Cover Letter for Writing Sample

I put two paper drafts in this writing sample, the latest link will be in my resume, and also available on my website [here].

- Page [2-10]
  S. Liu, H. Wang, Y. Liang, A. Gitter. Avoiding Negative Transfer on a Focused Task with Deep Multi-task Reinforcement Learning.
- Page [11-46]
  Here I only include the main results in paper body, while the complete results in SI will make paper grow over hundred of pages, and I remove since it doesn't affect the conclusion. The full paper will contain all of them.
  S. Liu*, M. Alnammi*, S. Ericksen, A. Voters, H. Wu, J. Keck, M. Hoffman, S. Wildman, A. Gitter. In preparation. In-Vitro Chemical Screening Guided by In-Silico Learning: Bridging the Gap between Prediction and Practice.

AVOIDING NEGATIVE TRANSFER ON A FOCUSED TASK WITH DEEP MULTI-TASK REINFORCEMENT LEARNING

Shengchao Liu[1] Hongyi Wang[1] Yingyu Liang[1]
Anthony Gitter[1,2,3]
[1]Department of Computer Science, University of Wisconsin-Madison
[2]Department of Biostatistics and Medical Informatics, University of Wisconsin-Madison
[3]Morgridge Institute for Research
{shengchao, hongyiwang, yliang}@cs.wisc.edu, gitter@biostat.wisc.edu

1 INTRODUCTION

Multi-task learning aims to exploit useful information in related learning tasks to improve the generalization performance of all the tasks jointly. Deep multi-task learning has been successfully applied under various scenarios such as virtual chemical screening Ma et al. (2015); Ramsundar et al. (2015), image detection Lu et al. (2016); Misra et al. (2016), genomics Kelley et al. (2016); Zhou & Troyanskaya (2015), and health prediction Jaques et al. (2016). The benefits of multi-task learning come from transferring knowledge among related tasks, which can reduce overfitting, especially when the data for some tasks is limited Liu et al. (2015); Ruder (2017a). But the underlying assumption that all tasks are related is not always true.

Although the performance may improve on average over all tasks in a multi-task model, for some specific tasks, the multi-task performance can be worse than a single-task model. We refer to this decrease in performance in the multi-task setting as negative transfer, and the problem occurs naturally in real datasets. Despite abundant approaches for multi-task learning, few methods explicitly aim to boost multi-task performance while minimizing negative transfer on specific tasks. Previously, shallow models Kang et al. (2011); Kumar & Daumé (2012) applied sparsity and clustering constraints to guide the training strategy for dissimilar tasks, but how deep neural networks can adopt such ideas is not well studied.

Our work first proposes to solve negative transfer issue by applying reinforcement learning to control the training process. We will start by focussing on one task, and argue that policy can help guide the deep network to select only the important information transferred from other tasks. We come up with this deep multi-task reinforcement learning (DMTRL) framework, and try to generalize it to different domains.

2 RELATED WORK

Self-Paced Curriculum Learning Self-paced curriculum learning borrows idea from education: it is reasonable for people to take curriculum from easy ones to complex ones. Curriculum learning suggests using easy samples to train the model first, then continue by adding more complicated samples. The easiness is hard to determine, and Kumar et al. (2010) introduces self-paced learning, where the easiness is identified by the learned model.

Then following works extend this from single-task model to multi-task case. Li et al. (2017) introduces a joint objective function by adding weights and regularizers on each task. And Murugesan & Carbonell (2017) uses a threshold on residual of each task to guide which group of tasks are easy, therefore for gradient updates. Above methods need the assumption that all tasks are related, and Pentina et al. (2015) handles dissimilar tasks by applying multiple task learning sequences.

Clustering-based Multi-task Learning Many works have been proposed in literature that using clustered tasks, grouping tasks using different notion of grouping. Some methods assume that parameters of tasks that can be grouped in the same cluster are either close to each other in some distance metric or share a common probabilistic prior, tasks assigned in different clusters didn't interact with each other Jacob et al. (2009); Chen et al. (2011). Similar to method proposed in this paper, many methods are trying to find probabilistic model that attempt to extract covariance matrix among each task pairs and use it in training predictorsZhang & Schneider (2010); Guo et al. (2011).

Another assumption is clustering-based method is that task parameters lie in a low dimensional subspace, which captures model structure shared among all tasks. Some methods assume that some features (with all raw and transformed features contained) are inactive for all tasks. And thus all tasks parameters are pushed to a low dimensional subspaceArgyriou et al. (2008). Other method follows the low dimensional subspace assumption but allows the tasks in different task groups to overlap with each other in one or more bases Kumar & Daumé (2012).

Deep Multi-task Learning Recently, more works have been using deep neural network. It benefits from taking the raw data as input, extracting the latent feature, so as to make better predictions.

However, only a few works Ruder (2017b) have been focusing on a better generalized deep multi-task framework. Ensemble is one traditionally adopted option, and Lee et al. (2015) introduces a similar idea called TreeNets, an ensembled deep neural network. In TreeNets, first part of layers are shared among tasks, and following layers are trained independently. Lu et al. (2016) proposes a framework that is able to dynamically construct a hierarchical network structure and selectively share information among closely-related tasks. Both methods require large amount of computation memory, especially when the hidden space is in high dimension.

Meta Learning and Reinforcement Learning Meta learning on deep network is a new coming area. Meta learning, or learning to learn, aims at making model being able to learn the learning process, so it can generalize well on new tasks or new samples. When it comes to deep network framework, there are many interesting and potentially constructive points, like optimal learning rates, batch size, predicting loss and gradients. There have been some recent works on such ideas, Andrychowicz et al. (2016) uses a meta learner to learn the gradient, Ravi & Larochelle (2016) predicts the next step hidden layer parameter with recurrent neural network.

Reinforcement learning provides another option for meta learning. The intuition behind reinforcement learning is once a model or agent observed the environment, with the feedback, it can update its policy space to make decision towards higher expected rewards. So a policy agent can fit well as meta learner, considering that agent can Finn et al. (2017) applies meta learning to find a better base model, so as being able to quickly transfer to new unseen but related tasks under some task distribution, where only a few data points are available. This setting is also called the few-shot learning, where data insufficient is very common on new tasks, and models are supposed to converge well within a few epochs on a limited number of data. Distral Teh et al. (2017) proposes a framework for simultaneously training multiple reinforcement tasks by learning a distilled policy, but is not for solving the multi-task classification problems.

Virtual Screening Deep learning methods showed overwhelming results starting from Merck (2012); Dahl (2012) Merck Molecular Activity Challange, 2012. The goal is to get a discriminator being able to tell if a given small molecule has active interactions against the target protein. And recent works Mayr et al. (2016); Dahl et al. (2014); Ma et al. (2015); Unterthiner et al. (2014); Ramsundar et al. (2015); Kearnes et al. (2016) have been investigated multi-task deep neural network and proved its outstanding performance compared with classical machine learning methods.

Fingerprint encodes each molecule structure into 1024 binary bits, each bit represents one substructure. Besides, SMILES can be used to represent the atom sequential orders, and therefore feed in as the model input. Jastrzębski et al. (2016) makes model comparison based on input features, including Recurrent Neural Network Language Model and Convolutional Neural Networks with SMILES, and shows that CNN is best when evaluated log-loss. Gomes et al. (2017) proposes Atomic Convolutional Networks (ACNN), which encodes the 3D neighborhood relation into 2D structure, so as feed in to the CNN. Ramsundar et al. (2017) proposes progressive and bypass network models, with soft parameter sharing to communicate shared knowledge among tasks. It also observes the negative transfer issue, but lacks promising solutions.

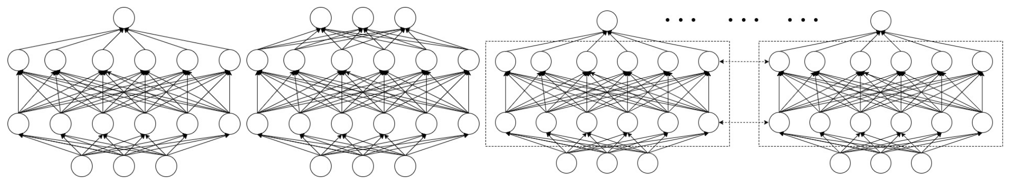
3 PROBLEM BACKGROUND

3.1 ANNOTATION

Suppose we have T tasks, corresponding to T datasets, and each dataset $S_t = \{X_t, Y_t\}$, $t \in \{0, 1, \ldots, T-1\}$. All the instance-task labels can compose a complement matrix $C \in \mathbb{R}^{n \times T}$, where n is the number of union of instance set and T is the task number. This complement matrix C fills in 'NaN' for missing items when combined all $\{Y_0, Y_1, \ldots, Y_{T-1}\}$ indexed by instance. In the context of stochastic gradient descent, one epoch, or one data pass, refers to using all samples for gradient updates once. W is the complete weight parameter for neural network.

3.2 DEEP SINGLE-TASK AND MULTI-TASK LEARNING

Classical neural networks include single-task models and multi-task models with hard or soft parameter sharing (Figure 3). We emphasize the hard-sharing parameter model, which requires less memory and exhibits negative transfer.



