Preprint for ACM Conference on Bioinformatics, Computational Biology and Biomedicine (ACM BCB 2013) Washington D.C. Sept. 2013. pp324-332.

Performance Model Selection for Learning-based Biological Image Analysis on a Cluster

Jie Zhou, Anthony Brunson, John Winans, Kirk Duffin and Nicholas Karonis
jzhou@niu.edu, brunsonac@gmail.com, jwinans@niu.edu, kduffin@niu.edu, karonis@niu.edu
Department of Computer Science
Northern Illinois University
Dekalb, IL 60115, USA

ABSTRACT
Microscopic images with increased scale and content call for high performance computing when applying automatic tools for biological image analysis. Speed of analysis can be improved at various stages. In learning-based models, selecting suitable algorithms for a given problem can be a lengthy process given the large pool of algorithms and the variety of biological problems. In this paper, we describe a portable method for efficiently and adaptively selecting an effective model for biological image classification as a step toward the goal of achieving high throughput biological image analysis. We implemented a high performance tool which extends the bioimage classification and annotation platform BIOCAT by deploying the model selection process on a cluster using a distributed design based on remote method invocation. The high performance model selection, when tested and compared using ten benchmarking data sets, is shown to not only dramatically increase the speed of the learning process, but also bring improved accuracy to several state-of-the-art data sets for bioimage classification. These achievements are attributed to the combined power of BIOCAT's adaptive model selection as well as the capability of distributed model evaluation. The tool is deployable to various types of distributed environments.

Categories and Subject Descriptors
B.8.0 [**Performance**]: General

Keywords
Biological Image Classification, Cluster Computing, Pattern Recognition, Remote Method Invocation, BIOCAT

1. INTRODUCTION
Analysis of microscopic images, including categorization, annotation and quantification, has been an important task since the first day of modern biology. With the recent availability of large scale and high content biological microscopic images, high-throughput analysis has become the bottleneck of many scientific discoveries [1]. In neuroscience, confocal imaging can now produce high-dimensional multiple channel images of neurons. Some of them present complex arbor structures, such as hippocampal pyramidal cells in mammals and lobula plate tangential cells (LPTCs) in *Drosophila* [2]. A genetic screen on the morphological features of such images would enable new discovery for neuronal development mechanism, functionality or disease cause. For example, the identification of regulators for synapse development needs extensive examination of high dimensional neuronal images and quantification of the essential morphometrics [3][4]. In cell biology, identification of mutant phenotypes using a large number of cultured C. elegans cells has become important for the study of metabolism and drug screening [5]. Similar examples demanding efficient analysis of images can be found in various subfields of biology. However, such analysis are commonly manual or semi-manual, which are low throughput and often subjective. The problem is becoming more prominent with the ever-increasing volume and content of biological images. If automated software tools can scale up to match the performance requirement, high-throughput image analysis can benefit various research fields and likely lead to new breakthroughs in a more timely fashion.

Categorization based on learning has been playing an important role in automated biological image analysis for assigning a category to an image or a region of interest in the image. Recently, learning-based algorithms and software for biological image analysis have been developed, including BIOCAT, Wndcharm, CellProfiler, PSLID, FARSIGHT [6][7][8][9]. These tools help various tasks of biological image analysis and are successful for their respective purpose. Among them, BIOCAT, for BIOimage Classification and Annotation Tool, (http://faculty.cs.niu.edu/~zhou/tool/biocat/) provides the advantages of portability, extensibility, a convenient user interface and adaptive model selection for a given problem.

At the same time, high performance computing (HPC) clusters have become increasingly prevalent, ranging from tightly connected powerful supercomputers to loosely connected low cost processors. Despite a few efforts focusing on batch processing images, little attention has been paid to other aspects of making effective use of high performance distributed computing. Given the volume of the data for the problems of interest in biological image analysis, high-throughput applications can be intractable on sequential desktop computers. With the expected grow of high content images, it can be more so tomorrow. To address this problem, we propose a methodology to harness the power of HPC cluster for learning-based biological image analysis.

In this paper, we present a high performance tool for automatic model selection of biological image classification, built on top of the BIOCAT platform. Due to the large pool of candidate learning algorithms and variety of problems being studied, such effort can potentially bring dramatic enhancement to the learning process when a suitable design is available. Specifically, we implement a distributed scheme that enables BIOCAT to work efficiently with large sets of images and candidate algorithms, for adaptively selecting the most suitable model among the candidate learning models for a given task. We also show through extensive testing on benchmarking data sets that the high performance model selection not only dramatically increases the speed of the learning process, but also brings improved accuracy to several state-of-the-art data sets for bioimage classification. These achievements are attributed to the combined power of BIOCAT's adaptive model selection as well as the capability of distributed model evaluation.

