Learning from the Harvard Clean Energy Project: The Use of Neural Networks to Accelerate Materials Discovery

*Edward O. Pyzer-Knapp, Kewei Li, and Alan Aspuru-Guzik**

Here, the employment of multilayer perceptrons, a type of artificial neural network, is proposed as part of a computational funneling procedure for high-throughput organic materials design. Through the use of state of the art algorithms and a large amount of data extracted from the Harvard Clean Energy Project, it is demonstrated that these methods allow a great reduction in the fraction of the screening library that is actually calculated. Neural networks can reproduce the results of quantum-chemical calculations with a large level of accuracy. The proposed approach allows to carry out large-scale molecular screening projects with less computational time. This, in turn, allows for the exploration of increasingly large and diverse libraries.

1. Introduction

The most common bottleneck of molecular materials design is the experimental synthesis and characterization of a material. Hence, the problem of selecting the most promising material to explore experimentally is at the forefront of materials discovery. Whilst theoretical techniques have shown some significant successes in narrowing a potential list of candidate molecules,[1–9] chemical space is simply too large for a brute force technique to be employed for the vast majority of problems.[10] Indeed in the Harvard Clean Energy Project (CEP), a exploration of a library of just 26 fragments using only two basic molecular combination rules resulted in the calculation of properties for over 3.5 million molecules and required a specialized distributed computing framework to compute the 30 000 CPU years expended.[3] Since chemical space—even that localized to some specific desired property—is many orders of magnitude larger than the 26 fragments studied in the Clean Energy Project, and can be combined in countless more ways, a way to search greater areas of chemical space without the resultant explosion of computational expense is required for high throughput virtual screening to fulfill its great potential.

In a recent paper,[11] we discussed the four philosophies of high throughput virtual screening, which include the use of a so-called "computational funnel." This is illustrated in **Figure 1**. In a computational funnel, a large number of starting molecules are gradually reduced to a very small number, which can be investigated experimentally, through the use of increasingly expensive calculations. One of the key concepts of the computational funnel is the balance between speed and accuracy at each of the sifting "layers," since the strictness of the filtering criteria are directly related to the accuracy of the methods employed. It is generally assumed that the more expensive a calculation is, the greater its filtering efficiency, i.e., the smaller the error bars on the predicted property of interest. Finding calculations that maximize this efficiency whilst minimizing computational expense is integral to the success of the process, since this results in a greater number of initial molecules; hence increasing the chances of finding a desirable product.

The pharmaceutical industry has had great success with using QSAR/QSPR models to quickly scan very large libraries of compounds for so-called active pharmaceutical ingredients. Like any parametrized model, these methods work very well on molecules which are closely related to those used in the development of the model, but have weaker performance when molecules which are very different from this initial set are calculated. This is to say that they are mainly interpolative, rather than extrapolative models. We believe that a "tailor made" QSPR model, generated on the fly from data calculated as part of a high-throughput virtual screen, can give both accuracy, and the desired speed of calculation; making it an ideal choice for the initial "screen" in the funnel.

In order to generate this tailor-made QSPR, we utilize multilayer perceptrons (MLPs), a specific class of artificial neural network (ANN). In this study, we introduce the concepts of MLPs, and show how that, given a suitable amount training examples, they can yield highly accurate predictions on a variety of properties, making them ideal for use in a high-throughput virtual screen.

2. Machine Learning Techniques in High-Throughput Virtual Screening

Machine learning techniques have been utilized in chemistry, particularly pharmaceutical chemistry, to produce predictive models.[12–14] It has been especially used for properties which are considered hard to model, such as chemical reactivity,[15] melting point,[16] and solubility.[17,18] Recently, these methods have also shown some promise in predicting elements of the electronic structure of molecules.[19–21]

Dr. E. O. Pyzer-Knapp, K. Li, Prof. A. Aspuru-Guzik
Department of Chemistry and Chemical Biology
12 Oxford Street, Cambridge, MA 02138, USA
E-mail: aspuru@chem.harvard.edu

DOI: 10.1002/adfm.201501919

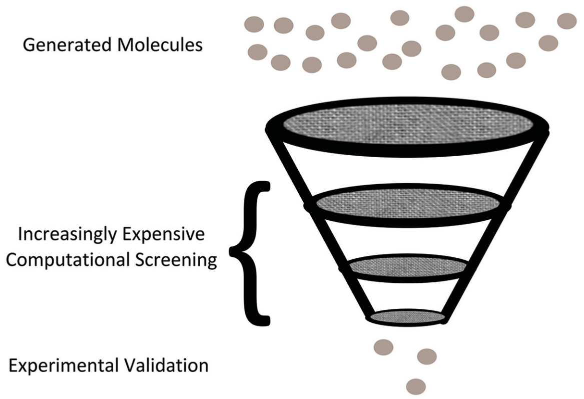

Figure 1. A graphical representation of a computational funnel as used in high-throughput virtual-screening projects.

2.1. Scalability Considerations for Machine Learning on Large Data Sets

For a machine learning approach to be successful in a high-throughput context it is highly desirable for it to scale well with the number of inputs. In other studies, we have found that simple Gaussian processes[22] work very well with smaller problems; for instance when the size of the chemical space that is to be explored is small[23] or for calibrating quantum results to experimental values (the number of experimental values used is rarely about a thousand).[24] Unfortunately, Gaussian processes do not scale as favorably with the number of data inputs, due to the inversion of the covariance matrix which scales as $O(N^3)$ in time and also $O(N^2)$ in memory. In these investigations, we have seen that the largest set of inputs that can be handled reasonably is of the order of 10 000. Since there is no expensive operations in a neural network, they scale significantly better than this and hence are more suited to this kind of study, however this comes at the cost of knowing the uncertainty associated with each prediction. Bayesian neural networks[25] potentially represent a good compromise solution for this task and are under active investigation.

