GPU-Accelerated Machine Learning Techniques Enable QSAR Modeling Of Large HTS Data

Edward W. Lowe, Jr., Mariusz Butkiewicz, Nils Woetzel, Jens Meiler

Abstract—**Quantitative structure activity relationship (QSAR) modeling using high-throughput screening (HTS) data is a powerful technique which enables the construction of predictive models. These models are utilized for the *in silico* screening of libraries of molecules for which experimental screening methods are both cost- and time-expensive. Machine learning techniques excel in QSAR modeling where the relationship between structure and activity is often complex and non-linear. As these HTS data sets continue to increase in number of compounds screened, extensive feature selection and cross validation becomes computationally expensive. Leveraging massively parallel architectures such as graphics processing units (GPUs) to accelerate the training algorithms for these machine learning techniques is a cost-efficient manner in which to combat this problem. In this work, several machine learning techniques are ported in OpenCL for GPU-acceleration to enable construction of QSAR ensemble models using HTS data. We report computational performance numbers using several HTS data sets freely available from PubChem database. We also report results of a case study using HTS data for a target of pharmacological and pharmaceutical relevance, cytochrome P450 3A4, for which an enrichment of 94% of the theoretical maximum is achieved.**

Keywords-Machine learning, GPU-Accelerated, High-throughput screening, quantitative structure activity relationship, OpenCL

I. Introduction

Quantitative structure activity relationships (QSAR) correlate complex non-linear relations between chemical structure and biological activity [1, 2]. In its conception, QSAR investigated biological activity of compounds as related to their physicochemical properties using linear regression [3, 4]. Modern QSAR techniques couple machine learning with 2-D and 3-D molecular descriptors [5, 6]. QSAR modeling has proven useful as a virtual screening tool in drug discovery [1]. Machine learning techniques as applied to QSAR modeling have been successful not only in virtual screening, but also for the prediction of parameters of pharmacological and pharmaceutical relevance[7].

Machine Learning algorithms have shown exciting potential in developing QSAR models to predict biological activity data [8-10]. The algorithms derive a model based on data acquired from experimentally screened compound libraries. These models are then used to screen large virtual libraries *in silico.* The result of this virtual screen is a prioritized subset of compounds for acquisition or synthesis that is enriched for active molecules [9].

Machine learning model ensembles can reduce prediction error by compensating misclassification with the consensus of all trained models [11]. Previous studies have shown that the performance of ensemble classifiers trained on HTS data can be significantly better than single models alone [8, 12, 13].

As high-throughput screening (HTS) data becomes more readily available through such initiatives as PubChem[14], the impact of QSAR modeling becomes increasingly apparent [8, 9]. While the chemical screening libraries used in HTS campaigns are on the order of 10^3 – 10^6 molecules, the available chemical space is much greater (10^8), with over 60,000,000 substances currently catalogued in the Chemical Abstracts database [15]. QSAR models constructed using machine learning techniques can be used to virtually screen the available chemical space for which experimental methods are prohibitively time and cost expensive.

While prediction with machine learning models for virtual screening campaigns is both cost and time efficient, extensive feature selection with cross validation becomes computationally expensive. As the size of HTS libraries continues to grow, complete optimization of these QSAR models becomes more difficult due to this computational intensity. One method of ameliorating this expense is through leveraging massively parallel architectures such as graphics processing units to accelerate the training algorithms.

Here, we present results for several machine learning techniques accelerated using OpenCL for training machine learning models on HTS data implemented in the BioChemistry Library (BCL), a C++ library developed at Vanderbilt University. The BCL provides functionality for generating molecular descriptors, training artificial neural networks, support vector machines with the extension for regression, kappa nearest neighbor, Kohonen networks, and decision trees, as well as all tools necessary for analysis, virtual screening, and similarity analysis. Virtual screening tools and similarity measures are also accelerated using OpenCL implementations. The performance improvements achieved through this GPU-acceleration of these methods is up to 2 orders of magnitude (283x). The advantages over existing implementations of GPU-accelerated machine learning techniques are the use of the open standard OpenCL where most implementations utilize CUDA or CUDA libraries, and also tight integration with cheminformatics specific tools

This work is supported by 1R21MH082254 and 1R01MH090192 to Jens Meiler. Edward W. Lowe, Jr. acknowledges NSF support through the CI-TraCS Fellowship (OCI-1122919).