(a) Single-task (b) Multi-task Hard (c) Multi-task Soft

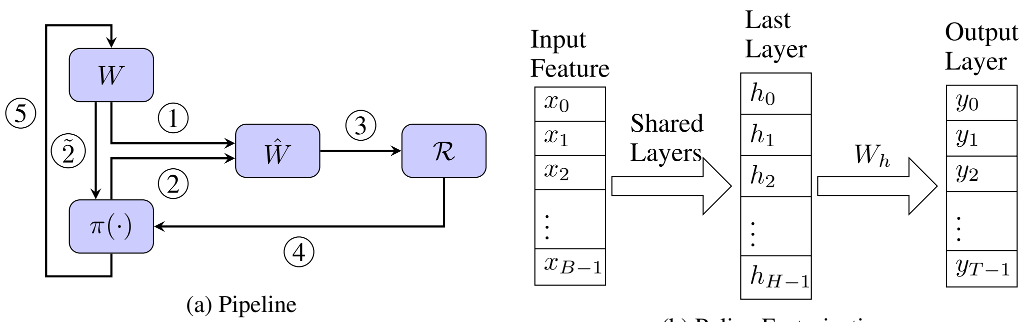
Figure 1: (a) contains one unit in the output layer representing a single task. (b) shares all parameters among tasks except at the output layer, where multiple predictions are made. (c) is a soft-sharing model. Each task has an identical layer structure with parameter similarity constraints.

The intuition behind MTL approaches is to transform knowledge among tasks, and improve performance especially when they are highly correlated. Besides, when data is insufficient for some tasks, learning can benefit from transferring representation from closely related ones. This situation can be extended to few-shot learning.

4 METHODOLOGY

Here we propose an algorithm, deep multi-task reinforcement learning (DMTRL), to avoid negative transfer on pre-defined *focused task* during the training process. Focused task is one for which we prioritize performance while using the remaining tasks to constrain and guide the model training. Because negative transfer can happen when tasks are not sufficiently related, we propose to consider only one subgroup of similar tasks for each gradient update during neural network training, learning which other tasks are most informative for the focused task. We use reinforcement learning to execute this strategy. Reinforcement learning has been used for fast convergence in few-shot learning Finn et al. (2017) and learning gradients by meta learning Andrychowicz et al. (2016). Our DMTR-L policy maps the multi-task network state to an action ($\pi(\cdot) : \mathcal{S} \to \mathcal{A}$, in which $\mathcal{S}, \mathcal{A}$ are state, action spaces). The policy is a binary bit vector representing one subgroup of tasks to be selected when updating the gradient. By combining domain knowledge, like task similarities, into the reward function, the DMTRL policy can help choose which subgroup to train on in iterations of each epoch.

The detailed DMTRL pipeline is illustrated in Figure 2a. At the beginning of each epoch, we sample actions under some distribution and get rewards after trial epochs. The value-based reinforcement learning assumes all actions are distributed uniformly and exhaustively enumerates them to select the action with highest reward as the best policy. While the value-based method can find the best policy, it is impractical in reality, because the time complexity is exponential in the number of tasks. A solution to this is to apply a policy gradient method, which allows us to parameterize the multi-task network by Θ and sample actions under the dynamically updated distribution. Policies are then updated with stored action-reward pairs, and up-to-date actions will be generated to guide multi-task training for each epoch.



(a) Pipeline

(b) Policy Featurization

Figure 2: In figure 2a, at the start of epoch t, $W = W_t$ is the multi-task network parameters. Before the gradient update, there are several trial episodes. $\textcircled{1}$ update $\hat{W}$ by taking current state W_t and $\textcircled{2}$ the sampled policy $\pi(\cdot)$. $\textcircled{3}$ get the rewards and $\textcircled{4}$ use them as feedback to select the best policy $\pi(\cdot)$. $\textcircled{5}$ use this optimal policy to update the gradient. For the policy-based method parameterized by Θ, the extra step $\tilde{\textcircled{2}}$ maps from state W_t to an action. In step $\textcircled{4}$ $\pi(\cdot)$ will be updated once receiving rewards. Following in figure 2b, we highlight the featurization part $\textcircled{1}$. For each batch with size of B, the input data $x_i \in \mathbb{R}^d$, where d is feature dimension. After some shared layers among tasks, it will go to last layer with H hidden units. Final output is $y = \sigma(W_h \cdot [h_0, \dots h_{H-1}])$, where $W_h \in \mathbb{R}^{H \times T}$, T is the task number. Here we can clearly observe that among different tasks, all parameters are shared except the last layer W_h, so we will use it as feature to feed in to our policy.

4.1 Policy Setup

In this section, we will elaborate how we set up policy, the key component in DMTRL. As illustrated in Figure 2, the projection parameter from last layer will be used for featurization. Two options are offered: the most brute-force way is to directly use parameter W_h as input feature for policy, but this can bring some problems. (1) the memory and computation cost will grow linearly as task number increases. (2) this cannot handle the *temporal dependency* issue. Suppose we have 2 tasks, $W_h \in \mathbb{R}^{H \times 2}$. At epoch τ, only first task ($W_H^{(0)}$) gets updated, and the second task ($W_H^{(1)}$) remains invariant. Then continue to epoch $\tau + 1$, we still have same input feature for second task ($W_H^{(0)}$), but the corresponding reward and label can be different. Based on this, the ideal featurization should be able to catch the temporal dependency. So we introduce using the gradient ∇W_h instead of W_h as input feature, which can take over the temporal dependency issue better.

Next we will introduce both model-free and model-based policies, and the latter one contains two variants: value-based and policy-based methods.

4.1.1 Model-free Policy

Model-free applies heuristic strategies, and it gets rid of the policy trial and update stages in figure 2a. Recall that the goal is to avoid negative transfer, and to reach this, in each stochastic gradient descent step, we hope the gradients can go to the optimal solution, and ignore the impact from unrelated tasks. One natural solution is to use class weight to control this process, where we interpret the class weight as the relevance from referenced task to focused one. To be more specific, the class weight of some reference task will be set to 0 if it is not related to focused one, so more related task will have higher class weight. We will include two types of featurization: the parameter of last layer W_h and gradient of last layer ∇W_h, and class weight is the cosine similarity based on that.

4.1.2 Model-based Policy

Contrary to the model-free strategy, policy can be modularized. Here we will apply T independent classification models corresponding to each row in $W_h \in \mathbb{R}^{H \times T}$.

Value-based model is to learn the policy by maximizing the expected state-function values. At time step τ, we choose the best policy π and use this policy to update the multi-task network weights.

$$\pi_{\tau+1}(W_\tau) = \arg\max_a Q(W_\tau, a)$$

The $Q(W_\tau, a)$ is a value function which maps the state space and action space to the reward space; and in this context, it is a mapping from deep network model W and class weight $\{0,1\}^T$ to the evaluation metric value on the focused task, like AUC[PR]. Sutton & Barto (1998) TD and MC are two traditional methods to approximate Q-value. Details are described in appendix Algorithm 1.

If the action space is discrete, we can enumerate all possible policies. But if the action space is too large, or it is continuous, Monte-Carlo is an alternative option.

The classical value-based methods, like Q-learning, SARSA, has one biggest drawback: given discrete action space, the computation time will grow exponentially, as the number of tasks increases. One solution is to use a parametric model, like neural network in deep Q-network (DQN) Mnih et al. (2013) to approximate Q-value. To be more specific, we will use a meta-RL agent to parameterize the policy, π_θ. The input is the model state (like hidden layer parameter), and output is the probability of each task applied for update in current epoch.

Policy-based model is another option. Once we have a parametric model to approximate the Q-value, instead of applying it in the bootstrap framework for policy selection, one natural thinking would be directly output the policy. And update-to-date method is to use policy gradient to train π_θ, where the goal is to maximize the expectation $\mathbb{E}_{a\sim\pi_\theta}[Q(W,a)]$. Here is the gradient:

$$\nabla_\theta \mathbb{E}_{a\sim\pi_\theta}\big[Q(W,a)\big] = \mathbb{E}_{a\sim\pi_\theta}\big[Q(W,a)\nabla_\theta \log \pi_\theta(a \mid W)\big]$$

Applying this for gradient updates, and at each time step t, we draw next action according to this newly updated distribution. Details are described in appendix Algorithm 2.

5 Experiments and Results

We test our DMTRL algorithm on virtual chemical screening tasks, where the goal is to predict the biochemical activity or other properties of a chemical compound. In this domain, the number of labeled training examples is typically limited, making multi-task learning appealing. We are conducting experiments on two virtual screening datasets (details in the supplement).

In the **Kaggle Merck Challenge dataset** Merck (2012), we predict real values for the provided chemicals on 15 tasks. Each task represents the inhibition or binding of a specific protein target or some other biochemical property.

In the **PubChem BioAssay (PCBA) dataset** Wang et al. (2012), each task is a binary classification problem. We predict whether a given chemical is active for 128 target proteins. Previous studies of these datasets used multi-task neural networks, and negative transfer was observed for some tasks Ramsundar et al. (2015; 2017). This makes them appealing for our efforts to eliminate negative transfer with DMTRL.

5.1 PCBA

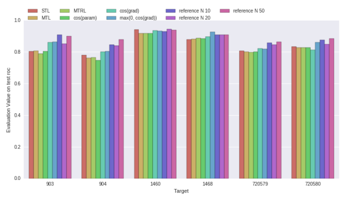
We selected 3 pairs of tasks that can be highly related based on domain knowledge. For evaluation metrics, AUC[ROC], AUC[BEDROC] and AUC[PR] are most widely used.