Artificial neural networks (ANNs) are computational models which are based upon the structure of the central nervous system of animals—biological neural networks[26]—and are commonly used for pattern recognition and function approximation (regression).

2.2. Structure of a Neural Network

ANNs consist of interconnected layers of neurons, so called since they represent the vast network of neurons in a brain, which are connected to each other along paths. These layers fall into one of three major classes: input, hidden, and output. Features are read in through the input layer, pass through the hidden layers (so called because they are hidden from the outside world), and are combined into a result at the output layer. The structure of a neural network is shown graphically in **Figure 2**.

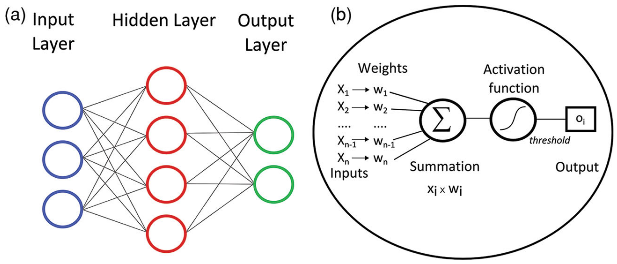

Figure 2. a) In a neural network, features typically enter through a linear input layer (blue) pass through a nonlinear hidden layer (red) before being combined by a linear output layer (green), which produces the target estimation. b) The operation of a single neuron, where inputs are combined with weights, pass through an activation function and produce an output that can be connected with further neurons.

2.3. Activation Functions

As can be seen from Figure 2, the output of a neuron, given an input, is defined by its activation function which models the firing of a synapse. In biological neural networks, the output is binary—either the synapse fires or it does not— and so initial ANNs used functions with forms similar to

$$\varphi(\alpha) = \begin{cases} 1, & x > 0 \\ -1, & x \le 0 \end{cases} \tag{1}$$

This is the so-called binary or threshold activation function. Whilst ANNs with this form have been used with some success for classification problems, the number of neurons required for successful use in regression problems means this is not commonly used in hidden layers. A more common function is the “tanh” sigmoidal activation function,

$$\varphi(\alpha) = \tanh(\alpha) \tag{2}$$

This model, along with the logistic function,

$$\varphi(\alpha) = \frac{1}{1 + e^{-\alpha}} \tag{3}$$

is frequently employed in ANNs and is used to add nonlinear hidden layers to the *multilayer perceptron* model for neural networks which will be employed in this study. It is essential that these hidden layers be nonlinear, since, due to their linear nature, an ANN with multiple linear hidden layers can be reduced to a simple one-layer ANN.

2.4. Multilayer Perceptrons

In this study, we utilize a class of ANNs known as MLPs, which are known to be particularly good at supervised regression problems. For an ANN to be classed as a MLP, it must fulfill three major criteria.

- It must have at least three layers (including the input and output layer).
- The hidden layers must consist of nonlinear functions, and this nonlinearity must be smooth (i.e., differentiable everywhere).

- The network must exhibit a high degree of connectivity.

Machine learning methods, and in particular neural networks, have enjoyed fluctuating popularity in chemistry over the years. In 1991, Zupan and Gasteiger posed the question "[Are neural networks] a new method for solving chemical problems, or just a passing phase?."[27] Some of the major challenges which have sometimes discouraged the use of neural networks in chemistry include the worry that large neural networks will over fit data. Another concern is the amount of time it can take to converge the training of a large neural network. Finally, neural networks rely on the amount of available, reliable data for training.

2.5. Avoiding Overfitting

One problem that is frequently encountered in the training of neural networks is overfitting. This is when the network can reproduce training examples with low error, but when presented with new features, the error is high. In effect, the network has memorized, rather than learned from, the data. In other words, it has learned the noise.

There are a number of ways to avoid overfitting,[28] and in this study we have opted to use a method known as *early stopping*.[29] This algorithm works by randomly selecting a subset of molecules from the training set each epoch, and validating the new weights against this set. If the error on this set starts to increase before the maximum number of epochs is reached, the training is terminated early. Additionally, for this study, errors will be reported in both the training set, and a separate validation set of 50 000 molecules—demonstrating the validity of the network for molecules which it has not been trained on.

2.6. Accelerating the Convergence of the Training of a MLP with the Stochastic Gradient Descent and RMSProp Algorithms

The back propagation algorithm is a commonly used technique to train ANNs,[30] and is used in conjunction with some optimization method, commonly gradient descent. It is generally divided into two stages: First, the training features are propagated through the ANN in order to generate the ANNs output for each feature. Second, for each step in training, the gradient of a loss function with respect to each of the weights is calculated. This gradient is then used in conjunction with a learning rate to modify the weights for each layer from the output to the input, hence the term back propagation. This procedure is repeated until the errors in the network drop below some predefined critical value. The way that each weight parameter is updated is dependent on a set of hyper-parameters (e.g., learning rate, which is akin to a step size in a more traditional optimization), which can themselves be optimized.

If all of the data are used at once during back propagation, it can take a significant amount of time to process each epoch. Thus when training this network, stochastic gradient descent[31] was employed to accelerate convergence. In this method, small stochastically selected subsets of size n are selected from the total N inputs and are trained, thus providing a noisy value for the back propagation algorithm to optimize against. Thus, many noisy updates are applied per epoch, providing a more efficient optimization—especially in data-sets with a high level of redundancy. For this method one "epoch" is reached when n/N batches have been processed.