Edward W. Lowe, Jr., Ph.D., is a Research Assistant Professor of Chemistry at the Center for Structural Biology at Vanderbilt University, Nashville TN, 37232;

Mariusz Butkiewicz is a graduate student in Chemistry at Vanderbilt University, Nashville TN, 37232;

Nils Woetzel, Ph.D. is a postdoctoral fellow at the Max Planck Institute for Developmental Biology, Tuebingen, 72076, Germany

Jens Meiler is an Associate Professor of Chemistry at the Center for Structural Biology at Vanderbilt University, Nashville TN, 37232;

Corresponding author: Edward W. Lowe, Jr., edward.w.lowe@vanderbilt.edu;

within the BCL. Model performance results are also provided for a case study of a pharmacologically relevant target, cytochrome P450 3A4.

II. Machine Learning Techniques

Artificial neural networks and support vector machine algorithms have OpenCL implementations allowing them to run on a graphics processing unit (GPU), and are also threaded using posix threads. These algorithms are well-established in chemistry and biology for quantitative structure property relationship modeling[7] and have been successfully applied to quantitative structure activity relationship (QSAR) modeling using HTS data[8, 9]. The BCL also contains implementations of kappa nearest neighbor, Kohonen Network, and decision trees. Kohonen Networks and decision trees implemented in the BCL are not multi-threaded nor are they OpenCL enabled. Kappa nearest neighbor does have an OpenCL implementation, but is not multi-threaded using posix threads. This is reflected in the timings in Table III. These methods in the context of QSAR have been described in our previous work[7-9, 16].

A. *Artificial Neural Network*

Artificial neural networks (ANN) [17-20] are modeled after the brain in that layers of neurons are linked by weighted connections. Input data are summed by their weights, an activation function is applied, and that output is used as input into the next layer, and so on. This is described as follows:

$$y_j = f(net_j) = f(\textstyle\sum_{i=1}^{d} x_i w_{ji}) \quad 1 \le j \le n_H \tag{1}$$

$$z_k = f(net_k) = f(\textstyle\sum_{j=1}^{n_H} y_j w_{kj}) \quad 1 \le k \le c \tag{2}$$

where d is the number of features or dimensionality, $f(x)$ is the activation function i.e. sigmoid or Gaussian, n_H is the number of neurons in the hidden layer, c is the number of outputs, w_{ji} are the weights, and x_i is the input data. In this supervised training implementation, the differences between the calculated z_k output and the known experimental value t_k allow for error back-propagation:

$$\Delta w_{kj} = \eta(t_k - z_k) f'(net_k) y_j \tag{3}$$

$$\Delta w_{ji} = \eta\left[\textstyle\sum_{k=1}^{c} w_{kj} (t_k - z_k) f'(net_k)\right] f'(net_j) x_i \tag{4}$$

The weight changes attempt to minimize the root mean square deviation (*rmsd*) between calculated and target values.

B. *Support Vector Machine with Extention for Regression*

SVM learning with extension for regression estimation (SVR)[21] represents a supervised machine learning approach successfully applied in the past[22]. The main principles focus on linear functions defined in high-dimensional feature space [23], risk minimization according to Vapnik's ε - intensive loss function, and structural risk minimization [24] of a risk function consisting of the empirical error and the regularized term.

Given an error threshold ε, SVRs seek to approximate a function resembling in a ε –tube around the approximation that maximizes the coverage of training data points in feature space. The applied distance measure in the high-dimensional space is a Radial Basis Function Kernel.

$$K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\gamma^2}} \tag{5}$$

III. Virtual High-Throughput Screening

Upon successful construction of optimized models, a virtual high-throughput screen (vHTS) can be performed. Optimized models are used to calculate a predicted activity of molecules screened. This has been implemented in OpenCL for ANN and SVM to achieve hardware-acceleration.

IV. Similarity Analysis

Similarity analysis is common practice in both machine learning and cheminformatics alike. Chemical similarity measures are the basis for database screening, and clustering. Here, we provide a GPU-accelerated tool for calculating several similarity coefficients (Table I) which serve as distance measures for bcl::Cluster[25]. This similarity application can operate on either feature vectors directly or SD files, generating the desired feature vectors for molecules on-the-fly. SD files are an industry standard file format for describing molecules and their atomic coordinates.