Results:

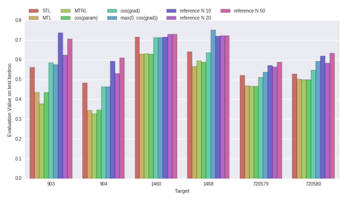
1. Focused > STL > MTL
2. For randomly sampled ones, N=50 > N=20 > N=10 > N=1 does not hold.

5.2 Kaggle

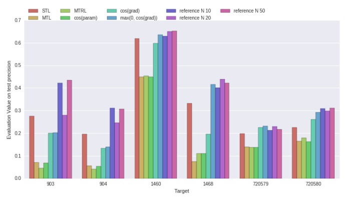
TBA



(a) AUC[ROC]



(b) AUC[BEDROC]



(c) AUC[PR]

Figure 3: These are performance on 6 selected tasks.

6 Conclusion

We get some quite promising and robust results using focused learning to handle the negative transfer issue. But there are still some other challenges.

Now we fix the negative transfer for focused task case, and if we want to generalize this to more focused tasks, that can trigger more open questions. Of course we can always separate n-focused-task problem into n independent focused training problems, but how to simultaneous get them is not trivial.

Another interesting issue remains like how to handle the *temporal dependency*. Fully utilize the gradient is one powerful way, and how to rigorously prove this will be included in the full paper.

All the code is available at GitHub Repository.

References

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

Andreas Argyriou, Andreas Maurer, and Massimiliano Pontil. An algorithm for transfer learning in a heterogeneous environment. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, ECML PKDD '08, pp. 71–85, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-87478-2. doi: 10.1007/978-3-540-87479-9_23. URL `http://dx.doi.org/10.1007/978-3-540-87479-9_23`.

Jianhui Chen, Jiayu Zhou, and Jieping Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pp. 42–50, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0813-7. doi: 10.1145/2020408.2020423. URL `http://doi.acm.org/10.1145/2020408.2020423`.

George Dahl. Deep learning how i did it: Merck 1st place interview. *Online article available from http://blog. kaggle. com/2012/11/01/deep-learning-how-i-did-it-merck-1st-place-interview*, 2012.

George E Dahl, Navdeep Jaitly, and Ruslan Salakhutdinov. Multi-task neural networks for qsar predictions. *arXiv preprint arXiv:1406.1231*, 2014.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Joseph Gomes, Bharath Ramsundar, Evan N Feinberg, and Vijay S Pande. Atomic convolutional networks for predicting protein-ligand binding affinity. *arXiv preprint arXiv:1703.10603*, 2017.

Shengbo Guo, Onno Zoeter, and Cédric Archambeau. Sparse bayesian multi-task learning. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 24*, pp. 1755–1763. Curran Associates, Inc., 2011. URL `http://papers.nips.cc/paper/4242-sparse-bayesian-multi-task-learning.pdf`.

Laurent Jacob, Jean philippe Vert, and Francis R. Bach. Clustered multi-task learning: A convex formulation. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (eds.), *Advances in Neural Information Processing Systems 21*, pp. 745–752. Curran Associates, Inc., 2009. URL `http://papers.nips.cc/paper/3499-clustered-multi-task-learning-a-convex-formulation.pdf`.

Natasha Jaques, Sara Taylor, Ehimwenma Nosakhare, Akane Sano, and Rosalind Picard. Multi-task learning for predicting health, stress, and happiness. In *NIPS Workshop on Machine Learning for Healthcare*. 2016.

Stanisław Jastrzębski, Damian Leśniak, and Wojciech Marian Czarnecki. Learning to smile (s). *arXiv preprint arXiv:1602.06289*, 2016.

Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 521–528, 2011.

Steven Kearnes, Brian Goldman, and Vijay Pande. Modeling industrial admet data with multitask networks. *arXiv preprint arXiv:1606.08793*, 2016.

David R. Kelley, Jasper Snoek, and John L. Rinn. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Research*, 26(7):990–999, July 2016. ISSN 1088-9051, 1549-5469. doi: 10.1101/gr.200535.115. URL `http://genome.cshlp.org/content/26/7/990`.

Abhishek Kumar and Hal Daumé, III. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pp. 1723–1730, 2012. ISBN 978-1-4503-1285-1. URL `http://dl.acm.org/citation.cfm?id=3042573.3042793`.

M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pp. 1189–1197, 2010.

Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015.

Changsheng Li, Junchi Yan, Fan Wei, Weishan Dong, Qingshan Liu, and Hongyuan Zha. Self-paced multi-task learning. In *AAAI*, pp. 2175–2181, 2017.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 912–921, May–June 2015. URL `http://www.aclweb.org/anthology/N15-1092`.

Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. *arXiv preprint arXiv:1611.05377*, 2016.

Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 55(2):263–274, 2015.

Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. Deeptox: toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80, 2016.

Merck. Merck molecular activity challenge. *https://www.kaggle.com/c/MerckActivity*, 2012.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3994–4003, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Keerthiram Murugesan and Jaime Carbonell. Self-paced multitask learning with shared knowledge. *arXiv preprint arXiv:1703.00977*, 2017.

Anastasia Pentina, Viktoriia Sharmanska, and Christoph H Lampert. Curriculum learning of multiple tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5492–5500, 2015.

Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande. Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072*, 2015.

Bharath Ramsundar, Bowen Liu, Zhenqin Wu, Andreas Verras, Matthew Tudor, Robert P Sheridan, and Vijay S Pande. Is multitask deep learning practical for pharma? *Journal of Chemical Information and Modeling*, 2017.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017a. URL `http://arxiv.org/abs/1706.05098`.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017b.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *arXiv preprint arXiv:1707.04175*, 2017.

Thomas Unterthiner, Andreas Mayr, Günter Klambauer, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, and Sepp Hochreiter. Deep learning as an opportunity in virtual screening. *Advances in neural information processing systems*, 27, 2014.

Yanli Wang, Jewen Xiao, Tugba O Suzek, Jian Zhang, Jiyao Wang, Zhigang Zhou, Lianyi Han, Karen Karapetyan, Svetlana Dracheva, Benjamin A Shoemaker, et al. Pubchem's bioassay database. *Nucleic acids research*, 40(D1):D400–D412, 2012.

Yi Zhang and Jeff Schneider. Learning multiple tasks with a sparse matrix-normal penalty. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems*, NIPS'10, pp. 2550–2558, USA, 2010. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=2997046.2997180`.

Jian Zhou and Olga G. Troyanskaya. Predicting effects of noncoding variants with deep learning-based sequence model. *Nature Methods*, 12(10):931–934, October 2015. ISSN 1548-7091. doi: 10.1038/nmeth.3547. URL `http://www.nature.com/nmeth/journal/v12/n10/abs/nmeth.3547.html`.

A Algorithm for Model-based Policy

Algorithm 1 Value-based Method

```
Initialize Neural Network $W_0$, initialize $\pi_0 = \{1\}^T$
repeat
    $\pi_{\tau+1}(W_t) = \arg\max_{a \in \{0,1\}^T} \; Q(W_t, a)$
    $\mathcal{L} = \sum_{i=1}^{T} \pi_{\tau+1}^{(i)} \cdot \mathcal{L}(\mathcal{Y}^{(i)}, f(\mathcal{X}^{(i)}, W_t^{(i)}))$
    update weight $W_{\tau+1}$
until Convergence
```

Algorithm 2 Policy-based Method

```
Initialize Neural Network $W_0$, and $\pi_\theta$
repeat
    $\theta_{\tau+1} = \theta_t + \alpha \cdot \mathbb{E}\big[Q(W_\tau, a) \nabla_{\theta_\tau} \; \log \, \pi_{\theta_\tau}(a \mid W_\tau)\big]$
    $\mathcal{L} = \sum_{i=1}^{T} \pi_{\theta_{\tau+1}}^{(i)} \cdot \mathcal{L}(\mathcal{Y}^{(i)}, f(X^{(i)}, W_\tau^{(i)}))$
    update weight $W_{\tau+1}$
until Convergence
```

In-Vitro Chemical Screening Guided by In-Silico Learning: Bridging the Gap between Prediction and Practice

Shengchao Liu[1] * Moayad Alnammi[1] * Spencer Ericksen[2,3] Andrew Voter[5]
Haozhen Wu[2,4] James Keck[5] Michael Hoffmann[2,6] Scott Wildman[2]
Anthony Gitter[1,3,7,8]
[1] Department of Computer Sciences, University of Wisconsin-Madison
[2] Small Molecule Screening Facility, University of Wisconsin Carbone Cancer Center
[3] Center for Predictive Computational Phenotyping, University of Wisconsin-Madison
[4] Department of Statistics, University of Wisconsin-Madison
[5] Department of Biomolecular Chemistry, University of Wisconsin-Madison
[6] McArdle Laboratory for Cancer Research, University of Wisconsin-Madison
[7] Department of Biostatistics and Medical Informatics, University of Wisconsin-Madison
[8] Morgridge Institute for Research

ABSTRACT

In a drug discovery pipeline, once a disease-relevant protein target has been identified, researchers face the daunting task of identifying chemical compounds that effectively modulate that target. Experimental phenotypic screening of thousands or millions of small molecules is time-consuming and expensive, whereas virtual (computational) screening can provide a small set of promising molecules that are more likely to be active towards the target protein. It acts as a pre-processing step for filtering the extremely large number of candidate chemicals.