When using a stochastic gradient descent model, it is important to reduce the learning rate toward the end of the training, so that the noise from the minibatches does not adversely influence the end result. This can be difficult to successfully achieve using a global learning rate, and so a technique known as RMSProp was utilized to aid the training of the weights[32] RMSProp evolved from the observation that the magnitude of the gradient can vary greatly for different weights, and can alter during the training procedure, hence making the selection of a single learning rate difficult. In RMSProp, the gradient is modified using a moving average of the squared gradient for each rate:

$$G(w,t) = 0.9G(w,t-1) + 0.1\left(\frac{\delta\varepsilon(n)}{\delta w(n)}(t)\right)^2 \tag{4}$$

RMSProp has the additional advantage than networks which utilize this method tend to need less hyper-parameter optimization (i.e., are more robust to nonoptimal hyper-parameters).[33]

2.7. Speeding Up the Overall Training of the Network with Parallelization Using the Hogwild Algorithm

Due to the large size of the training data set, parallelization of the training becomes very important to achieve reasonable training times. One major limitation of stochastic gradient descent when data sizes are large is its inherently sequential nature; often necessitating a data-gathering step for parallel computation. A consequence of this is that the bottleneck of computation now lies with the synchronization of threads, which can have significant overhead. In order to remove this bottleneck we use a variant of the Hogwild algorithm[34] as implemented in Microsoft's Project Adam.[35] Hogwild provides a means to avoid thread synchronization by splitting the data into batches, each trained on a thread. Each thread updating the weights as it trains in a nonsafe manner (i.e., no locking); it is possible to think of this as analogous to another layer of batching. We have observed stability in this algorithm in our data-set with up to 30 threads.

Before implementing this algorithm, training times for the network (200 000 data points) were in the order of two weeks, even when compromises were made with the hyper-parameters (such as learning rate) which can lead to faster—albeit less accurate—training. This is simply too long for a high throughput setting, and acts as a ceiling to the size of the dataset that can be trained; an obviously undesirable effect. When using our Hogwild implementation, however, training time was reduced to around 4 h; a significant improvement. It was observed that when using this implementation, fewer epochs were needed to train the network, and the mean absolute error (MAE) of the network was significantly reduced. We include a comparison of our new methods with an existing python implementation of an artificial neural network for a set of 50 000 molecules in the supporting information (Figures S1–S3, Supporting Information).

In a computational funnel, increasingly expensive calculations are performed on diminishing sets of molecules in order to increase the efficiency of computational deployment. That is to say, the computationally intensive calculations are only performed on molecules which are believed to have a reasonable chance of displaying the desired properties. In this framework, the Holy Grail is a calculation which is computationally inexpensive, but has good accuracy. This will allow a large number of molecules to be screened (it is a cheap calculation), with a high dropout rate (since we are confident in the result).

2.8. Use of Machine Learning Techniques in a Computational Funnel

We believe that machine learning (ML) based techniques offer the potential to fulfill this criteria since the initial expense of training the model is far outweighed by the increase in performance of this model over other similar cost approaches. The use of a neural network inside such a funnel also answers one of the concerns raised earlier in this study: the availability of high-quality data. Due to its position in the funnel, the neural network will have an abundance of high-quality relevant calculated data with little or no noise—since the output of the calculation is the precise value that the neural network is trying to predict. This is unlike other methods, where neural networks are used to predict spectroscopic data, or experimental data, which have inherent noise.

Employed in this way, once trained, these techniques calculate the target values much faster than current semiempirical methods and, as we will show, can approach the accuracy of DFT methods; truly a best-of-both-worlds outcome.

As an example of the use of ML techniques in High Throuput Virtual Screening (HTVS), we examine the Harvard Clean Energy Project (CEP)—a high-throughput virtual screening effort for the discovery of high-performance organic photovoltaic materials (OPV). The figure of merit utilized in this project is the power conversion efficiency (PCE), which is derived from a model described by Scharber et al.[36] and is based upon the highest occupied molecular orbital (HOMO) and lowest unoccupied molecular orbital (LUMO) energy levels, coupled with reasonable approximations for some empirical parameters such as the fill factor and the internal quantum efficiency. We will therefore examine the potential for learning the energy levels of the HOMOs and LUMOs of a random selection of these molecules, as well as attempting to learn the PCE directly.

3. Computational Methods

3.1. Selection of Input Feature Representation

The choice of how to describe the molecules within the library—the so-called input feature representation—is critical to the success of the neural network. Whilst the Coulomb matrix, and its variants, have been used successfully,[19,37,38] they require a knowledge of the 3D coordinates of the molecule, preferably of a high quality. Whilst this makes them ideal for some tasks, in which the 3D coordinates are already known, they are less than ideal for a HTVS workflow, since the ideal position for a filter based upon these neural networks is before any generation of 3D coordinates has occurred. Additionally, for problems in which the response of a material is based upon a combination of many different conformers, the training set for a neural network would have to include multiple conformers for each molecule, significantly increasing its size and thus dramatically slowing down the training time.