Table I
Similarity Coefficients

Similarity Measure	**Description**
Tanimoto	$\frac{\sum_{i=1}^{n} x_{iA}x_{iB}}{\sum_{i=1}^{n}(x_{iA})^2 + \sum_{i=1}^{n}(x_{iB})^2 - \sum_{i=1}^{n} x_{iA}x_{iB}}$
Cosine	$\frac{\sum_{i=1}^{n} x_{iA}x_{iB}}{\sqrt{\sum_{i=1}^{n}(x_{iA})^2 \sum_{i=1}^{n}(x_{iB})^2}}$
Dice	$\frac{2\sum_{i=1}^{n} x_{iA}x_{iB}}{\sum_{i=1}^{n}(x_{iA})^2 + \sum_{i=1}^{n}(x_{iB})^2}$
Euclidean	$\frac{1}{1+\sqrt{\sum_{i=1}^{n}(x_{iA}-x_{iB})^2}}$
Manhattan	$\frac{1}{1+\sum_{i=1}^{n}\lvert x_{iA}-x_{iB}\rvert}$

V. OpenCL Implementations

Hardware acceleration of several machine learning techniques of interest in this work, ANN, SVM, and kNN, enable thorough feature selection and parameter optimization of large HTS data sets which would otherwise be computationally prohibitive. Here, we provide implementation details for the OpenCL hardware-acceleration achieved in this work for ANN and SVM as the kNN implementation is taken from previous work by Garcia *et al*[26]. All reduction and matrix-vector operation OpenCL kernels use a block size of 256x1 while transposition, matrix-matrix multiplication, and pairwise similarity measurement OpenCL kernels use a block size of 16x16. Shared memory and register tiling is used whenever possible.

A. *Artificial Neural Network Training*

During ANN training using OpenCL, nearly all calculations are performed on the GPU with the exception of partial reductions which are completed on the CPU. The training proceeds through batch training such that matrix-matrix operations are leveraged. The OpenCL kernels written for this implementation include transpose, matrix-matrix multiplication, sigmoid, derivative of sigmoid, matrix-matrix addition, calculation of error terms *k* and *j*, column-wise reduction sum, calculation of change terms, and *rmsd* calculation. The implementation is outlined in Figure 1.

$$Hidden = N * W_{JI}^{T}$$

$$Hidden \mathrel{+}= Bias_J$$

$$Y = Sigmoid(Hidden)$$

$$Output = Y * W_{KJ}^{T}$$

$$Output \mathrel{+}= Bias_k$$

$$Z = Sigmoid(Output)$$

$$Error_K = (T - Z) \cdot Sigmoid'(Z)$$

$$I = Error_K * W_{KJ}$$

$$Error_J = I \cdot Sigmoid'(Y)$$

$$SlopeBias_J = ColReduction(Error_J)$$

$$SlopeBias_K = ColReduction(Error_K)$$

$$SlopeW_{JI} = Error_J^T * N$$

$$SlopeW_{KJ} = Error_K^T * Y$$

$$\Delta Bias_J = \eta * SlopeBias_J + \alpha * \Delta Bias_J$$

$$\Delta Bias_k = \eta * SlopeBias_K + \alpha * \Delta Bias_K$$

$$\Delta W_{JI} = \eta * SlopeW_{JI} + \alpha * \Delta W_{JI}$$

$$\Delta W_{KJ} = \eta * SlopeW_{KJ} + \alpha * \Delta W_{KJ}$$

$$Bias_J \mathrel{+}= \Delta Bias_J$$

$$Bias_K \mathrel{+}= \Delta Bias_K$$

$$W_{JI} \mathrel{+}= \Delta W_{JI}$$

$$W_{KJ} \mathrel{+}= \Delta W_{KJ}$$

Figure 1: One epoch of the OpenCL ANN training algorithm where N is the feature matrix of training data, W_{JI} and W_{KJ} are weight matrices, $Bias_J$ and $Bias_K$ are bias vectors, η is the training rate, α is the training momentum, and (·) denotes element-wise operation

B. *Support Vector Machine Training*

For the SVM training algorithm, only the more computationally expensive portions of the algorithm are offloaded to the GPU for acceleration. The implementation uses sequential minimal optimization[27] and offloads kernel matrix calculation (RBF, Polynomial, Linear), all ArgMin and ArgMax functions, bias calculations, and the updating of alpha status and gradients.

C. *Prediction*

For prediction using ANN models, the first six functions outlined in Figure 1 are used to arrive at the predicted output. This leverages the most efficient OpenCL kernels in the algorithm and achieves significant performance acceleration.