Here we focus on the SSB-PriA and RMI-FANCM targets, and propose a standard pipeline in the real scenario. We also argue that the most popular evaluation metrics in this domain, area under the receiver operating characteristic curve, can be misleading and compare it with other evaluation metrics, showing which provide real-world value. Furthermore, we apply the most up-to-date models, including deep neural networks and recurrent deep networks, and compare these models in a real-world setting by assessing their ability to prospectively prioritize active compounds. We stated that ensemble models can show better performance, and first figured out the key components that make ensemble models outperform others. We utilized a **Simple Ensemble** model which can reach best performance to substantiate that the usage of both binary and continuous labels is most important. Moreover, we present a user-friendly framework for virtual screening tasks based on Keras, a neural network library built on top of Theano and Tensorflow.

*Authors contributed equally

1. INTRODUCTION

Drug discovery is a very timely and expensive challenge. The process starts by first identifying a target protein for which we would like to induce an altering effect upon via interaction with a compound. The interacting compounds are identified by screen-testing tens of thousands of candidate compounds with the target via a process called High-Throughput Screen (HTS) in the pharmaceutical industry. These tests produce a wealth of information that can be used for learning concepts in the HTS domain. The tests themselves are automated, but blindly testing millions of compounds can be timely and costly in the long run. Thus, there is a crucial need for a virtual screening process that acts as a preliminary step for prioritizing among the candidate compounds.

Virtual screening includes two categories, structure- and ligand-based methods. Structure-based methods will consider the target structure and simulating the 3D structural interactions of the target and compounds. This requires knowledge of the structural properties and does not make use of historical screening data in its decision process, forgoing the ability to learn from the past. Alternatively, ligand-based methods assume no structural information on the target and uses the data generated from the HTS process along with machine learning techniques in order to learn concepts (e.g. if a compound will most likely bind with a target or not).

In this paper we will introduce a complete process to develop a ligand-based model on a newly generated screening benchmark. Apriori, we identify classes of models we want to try. Each class defines a set of models that can be tuned via parameters, and so, in the first stage we manually select a subset of these parameters. We further prune these selected models using a subset of the dataset and advance them to the next stage. Models that advance are then scrutinized further by k-fold cross-validation and hypothesis testing to better assess generalization and prospective performance. In the final stage, we re-assessed our model generalizations on a hold-out dataset, which is generated in parallel to previous stages.

In this paper, we make the following contributions:

- An in-depth multi-stage approach to virtual screening in a collaborative setting between in-vitro and in-silico groups. In-vitro results are handed to the in-silico group for analysis, model-training, and future hit proposals. We further analyze the next batch of tests to judge the generalization of the initial models.
- We put different classes of models against each other on a real-world screening dataset. All illustrate ligand-based machine learning models outperform structure-based docking models.
- We analyze the different metrics used in the field of virtual screening and conclude which metric best coincides with the number of hits found. Unsurprisingly, each metric can attribute to bias for model selection, but the choice of metric indirectly affects model selection.
- We carefully scrutinize the prospective screening results to showcase that ensemble methods can work better, and introduce a **Simple Ensemble** model that can reach best performance. By this, we illustrated that where both binary and continuous labels are used yield most powerful models.

2. BACKGROUND

2.1 Dataset

Our case study is on a newly generated dataset [12, 34] **SSB-PriA and RMI-FANCM dataset**. The Keck laboratory has conducted in-vitro high-throughput screening on two interactions: SSB-PriA and RMI-FANCM. SSB and PriA are two proteins, and the target is whether or not the involvement of a molecule can prevent their binding. Similar cases for RMI and FANCM. This dataset consists of 5 labels per molecule: PriA-SSB AS , PriA-SSB FP , PriA-SSB AS %inhibition, RMI-FANCM and RMI-FANCM % inhibition.

PriA-SSB alpha screen (AS) Retest: The alpha screen assay was run initially on all 75k compounds as a single test. Those that tested above a certain threshold (35% inhibition) and pass chemical structural filters were tested a second time in the same assay. Those that were confirmed in a secondary AS screen (again above 35%) were marked as actives in the binary labels. We considered an additional 25k compounds for the prospective screen. In this set, actives were defined as those with at least 35% inhibition that passed the structural filter. We did not confirm hits with a secondary screen.

PriA-SSB fluorescence polarization (FP): This is a separate assay for the same target run only on the initial hits from the AS assay. Those that passed a threshold similar to the AS (30%) were declared to be hits. The remaining values (binary labels only) were set to zero.

RMI-FANCM : TBA.

In SSB-PriA and RMI-FANCM , the continuous data, % inhibition, corresponds to the AS primary screening values. Because secondary screens and structural filters are used to define a high-confidence set of active compounds, there is so single % inhibition threhsold that separates actives from inactives. If we sort the compounds by % inhibition, the binary labels will be segmented into pieces (Table 1), and comparing to hard-thresholding binary labels, this is more closely related to reality.

Table 1. Some examples onPriA-SSB AS . Molecule ID, actual binary label, and corresponding % inhibition values.

molecule ID	binary label	% inhibition
SMSSF-0548062	1	41.611
SMSSF-0018649	0	50.607
SMSSF-0018695	0	61.131
SMSSF-0019318	0	70.299
SMSSF-0548079	1	71.333

To help learn a better chemical representation with multi-task neural networks, we need more comprehensive screening contexts to transfer useful knowledge. We use **PubChem BioAssay (PCBA)** [36] for this purpose. PCBA uses a pre-determined hard threshold on IC_{50}. All the molecules above this threshold are active and all below are inactive. The details are in the appendix B.

2.2 Feature Representation

2.2.1 *Extended Connectivity Fingerprint*

Extended Connectivity Fingerprint (ECFP) [28] is a widely accepted featurization mechanism to convert molecules to fixed-length bit strings. It is an iterative algorithm that encodes the circular substructures of the molecule as identifiers at increasing levels with each iteration. In each iteration, hashing is applied to generate new identifiers, and thus, there is a chance that two substructures are represented by the same identifier. In the end, a list of identifiers encoding the substructures are folded to bit positions of a fixed-length bit string. A 1-bit at a particular position indicates the presence of a substructure (or multiple substructures) and a 0-bit indicates the absence of any substructure. The number of iterations, also called the diameter, d and length of the bit string l is set by the user. We used the common setting of $d = 4$ and $l = 1024$. Figure 1 illustrates the concept with a small fixed-length bit string.

ECFPs are commonly used as features for molecules in predicting drug activity. [21] report that a Deep Neural Network trained on ECFPs had similar performance to one trained on molecular descriptors. We use them in a supervised learning setting where the input features are the ECFP fingerprints and the target activity are the output labels. The goal is to train a supervised learning model that is able to generalize to unseen fingerprint instances.

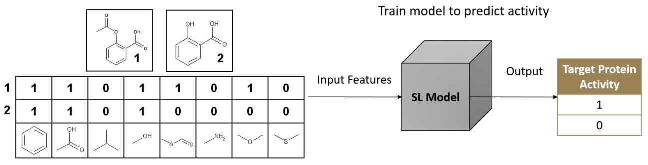


Figure 1. ECFP fingerprints used in a supervised learning (SL) setting for learning drug activity. .

2.2.2 *Simplified Molecular Input Line Entry System*

The second option for feature representation is [37] Simplified Molecular Input Line Entry System(SMILES). Each molecule can be represented via a SMILES sequence, which consists of around 35 different characters. For example, c1cc(oc1C(=O)Nc2nc(cs2)C(=O)OCC)Br is a canonical SMILES for the molecule in Fig. 2. Each alphabet represents an atom, except for Br, Cl, and Si, any sequence between two same number is a ring.

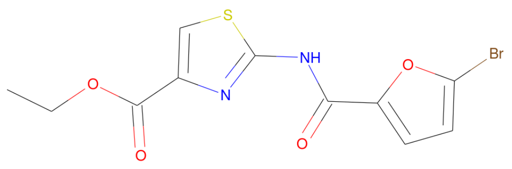


Figure 2. Canonical SMILES is c1cc(oc1C(=O)Nc2nc(cs2)C(=O)OCC)Br.

Data augmentation has been widely accepted in [35] image classification problem, like rotating, mirroring, adjusting contrast images. The fundamental ideas behind data augmentation is that input data are first mapped into a latent feature space so that predictions are made on this space. The usual way people do is manual feature extraction, which in theory is worse than this learned latent feature space. This latent space possess the advantages of both fully representing the data and well understood by computer.

[14] Data augmentation has also been applied in virtual screening works to get more SMILES strings. There are specific rules to generate canonical SMILES , and we randomly pick up some starting points to go over the graph for augmentation.

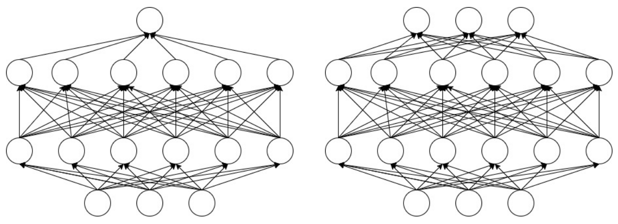
3. METHODOLOGY

3.1 Models

We selected a large number of existing virtual screening approaches for our benchmarks and prospective predictions. These included a variety of supervised learning approaches, structure-based docking, and a chemical similarity baseline. To elaborate which model require what kind of labels and targets, we specify it in Appendix. F.