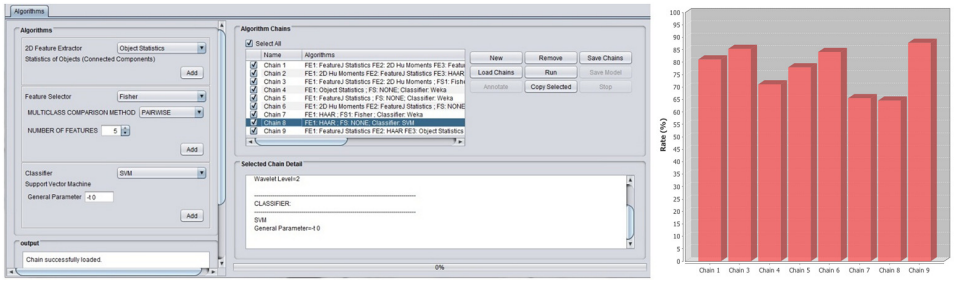

Figure 1: GUI of BIOCAT for building model chains and output of model comparison and selection.

The paper is organized as follows: Section II will introduce the background of BIOCAT and related tools/algorithms. Section III will detail the design of high performance model selection tool on a cluster. Section IV has experimental results, followed by discussions and conclusions.

2. Related Work

2.1 BIOCAT

BIOCAT (for BIOimage Classification and Annotation Tool) is a tool originated from our work of biological image annotation [11][12]. It provides a flexible and extensible platform for biological image classification based on learning models, handling both image sets and the regions of interest (ROI) in images. One related tool in the field is Wndchrm [6], which is a pioneering tool in bioimage classification. (Also related but with different focus are newer tools for image segmentation such as CellProfiler, Ilastik and Icy [13][14].) Different from Wncharm which is a command-line based tool and extracts the same number of features regardless of the problem, the regular release of BIOCAT includes a user-friendly graphical interface which facilitates the domain user (typically biologists) who are not computer scientists. BIOCAT has been implemented in Java to allow interoperability with ImageJ [15], which is a very popular and powerful library for biomedical image manipulation. The language also brings portability to the tool.

The ever-increasing volume of image data begs the question as to whether it is possible to employ high performance computing to speed up the biological image analysis of BIOCAT. High performance computing can be used at different stages of the analysis. Here we focus on addressing the efficient building of an image learning model, particularly, the speedup of the model selection process during training and model validation.

In this section, we will start with the introduction of the process that BIOCAT performs for adaptive model selection, then explain why the model is suitable for a distributed implementation on a cluster which will improve the performance dramatically.

2.2 Model Selection in BIOCAT

One important feature of BIOCAT compared to alternatives is its capability to adaptively select models for different problems. Currently, the common practice for model selection in the pattern recognition and learning community is often a manual, trial-and- error comparison, largely based on the practitioner's level of experience. On the other hand, some algorithms/tools chooses to include all the algorithm modules regardless of the problem at hand, for example the inclusion of a long list of various feature extractors. The former is labor

intensive, while the latter is computationally expensive during both training and application stages and not suitable for high-throughput analysis.

BIOCAT uses a different strategy: It promotes an adaptive model search. It defines a so-called algorithm chain: Consisting of individual algorithm modules as Lego-like candidate components, an algorithm chain represents the process flow in a model for image classification. It contains zero to many feature extractions/selectors and a classifier. The extracted features from multiple extractors are pooled, they can then run through selectors sequentially and eventually be classified. It can be conveniently built using the BIOCAT GUI and saved into chain files. Each chain represents a model.

A model search is then performed by going through the chains, evaluating and comparing their accuracies, using either training/testing sets or cross-validation, based on image sets or ROIs in the images according to the given problem. The model is thus considered adaptive for the problem since multiple combinations of algorithms (chains) are compared and the best performing one is picked. For the single desktop (i.e., PC) release of BIOCAT, chains are exhaustively evaluated in a linear fashion, then the conclusions are summarized and reported.

Figure 1 shows the GUI of BIOCAT tool for building the algorithms as well as the output that compares the accuracy of various chains involved in evaluation.

2.3 Algorithms in the Model

State-of-the-art algorithms from pattern recognition and machine learning community are included in BIOCAT as pluggable components. These components include feature extractors such as invariant feature transform, wavelet transform, Hessian and structural features as well as feature selectors and classifiers. While the extensible design of BIOCAT has made adding new algorithms as plug-ins easy (such as wrapping exiting libraries of FeatureJ [16] and Weka [17]), many algorithms are re-implemented since multidimensional bioimages require special attention to algorithm efficiency compared to other libraries. BIOCAT currently contains about 20 algorithms and it is a growing list. We briefly explain the following algorithms which are used in the experiments:

Feature Extractor:

- [I] Image Statistics: 15 features based on pixel intensity statistics including mean, standard deviation, median etc. [16].
- [H] Hu Moments: 8 Hu moments invariant to translation, scale and rotation.
- [W] Wavelet: HAAR wavelet coefficients. The number of features are the same as the original image size.
- [O] Object statistics: 8 features including statistics of the foreground objects in the image, such as their sizes and locations of centers of mass.
- [HE][S] Hessian and Structure Features [16].
- [3H] 3D Hu Moments: 3D extension of 2D Hu Moments. 8 features.
- [3W] 3D Anisotropic Wavelet: Haar Wavelet are extracted from each x-y plan then weighted sums are obtained for the z direction. The number of features is the same as the size of one slice.

