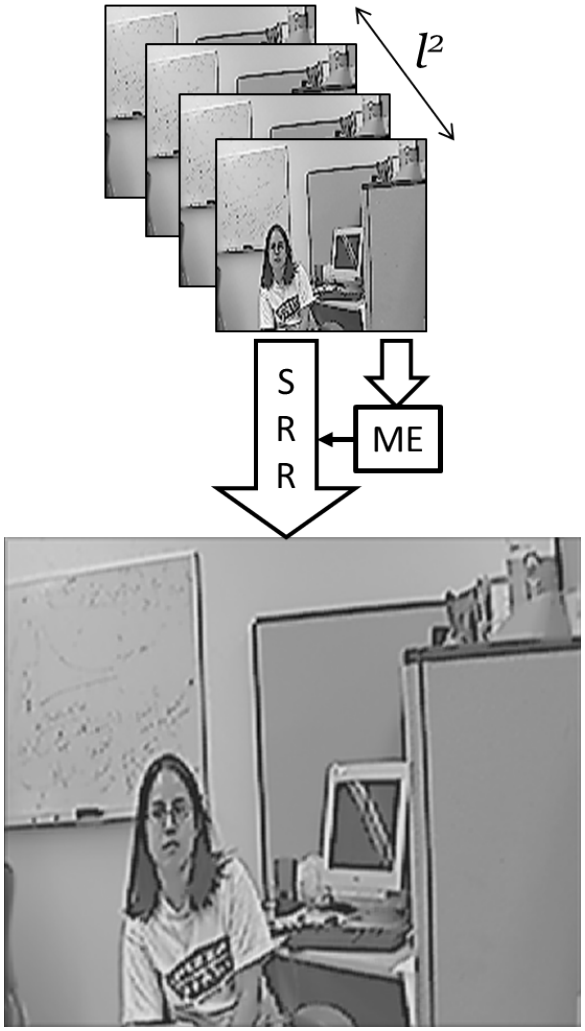

Fig. 4. **Super-Resolution:** 16 distinct LR images of size (128x96) are used to generate one HR image of size (512x384). Datasets courtesy: Multi-Dimensional Signal Processing (MDSP) research lab at the University of California at Santa Cruz [19]

This specific program is obtained from David Bleis LDA C implementation[20]. It features a variable number of topics N, an initial value for the alpha hyperparameter, and random or seeded sampling methods. The alpha hyperparameter provides a prior distribution for the number of topics per

document. Increasing the value for alpha allows documents to belong to more topics. Increasing the number of topics allows the algorithm to find more word groupings. Finally, using random or seeded allows the user to specify a seed for the random number generator used for Gibbs sampling.

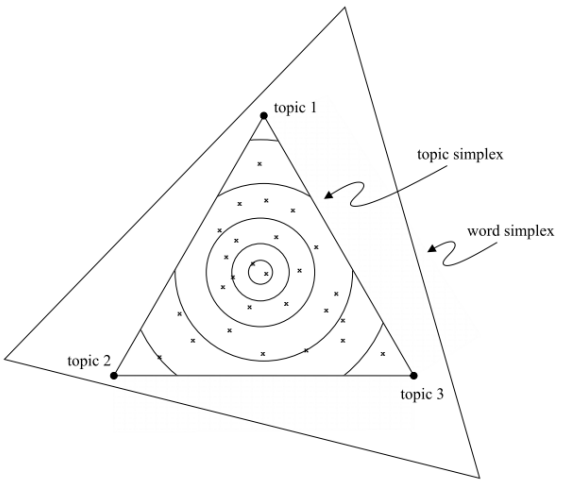

Fig. 5. **Latent Dirichet Allocation[20, pg. 1002]:** Topic simplex for three topics inside of a word simplex for three words. Each corner in the topic simplex represents a specific topic and each point represents a single document. Each point inside of the topic simplex denotes a probability towards each topic, where each corner represents a probability of one for a certain topic.