D. *Similarity Measures*

The similarity measures used in this work utilize a general pairwise OpenCL kernel scheme reported by

```
__kernel void Dice
(
  __global float* OUTPUT,
  __const __global float* INPUT,
  __local float* As,
  __local float* Bs,
  __const int ROWS,
  __const int COLS
)
{
  int bx = get_group_id( 0), by = get_group_id( 1);
  int tx = get_local_id( 0), ty = get_local_id( 1);
  int A_Begin = by * BLOCK_SIZE * COLS;
  int B_Begin = bx * BLOCK_SIZE * COLS;
  int A_End = A_Begin + COLS – 1, A_idx, B_idx, k, o;
  float a_b_sum = 0.0, a2_sum = 0.0, b2_sum = 0.0, a = 0.0, b = 0.0;
  for( A_idx = A_Begin, B_idx = B_Begin; A_idx <= A_End;
A_idx += BLOCK_SIZE, B_idx += BLOCKSIZE)
  {
    As[ ty * BLOCK_SIZE + tx] = INPUT[ A_idx + ty * COLS + tx];
    Bs[ tx * BLOCK_SIZE + ty] = INPUT[ B_idx + ty * COLS + tx];
#pragma unroll
    for( k = 0; k < BLOCK_SIZE; k++)
    {
      a = As[ ty * BLOCK_SIZE + k];
      b = Bs[ k * BLOCK_SIZE + tx];
      a_b_sum += a * b;
      a2_sum += a * a;
      b2_sum += b * b;
    }
    barrier( CLK_LOCAL_MEM_FENCE);
  }

  o = by * BLOCK_SIZE * ROWS + ty * ROWS + bx * BLOCK_SIZE + tx;
  OUTPUT[ o] = 2 * a_b_sum * native_recip( a2_sum + b2_sum);
}
```

Figure 2: OpenCL kernel for the pairwise calculation of Dice coefficient.

Chang *et al*[28] for the CUDA implementation of Manhattan distance calculation. Each block of threads calculates a sub-matrix of the resulting output where each thread is responsible for one output value. The OpenCL kernel for Dice is shown in Figure 2.

VI. General Performance

Performance measurements were taken using several HTS data sets acquired through PubChem. These data sets are summarized in Table II. Actives are those molecules which elicit the desired response in the biological experiments while inactives do not. Performance numbers are reported for each machine learning technique for both CPU and GPU implementations where appropriate. Timings were performed on a Dell T3500 workstation equipped with 12GB RAM and an Intel(R) Xeon(R) W3570@3.20GHz running 64-bit CentOS 5.2. The OpenCL GPU implementation timings were performed on NVIDIA GTX 480.

Table II
Data Set Composition

Data Set ID	Actives	Inactives
AID884	3,438	7,066
AID893	5,398	65,259
AID1445	883	206,897

A. *Training*

Each data set was obtained from the PubChem database and was trained using the machine learning methods available within the BCL to predict the inhibition values (IC_{50}) as output. Thus, these are not simply classifications as active/inactive.

The resulting performance numbers are shown in Table III. For ANN, a three layer network was trained using simple propagation for 100 epochs with 16 neurons in the hidden layer. SVM was trained 100 iterations with a radial basis function kernel with gamma of 0.1 and a cost of 1.0. KNN was performed using a kappa of 11. Kohonen networks used a map of 20 x 20 nodes for 100 iterations. DT was performed using information gain. The objective function used for all methods was *rmsd.* Times reported in Table III are in seconds with CPU/CPU_6-core/GPU. These results are to summarize the performance of the implementations and specifically the acceleration achieved through threading and OpenCL acceleration. As such, the number of epochs may not have been sufficient to fully train the models, which was not the intent of this performance evaluation since the OpenCL implementations give identical results to the C++ implementations.

TABLE III
TRAINING PERFORMANCE IN SECONDS

ML Method	AID884	AID893	AID1445
ANN	109/21/1	1151/193/10	3660/661/32
SVM	14/6/0.4	145/61/5	441/200/14
KNN	7/*/0.4	714/*/25	6118/*/90
Kohonen	289/*/*	4184/*/*	14906/*/*
DT	1/*/*	495/*/*	1801/*/*

Table III: 1-Core CPU/6-Core CPU/GPU times where (*) denotes that the implementation is not available.

B. *Prediction*

Prediction speeds were then assessed using a trained model for vHTS. Numbers reported in Table IV are in molecules predicted per second. GPU performance is reported in parenthesis for ANN and SVM.