3.1.1 *Neural Networks*

Deep learning is a powerful machine learning method that has benefited greatly from advancements in training algorithms, GPU architecture, large amounts of labeled data, and software frameworks. One of the potential powers of deep learning is its ability to extract latent features. In traditional machine learning methods, the model build-up has two parts: feature engineering and model training; but in deep learning, it allows modelers to focus less on feature extraction, and instead provide the "raw" feature, like image pixels in image recognition and words in end-to-end natural language models. The deep network can automatically learn the representative features for the modeling task.



(a) Single-task Neural Network (b) Multi-task Neural Network

Figure 3. Deep Neural Network Structures. Only consider two layers, X as input layer and Y as output layer, and parameters W between two layers. $\hat{Y} = \sigma(WX + b)$. Fig. 3(a) has only one unit on output layer. Fig. 3(b) has multi units on output layer representing different targets.

Single-task neural network: Network structure is described in Figure 3(a). For the three binary SSB-PriA and RMI-FANCM targets, take ECFP as input features, train a neural network on each of the three binary targets. In regression problem, we can apply the same network structure as before, but use % inhibition instead of binary labels. It can enrich the model performance by providing continuous labels.

Multi-task neural network: The very first idea was applied by winner in [5,6] Merck Molecular Activity Challenge. As showed in Figure 3(b) , the intrinsic idea is transferring knowledge among tasks can improve the overall performance, which is so called the multi-task effect.

Single-task Atom-level LSTM: Long Short Term Memory (LSTM) is one of most prevalent recurrent neural network (RNN) models. RNN takes advantage of four inner cell units, which can keep memories from past history then make predictions for the next units. We take SMILES as input feature, and apply [24] skip-gram language model. First for each molecule, use one-hot vector to represent each atom. It takes the context for each character and embeds to a feature space. The output prediction is binary label against the target.

Influence Relevance Voter (IRV): IRV, introduced by [19,32], is a hybrid between k-nearest-neighbors and neural

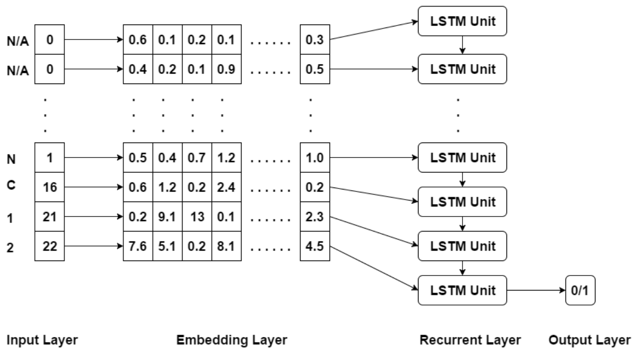


Figure 4. LSTM Structure. Input Layer is padding each molecule compound to a fixed size $[91 \times 1]$ vector. Embedding layer is to embed each atom in SMILES to a fixed vector, and in our case is $[10 \times 1]$ vector. So the whole matrix for one compound is $[91 \times 10]$. For recurrent layer, we apply LSTM units. Output layer is simply the active or inactive label prediction.

networks. The idea is that a molecule's label prediction is computed by nonlinearly combining the predictions of its nearest neighbors. This nonlinear combination takes into account the predictions and the similarity of the neighbors. The weights of combining the neighbor predictions are represented by a simple neural network.

3.1.2 Ensemble Models

Ensemble model comes from the idea that different models can have preference to different kinds of predictions. And when considering all the models together, we might be able to reach better performance.

Random Forest: Randomized decision tree ensembles have been successful on many problems due to its variance reducing power. The idea is to build n decision trees with random subsamples of training data and random subset of the features. The classification results on a new point are then averaged from the n decision trees.

Calibrated Boosting-Forest: [39] builds a two-layer ensemble XGBoost framework [3]. The first layer consists of four base models: boosted tree and linear regressor with binary and continuous label, while the second layer perform as an ensemble model on top of it.

Simple Ensemble: We introduce a very simple ensemble method. First we train some base models, like single-task classification network, single-task regression networks, or random forest. Then for each compounds, get the higher rank on each of the base models and use this newly generated order as the prediction confidence on each compounds.

3.1.3 Structured-based Method: Docking

Extension of work from [9] previous paper on traditional and advanced consensus scoring methods for docking-based VS.

3.1.4 Chemical Similarity Baseline

In the prospective screening stage, we introduced a simple baseline based on chemical similarity to the known active compounds, which is representative of standard practice in high-throughput screening. For each of the 25k compounds in the prospective screening set, we computed the Tanimoto similarity with all of the PriA-SSB AS actives using the ECFP fingerprints. We kept the best similarity over all PriA-SSB AS actives as the compound score and ranked compounds to prioritize those that were most similar to a known active.

3.2 Evaluation Metrics

The area under the receiver operating characteristic curve (AUC[ROC]) has been widely accepted in [16, 18, 22, 26, 27]. ROC curve plots the relationship between true positive rate (TPR) or sensitivity and false positive rate (FPR) or specificity, which is defined in equation (1). As the false positive (FP) rate goes to 100%, all ROC curves will converge, so we may as well focus on the low FP rate part which can be more distinguishable among different ROC curves. Thus we consider the concentrated ROC (BEDROC) introduced in [31]: it enlarges the early ROC curve by some scaling function.

$$TPR = \frac{TP}{TP+FN}, \quad FPR = \frac{FP}{FP+TN} \tag{1}$$

$$Recall = \frac{TP}{TP+FN}, \quad Precision = \frac{TP}{TP+FP} \tag{2}$$

Area under the Precision-Recall curve (AUC[PR]) defined by equation (2) is another option. AUC[PR] has the advantage of highlighting classifier performance on identifying a class of interest, particularly in highly skewed datasets where the focus might be on positive instances. It is worth noticing that [8] proves there exists interpolation issues in computing AUC[PR] and provide an improved algorithm, and we also apply it.

Another common metric is enrichment factor (EF) which is the ratio between number of actives found in the top R predictions vs. the number of actives found at random. In other words, how much better does the method perform over random guessing based on the distribution of actives. Let $R \in [0\%, 100\%]$ is a pre-defined float number.

$$EF_R = \frac{\text{\# actual pos present in top R ranked predictions}}{\text{\# actual pos} \times R} \tag{3}$$

$$EF_{max,R} = \frac{min\{\text{\# actual pos, sample size} \times R\}}{\text{\# actual pos} \times R} \tag{4}$$

$EF_{max,R}$ represents the maximum enrichment factor possible at R. Difficulty arises when interpreting EF scores as they vary with the dataset and ratios. We introduce the normalized enrichment factor (NEF) defined below:

$$NEF_R = \frac{EF_R}{EF_{max,R}} \tag{5}$$

As $NEF_R \in [0,1]$, this makes it easier to compare performance based on enrichment factor; 1.0 is perfect enrichment factor. Furthermore, we can plot NEF_R vs. $R \in [0\%, 100\%]$ to get AUC[NEF] $\in [0,1]$.

We train regression models for the continuous % inhibition scores. The output of these models are then used as if they were confidence scores analogous to probability scores; higher scores equate to more confidence in being active. These confidence scores are then corresponded with the binary labels. With this treatment, the introduced evaluation metrics apply directly.

3.3 Pipeline

Our virtual screening assessment contains three stages:

1. Tune hyperparameters in order to prune the model search space.
2. Rank models on 5-fold cross-validation results and apply hypothesis testing for different metrics.
3. Assess all models' ability to prospectively identify active compounds from a new set.

In contrast to most other virtual screening studies, the prospective screens were not conducted until after all models were trained and evaluated in the cross-validation stage.

3.3.1 Hyperparameter Sweeping Stage

Hyperparameters are the parameters of a model that can be set by an expert as opposed to the weights or parameters that are learned during training. In the context of deep networks, the hyperparameters can be the number of hidden layers, the number of hidden units in each layer, types of activation functions, drop out ratios, types of regularizer, etc. In random forest, hyperparameters can be the number of trees, the size of the subsamples, the size of the subset of features, etc. In this stage, we apply grid search on a manually defined set of hyperparameters in order to prune the models considered for subsequent stages.

We first split the 75k SSB-PriA and RMI-FANCM dataset into 5 stratified folds as described in Appendix A. In this stage, we will use first 4 folds for hyperparameter sweeping, and all details are shown in Appendix G. The purpose of this stage is pruning models, so it can guarantee that our model is not chosen randomly.

3.3.2 Cross-Validation Stage

We will apply models described in section 3.1 after tuning hyperparemeters, and the classical cross-validation training strategy will be applied. The goal of this stage is to filter out the most promising models for the real application.

One of the biggest challenges is to select best models. Ideally, the best model can have overwhelming performance on all evaluation metrics, but this is hard to reach with existing models. And this problem can get more complicated when we use different evaluation metrics, for each of them may reveal different model performance ranks. We will illustrate how to choose the optimal models and evaluation metrics under various circumstances.

3.3.3 Prospective Screening Stage

In the real setting, a small molecule screening facility has limited funds to purchase compounds. We adopt this setting by providing each model a budget of 250 compounds, screening their predictions, and assessing which identified the most active compounds. The optimal models will find as many positive compounds as possible within the funding restriction.

This stage can also further verify our conclusions from previous stages. We show empirically that under this setting, NEF can better represent the model's learning ability.