Singular Value Decomposition (SVD) is a powerful rank reduction algorithm used in many artificial intelligence, signal processing, and computer vision applications [22]. SVD is a factorization of a matrix into three matrices. The resulting three matrices have useful information about the original matrix and can be multiplied together to obtain the original matrix back. SD-VBS utilizes SVD for Latent Semantic Analysis (LSA) [23]. LSA is a technique used in natural language processing to analyze relationships between a set of documents and the terms they contain. LSA can be used to create a simple search engine.

SD-VBS utilizes SVD for LSA. The input is a corpus of articles in a bag-of-words format. The bag-of-words format is a matrix with words represented in rows and articles represented in columns, and each entry contains the number of times that word appears in that article. We use the bag-of-words data of Daily Kos website from University of California, Irvine. SVD involves several matrix operations to find eigenvectors and eigenvalues, which overall makes it computationally intensive with respect to the size of the input matrix. To make the benchmarks running time feasible, we reduce the size of the input matrix.

Principle Component Analysis (PCA) is one of the most versatile and widely used statistical techniques for feature extraction in multivariate datasets. When dealing

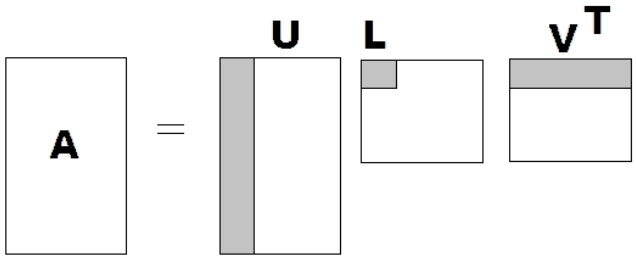

Fig. 6. **SVD:** SVD decomposes a matrix into three matrices: U, S, and V. The matrices U and V contain information about the spread of the data, while the matrix S contains the significance of the information provided in U and V.

with high-dimensional data, one needs to consider the weight and importance of variables for accuracy and run-time considerations. PCA allows high-dimensional data to be reduced to an orthogonal lower-dimensional data-set by leveraging the dependencies between variables and identifying the most important ones. As a prepossessing technique, PCA has found applications ranging from computer vision to machine learning and has a run-time of O(n^3) where n is the number of points.[cite?]

Our implementation of PCA is based on Carnegie Mellon Universities StatLib library, modified to reduce large number of static array references and memory operations. The algorithm includes operations for finding correlation analysis, covariance analysis, and SSCP analysis.

Motion Estimation is the process of determining motion vectors that describe the transformation of one 2D image to another. It is an inverse, ill-posed problem as the motion is in 3D but the images are the projection of the 3D scene onto a 2D plane. Motion Estimation is an essential element in image and video processing. It is a key part of the video coding and applications such as frame rate up conversion, image super resolution etc. Performance of motion estimation can directly affect the performance of these applications.

We implement the fast sub-pixel motion estimation algorithm proposed by Stanley H. Chan $et.al.$ [25] This is a hybrid algorithm which combines two popular techniques of motion estimation namely block matching and optical flow. As the name suggests, the block matching algorithm finds the best matched block in the search space. While the block matching algorithm is the most accurate and easiest to parallelize, it is a bad candidate for the sub-pixel motion estimation because it must interpolate the image by a certain factor depending on the desired sub-pixel accuracy. For example if we want to achieve 0.25 pixel accuracy, then we need to enlarge the image by 8 times along each direction. Although state-of-art algorithms can selectively choose where to interpolate, their cost is still high. Hence the implementation uses block matching algorithm for full-pixel