TABLE IV
PREDICTION PERFORMANCE IN MOLECULES / SECOND

ML Method	Molecules / second (10^3)
ANN	7.1(43)
SVM	2.9(10)
KNN	0.1(15)
Kohonen	2.5
DT	180

C. *Similarity Analysis*

The active data sets for each screen were then assessed using the 5 similarity measures described in Table I. The performance of these calculations is reported in Table V. The CPU time is reported in seconds with the GPU time reported in parenthesis. Maximum speed-up achieved is also reported for each measure.

TABLE V
PREDICTION PERFORMANCE IN SECONDS

Similarity Measure	AID884	AID893	AID1445	Max Speed-up
Tanimoto	53(0.2)	147(0.55)	3.4(0.02)	267
Cosine	47(0.2)	150(0.53)	3.5(0.02)	283
Dice	52(0.2)	145(0.54)	3.9(0.02)	269
Euclidean	27(0.2)	95(0.51)	2.3(0.01)	230
Manhattan	20(0.2)	56(0.52)	1.6(0.01)	160

VII. CASE STUDY: CYP3A4

The data for this case study was acquired through PubChem bioassay ID 884. The high-throughput screen assayed the inhibition of cytochrome P450 3A4 (CYP3A4) on a data set of 10,500 molecules. CYP3A4 is involved in the metabolism and clearance of xenobiotics.

A. *Data Set*

The data set contains 3,438 active and 7,066 inactive molecules. The molecules and bioactivities were downloaded from the PubChem database and 3D coordinates were generated with Corina[29]. Atomic and molecular properties, and descriptors using these properties, were then generated using the BCL and stored in a specialized binary format for rapid descriptor subset generation and to reduce I/O.

B. *Training Methods*

This case study involved the generation of optimized ANN, SVM, KNN, Kohonen, and DT models for the prediction of CYP3A4 inhibition. Additionally, consensus ensemble models using equal weighting were investigated and compared against single algorithm predictive models.

1) *Feature Selection*

Feature selection is a critical component of model optimization during QSAR model construction. A total of 1,284 features were generated for each molecule comprised of 60 groups as described in previous work[7]. Sequential forward feature selection (FFS) was then performed for each machine learning technique individually with 5-fold cross validation using average enrichment as the final objective function:

$$Enrichment = \frac{TP/(TP+FP)}{P/(P+N)} \tag{6}$$

where *TP* = true positive, *FP* = false positive, *P* = actual positives, and *N* = actual negatives. The average is taken from the enrichments achieved by the 5-fold cross validation trainings.

Forward Feature selection describes a deterministic greedy search algorithm among all feature groups. First, every single feature is selected and trained followed by the evaluation of respective objective functions. The top performing feature is elected as the starting subset for the next round of FFS. Next, each remaining feature is added to the current subset in an iterative fashion resulting in N-1 feature sets.

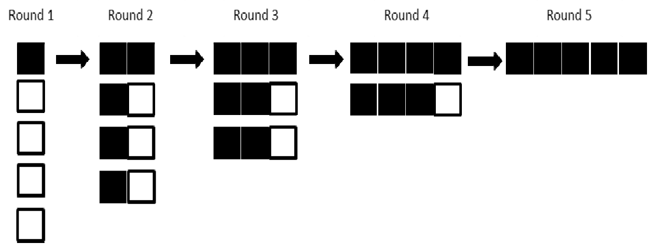

Figure 3: Schematic example of Forward Feature selection with 5 descriptor groups. Thus, 9,150 models are trained for each machine learning technique during this process as $cv * \frac{n(n+1)}{2}$ where cv is the cross validation number and n is the number of feature categories (60).

The best performing feature set is chosen again as the starting set for the next round. This process is repeated until all features are selected and the best descriptor combination is determined as shown in Figure 3.

2) *Model Optimization*

Using the optimized feature set for each algorithm, parameters for the algorithms were then optimized. This included the optimization of η and α for ANN, *C* and γ for SVM with RBF, and kappa for KNN. Upon completion of parameter optimization, the models were fully cross validated which involved 10 rounds of 5-fold cross validation such that every molecule in the data set was used in the independent test set once.

3) *Ensemble Predictors*

The use of ensemble predictors with multiple machine learning techniques was then investigated by analyzing the predictive performance of every unique combination of optimized machine learning models in a consensus approach. Thus, the average prediction of the 50 models generated for each machine learning technique during the final cross validation process is used as a single prediction for that method. Average predictions using every unique combination are then calculated and analyzed by average enrichment. The different machine learning models have equal weighting in ensemble predictions.