Feature Selector:

- [F] Fisher Feature Selector. It selects the subset of extracted features based on Fisher's Criterion.

Classifier:

- [N][R] Nearest Neighbor (3NN by default) and Random Forest Classifier [17].
- [S] Support Vector Classifier (linear kernel by default) [18].

The advantage of such adaptive model selection, as stated before, is that the resulting model from selection is often a suitable model for the given problem. The outcome model of a selection is not necessarily complex in order to be effective for a given problem. In fact, sometimes a simple model can be very good (see the section of Experimental Results). Simpler models add great value to improve the performance of subsequent image analysis stages. However, the computational complexity of model selection process itself can be large due to two factors: one is related to the search strategy, which is linear to the number of chains being evaluated for the PC release, the

other is the complexity of the algorithms in the chain. Some of the learning algorithms are computationally complex. For example, SVM classifier can be slow when the data dimensionality is high and the number of samples is large (such as sets with a dimensionality greater than 1Meg and hundreds of images or more). If the extracted features are correlated to the original image size (e.g. Wavelet) then the related chain will be resource expensive for large sets and/or large image sizes. When multiple chains are involved and the computation is executed on PC, the time to select an appropriate model can become prohibitively long (see Section 4). On the other hand, since the chains can be evaluated independently, the chain-based model search of BIOCAT provides a good foundation for distributed computing on a high-performance cluster. In the next section, we detail a distributed design for high performance model selection of BIOCAT.

3. DESIGN

The BIOCAT's training logic is implemented such that a set of independent chains are evaluated serially. When a chain is evaluated, one or more algorithms are performed on the image set. In order to reduce the run time of the training application we focus on reducing the execution time by distributing the chain analysis across multiple processor cores and computing many simultaneously (i.e., in parallel).

When chains are evaluated in parallel, the theoretical performance improvement is proportional to the number of chains. In the parallelization strategy we present here each chain represents a unit of work and all such chains can be processed simultaneously. Therefore, when comparing the time it takes to process a given set of N chains sequentially to the time it takes to process the same set of chains in parallel the theoretical limit of the expected reduction in execution time is proportional to $1/N^{th}$ the time it takes to process the chains sequentially. The details of our parallelization method is the focus of the next section.

3.1 Using a high-performance cluster

The tool is deployed and run on a computer cluster at Northern Illinois University named Gaea. The cluster has sixty compute nodes and two login/administrative nodes. The compute nodes are connected via a full 1:1 non-blocking Infiniband and Ethernet switch interconnects. Each compute node has two 6-core CPUs (Intel Xeon X5650, 2.67GHz), 72 GB RAM, two 448-core GPUs (NVIDIA Tesla M2070, 6 GB

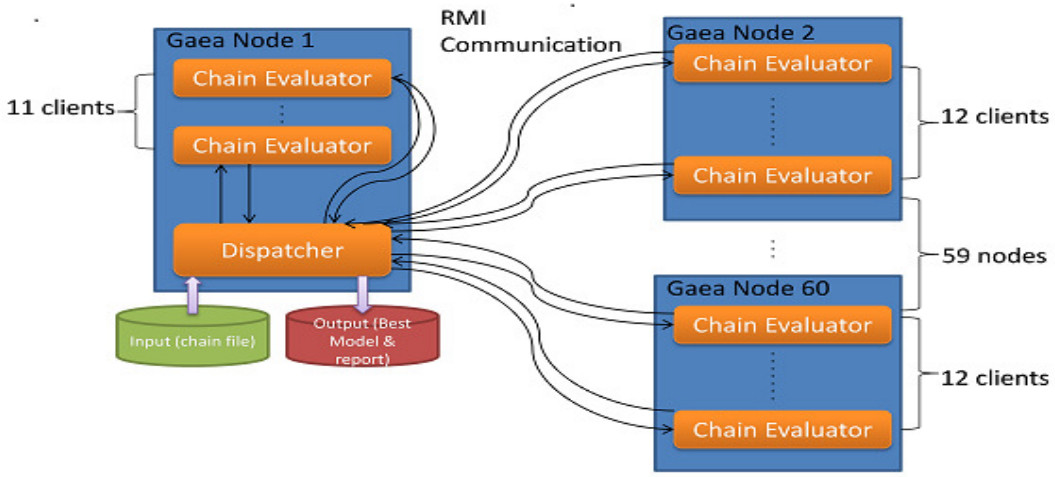

Figure 2: The communication diagram of distributed model selection.