4. CROSS-VALIDATION RESULTS

In this stage we want to assess and compare 41 models using the 5 folds across different metrics. The following summarizes the steps for this stage:

1. Models: 8 deep neural networks (DNN), 8 random forests (RF), 5 IRV, 6 calibrated boosting-forest(CBF), and 14 docking.
2. 5-fold cross-validation is done for RF, CBF, IRV and docking. 4 by 5-fold cross-validation is done for DNNs, since neural network requires an extra dataset as early stopping criterion. Each produces a test set score. We will discuss the plottings in
3. For each fold iteration, on the test set, we record: AUC[ROC], AUC[PR], AUC[NEF], and EF/NEF at the following percentiles $[0.001, 0.0015, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2]$.

For each iteration of k-fold cross-validation, in deep neural networks and IRV we pick 3 folds for training and 1 fold for validation and early stopping. For random forest, since the sklearn implementation does not implement early stopping, we train on the complete 4 folds. This may cause slightly biased decisions, and we will discuss further in Section 4.

To compare multiple models, we can use statistical tests that account for the multiple testing problem. We use Tukey's range test for pairwise comparison to assess whether the mean metrics of two models are significantly different. In the event that the Tukey test does not produce significance, and seeing how we are trying to mimic a real scenario, we perform an ad-hoc comparison based on the absolute metrics. We propose the following for *each metric*:

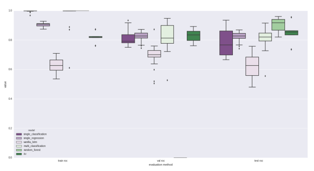
1. Compare models using Tukey's range test on the test set scores. The result of this step assigns wins to models for which Tukey's test finds significance.

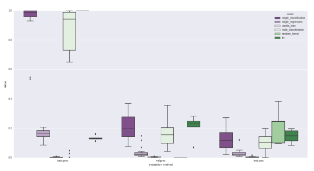
2. Rank models based on results of Tukey's test.

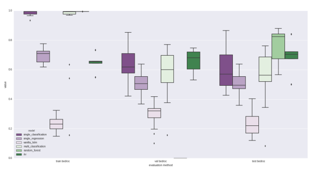
With this, we hypothesize that these top performers will perform the best on the new 25k molecule dataset. We will affirm this statement when we go to the prospective screening stage.

4.1 Evaluation Metric Plotting

We select 6 out of 41 models, and plot the corresponding evaluation values on training, validation, and test dateaset in Figure 5.


(a) AUC[ROC]


(b) AUC[PR]


(c) AUC[BEDROC]

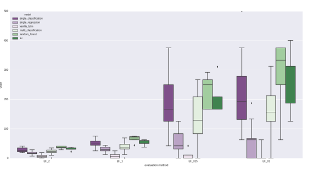

(d) EF

Figure 5. Six models trained on task PriA-SSB AS . Plot evaluations on training, validation, and test set. Random forest model does not require a specific validation set for early stopping, so its corresponding evaluation values are set to 0.

The overall performance using AUC[ROC] are comparatively promising on all models, where all but LSTM can reach around 0.8. And among all, random forest shows to be the best model, reaching approximately 0.9. This conclusion still holds when we turn to AUC[PR] and AUC[BEDROC], but they also introduce another problem: as one noticing observation from the plot, the huge gap between training, validation, and test set on most models. Obviously single-task and multi-task networks and random forest get overfitted on training set, reaching almost 100% AUC, and it drops dramatically on both validation and test set.

One possible explanation is the inappropriate representation of compounds. Given the fact that we use only 1024 fingerprints, the reason can be either the bits of fingerprints is not large enough, or fingerprint alone cannot fully represent the compound from the machine learning models' aspect. To check the first reason, we try to extend the number of bits, like 4096, but it has show no remarkable improvements. In addition, since fingerprints are generated from canonical S-MILES, which can be treated as a sequence of atoms, we may as well add the recurrent network to check the second reason. And as in Figure 5, vanilla-LSTM does show smaller gap, but its performance is worst among all the models plotted. So from the aspect of stability, vanilla-LSTM proves to be promising, but when considering the model accuracy, it is far from expectation. Interestingly, both single-regression and IRV present comparatively stable performance. The reason that regression models can be helpful here is because our virtual screening is rank-based, and it may better discovery the information decoded in the continuous %inhibition. For IRV model, it finds the nearest neighbors based on a similarity score. The input to the IRV model will then be the similarity scores of the k-nearest-neighbors. This can help explain the stability since IRV does not directly learn from the fingerprints.

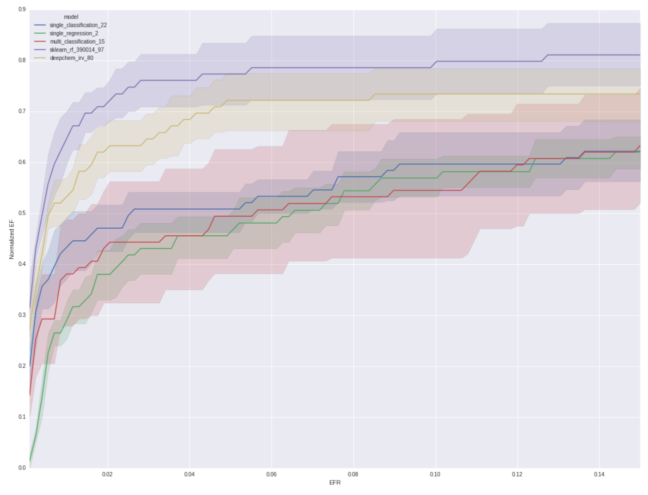


Figure 6. Normalized enrichment factor (NEF) on task PriA-SSB AS . NEF is defined in Equation 5. We plot the corresponding continuous values $R \in [0, 0.15]$.

The normalized enrichment factor curve in Figure 6 is another interesting metric. We can see random forest consistently outperforms other methods by a considerable margin even as we increase the percentiles.

4.1.1 *Can AUC[ROC] be misleading?*

We argue that AUC[ROC] is not a representative evaluation metric, sometimes even misleading. We can observe that under some circumstances, other evaluation methods like [8] AUC[PR], [31]AUC[BEDROC], and AUC[NEF] can reveal quite different attributes of models. We draw attention to several empirical evidence we have found.

On the previous hyperparameter sweeping stage, the decision will be hard to make if we are focusing on AUC[ROC]. See appendix G. However, once we also consider AUC[PR], AUC[BEDROC], and EF, the difference between models is much more obvious.

In the cross validation results, most models reach promising performance on AUC[ROC], and the top models have approximately same evaluation values. But when focusing on AUC[PR], AUC[BEDROC], EF, and AUC[NEF], the difference among such top models can be huge. Brief understanding is that, compared to PR and EF, ROC focuses more on true negative(TN) case. While in the virtual screening tasks, due to the highly skewed data, most predictions will be close to 0 or negative. So focusing too much on TN will give a biased performance evaluation.

4.1.2 Multi-task Effect

In general, multi-task network can achieve better performance than single-task network due to its ability to transfer knowledge among all tasks. And such benefit is called multi-task effect. [27] explains the multi-task effect: similar tasks can benefit from training on shared active compounds.

In our experiment, we combine the Keck task with 128 PCBA tasks during multi-task training, and as shown in 5, it does not present considerable improvements comparing with single-task. If we assume multi-task effect comes from learning a better and generalized latent space, then one possible explanation is that multi-task model can extract better feature representation only when tasks are highly correlated, and all three newly generated Keck targets have no similarity with 128 PCBA tasks. To verify this, there is no sharing compounds between PriA-SSB AS and PCBA, so from the data's aspect, these two sets of tasks share no similarity; and based on the domain knowledge, PriA-SSB AS is not related to PCBA.

4.1.3 RF outperforms other models

As to what makes random forest best, we have two assumptions: (1) The 3 tasks possess very extreme label imbalance issue, and in methods like deep neural network, validation set are required either for early stopping or layer prediction. (2) Random forest is better than other methods due to its model design and dataset properties. Because [6] all previous models are using hard threshold, they are able to conclude that neural network is better.

4.2 Tukey-Based Universal Confidence Intervals

We conduct Tukey's range test to compare our models. One way to summarize the results of this test is to use universal confidence intervals as introduced in [13]. It plots the mean and confidence intervals based on Tukey's Q critical value of each model. If the universal confidence intervals of two models overlap, then there is significance. Otherwise, no significance is detected. Figure 7 showcases the plots for three metrics-label pairs. The total 54 plots for each metric-label pair can found at **URL** and the appendix L.

4.2.1 Model Comparison Results

We rank models for each metric-label pair based on Tukey's test results. This can be thought of assigning "win" points to each model. We summarize these results in a table at **URL** and the appendix I. Random Forest and CBF models consistently place in the top 5 ranks for each metric-label pair. Table 2 shows the percentage of model overlap in the top 5 of each metric-label pair.