4) *Model Analysis*

Model analysis is a critical phase during the construction of QSAR models to evaluate the predictive power of the trained models. In this work, receiver operating characteristic (ROC) curves are used to evaluate machine learning performance ROC curves are plots of true positive rates (TPR) versus false positive rates (FPR) when calculated outputs are sorted:

$$TPR = \frac{TP}{P} \tag{7}$$

$$FPR = \frac{FP}{N} \tag{8}$$

Additionally, average enrichment is also calculated and used for model performance analysis.

C. Results

The final objective function values of average enrichment for the optimized models are shown in Table VI. The maximum theoretical enrichment possible is 3.06.

Table VI
Optimized Feature Set Performance

Method	Average Enrichment	Number Features
ANN	2.78	298
SVM	2.67	392
KNN	2.78	73
Kohonen	2.71	94
DT	1.43	332

Algorithmic parameters used during feature selection and those achieved through optimization for ANN, SVM, and KNN using the optimized feature sets for the respective algorithms are shown in Table VII.

Ensemble predictors were then constructed using every unique combination of optimized models. The results of the objective function evaluations using these ensemble predictors are shown in Table VIII.

Table VII
Optimized Parameters

Step	ANN (η,α)	SVM (*C*, γ)	KNN (*k*)
FFS	0.1, 0.5	10,0.5	9
Optimized	0.0625,0.25	0.00195,1.0	17

The best performance as evaluated by average enrichment, AUC, and RMSD was the ensemble predictor ANN/KNN/Kohonen which achieved an average enrichment of 2.89, 94% of the theoretical maximum. The ROC curve for this ensemble predictor is shown in Figure 4.

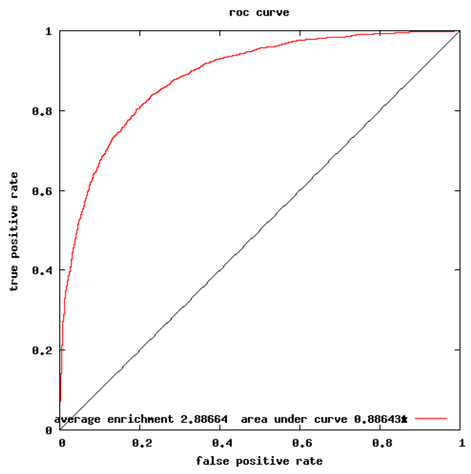

Figure 4: ROC curve illustrating the performance of the best performing ensemble model, ANN/KNN/Kohonen, achieving an average enrichment of 94% of the theoretical maximum enrichment.

Table VIII
Ensemble Predictor Performance

Method	Enrichment	AUC	RMSD
ANN	2.78	0.87	0.51
ANN/DT	2.79	0.87	0.57
ANN/KNN	2.87	0.89	0.49
ANN/KNN/DT	2.87	0.89	0.52
ANN/KNN/Kohonen	**2.89**	**0.89**	**0.51**
ANN/KNN/Kohonen/DT	2.89	0.89	0.52
ANN/Kohonen	2.87	0.87	0.53
ANN/Kohonen/DT	2.87	0.87	0.55
ANN/SVM	2.83	0.87	0.54
ANN/SVM/DT	2.83	0.88	0.57
ANN/SVM/KNN	2.87	0.89	0.51
ANN/SVM/KNN/DT	2.87	0.89	0.53
ANN/SVM/KNN/Kohonen	2.88	0.89	0.52
ANN/SVM/KNN/Kohonen/DT	2.88	0.89	0.53
ANN/SVM/Kohonen	2.87	0.87	0.55
ANN/SVM/Kohonen/DT	2.87	0.87	0.56
DT	1.43	0.95	0.77
KNN	2.78	0.87	0.53
KNN/DT	2.79	0.87	0.57
KNN/Kohonen	2.84	0.87	0.53
KNN/Kohonen/DT	2.84	0.87	0.55
Kohonen	2.71	0.82	0.59
Kohonen/DT	2.71	0.82	0.61

SVM	2.67	0.83	0.64
SVM/DT	2.68	0.84	0.64
SVM/KNN	2.83	0.88	0.54
SVM/KNN/DT	2.83	0.88	0.56
SVM/KNN/Kohonen	2.86	0.87	0.55
SVM/KNN/Kohonen/DT	2.86	0.88	0.55
SVM/Kohonen	2.80	0.84	0.60
SVM/Kohonen/DT	2.80	0.84	0.60

D. *Conclusion*

Here, we have utilized GPU-accelerated machine learning techniques implemented in the BCL to demonstrate the utility of ensemble predictors for QSAR modeling and shown the significant performance improvements that can be achieved over single algorithms alone. Additionally, we have constructed a high-performance predictive model for CYP3A4 inhibition, a valuable tool in drug discovery and pharmacology. This model will be made freely available for academic use at www.meilerlab.org.