RAM), and 2 TB of local storage. The cluster also has a 192 TB disk storage array with an I/O server and data passes directly to/from the compute nodes via the Infiniband switch.

By allowing the chains to be evaluated in parallel, as many as 720 can be evaluated simultaneously across Gaea's CPU cores. Given that the computational complexity of the chains are not identical, the total execution time for such a parallel job will be that of the single longest chain. As discussed in [19] and [20], it is not clear how well multithreaded Java applications scale on SMP. We, therefore, only consider single-threaded Java applications where only one is executed per cluster node in this study.

Gaea is used as a multi-user general purpose cluster. Jobs are executed on its computing nodes using a batch queuing system named Torque (adaptivecomputing.com). In the simplest terms, when a job is started on the cluster one single node runs a Portable Batch System (PBS) shell script responsible for:

- Distributing and starting the application processes across multiple computing nodes.

- Orchestrate any required clean up when the application has completed.

3.2 Distributed Model Selection

To operate in this environment the BIOCAT application was altered by adding a second entry point that allows the 'Chain Evaluator' training logic to be executed directly from the command line (sans GUI) and report the evaluation results to a central 'Dispatcher' process to record the best model and generate a statistical report of the evaluated chains (Figure 2).

Distributing the BIOCAT Java application across the compute nodes requires the starting of one or more JVMs on each compute node and executes an application instance, providing each such application instance with a chain to evaluate and a method to collect and record the results from each evaluation.

Placing an ssh command in a loop within the PBS script is a convenient method for starting and passing an ID to each application instance across the cluster. The ID is used as an index into a vector of all chains to be evaluated.

For training, the total number of tasks is given by the number of chains that are to be evaluated. The chain evaluation process requires no communication between.

The application with task ID zero is called the 'Dispatcher' (Figure 2) and is assigned the job of collecting and reducing the results from all chain evaluations into a sorted list of chains where the most fit (best) chain is first. The other tasks are 'Chain Evaluators' and are assigned ID numbers from one through the number of chains to be evaluated. The chain assigned to each Chain Evaluator is given by the ID of the task.

Communication between the Dispatcher and the Chain Evaluators is performed using the Java Remote Method Invocation (RMI) package. RMI provides a mechanism where a member method in an object instance in one JVM can be called from code in a different JVM, possibly executing on a different computing node.

The distributed version of BIOCAT assigns the Dispatcher the role of an 'RMI server' and the Chain Evaluators the role of 'client'. As such the Dispatcher registers itself as a service to the RMI registry so the Chain Evaluators' can locate it. Note that the Chain Evaluators are provided the node name where the registry is deployed by the PBS script when they are started as is required when using RMI.

Once the services provided by the Dispatcher have been located, the Chain Evaluators invoke a function on the Dispatcher and pass it an object with a reference back to itself allowing the Dispatcher the option to then invoke methods on the Chain Evaluator at a later time.

The Chain Evaluators then proceed to process the chain whose ID matches that of itself. As depicted in Figure 2, each Chain Evaluator executes on one of the cluster's compute nodes. It reads its chain directly from the cluster's disk server node over the Infiniband network. Thereafter processing each chain yields either a fitness ranking of the chain or a processing error of some kind (illegal training image format, invalid chain definition, depletion of a system resource, etc.). The result is then reported back to the Dispatcher and the Chain Evaluator task terminates.

As the Dispatcher collects the results it sorts them by accuracy on testing set or cross-validation. When all Chain Evaluators have reported their results, the Dispatcher logs the results and generates a graphical report. Should any Chain Evaluator report an error condition, the problem is logged and all Chain Evaluators are notified that the job has failed terminating the application immediately.

Table 1. Biological Image Data Sets.

Image Set	Problem Domain	Modality	Training Images	Testing Images	# of Classes	Image Size in Number of Pixels and Dimension	Image Set Size in Bytes
K150	Expressed Neurons of Drosophila Brain	Fluorescence	12	8	4	500K (530*998 24 bit)	30M
Pollen	Subcellular pattern	Fluorescence	630	-	7	225 (25*25)	142K
RatLiver	Liver tissue of different gender	Brightfield	256	-	2	1.4M (1388*1040 32 bit)	1.5G
CHO	Subcellular pattern	Fluorescence	200	-	5	190K (512*382)	39M
Muscle Age	C. elegans muscles at different ages	Fluorescence	252	-	4	1.9M(1600*1200 16 bit)	1G
Binucleate	Binucleate and regular cells	Fluorescence	40	-	2	1.3M (1280*1024 16 bit)	105M
RNAi	phenotypes of drosophila cells	Fluorescence	200	-	10	1M (1024*1024 16bit)	400M
Hela	Subcellular pattern	Fluorescence	860	-	10	146K(382*382 16bit)	250M
Termbulb	C. elegans terminal bulb at different ages	DIC	970		7	90K(300x300 16 bit)	175M
K1503D	Expressed Neurons of Drosophila Brain	Fluorescence	12	8	4	28M (512*512*108 24 bit)	1.7G