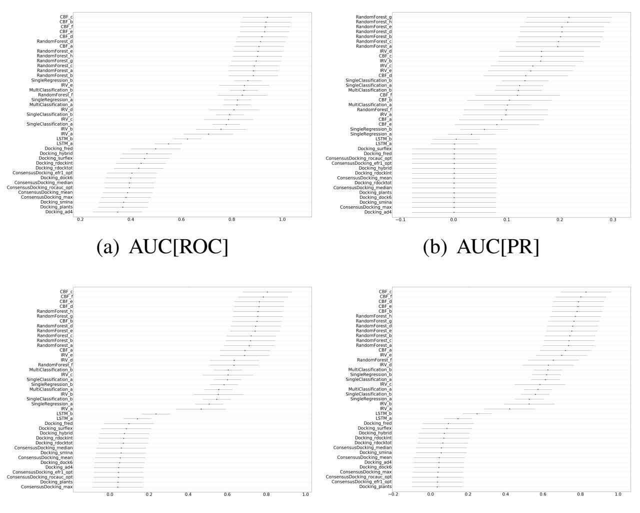

(a) AUC[ROC] (b) AUC[PR]
(c) AUC[BEDROC] (d) AUC[NEF]

Figure 7. Tukey-Based Universal Confidence Interval for PriA-SSB AS .

With this, we have a ranking among models for each metric-label pair that we can revise when we perform prospective screening.

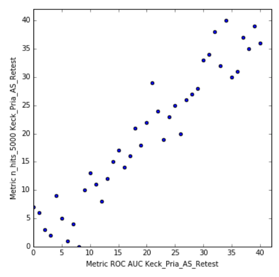
Table 2. Percentage of model appearance in the top 5 over all metric-label pairs in cross-validation stage.

Model	Overlap Percentage	Model	Overlap Percentage
CBF_c	54.3%	**SingleRegression_b**	23.9%
RandomForest_h	53.2%	**RandomForest_d**	21.7%
RandomForest_g	46.7%	**RandomForest_b**	19.5%
CBF_b	41.3%	**SingleRegression_a**	18.4%
CBF_d	40%	**CBF_a**	17.3%
CBF_f	36.9%	**RandomForest_c**	15.2%
RandomForest_e	28.2%	**RandomForest_a**	14.1%

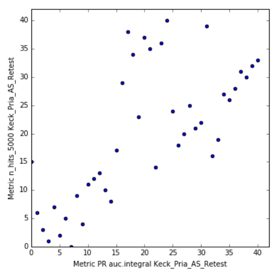
4.3 Metric Discussion

In section 4.1.1 we describe one case where we observe AUC[ROC] cannot fully explain the model performance. And to apply for the ultimate goal, we introduce n_{hits} as ground truth. In order to determine which metric relates to n_{hits} in a more rigorous and comprehensive way, we compare the model ranking induced by each metric with the model ranking induced by n_{hits}. Figure 8 are sample plots showcasing the correlation between n_{hits} and different metrics.

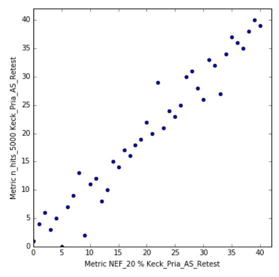
If all points lie on the $x = y$ curve, then the metric coincides perfectly with n_{hits}. As what we observe from Figure 8, in general, the metric reveals that enrichment factor proves to be a more promising evaluation metric. This can be caused by the fact that in the real scenario, the number of positive hits in top ranked compounds is what chemistry people care about, and by definition, enrichment factor and normalized enrichment factor are the two closest evaluation metrics. Of course such conclusions may change as we come up with more insightful feature representation, machine learning algorithm, and domain requirement. Just for the current setting, we may



(a) n_{hits} and AUC[ROC]



(b) n_{hits} and AUC[PR]



(c) n_{hits} and NEF

Figure 8. Metrics comparison on task PriA-SSB AS . Fig. 8(a) to Fig. 8(c) correspond to the model ranking comparison between $_{hits} = 5000$ and AUC[ROC], AUC[PR], and NEF respectively.

as well conclude that with fingerprint as input feature, and all the models we have trained, combine with the application which focuses more on top hits, enrichment factor shows to be a more illustrative and promising evaluation metric.

Note that we perform these comparisons with n_{hits} at various n_{tests}. To get an effective score for ranking metrics, we use Spearman's rank correlation coefficient based on the rankings induced by the metric of concern vs. n_{hits} at a specific n_{tests}. We can then rank the metrics based on their correlation coefficient. The entire metric ranking results can be found at **URL** and the appendix J. The main takeaway is that ranking of metrics varies by n_{tests} performed; some metrics overtake one another as we increase or decrease n_{tests}. However, NEF_R seems to be consistently placing in the top ranks in such a manner that R coincides with n_{tests}. This is evident when we just focus on a single label and see the top ranking metrics for $n_{tests} \in [100, 250, 500, 1000, 2500, 5000, 10000]$. This suggests that if we know apriori how many n_{tests} we'd like to perform, then NEF_R at a suitable R is an appropriate metric. With this, we have a ranking among metrics at various n_{tests} that we can revise when we perform prospective screening.

4.4 Closer Look at STNN and RF

We will investigate the performance different between single-task neural network (STNN) and random forest (RF) starting from comparing the predicted results on target PriA-SSB AS .

As what can be observed from Table 3, the predicted val-

Table 3. This is the comparison between single-task neural network (STNN) and random forest (RF) on the testset. We picked first 4 folds for training and validation, last fold as test set, and all the positive molecules listed here are positive towards target PriA-SSB AS . The column pred represents the predicted values, and rank is the ranking of corresponding predicted values out of all the 14486 molecules in testset.

molecule	STNN		RF	
id	pred	rank	pred	rank
14425	0.000002	14219	0.000000	8666
14427	0.160746	8	0.251750	1
14428	0.000026	1261	0.001500	1440
14429	0.708671	5	0.147250	7
14431	0.000028	1088	0.004000	442
14436	0.920472	4	0.169500	5
14437	0.069630	10	0.122250	14
14438	0.000193	116	0.089500	20
14439	0.040926	13	0.082250	25

ues on some molecules, STNN is more confident and closer to 1, although the ranking are almost the same with RF, like molecule 14429. But there are also some molecules predicted to be closer to negative on both models, but getting comparatively higher ranking in random forest. We collect these molecules set as $S = [14425, 14428, 14431, 14438]$. One natural assumption is that maybe the neural network is trained so well, that for each molecule in S, the most similar molecules in the training set are actually negative. This assumption is reasonable considering that using 1024 fingerprints as feature may lose some information and cannot fully reveal the molecule structure. And Table 4 verifies our assumption. We have more comprehensive tables in **URL** .

Table 4. We pick up top 10 molecules in the training set that are closest to molecule 14425 (active compound). The similarity is Tanimoto similarity. The RF can return only the important features, which is the set of important bits in 1024 fingerprint, and the second similarity is based only on such important bits. In the column true label, 0 and 1 represent inactive and active respectively. Here the closest 10 molecules towards 14425 are all negative.

molecule id	similarity	similarity (important)	true label
33815	0.910714	1.0	0
33834	0.786885	0.928571	0
33792	0.774194	0.928571	0
19401	0.762712	0.666667	0
925	0.704918	0.733333	0
33820	0.693548	0.588235	0
33825	0.683333	0.615385	0
33833	0.681818	0.714286	0
19423	0.681818	0.8	0
4912	0.677419	0.666667	0

The third column in Table 4 highlights feature importance in tree-like drug discovery models. Combining Table. 3, we may conclude:

1. For the molecules that have very similar molecules (like where Tanitomo similarity is over 0.7) in the training set,

both models can perform quite accurate predictions on some molecules. And NN is more confident, almost all above 0.8. The predicted values generated by tree-like models are almost less than 0.6.

2. For the molecules that have less similar molecules in the training set, both models predict badly. RF model has higher predicted values and ranks, but STNN will more confidently predict them to be negative. And this explains what we observe from Figure 5, 6, and 7, that the evaluation methods on tree-like models are better.

So we may as well guess that, the predicted results from single-task NN have more correlation to its counterparts in the training set, while tree-like models somehow are less likely to attach to the training set. That's why for some true-positive predictions, NN is more confident. If we can have more generalized and perfect featurization strategy, the performance on STNN can get improved.

Also, we may ask this following questions: If the predicted values are so small, less than 0.01, why do we take this part to evaluate the performance. For example, for molecule 14425, NN predicts it with probability 0.000002 to be true, and RF predicts it to be 0.00063. In the real setting, it will not matter, because both are not trustworthy. And coming back to our choice on the evaluation metrics, the EF with low EF ratio can overcome this problem, while AUC will be over-optimistic on the RF.

5. PROSPECTIVE SCREENING RESULTS

A molecule facility would like to in-vitro test compound activity bindings to a protein. The molecule facility has funds to purchase n_{tests} compounds, and expect to maximize their return on investment by having most numbers of compounds with an active bind. Define n_{hits} as the number of active hits from the n_{tests} tested compounds. This scenario illustrates the importance of maximizing the n_{hits} for tests, as active hits can help guide the search for more actives. As discussed in 3.2, different evaluation metrics have been used to assess virtual screening methods. For our model comparisons in this stage, we will use n_{hits} as a means to assess each evaluation metric in terms of how they translate to real-world value. We want to identify metrics that are positively and linearly correlated with n_{hits}. The following summarizes the prospective screening stage setting:

1. We will continue running the 41 models we have in cross validation stage. But to clarify some statements, we also add two extra methods, the ensemble neural network and chemical similarity baseline.
2. Each model is trained on the initial 75k molecule dataset with the later 25k molecule dataset as the test set.
3. On this test set, we record: AUC[ROC], AUC[PR], AUC[NEF], and EF/NEF at the following ratios $[0.001, 0.0015, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2]$.
4. Use fourth library as test set, and record n_{hits} on the top n_{tests} ranking probability predictions from the models. We compute n_{hits} for $n_{\text{tests}} \in [100, 250, 500, 1000, 2500, 5000, 10000]$.