VIII. Discussion

Here, we have discussed the utility of GPU-accelerated machine learning algorithms for QSAR modeling using HTS data. These hardware-accelerated algorithms enable the construction of high-quality, high-performance QSAR models while also enabling rapid virtual screening campaigns due to this acceleration. Additionally, we have shown a case study using a target of pharmacological and pharmaceutical relevance using freely available data from PubChem. With the accelerated performance achieved in this work, construction of optimized models shows a speed up of over two orders of magnitude allowing the construction of optimized models on a GPU-equipped workstation in only 24 hours using large data sets. Virtual screening has also been accelerated such that the entire PubChem database of 31+ million compounds can be screened in under an hour.

Acknowledgment

This work is supported by 1R21MH082254 and 1R01MH090192 to Jens Meiler. Edward W. Lowe, Jr. acknowledges NSF support through the CI-TraCS Fellowship (OCI-1122919). Nils Woetzel was supported by the Warren research fellowship of the Chemistry Department at Vanderbilt University. The authors thank the Advanced Computing Center for Research & Education at Vanderbilt University of hardware support.

References

[1] A. Tropsha, "Best Practices for QSAR Model Development, Validation, and Exploitation," *Molecular Informatics,* vol. 29, pp. 476-488, 2010.

[2] R. Perkins, H. Fang, W. Tong, and W. J. Welsh, "QUANTITATIVE STRUCTURE–ACTIVITY RELATIONSHIP METHODS: PERSPECTIVES ON DRUG DISCOVERY AND TOXICOLOGY," *Environmental Toxicology and Chemistry,* vol. 22, pp. 1666-1666, 2003.

[3] C. Hansch, P. P. Maloney, T. Fujita, and R. M. Muir, "Correlation of Biological Activity of Phenoxyacetic Acids with Hammett Substituent Constants and Partition Coefficients," *Nature,* vol. 194, pp. 178-180, 1962.

[4] C. Hansch, "Use of quantitative structure-activity relationships (QSAR) in drug design (review)," *Pharmaceutical Chemistry Journal,* vol. 14, pp. 678-691, 1980.

[5] T. Scior, J. L. Medina-Franco, Q. T. Do, K. Martínez-Mayorga, Y. Rojas, and P. Bernard, "How to Recognize and Workaround Pitfalls in QSAR Studies: A Critical Review," *Current medicinal chemistry,* vol. 16, pp. 4297-4313, 2009.

[6] A. Hillebrecht and G. Klebe, "Use of 3D QSAR models for database screening: a feasibility study," *J. Chem. Inf. Model,* vol. 48, pp. 384-396, 2008.

[7] E. W. Lowe, Jr., M. Butkiewicz, M. Spellings, A. Omlor, and J. Meiler, "Comparative Analysis of Machine Learning Techniques for the Prediction of LogP," presented at the SSCI 2011 CIBCB - 2011 Symposium on Computational Intelligence in Bioinformatics and Computational Biology, Paris, France, 2011.

[8] M. Butkiewicz, R. Mueller, D. Selic, E. Dawson, and J. Meiler, "Application of Machine Learning Approaches on Quantitative Structure Activity Relationships," presented at the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, Nashville, 2009.

[9] R. Mueller, A. L. Rodriguez, E. S. Dawson, M. Butkiewicz, T. T. Nguyen, S. Oleszkiewicz, A. Bleckmann, C. D. Weaver, C. W. Lindsley, P. J. Conn, and J. Meiler, "Identification of Metabotropic Glutamate Receptor Subtype 5 Potentiators Using Virtual High-Throughput Screening," *ACS Chemical Neuroscience,* vol. 1, pp. 288-305, Apr 21 2010.

[10] A. Bleckmann and J. Meiler, "Epothilones: Quantitative Structure Activity Relations Studied by Support Vector Machines and Artificial Neural Networks," *QSAR Comb. Sci.,* vol. 22, pp. 719-721, 2003.