3.3 Design Notes

The choice of RMI was made by assessing the applicability of the communication packages currently available starting with that provided in the JDK. RMI is part of the core JDK functionality and addresses the needs of BIOCAT. RMI has been part of the JDK since version 1.1 [Java 2 Platform API Specification] and is therefore available everywhere without the need for installing extra packages. The BioCat training application exchanges at most three small messages between the Dispatcher and each Chain Evaluator thus the performance of the communication system is insignificant on the overall performance.

Using the PBS script to distribute the application across the cluster nodes allows the geometry of the application to be managed by the same PBS script used by Torque. This is easily and quickly changed to suit the needs of the user. With simple modification the same script can be used on a COW (collection of workstations) or other clusters lacking a scheduling system.

The changes made to the BIOCAT tool to allow distributed operation were to add a second entry point allowing the execution of the training logic without a user interface. This approach allows the same source and executable to run on the desktop or computer cluster distributed or not.

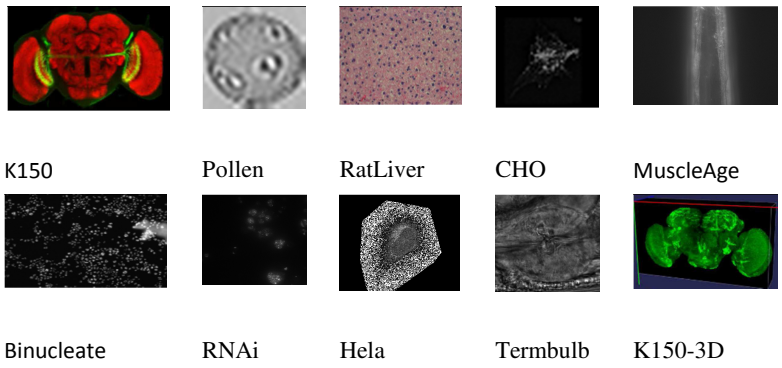

Figure 3. Example images in the data sets.

4. EXPERIMENTAL RESULTS

We conducted the experiments on ten data sets, including eight from the well-known benchmarking biological image dataset IICBU [21] and two additional neuronal image sets for both 2D and 3D K150 data sets. Table 1 lists the details of the data sets. Figure 3 shows some example images of the sets. As we can see, the sets are from different biological domains with different characteristics and some have a large image size in terms of number of pixels that need to be processed. For sets without the separation of training and testing, the speed and recognition accuracy for model selection are both reported based on five-fold cross-validation. For sets that are too small to split into five subsets (k150, k1503D), a separation of training and testing sets is used.

Ten algorithm chains are built for the purpose of comparing the performances of on regular model selection in BIOCAT versus the HPC model selection. Among the chains, eight contain feature extracting algorithms for 2D data sets, the rest are for 3D data sets. These candidate chains are chosen due to their proven effectiveness on some datasets when tested using the regular model selection on a PC.

Details of the pattern recognition algorithm modules in each chain are shown in Table 2. Based on the included algorithms modules, some chains are more computationally complex than others and some may have a higher memory requirement. For example, the object statistics extractor does several scans of the image to yield features based on the foreground objects and their centers of masses. It is typically slow on large images and sets which are used in Chain #6, #7, and #8. On the other hand, the HAAR wavelet features yield the same number of features as the number of original pixel sizes, which imposes a high memory requirement for large images, especially when no subsequent feature selection is performed, such as Chain #5. The support vector machine classifier is computationally complex and takes more time on large sets, which is also in Chain #5.

Table 2 Algorithms in the chains (acronyms are explained in Section 2.3).

Chain #	I	H	W	O	HE	S	3H	3W	F	N	R	S
1	X	X							x		X	
2	X									X		
3	X	X									X	
4			X						x	X		
5			X									X
6	X		X	X					x		X	
7				X							X	
8	X	X		X	X	X			x		X	
9							x				x	
10								x				x

Figure 4 shows the speeds of the regular sequential model selection on PC and the high performance distributed model selection for different chains and on different databases. In all graphs, the X coordinate is the number of chains compared during the model selection process and Y coordinate is the time needed to complete the process. The performance for the regular sequential model selection tool is reported on the same computer as the parallel version (Intel Xeon 2.67GHz) for a fair comparison. In Figure 4, models with up to eight chains are compared on data sets Pollen, MuscleAge, RNAi and CHO. Other data sets present a similar speed trend and are not included in the paper to save space.