Thus, this study focused upon fingerprints based upon simply the molecular graph, i.e., a knowledge of the type and connectivity of atoms. In order to be easily fed into a neural network, it is also desirable for the feature representation to be of a fixed length. Contained within the RDKit[39] are four fingerprints which satisfy our selection criteria: the hashed atom-pair fingerprint,[40] the MACCS key-based fingerprint,[41] the Morgan circular fingerprint,[42] and the hashed topological torsion fingerprint.[43] In order to choose a representation, a small subset of 20 000 molecules was selected from the set of 200 000 used in this study, and randomly split into a training and validation set, with a 90:10 split. For each molecule the HOMO, LUMO, and power conversion efficiency were predicted using each feature representation. The length for all the features was set to 1024 bits, except for the MACCS keys, which have a fixed length of 167 bits. The results are shown in **Figure 3**. It can be seen from Figure 3 that there is a great deal of difference between the performances of each feature representation. Whilst for this problem, the hashed atom-pair and MACCS key representation performed poorly, the Morgan circular fingerprint and the hashed topological fingerprint had similar performance, with the Morgan circular fingerprint performing slightly better than the hashed torsional fingerprint for the HOMO and LUMO tasks, and slightly worse for the PCE task. Since it was found that the Morgan circular fingerprint is slightly faster to calculate than the hashed topological fingerprint (4.6 s per 10 000 vs 6.0 s per 10 000 averaged over 10 runs), it was decided that the

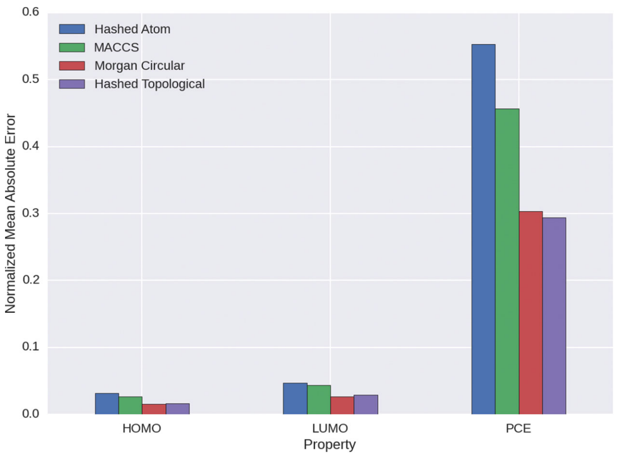

Figure 3. The performances of the four studied input feature representations available in the RDKit software toolkit. For each case the MAE was normalized to the mean of the absolute values for the molecules studied.

Morgan circular fingerprint was a good choice for feature representation, and use it exclusively for the rest of this study.

3.2. Training the Network

A set of 200 000 molecules was randomly selected from the Clean Energy Project Database (CEPDB) for training, and a further 50 000 were randomly selected to make up the validation set. For each of these molecules, a 1024-bit, radius = 2, Morgan circular fingerprint[42] was created using the functionality provided by the RDKit[39] for use as the input feature. The radius was chosen since it is fast to compute, yet still captures enough of the intrinsic chemical information to be predictive.[44] Whilst the HOMOs, LUMOs, and PCE provided on the open-access website *cepdb.molecularspace.org* are averaged over conformers, and calibrated against experimental values, it was decided that it was more illuminating to instead use the initial quantum-chemical results calculated using Q-Chem[45] at the BP86[46,47]/def2-SVP[48] for the lowest energy conformer as targets. This removes extra noise from the calibration procedure, and the exploration of the conformational landscape; allowing a clear demonstration of the predictive power of the MLP. We built our multilayer perceptron with three hidden layers, using linear input and output nodes and three hidden layers consisting of 128, 64, and 32 nodes and constructed using the logistic function, with randomly initialized weights.

The neural networks implementation used here is available as part of the package MAML which was developed by the authors.[49] All targets were predicted simultaneously, as this has been shown to lead to a more robust network.[50]

4. Results and Discussion

4.1. Predicting Targets

As mentioned earlier, using the Scharber model,[36] the HOMO and LUMO level of a molecule can be related to the PCE, the “fitness function” for molecules in the CEP.[51] Training the model on 200 000 structures and testing on 50 000 structures produces the following results, which are shown in **Table 1** and plotted in **Figure 4**.

These results are favorable for the use of neural networks. Machine learning efforts which have previously attempted to learn the HOMO and LUMO have reported testing errors of 0.15 and 0.12 eV, respectively;[19] and hence this effort displays a circa 6× and 4× improvement for these values—something of significance for having reliable screening layers. It should be noted that PCE values below 0.0% are an artifact of the Scharber model and, whilst unphysical, expected. When the errors in the HOMO and LUMO prediction are propagated through the test set, an uncertainty in the PCE of ±0.28% was calculated. This is very close to the value predicted by the network trained on to predict the PCE without any quantum-chemical information.

Table 1. Mean absolute error (MAE) for the predictions for the highest occupied molecular orbital, lowest unoccupied molecular orbital, and power conversion efficiency provided by a MLP trained on 200 000 molecules for a validation set of 50 000 molecules which did not appear in the training set.

Target	Training MAE	Testing MAE
HOMO	0.019 eV	0.028 eV
LUMO	0.024 eV	0.032 eV
PCE	0.17%	0.28%

Whilst we would expect a small decay in performance toward the higher end of the PCE range, due to the relative lack of data on high-performance materials in the database— which can be observed from the distributions of points shown on the axis of these plots—this is not seen to be problematic. In the context of high-throughput virtual screening, it is acceptable for a model to over-estimate a small number of targets, so long as this does not significantly affect the efficiency of the screening process. Additionally, some underestimation can be tolerated since it is argued that the larger libraries that are made available by a fast screening method more than compensate for incorrectly filtering a small number of promising molecules. We are working on active learning approaches to improve our search based on these observations.