[11] L. Shao, L. Wu, X. Fan, and Y. Cheng, "Consensus Ranking Approach to Understanding the Underlying Mechanism With QSAR," *Journal of Chemical Information and Modeling*.

[12] K. Simmons, J. Kinney, A. Owens, D. A. Kleier, K. Bloch, D. Argentar, A. Walsh, and G. Vaidyanathan, "Practical Outcomes of Applying Ensemble Machine Learning Classifiers to High-Throughput Screening (HTS) Data Analysis and Screening," *J Chem Inf Model,* Nov 5 2008.

[13] K. Simmons, J. Kinney, A. Owens, D. Kleier, K. Bloch, D. Argentar, A. Walsh, and G. Vaidyanathan, "Comparative Study of Machine-Learning and Chemometric Tools for Analysis of In-Vivo High-Throughput Screening Data," *Journal of Chemical Information and Modeling,* vol. 48, pp. 1663-1668, 2008.

[14] *PubChem Database*. Available: http://pubchem.ncbi.nlm.nih.gov/

[15] A. B. Wagner, "SciFinder Scholar 2006: an empirical analysis of research topic query processing," *J Chem Inf Model,* vol. 46, pp. 767-74, Mar-Apr 2006.

[16] E. W. Lowe Jr, M. Butkiewicz, Z. White, M. Spellings, A. Omlor, and J. Meiler, "Comparative Analysis of Machine Learning Techniques for the Prediction of the DMPK Parameters Intrinsic Clearance and Plasma Protein Binding," presented at the 4th International Conference on Bioinformatics and Computational Biology, Las Vegas, NV, 2012.

[17] T. Fox and J. M. Kriegl, "Machine learning techniques for in silico modeling of drug metabolism," *Curr Top Med Chem,* vol. 6, pp. 1579-91, 2006.

[18] J. Meiler, "PROSHIFT: Protein Chemical Shift Prediction Using Artificial Neural Networks," *J. Biomol. NMR,* vol. 26, pp. 25-37, 2003.

[19] W. P. Walters and M. A. Murcko, "Prediction of 'drug-likeness'," *Adv Drug Deliv Rev,* vol. 54, pp. 255-71, Mar 31 2002.

[20] I. V. Tetko, V. V. Kovalishyn, and D. J. Livingstone, "Volume Learning Algorithm Artificial Neural Networks for 3D QSAR Studies," *Journal of Medicinal Chemistry,* vol. 44, pp. 2411-2420, 2001.

[21] B. C. Drucker H and V. V, "Support vector regression machines."

[22] R. N. Jorissen and M. K. Gilson, "Virtual screening of molecular databases using a support vector machine," *J Chem Inf Model,* vol. 45, pp. 549-61, May-Jun 2005.

[23] B. Schoelkopf and A. J. Smola, *Learning with Kernels*. Cambridge, Massachusetts: The MIT Press, 2002.

[24] X. Y. He, J. Wegiel, Y. Z. Yang, R. Pullarkat, H. Schulz, and S. Y. Yang, "Type 10 17beta-hydroxysteroid dehydrogenase catalyzing the oxidation of steroid modulators of gamma-aminobutyric acid

type A receptors," *Mol Cell Endocrinol,* vol. 229, pp. 111-7, Jan 14 2005.

[25] N. Alexander, N. Woetzel, and J. Meiler, "Bcl::Cluster: A method for clustering biological molecules coupled with visualization in the Pymol Molecular Graphics System," in *Computational Advances in Bio and Medical Sciences (ICCABS), 2011 IEEE 1st International Conference on*, 2011, pp. 13-18.

[26] V. Garcia, E. Debreuve, and M. Barlaud, "Fast k Nearest Neighbor Search Using GPU," in *CVPR Workshop on Computer Vision on GPU*, Anchorage, Alaska, USA, 2008.

[27] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in kernel methods: support vector learning*, ed Cambridge: MIT Press, 1999, pp. 185-208.

[28] C. Dar-Jen, A. H. Desoky, O. Ming, and E. C. Rouchka, "Compute Pairwise Manhattan Distance and Pearson Correlation Coefficient of Data Points with GPU," in *Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, 2009. SNPD '09. 10th ACIS International Conference on*, 2009, pp. 501-506.

[29] J. Sadowski and J. Gasteiger, "From Atoms and Bonds to Three-Dimensional Atomic Coordinates: Automatic Model Builders," *Chem. Reviews,* vol. 93, pp. 2567-2581, 1993.