We can see from Figure 4 that the high-performance tool's time with respect to the number of chains involved in selection (the solid lines in the figure) is bounded by constant time on all sets. It is essentially bounded by the runtime of the longest single job (chain 5 in our experiments). This is due to the parallel design of the tool as explained in Section III. It brings about four fold performance increase when eight chains are evaluated during selection. For example, for the data set RNAi, it takes the regular model selector 10800 seconds (3 hours) to finish selecting the best model out of the eight chains while it only takes the high-performance version 2700 seconds (45 minutes), which is a 8100 second speedup and 1/4 of the time of the regular version. The more chains are included in the comparison, the more dramatic the reduction in execution time. We expect this trend to continue to the capacity of the parallelism offered the cluster.

Table 3 lists the accuracy we obtained on the data sets. The accuracy is reported as the proportion of correctly classified images over the total number of images in the testing set. In the case of cross-validation, the average accuracy over all folds is reported. The best result of model selection is reported in the row of "Accuracy". The row of "Literature" shows the result of the tool Wndcharm or the original literature where the data set was first reported, whichever is higher (k150 sets do not have literature results and for binucleate we reported results up to four chains which already yields 100% accuracy). From the results we have several observations:

1. BIOCAT obtains the same or better accuracy than literature on seven out of ten data sets (including k150 sets which we obtained 100% accuracy). For the MuscleAge data set, an accuracy of as high as 90% is obtained by chain #6, whereas the state-of-the-art accuracy is only 53%. We also note that Wndchm is a computationally complex tool. To achieve the reported results, thousands of features including Zernike moments of factorial complexity are extracted for any given problem. The results we obtain are especially promising considering that we are only comparing up to eight chains during the model selection (for 2D image sets). BIOCAT and the high-performance model selection tool can allow a much larger scale of comparison, which is expected to deliver more benefits in both efficiency and effectiveness.

2. Different models are suitable for different classification problems. Winning chains differ on various data sets. A chain that works very well on one problem may perform poorly on another problem and vice versa. The conclusion seems obvious from the viewpoint of pattern recognition since a domain demands its own set of suitable features and classifiers. However, BIOCAT facilitates such comparison by running the evaluation automatically and a model can be chosen adaptively for the given problem, which alleviates the

manual labor or blind inclusion of unsuitable algorithms. High performance selection, as described in this paper, scales up the possibility of the ranges of the models that can be compared in an efficient way.

3. Simple models can be powerful. One interesting observation, related to the above point of "various algorithms having various applicability" is that simple chains can also work well. As an example, Chain 1 -- with has a low computational complexity -- gains very good performance on Pollen and CHO sets. Chain 2, also a simple chain, works well on the RatLiver set. Such a model selection approach shifts the complexity of learning toward the selection stage and can bring performance benefits for later stages when a non-complex model is applied to classifying images especially in high throughput scenario. Such advantage is also beneficial for processing and annotating regions of interest in large images with higher dimensions. As we are aware, high throughput and high content are the trends of biological imaging, so informatics tools must scale up correspondingly. Comparatively, alternative tools using a large number of features regardless of the problem may be an expensive operation and not suitable for high-throughput setting.

5. CONCLUSIONS

The high performance model selection tool for learning-based biological image classification is able to run on a high performance cluster and scale bounded by constant time due to its distributed design. The design of the tool makes it possible to reuse the existing BIOCAT platform without impacting its internal architecture.

BIOCAT's adaptive model selection, tested on ten data sets, shows that a simple model may suffice, which will be very helpful to the later stage of applying the model to large sets of images or ROIs of a high-dimensional image. It will facilitate the high-throughput analysis that is critically needed for biological image analysis. The model also yields some improved accuracy on several benchmarking data sets,