In addition to exploiting the “active space” afforded by the HTVS paradigm, we believe that there are additional factors that explain the strong performance of our network. As previously discussed, we have taken care to select a powerful feature representation, which likely encapsulates the relevant features of each molecule, such as functional groups, which have an effect upon the predicted properties. Due to the high-dimensional nature of the optimization of the network's weights, our use of stochastic gradient descent with minibatches allows a noisy optimization, which includes more updates to the weights per epoch than the equivalent online method and results in a lower MAE for the network. This is further enhanced by the use of the Hogwild algorithm for alleviating thread-locking during the parallelization of stochastic gradient descent which, as a by-product, includes an additional layer of batching to the minimization, which we have found to be helpful for further lowering the MAE of the network, especially on larger set sizes which have high redundancy. Within the stochastic gradient descent algorithm, the use of the RMSProp algorithm allows the network to quickly adjust weights that are deemed to be far from their optimum value without undue adjustments to those that are close. RMSProp also allows the opportunity to “tune down” the noise from stochastic gradient descent as convergence is approached. We have found that RMSProp both increases the speed of convergence for the network, and also improves the final MAE.

4.2. Dependence upon Training Set Size

To investigate the dependence of the method on the size of the training set, a set of MLPs were trained in the same manner, using 12 different set sizes spaced between 10% of the 200 000 training set and 99% of the training set; with the resulting MAEs averaged over five runs. A fixed random seed was used to for each run ensure that the smaller training sets were exact subsets of the larger ones, in order to make a direct comparison. The resulting MAEs for HOMO, LUMO, and PCE are shown in

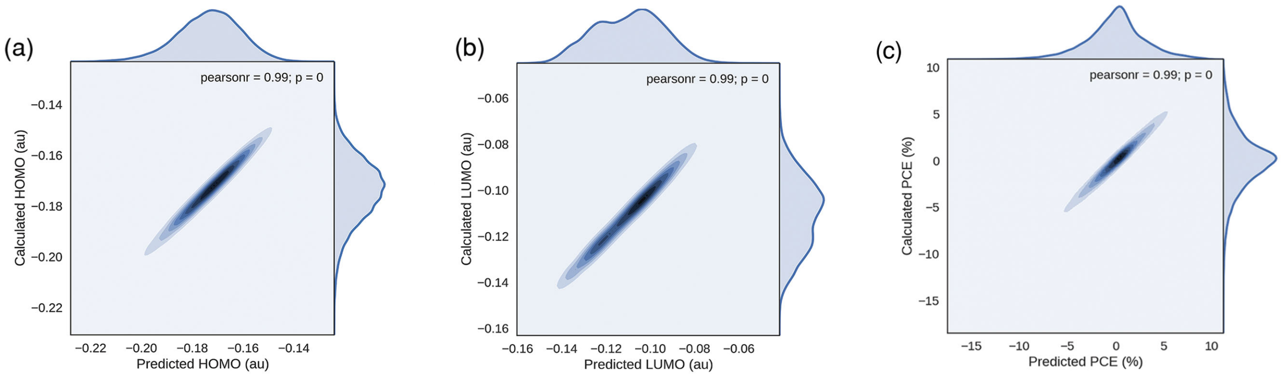

Figure 4. The results of validating the neural network trained from the CEP data on 50 000 randomly selected data points for a) highest occupied molecular orbitals, b) lowest unoccupied molecular orbitals, and c) power conversion efficiencies. It was ensured that no molecule which was contained within the training set was found in this validation set. It should be noted that negative power conversion efficiencies are an artifact of the Scharber model, and whilst unphysical, expected.

Figure 5, with the variance being drawn from the standard deviation of values over the five runs. It can be seen from Figure 5 that reasonable performance can be obtained when small training sets are used, with relative improvements decaying in an exponential manner. It can also be seen that the variability in the performance is much higher when a small training set is used, emphasizing the importance of how the training sets are selected—since the smaller the training set size, the greater the importance of each molecule contained within in. To this end, we are currently investigating the use of active learning methods, to improve the selection of training sets, and hope to report these results in a publication in the near future.

4.3. Speeding Up High-Throughput Virtual Screening

The large number of quantum chemical calculations performed by the CEP has only been possible due to the use of IBM's World Community Grid,[52] and is possibly beyond the limits of most high-throughput screening groups.

We have shown in this paper that a neural network trained on a fraction of this data can allow the accurate prioritization of promising molecules and the discarding of less promising ones, placing the computation within the reach of groups with more moderate computing resources. For context, a single average sized molecule will take in the order of hours to calculate using quantum-chemical methods, whilst the neural network trained in this study can return predictions on a set of 50 000 molecules in less than 1.5 s. This prediction is performed using a single core, with low RAM requirements, which themselves scale linearly with the size of the set of inputs to be predicted. Additionally, the network used in this study can be stored on disk in 1 MB, although the exact size of storage will differ from network to network, since it is dependent on the number of weights, and hence the network architecture.

The strength of the prioritization afforded by this network is illustrated in **Figure 6** which shows how the filter employed can quickly cut the number of potential molecules to be calculated. Of course, the calculations needed to train a neural network should also be taken into account, and the size of the training set should be adjusted to scale with the size of the resources available with the "confidence window" in which structures are not discarded adjusted accordingly.

This allows the use of initial molecular libraries which are many times greater in size than those used currently. By expanding these libraries, we reduce the risk of focusing the search in too narrow an area of chemical space and improve the chances that the screening will uncover desirable materials. It can also be seen how integrating this technique into molecular library generation, via say a Monte Carlo regime, could allow a noisy optimization of molecular structure with respect to any given property.