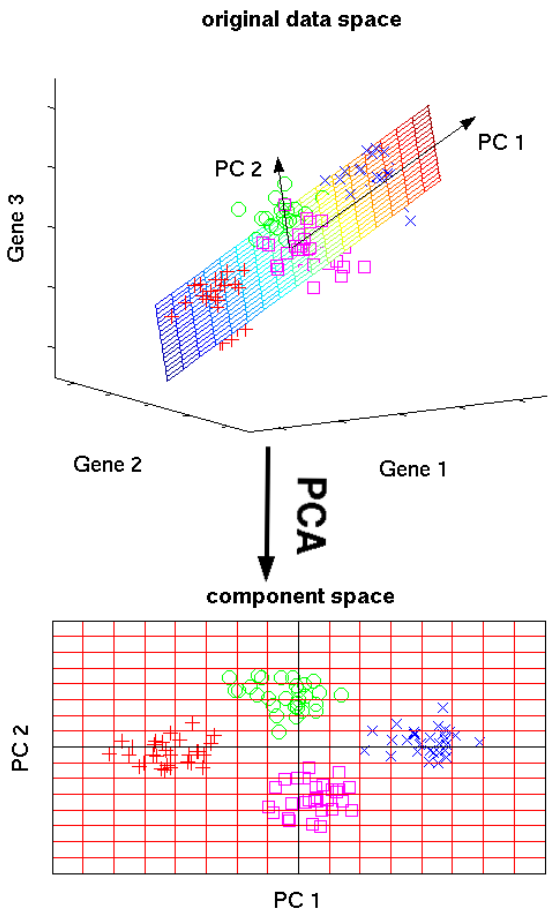

Fig. 7. **PCA:** Illustrated are three-dimensional gene expression data which are mainly located within a two-dimensional subspace. PCA is used to visualize these data by reducing the dimensionality of the data: The three original variables (genes) are reduced to a lower number of two new variables termed principal components (PCs). Top: Using PCA, we can identify the two-dimensional plane that optimally describes the highest variance of the data. This two-dimensional subspace can then be rotated and presented as a two-dimensional component space (bottom). Such two-dimensional visualization of the samples allow us to draw qualitative conclusions about the separability of experimental conditions (marked by different colors). [24]

part of motion estimation [26] and the sub-pixel part is computed using the optical flow [27]. Optical flow [28] is the pattern of apparent motion of objects, as perceived by the variation in the light intensity, caused by the relative motion between an observer, scene and the light source. Optical flow is based on the assumption of the brightness constancy constrain and holds true only for the small motions and hence is an excellent candidate for the sub-pixel motion.

Liblinear is a versatile library for large-scale linear classification with applications in computer vision, natural language processing, neuroimaging, and bioinformatic[]. The library has been used in a variety of applications from realtime object recognition to predicting protein solubility. The library supports both binary classification and multiclass classification with a focus in performance for data with a large amount of

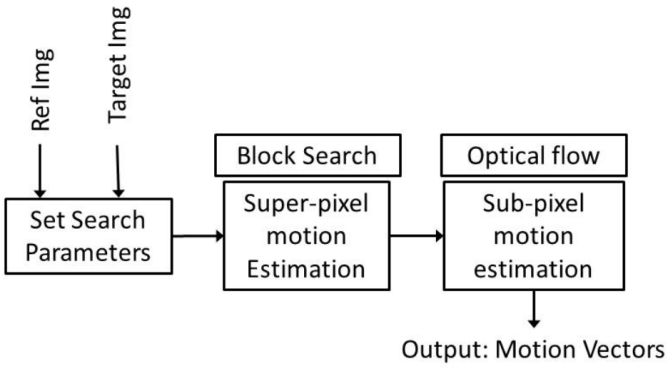

Fig. 8. **Motion Estimation:** Two images (ref, target) are input and output is a set of motion vectors. Full-pixel motion is estimated using block search algorithm. Sub-pixel motion is estimated using optical flow.

features and sparsity. Binary.Binary linear classification is a supervised learning model in which elements are [classified] into two groups of data based on some predetermined metric. The goal of binary classification is to take a set of n-features and create a binary response which can then be used to reason about future sets of related data. Liblinear also supports multiclass classification through the one-vs-all which is a subset of binary classification. In a one-vs-all algorithm, each class is separated from others in training step. In the prediction step a simple binary classifier is run over each class and the one with the highest probability is chosen.