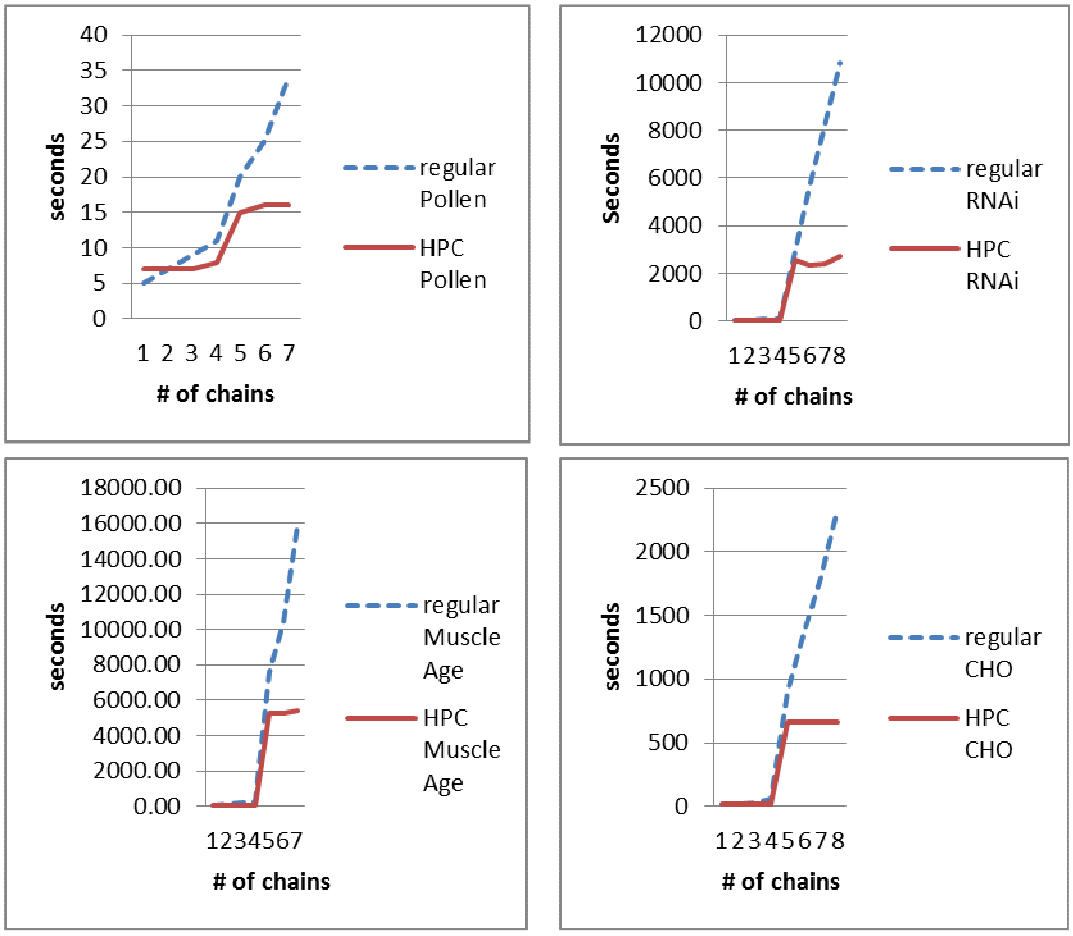

Figure 4: Speed comparison of regular and high-performance model selection. The dotted lines are for the regular version. The solid lines are for the HPC version.

which shows its power in effective model selection.

While the experiments in this paper has been conducted on a cluster (Gaea), the design employed in the paper can be extended to other (smaller and/or loosely connected) clusters due to the portable RMI-based strategy for distributed environment.

As next steps we will run larger scale experiments including those on high dimensional data sets. We will also explore high performance parallel strategies for applying algorithms when models are applied to regions in a large, high dimensional images for the purpose of object annotation and quantification.

Table 3: Accuracy of chains for each data set (%).

Image Set	K150	Pollen	RatLiver	CHO	Muscle Age	Binucleate	RNAi	HeLA	TermBulb	K1503D
Chain1	50	**88**	97	**94**	62	72	48	58	50	-
Chain2	63	71	**99**	90	56	85	46	52	45	-
Chain3	33	75	98	92	60	78	49	58	**51**	-
Chain4	33	83	85	67	84	**100**	21	32	28	-
Chain5	**100**	63	75	55	46	-	21	48	44	-
Chain6	50	80	96	90	**90**	-	42	62	44	-
Chain 7	75	84	96	91	54	-	38	63	38	-
Chain 8	36	84	98	93	89	-	**55**	**66**	49	-
Chain 9	-	-	-	-	-	-	-	-	-	65
Chain 10	-	-	-	-	-	-	-	-	-	**100**
Accuracy	**100**	88	**99**	**94**	**90**	**100**	55	66	**51**	**100**
Literature	-	97	99	93	53	100	82	84	49	-

6. ACKNOWLEDGMENTS

Our thanks to Dr. Julie H. Simpson at HHMI for providing the K150 data sets. Our thanks also to Professor Clyde Kimball at Northern Illinois University and his support in providing access to the cluster Gaea. This work was sponsored in part through US Department of Energy contract DE-SC0005135.

7. REFERENCES

[1] K. W. Eliceiri, M. R. Berthold, I. G. Goldberg, L. Ibáñez, B. S. Manjunath, M. E. Martone, R. F. Murphy, H. Peng, A. L. Plant, B. Roysam, N. Stuurmann, J. R. Swedlow, P. Tomancak, and A. E. Carpenter, "Biological imaging software tools," *Nature methods*, vol. 9, no. 7, pp. 697–710, Jan. 2012.