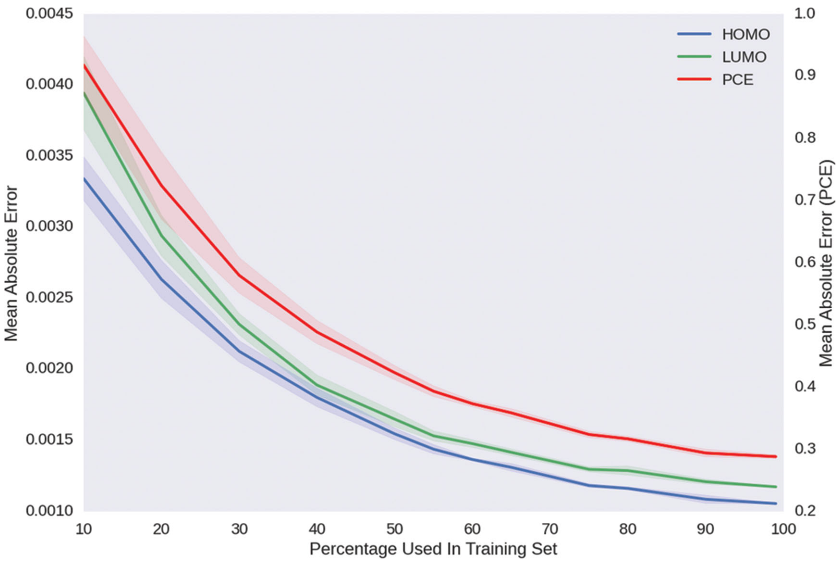

Figure 5. The MAE of the network as the data size is increased from 20 000 (10%) to 200 000 (100%) shows that reasonable performance can be obtained with relatively small sets, and that relative improvements decrease as 100% of the library size is approached.

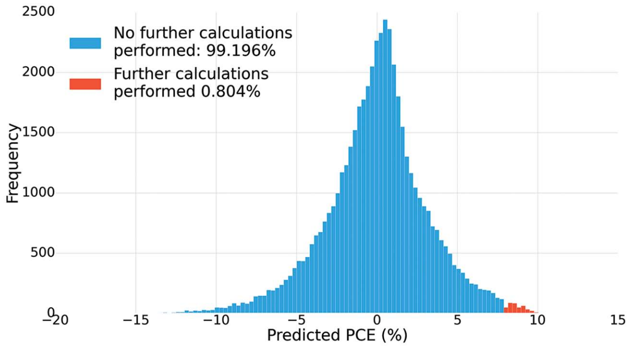

Figure 6. The number of results that can be excluded with even a conservative filter of $\mathrm{PCE}_{\mathrm{predicted}} > 8.0\%$ can vastly enlarge the scope of the problem that can be undertaken. Here, the red fill indicates those molecules which the neural network trained on 200k molecules would filter out if this criteria were applied.

It is important to note that this study does not purport to return a general neural network, from which many different sets of molecules can be predicted; rather it aims to exploit the local nature of the chemical space often utilized within HTVS efforts. Since machine learning techniques are by and large interpolative rather than extrapolative, this can lead to significant gains in predictive power on relevant molecules, if one is prepared to sacrifice the transferability of the network. Within a HTVS workflow, the library of molecules has often been constructed in such a way that this restriction is met, although it is also worth noting that neural networks will still be predictive in diverse libraries, only their predictive power is likely to be weakened by this situation.

We have demonstrated that within these restrictions and in the context of focused HTVS efforts, multilayer perceptrons can be trained to predict HOMOs, LUMOs, and PCEs to a very good accuracy; displaying a significant predictive power, even when small data-sets are used. It was also shown that learning the PCE directly is as successful as learning the HOMO and LUMO values, and calculating the PCE from these predicted values. Additionally, we have shown how the use of nonthread locking techniques significantly reduces the training time of the MLP, allowing the training of significantly larger data sets, which further increase the accuracy.

5. Conclusions

We propose the integration of multilayer perceptrons into the generation of large molecular libraries, which can be of a far greater size than can be reasonably computed, with the properties of new molecules interpolated from existing results. Thus any high-throughput virtual screening effort can enrich the sample of molecules selected for calculation to have a far greater proportion of desirable features than any traditional, random sampling technique could achieve.

Both the accuracy and the speed of the MLP both exposes the exciting area of high-throughput screening to those with less computational resources and increases the size of libraries that can be screened to those who currently occupy the area, allowing deeper and wider explorations of thus unseen chemical space.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

A.A.-G. and E.O.P.-K. acknowledge the Department of Energy through Grant No. DE-SC0008733 for funding. K.L. thanks the Harvard College Research Plan for financial aid. This research would not have been possible without the use of the Harvard FAS Odyssey Cluster and support from FAS Research Computing. The authors also wish to thank IBM for organizing the World Community Grid and the WCG members for their computing time donations.
Note: Figure 4a,b were amended on November 4, 2015.