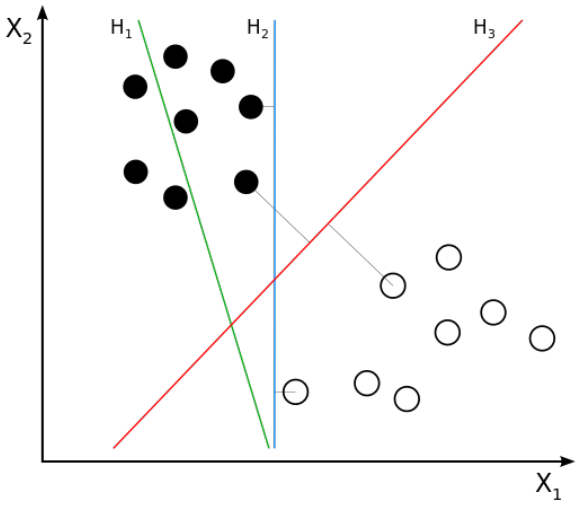

Fig. 9. **LibLinear:** [wikicopy] H1 does not separate the classes. H2 does, but only with a small margin. H3 separates them with the maximum margin.

III. Methodology

In the following section we profile the algorithms in the Brain Benchmark Suite and describe our evaluation parameters. The goal is to determine how scalable these applications are and where the major bottlenecks lie in the kernels for each of these algorithms. To focus on hot-spot analysis we utilized Intel's Vtune software to find bottlenecks in our

application. To determine scalability we use Kremlin, an open source program out of UC San Diego designed to assist the parallelization of serial program, utilizing hierarchical critical path analysis.[kermlin-copy] Using Kremlin, we run each benchmark with an OpenMP planner and simulate 4,32,and 1000 core (ideal) configurations. Our benchmarking platform is described in table 2. The benchmarks were run with all the datasets included (small,medium,large) with the benchmark suite and will be made public upon publication.

TABLE II
EVALUATION PLATFORM

Feature	Description
Operating System	Linux 3.13.8-1-ARCH
Processors	Intel Core i7-2620M CPU @ 2.70GHz
L1 Cache	32kB x2 @ 8-way Set-associative
L2 Cache	256kB x2 @ 8-way Set-associative
L3 Cache	4096kB Shared @ 16-way Set-associative
Memory	8GB
Memory Bus(?)	1333Mhz

A. Hotspot and Scalability Analysis

Superresolution Recontruction

The super-resolution reconstruction benchmark is the most computationally intensive application in our benchmark suite. About 20% of the total execution time is spent in the DIV unit of the processor, responsible for non-trivial mathematical operations such as divides and square roots.

Super-resolution has coarse and fine grained parallelism. The coarse grained parallelism is achieved by solving for all odd LR pixels in parallel and then solving for all even LR pixels in parallel. The fine grained parallelism is achieved by parallelizing the kernels such as Gauss-Seidel solver and matrix operations. About 35% of the time is spent performing the Gauss-Seidel and 65% of the time is spent on matrix operations. While all of these kernels can be parallelized, it is also possible to parallelize the calls to these kernels

Running SRR through Kremlin, we find that it ends. (kremlin results).

Sphinx

Sphinx speech recognition has a number hidden markov model computations that cause slowdown in the application. The HMM in this program take close to 30% of the total runtime, the majority of which is located in two main operations. This includes the evaluation of finding the optimal HMM sequence using the Viterbi algorithm [12] and the search of most likely sentence using n-gram model [14]. (maybe explain why running Viterbi on HMM is compute intensive? Viterbi algorithm is dynamic programming.

Running SRR through Kremlin, we find that (kremlin problems).