[2] W. B. Grueber, C.-H. Yang, B. Ye, and Y.-N. Jan, "The development of neuronal morphology in insects," *Current biology : CB*, vol. 15, no. 17, pp. R730–8, Sep. 2005.

[3] S. Lamichhane and J. Zhou, "Automatic Analysis of Dendritic Territory for Neuronal Images," in *International Symposium on Bioinformatics Research and Applications*, 2012.

[4] J. Zhou and H. Peng, "Counting cells in 3D confocal images based on discriminative models," in *ACM Conference on Bioinformatics, Computational Biology and Biomedicine (ACM BCB)*, 2011.

[5] T. ER, F. MA, W. NK, B. A, and A. FM, "Microsporidia are natural intracellular parasites of the nematode Caenorhabditis elegans," *PLoS Biol*, vol. 6: e309, 2008.

[6] L. Shamir, N. Orlov, D. M. Eckley, T. Macura, J. Johnston, and I. G. Goldberg, "Wndchrm – an open source utility for biological image analysis," *Source code for biology and medicine*, vol. 3, no. 1, p. 13, 2008.

[7] M. Velliste and R. F. Murphy, “Automated determination of protein subcellular locations from 3D fluorescence microscope images,” in *Proceedings of the IEEE International Symposium on Biomedical Imaging*, 2002, pp. 867–870.

[8] A. E. Carpenter, T. R. Jones, M. R. Lamprecht, C. Clarke, I. H. Kang, O. Friman, D. A. Guertin, J. H. Chang, R. A. Lindquist, J. Moffat, P. Golland, and D. M. Sabatini, “CellProfiler: image analysis software for identifying and quantifying cell phenotypes.,” *Genome Biology*, vol. 7, no. 10, p. R100, 2006.

[9] J. Luisi, A. Narayanaswamy, Z. Galbreath, and B. Roysam, “The FARSIGHT Trace Editor: An Open Source Tool for 3-D Inspection and Efficient Pattern Analysis Aided Editing of Automated Neuronal Reconstructions,” *Neuroinformatics*, vol. 9, no. 2–3, pp. 305–15, Sep. 2011.

[10] “BIOCAT: http://faculty.cs.niu.edu/~zhou/tool/biocat/.” .

[11] J. Zhou and H. Peng, “Automatic recognition and annotation of gene expressions of Fly Embryos,” *Bioinformatics*, vol. 23, no. 5, pp. 589–596, 2007.

[12] H. C. Peng, F. H. Long, J. Zhou, G. Leung, M. Eisen, and E. Myers, “Automatic image analysis for gene expression patterns of fly embryos,” *BMC Cell Biology*, vol. 8, 2007.

[13] F. de Chaumont, S. Dallongeville, N. Chenouard, N. Hervé, S. Pop, T. Provoost, V. Meas-Yedid, P. Pankajakshan, T. Lecomte, Y. Le Montagner, T. Lagache, A. Dufour, and J.-C. Olivo-Marin, “Icy: an open bioimage informatics platform for extended reproducible research,” *Nature methods*, vol. 9, no. 7, pp. 690–6, Jan. 2012.

[14] A. Kreshuk, C. N. Straehle, C. Sommer, U. Koethe, M. Cantoni, G. Knott, and F. a Hamprecht, “Automated detection and segmentation of synaptic contacts in nearly isotropic serial electron microscopy images.,” *PloS one*, vol. 6, no. 10, p. e24899, Jan. 2011.

[15] M. D. Abramoff, P. J. Magalhaes, and S. J. Ram, “Image Processing with ImageJ,” *Biophotonics International*, vol. 11, no. 7, pp. 36–42, 2004.

[16] E. Meijering, M. Jacob, J.-C. F. Sarria, P. Steiner, H. Hirling, and M. Unser, “Design and validation of a tool for neurite tracing and analysis in fluorescence microscopy images.,” *Cytometry. Part A : the journal of the International Society for Analytical Cytology*, vol. 58, no. 2, pp. 167–76, Apr. 2004.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA Data Mining Software: An Update,” *SIGKDD Explorations*, vol. 11, no. 1, 2009.

[18] C.-C. Chang and C.-J. Lin, “LIBSVM: a library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

[19] B. A. Caromel, V. Bodnartchouk, D. Taboada, C. Delbé, F. Huet, and G. L., “Current State of Java for HPC,” 2008.

[20] Z. CAO, W. HUANG, and J. M. CHANG, “A study of java virtual machine scalability issues on smp system,” *iiswc*, vol. 0, pp. 119–128, 2005.

[21] L. Shamir, N. Orlov, D. Mark Eckley, T. J. Macura, and I. G. Goldberg, “IICBU 2008: a proposed benchmark suite for biological image analysis.,” *Medical & biological engineering & computing*, vol. 46, no. 9, pp. 943–7, Sep. 2008.