Received: May 11, 2015
Revised: July 12, 2015
Published online: September 18, 2015

[1] E. O. Pyzer-Knapp, H. P. G. Thompson, F. Schiffmann, K. E. Jelfs, S. Y. Chong, M. A. Little, A. I. Cooper, G. M. Day, *Chem. Sci.* **2014**, *5*, 2235.
[2] B. Huskinson, M. P. Marshak, C. Suh, S. Er, M. R. Gerhardt, C. J. Galvin, X. Chen, A. Aspuru-Guzik, R. G. Gordon, M. J. Aziz, *Nature* **2014**, *505*, 195.
[3] J. Hachmann, R. Olivares-Amaya, A. Jinich, A. L. Appleton, M. A. Blood-Forsythe, L. R. Seress, C. Roman-Salgado, K. Trepte, S. Atahan-Evrenk, S. Er, S. Shrestha, R. Mondal, A. Sokolov, Z. Bao, A. Aspuru-Guzik, *Energy Environ. Sci.* **2014**, *7*, 698.
[4] M. D. Halls, K. Tasaki, *J. Power Sources* **2010**, *195*, 1472.
[5] S. Subramaniam, M. Mehrotra, D. Gupta, *Bioinformation* **2008**, *3*, 14.
[6] B. K. Shoichet, *Nature* **2004**, *432*, 862.
[7] V. Sorna, E. R. Theisen, B. Stephens, S. L. Warner, D. J. Bearss, H. Vankayalapati, S. Sharma, *J. Med. Chem.* **2013**, *56*, 9496.
[8] M. Korth, *Phys. Chem. Chem. Phys.* **2014**, *16*, 7919.
[9] J. J. Low, A. I. Benin, P. Jakubczak, J. F. Abrahamian, S. A. Faheem, R. R. Willis, *J. Am. Chem. Soc.* **2009**, *131*, 15834.
[10] J.-L. Reymond, L. Ruddigkeit, L. Blum, R. van Deursen, *Wiley Interdisc. Rev.: Comput. Mol. Sci.* **2012**, *2*, 717.
[11] E. O. Pyzer-Knapp, C. Suh, R. Gomez-Bombarelli, J. Aguilera-Iparraguirre, A. Aspuru-Guzik, *Annu. Rev. Mater. Res.* **2015**, *45*, 195.
[12] S. Agarwal, D. Dugar, S. Sengupta, *J. Chem. Inf. Model.* **2010**, *50*, 716.
[13] H. Gelernter, J. R. Rose, C. Chen, *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 492.
[14] J.-P. Vert, L. Jacob, *Comb. Chem. High Throughput Screening* **2008**, *11*, 677.
[15] M. A. Kayala, P. Baldi, *J. Chem. Inf. Model.* **2012**, *52*, 2526.
[16] M. Karthikeyan, R. C. Glen, A. Bender, *J. Chem. Inf. Model.* **2005**, *45*, 581.
[17] A. Lusci, G. Pollastri, P. Baldi, *J. Chem. Inf. Model.* **2013**, *53*, 1563.
[18] J. L. McDonagh, N. Nath, L. D. Ferrari, T. van Mourik, J. B. O. Mitchell, *J. Chem. Inf. Model.* **2014**, *54*, 844.