Restricted Boltzmann Machines

Due to the nature of the RBM algorithm (generative stochastic neural network) most of the execution time is spent in various activation functions for hidden and visible units, taking up 40% of the total execution time. In a neural network, each processing unit is a neuron which fire a binary signal when they have reached certain parameters. Based on the type of activation function in the network, a function is specified which cause the neuron to fire a '1' or a '0' dependent on the function threshold. When this happens, these signals propagate through the network and can cause other neurons to fire as well. (make sure this explanation is right. Enrico: I think this explanation is OK.). Activation functions are key in the training of neural networks.

Kremlin finds that all the aforementioned functions, train, activativeHiddenUnits, and activativeVisibleUnits are the most promising targets for parallelization. The entire train function can be be reduced by 36% with 4-cores with an ideal speed up of 44%. The activation functions can each be speed up by 14% and 11% with 4-cores and the ideal cause only provides a slightly increase to 17% and 14% execution time reduction.

Latent Dirichet Allocation

LDA[7] suffers from being relatively DIV active with 14% of the execution time going into these long latency mathematical operations. The digamma function, the logarithmic derivative of the gamma function, also accounts for 30% of the execution time. (explain more?)

We also find that the main computational function in LDA can be parallelized for four cores to reduce the runtime by 72% and at 92% under ideal circumstances.

Motion Estimation

We identify two primary kernels in motion estimation namely FullSearch and TaylorSeries. FullSearch is a block matching kernel for full-pixel motion estimation while TaylorSeries is the core of optical flow to find out sub-pixel motion. Both of these kernels can be parallelized. We find hierarchical parallelism such that multiple blocks can be searched in parallel (coarse grained) and matching operation in each block can be parallelized as well (fine grained). We find that parallelization can lead to a decrease in execution time by 73% with 4 cores and almost 90% in the ideal case.

LibLinear

We run Liblinear with an SVM workload and find that close to 90% of its execution time is tied up in a solver function that is responsible for several calculations including regularization and running a support vector clustering algorithm. The SVC algorithm is responsible for mapping data from a lower dimension to a higher one to find a suitable hyperplane that can provide a linear a classifier between sets of data. The L2 regularization is an optimization technique used to prevent overfitting and reducing the complexity of

the predictor.

Single Value Decomposition

SVD combines various linear algebra operations to perform rank reduction and the runtime of the algorithm is heavily data dependent (is this a thing?). In particular, 37% of the cycle time is tied up in QR decomposition used to solve the least squares problem. [8]

Parallelization of SVD provides significant oppertunity for speedup as the QR factorization works through an iterative method. In the 4 core case, we can obtain a speed up of almost 40% and in the ideal case the speed up possible is 50%.

IV. Related Work

Many benchmarking suites have been assembled in order to display the capabilities of machine learning algorithms which are tailored to specific tasks, such as computer vision. The San Diego Vision Benchmark Suite compiles a variety of computer vision algorithms, from SVM to Face Detection. The San Diego Vision Benchmark Suite gains an architectural understanding of a diverse set of computer vision algorithms and characterizes the computational properties of each algorithm. There are many other computer vision related benchmarks including MEVBench[1] and MediaBench[29]. All of the aforementioned benchmarking suites focus primarily on aspects of computer vision. Specifically, MEVBench focuses on mobile computer vision algorithms geared toward embedded systems and MediaBench focuses on video and multimedia processing.

Several other benchmarking suites have focused on the architectural characterization of various machine learning algorithms. Most notably, the MineBench[3] Benchmarking Suite has studied Data Mining and Bioinformatic workloads in order to characterize the branch and cache performance of each algorithm. The Splash-2[30] suite has characterized parallel applications in terms of the cache performance on multiprocessors.

The Brain Benchmark Suite differs from existing suites in that it completes the picture of a synthetic brain while also providing architectural analysis and an analysis of scalability of each algorithm. The datasets used in this benchmark suite provide real-world applications of each algorithm within the suite.

V. Conclusion and Future Work

References

[1] J. Clemons, H. Zhu, S. Savarese, and T. Austin, "Mevbench: A mobile computer vision benchmarking suite," in *Workload Characterization (IISWC), 2011 IEEE International Symposium on*, Nov 2011, pp. 91–102.