[19] G. Montavon, M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, K.-R. Muller, O. A. von Lilienfeld, *New J. Phys.* **2013**, *15*, 095003.
[20] R. Ramakrishnan, P. O. Dral, M. Rupp, O. A. von Lilienfeld, *J. Chem. Theory Comput.* **2015**, *11*, 2087.
[21] R. Ramakrishnan, O. A. von Lilienfeld, *Chim. Int. J. Chem.* **2015**, *69*, 182.
[22] C. E. Rasmussen, *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA **2006**.
[23] E. O. Pyzer-Knapp, C. Suh, R. P. Adams, A. Aspuru-Guzik, unpublished.
[24] G. Simm, E. O. Pyzer-Knapp, A. Aspuru-Guzik, unpublished.
[25] D. Mackay, *Netw. Comput. Neural Syst.* **1995**, *6*, 469.
[26] F. Rosenblatt, *Psychol. Rev.* **1958**, *65*, 386.
[27] J. Zupan, J. Gasteiger, *Anal. Chim. Acta* **1991**, *248*, 1.
[28] S. Lawrence, C. L. Giles, *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Networks IJCNN 2000*, IEEE, Piscataway, NJ **2000**, Vol. 1, pp. 114–119.
[29] L. Prechelt, *Neural Networks* **1998**, *11*, 761.
[30] D. E. Rumelhert, G. E. Hinton, R. J. Williams, *Readings in Cognitive Science*, Elsevier, Amsterdam/New York **1988**, pp. 399–421.
[31] L. Bottou, *Proc. COMPSTAT*, Springer-Verlag, Berlin/Heidelberg, Germany **2010**, pp. 177–186.
[32] G. Hinton, T. Tielman, Lecture 6.5 – rmsprop, COURSERA: Neural Networks for Machine Learning.
[33] T. Schaul, I. Antonoglou, D. Silver, **2013**, arXiv:1406.1231.
[34] B. Recht, C. Re, S. Wright, F. Niu, in *Advances in Neural Information Processing Systems 24* (Eds: J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, K. Q. Weinberger), Curran Associates, Inc., Red Hook, NJ **2011**, pp. 693–701.
[35] T. Chilimbi, Y. Suzue, J. Apacible, K. Kalyanaraman, *11th USENIX Symp. Oper. Syst. Design Implementation (OSDI 14)*, USENIX Association, Broomfield, CO, **2014**, pp. 571–582.
[36] M. C. Scharber, D. Mühlbacher, M. Koppe, P. Denk, C. Waldauf, A. J. Heeger, C. J. Brabec, *Adv. Mater.* **2006**, *18*, 789.
[37] G. Montavon, K. Hansen, S. Fazli, M. Rupp, F. Biegler, A. Ziehe, A. Tkatchenko, A. V. Lilienfeld, K.-R. Müller, in *Advances in Neural Information Processing Systems 25* (Eds: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger), Curran Associates, Inc., Red Hook, NJ **2012**, pp. 440–448.
[38] M. Rupp, A. Tkatchenko, K.-R. Müller, O. A. von Lilienfeld, *Phys. Rev. Lett.* **2012**, *108*, 058301.
[39] G. Landrum, *RDKit: Open-Source Cheminformatics*.
[40] R. E. Carhart, D. H. Smith, R. Venkataraghavan, *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 64.
[41] J. L. Durant, B. A. Leland, D. R. Henry, J. G. Nourse, *J. Chem. Inf. Comput. Sci.* **2002**, *42*, 1273.
[42] D. Rogers, M. Hahn, *J. Chem. Inf. Model.* **2010**, *50*, 742.
[43] R. Nilakantan, N. Bauman, J. S. Dixon, R. Venkataraghavan, *J. Chem. Inf. Comput. Sci.* **1987**, *27*, 82.
[44] D. Rogers, R. D. Brown, M. Hahn, *J. Biomol. Screening* **2005**, *10*, 682.
[45] Y. Shao, Z. Gan, E. Epifanovsky, A. T. B. Gilbert, M. Wormit, J. Kussmann, A. W. Lange, A. Behn, J. Deng, X. Feng, D. Ghosh, M. Goldey, P. R. Horn, L. D. Jacobson, I. Kaliman, R. Z. Khaliullin, T. Ku , A. Landau, J. Liu, E. I. Proynov, Y. M. Rhee, R. M. Richard, M. A. Rohrdanz, R. P. Steele, E. J. Sundstrom, H. L. Woodcock, P. M. Zimmerman, D. Zuev, B. Albrecht, E. Alguire, B. Austin, G. J. O. Beran, Y. A. Bernard, E. Berquist, K. Brandhorst, K. B. Bravaya, S. T. Brown, D. Casanova, C.-M. Chang, Y. Chen, S. H. Chien, K. D. Closser, D. L. Crittenden, M. Diedenhofen, R. A. DiStasio, H. Do, A. D. Dutoi, R. G. Edgar, S. Fatehi, L. Fusti-Molnar, A. Ghysels, A. Golubeva-Zadorozhnaya, J. Gomes, M. W. D. Hanson-Heine, P. H. P. Harbach, A. W. Hauser, E. G. Hohenstein, Z. C. Holden, T.-C. Jagau, H. Ji, B. Kaduk, K. Khistyaev, J. Kim, J. Kim, R. A. King, P. Klunzinger, D. Kosenkov, T. Kowalczyk, C. M. Krauter, K. U. Lao, A. D. Laurent, K. V. Lawler, S. V. Levchenko, C. Y. Lin, F. Liu, E. Livshits, R. C. Lochan, A. Luenser, P. Manohar, S. F. Manzer, S.-P. Mao, N. Mardirossian, A. V. Marenich, S. A. Maurer, N. J. Mayhall, E. Neuscamman, C. M. Oana, R. Olivares-Amaya, D. P. O'Neill, J. A. Parkhill, T. M. Perrine, R. Peverati, A. Prociuk, D. R. Rehn, E. Rosta, N. J. Russ, S. M. Sharada, S. Sharma, D. W. Small, A. Sodt, T. Stein, D. Stuck, Y.-C. Su, A. J. W. Thom, T. Tsuchimochi, V. Vanovschi, L. Vogt, O. Vydrov, T. Wang, M. A. Watson, J. Wenzel, A. White, C. F. Williams, J. Yang, S. Yeganeh, S. R. Yost, Z.-Q. You, I. Y. Zhang, X. Zhang, Y. Zhao, B. R. Brooks, G. K. L. Chan, D. M. Chipman, C. J. Cramer, W. A. Goddard, M. S. Gordon, W. J. Hehre, A. Klamt, H. F. Schaefer, M. W. Schmidt, C. D. Sherrill, D. G. Truhlar, A. Warshel, X. Xu, A. Aspuru-Guzik, R. Baer, A. T. Bell, N. A. Besley, J.-D. Chai, A. Dreuw, B. D. Dunietz, T. R. Furlani, S. R. Gwaltney, C.-P. Hsu, Y. Jung, J. Kong, D. S. Lambrecht, W. Liang, C. Ochsenfeld, V. A. Rassolov, L. V. Slipchenko, J. E. Subotnik, T. Van Voorhis, J. M. Herbert, A. I. Krylov, P. M. W. Gill, M. Head-Gordon, *Mol. Phys.* **2015**, *113*, 184.
[46] J. P. Perdew, *Phys. Rev. B* **1986**, *33*, 8822.
[47] A. D. Becke, *Phys. Rev. A* **1988**, *38*, 3098.
[48] F. Weigend, R. Ahlrichs, *Phys. Chem. Chem. Phys.* **2005**, *7*, 3297.
[49] E. O. Pyzer-Knapp, K. Li, A. Aspuru-Guzik, *MAML – Molecular Applied Machine Learning.*
[50] G. E. Dahl, N. Jaitly, R. Salakhutdinov, **2014**, arXiv:1406.1231.
[51] J. Hachmann, R. Olivares-Amaya, S. Atahan-Evrenk, C. Amador-Bedolla, R. S. Sanchez-Carrera, A. Gold-Parker, L. Vogt, A. M. Brockway, A. Aspuru-Guzik, *J. Phys. Chem. Lett.* **2011**, *2*, 2241.
[52] World Community Grid, http://www.worldcommunitygrid.org/ (accessed February 2015).