[2] S. Venkata, I. Ahn, D. Jeon, A. Gupta, C. Louie, S. Garcia, S. Belongie, and M. Taylor, "Sd-vbs: The san diego vision benchmark suite," in *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, Oct 2009.

[3] B. Ozisikyilmaz, R. Narayanan, J. Zambreno, G. Memik, and A. Choudhary, "An architectural characterization study of data mining and bioinformatics workloads," in *Workload Characterization, 2006 IEEE International Symposium on*, Oct 2006, pp. 61–70.

[4] D. Sculley and G. Inc, "Large scale learning to rank," in *In NIPS 2009 Workshop on Advances in Ranking*, 2009.

[5] H. Larochelle and Y. Bengio, "Classification using discriminative restricted boltzmann machines," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 536–543.

[6] R. Salakhutdinov and G. E. Hinton, "Replicated softmax: an undirected topic model." in *NIPS*, vol. 22, 2009, pp. 1607–1614.

[7] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[8] X. Huang, F. Alleva, H.-W. Hon, M.-Y. Hwang, K.-F. Lee, and R. Rosenfeld, "The sphinx-ii speech recognition system: an overview," *Computer Speech & Language*, vol. 7, no. 2, pp. 137–148, 1993.

[9] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: ACM, 2007, pp. 791–798.

[10] X. Huang, F. Alleva, H.-W. Hon, M.-Y. Hwang, K.-F. Lee, and R. Rosenfeld, "The sphinx-ii speech recognition system: an overview," *Computer Speech & Language*, vol. 7, no. 2, pp. 137–148, 1993.

[11] C. M. University, "Cmusphinx." [Online]. Available: http://cmusphinx.sourceforge.net/wiki/tutorialconcepts

[12] D. Jurafsky and J. H. Martin, *Speech & Language Processing*. Pearson Education India, 2000.

[13] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The Annals of Mathematical Statistics*, vol. 37, no. 6, pp. 1554–1563, 12 1966.

[14] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Comput. Linguist.*, vol. 18, no. 4, pp. 467–479, Dec. 1992. [Online]. Available: http://dl.acm.org/citation.cfm?id=176313.176316

[15] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: a technical overview," *Signal Processing Magazine, IEEE*, vol. 20, no. 3, pp. 21–36, May 2003.

[16] Q. Zhang, R. Guy, and R. Plemmons, "Matrix structures and parallel algorithms for image superresolution reconstruction," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 4, pp. 1873–1893, 2010.

[17] G.-Z. Y. Mirna Lerotic, "Reverse engineering of human vision and super-resolution." [Online]. Available: http://ubimon.doc.ic.ac.uk/isc/public/HPosters06_151-200/paper155.pdf

[18] R.-D. E. M. Farsiu, S. and P. Milanfar, "Advances and challenges in super-resolution," *Int. J. Imaging Syst. Technol.*, vol. 14, no. 2, p. 4757, 2004.

[19] P. Milanfar, "Mdsp super-resolution and demosaicing datasets." [Online]. Available: http://www.soe.ucsc.edu/~milanfar/software/sr-datasets.html

[20] M. I. J. David M. Blei, Andrew Y. Ng, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, January 2003.

[21] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[22] G. Strang, *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2003.

[23] S. T. Dumais, "Latent semantic analysis," *Annual Review of Information Science and Technology*, vol. 38, no. 1, pp. 188–230, 2004. [Online]. Available: http://dx.doi.org/10.1002/aris.1440380105

[24] M. Scholz, "Pca - principal component analysis." [Online]. Available: http://www.nlpca.org/pca_principal_component_analysis.html

[25] S. Chan, D. Vo, and T. Nguyen, "Subpixel motion estimation without interpolation," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, March 2010, pp. 722–725.

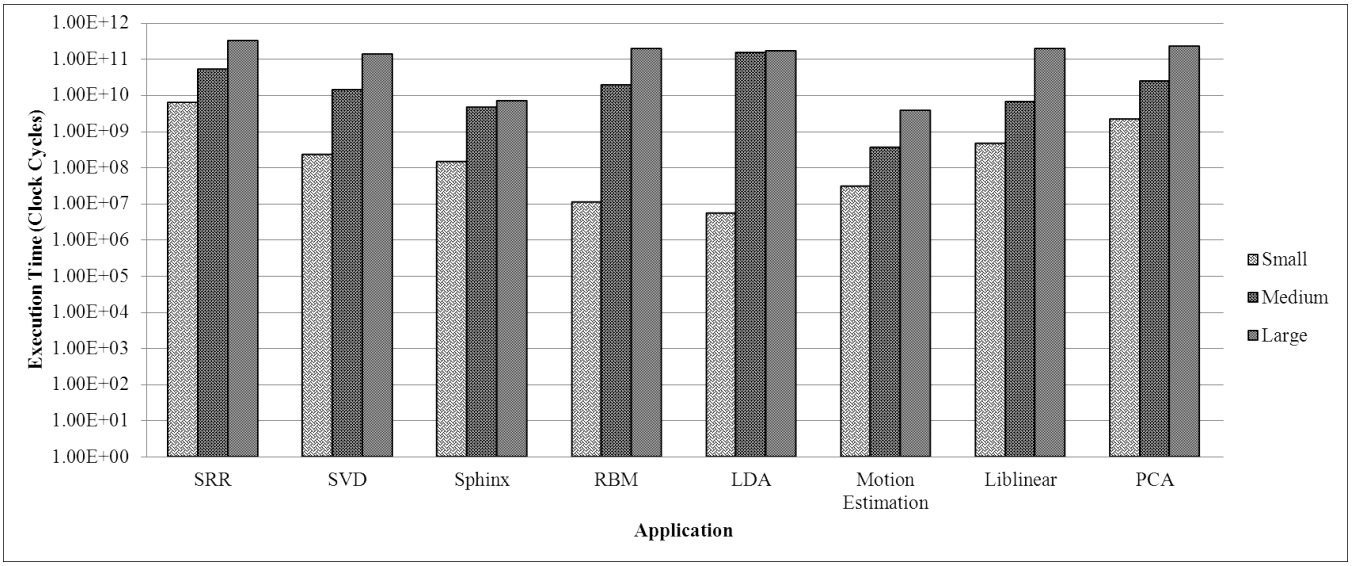

Fig. 10. Brain Benchmark Relationship

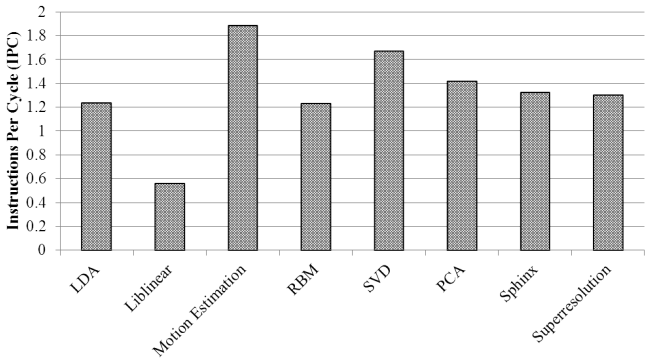

Fig. 11. Brain Benchmark Relationship

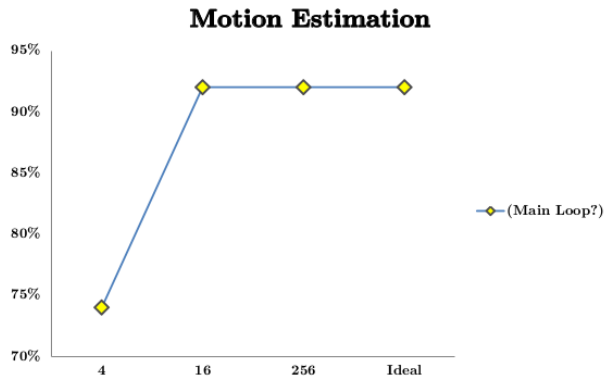

Fig. 13. Brain Benchmark Relationship

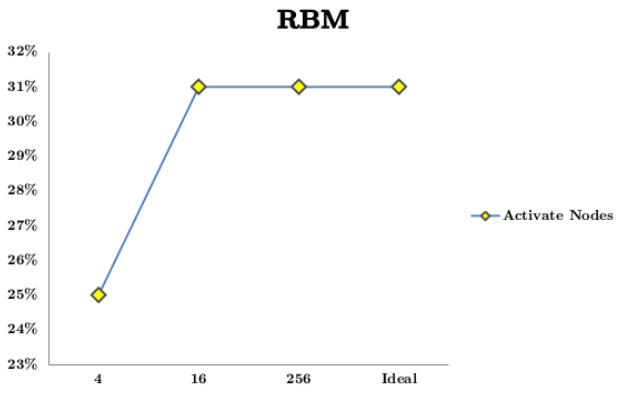

Fig. 12. Brain Benchmark Relationship

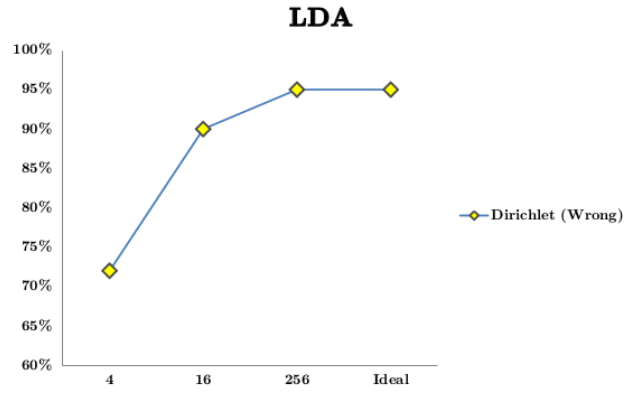

Fig. 14. Brain Benchmark Relationship

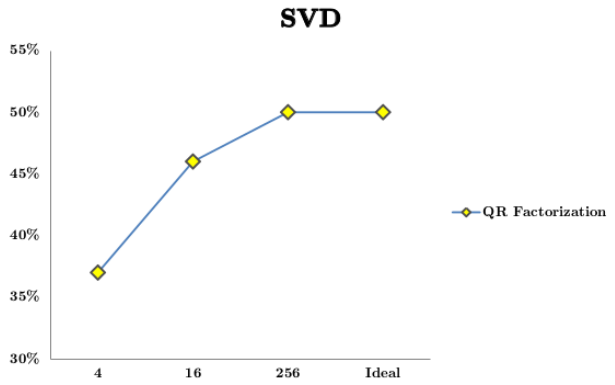

Fig. 15. Brain Benchmark Relationship

[26] V. Processing and Communications, *Yao Wang, Jorn Ostermann, and Ya-Qin Zhang*. Prentice Hall, 2002.

[27] M. Irani and S. Peleg, "Improving resolution by image registration," *CVGIP: Graphical Models and Image Processing*, vol. 53, p. 324335, 1993.

[28] B. K. Horn and B. G. Schunck, "Determining optical flow," pp. 319–331, 1981.

[29] C. Lee, M. Potkonjak, and W. Mangione-Smith, "Mediabench: a tool for evaluating and synthesizing multimedia and communications systems," in *Microarchitecture, 1997. Proceedings., Thirtieth Annual IEEE/ACM International Symposium on*, Dec 1997, pp. 330–335.

[30] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta, "The splash-2 programs: characterization and methodological considerations," in *Computer Architecture, 1995. Proceedings., 22nd Annual International Symposium on*, June 1995, pp. 24–36.