This stage has only one recording for each metric, thus, we can only compare models directly. We propose the following to assess the goals:

1. For each metric, identify the top M_{best} performers on the held-out dataset. Analyze the amount of overlap between the top M_{best} performers in CV stage and PS stage for the same metric.
2. For n_{hits} for $n_{\text{tests}} \in [100, 250, 500, 1000, 2500, 5000, 10000]$, identify the top M_{best} on the held-out dataset. For each metric in CV stage results, and for each n_{hits} in PS stage results, analyze the amount of overlap between the top M_{best} performers in CV stage and PS stage.

The first step helps answer the first goal by analyzing which models retained their ranking as top performers. The second step gives us a measure on how well each metric was able to reflect real-life value, i.e. if the top performing models based on a metric are still the top performing based on n_{hits}.

5.1 Focused Study: Hits in Top 250

In our last library of compounds, on target PriA-SSB AS , expected purchase limit allows us to screen 250 out of the overall 25k compounds. We strictly follow the model rank given by NEF in figure 7(d), and the corresponding hit numbers and similarity by clustering are presented in Table 5.

Table 5. Number of active hits in top 250 predictions on 8 selected models. The 8 selected models are best among each algorithm, and the number of hits on all models can be found in Appendix R. The last two columns correspond to two clustering methods, and we use this to show how diverse molecules our models can find. Both algorithms have 40 clusters in all. SIM was identified by Wards clustering based on Tanimoto from ECFP4 fingerprints. MSC identifies a maximum common substructure which will be further used to group compounds.

model name	number of hits	SIM	MCS
Baseline	33	16	18
Docking_fred	2	2	2
IRV_e	30	16	19
CBF_c	48	24	25
RandomForest_h	41	21	25
STNN-C_a	25	12	15
STNN-R_b	35	20	20

As we can see, almost all the supervised machine learning methods can beat baseline. As what is shown in Table 5, ensemble neural network shows best performance, following calibrated boosting-tree, random forest and STNN

Regression model. Ensemble neural network is taking the higher rank for each compound using SingleClassification_b and SingleRegression_b. Docking's performance is not very promising according to the number of hits. But when we plot the overlap of all the top 250 predictions (Figure 9) the docking program is able to find 2 unique active compounds, as shown in Figure 9.

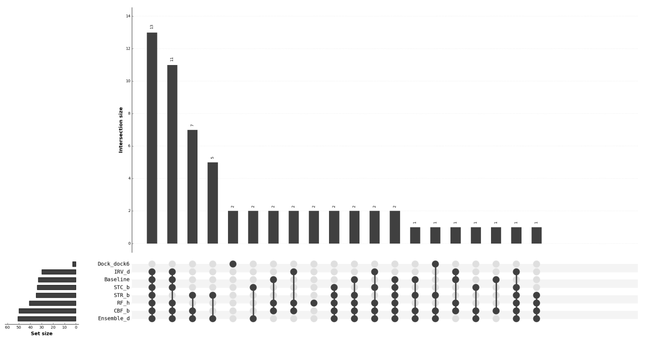


Figure 9. An UpSet plot showing the overlap between the 8 selected models. The plot generalizes a Venn diagram. Dock_dock6 is one docking program, STC and STR are single-task classification and regression respectively. RF is random forest, and CBF is calibrated boosting-forest. The complete intersections for all models are in Appendix S.

We can observe that supervised learning methods can reproduce similarity to some extent, but each of them can find unique actives as well. For example, most machine learning methods, including the baseline model, can agree on 27 active compounds, where single-task regression only share 14 among them. Random forest, on the other hand, is able to identify 2 unique actives, which is missing by other methods. And docking, even though the total hits is lower, but can still find two unique molecules. These convince us to keep using supervised methods. Then a natural and better solution comes to our mind is to use ensemble method, to combine the benefits from each models. In section 5.1.1, we will describe why ensemble method is better, and how to reach it.

5.1.1 *How Ensemble Model Helps*

In Figure 9, we can clearly see how different each model capture actives. Recall that both the ensemble neural network and CBF are ensembling classification and regression models. And we want to make a statement that blindly ensemble models may be helpful, but it can help most when we can ensemble both the regression and classification models. Here are two observations help substantiate this point. (1) Recall that random forest itself is an ensemble method, which averages over the predictions made by various trees. But all the trees are classifiers, no regression process is included. (2) The **Simple Ensemble**'s performance also verifies this point because it only includes one classification and regression model. The simple ensemble method identifies 49 active compounds in its top 250 predictions, covering 25 SIM clusters and 26 MCS clusters. This performance is comparable, and slightly better than, the best model in Table 5. We conclude that while ensembling classification and regression models, it can fully utilizes its data by considering both the binary and continuous labels. This can be explained from the machine learning theory's point, when we double the data for training, the PAC-bound becomes smaller.

We can draw another conclusion that, simply adding more base models won't help improve performance. The best calibrated boosting-tree has 10 base models, and Simple Ensemble can reach better performance with one classification and regression model. How to reach the best ensemble models is still not clear, but this is an interesting future work. Now we would recommend to follow the principle that simple is best, try different base models, select the best one classification and regression model and then do ensemble on them. We did not try other ensemble methods to keep our hypothesis space consistent and fixed, but it worth exploring different base models with different ensemble strategies.

5.2 Model Comparison

For the following sections, since the ensemble neural network consists of rankings, we will not calculate its evaluation metrics, and we will replace calibrated boosting-forest with ensemble method . We rank models for each metric based on the raw scores on the single test in Table 6.

Table 6. We show 3 main evaluation metrics on 6 selected models. Dock_6 is Docking_dock6, CBF is calibrated boosting-forest, RF is random forest, STC and STR are single-task classification and regression respectively. More comprehensive results can be found in **URL** and the appendix M.

model	AUC[ROC]	AUC[PR]	NEF@1%
Dock_6	0.566	0.153	0.036
IRV_d	0.753	0.526	0.357
CBF_b	0.917	0.743	0.595
RF_g	0.838	0.614	0.488
STC_b	0.750	0.476	0.405
STR_b	0.883	0.637	0.417

For this stage, ensemble method models place in the top 5 ranks for each metric. In the CV stage, we had good representation for random forest in top 5, but in this stage it lags behind. Table 7 shows the percentage of model overlap in the top 5 of each metric.

Table 7. Percentage of model appearance in the top 5 over all metrics in PS stage.

Model	**Overlap Percentage**
CBF_a	89.4%
CBF_f	89.4%
CBF_c	89.4%
CBF_e	84.2%
CBF_b	68.4%
CBF_d	52.6%
SingleClassification_b	10.5%
RandomForest_h	10.5%
RandomForest_g	5.2%

Recall from the CV stage we have a ranking among models for each metric based on the wins from the Tukey's test (see appendix M). We compare the rankings achieved for each metric in the CV stage vs. rankings in the PS stage. Again, one way to perform this comparison is to Spearman's rank correlation coefficient. The full results for this comparison can be found at P. To summarize, 14 out of 19 metrics achieve over 0.8 correlation. AUC[PR] achieves ≈ 0.5 correlation which indicates that it is not a consistent metric for judging future performance. This is because by definition AUC[PR] heavily depends on the class distribution (unline AUC[ROC]) which changed on the held-out test set.

5.3 Metric Ranking Comparison

As in the CV stage, we compare the rankings induces by each metric vs. n_{hits} at various n_{tests}. We then compare these rankings using Spearman's rank correlation coefficient. Results can be found at **URL** and the appendix N. Again, NEF_R seems to be consistently placing in the top ranks in such a manner that R coincides with n_{tests}.

We compare this with the metric rankings we achieved from the CV stage ((see appendix J). This can be achieved in two ways:

1. Take the difference between the correlation results of CV vs. PS. Order from smallest-to-largest difference to obtain ranking over metrics. See appendix Q. This allows us to rank metrics on how well they maintain similar correlation with n_{hits} in CV vs. PS.
2. Compare the rankings induced by n_{hits} at various n_{tests} in CV vs. PS using Spearman's rank correlation. See appendix Q. This tells us how well *all* metrics maintain their ranks in CV vs. PS.

We combine both approaches above. From the first approach, we see again that NEF_R places in the top 5 ranks. From the second approach, we see that the correlation scores are all above 0.49 except for n_{hits} and $n_{tests} = 10000$ which achieves 0.22 correlation. This is mainly due to metrics like AUC[PR] changing its ranking drastically from CV vs. PS. These two results promote the use of NEF_R with R set so that it coincides with the n_{tests} to be performed. In the CV stage, we also concluded that NEF_R should be used.

6. RELATED WORK

Virtual screening can be divided into structure-based and ligand-based methods. Structure-based methods use the 3d structure of drug drug target, and fits each of million of small molecules or compounds. . Instead of the 3d simulation, ligand-based methods do not use target information and focus molecules: extract features from each molecule that explain the activity towards the target.

Molecule Discrimination Deep learning methods showed overwhelming results starting from [5, 23] Merck Molecular Activity Challange, 2012. [7, 16, 20, 22, 27, 33] have been investigating multi-task deep neural network and proved its outstanding performance compared with classical machine learning methods. Label imbalance is one of the most common challenges, and [2] tries to solve it with one-shot learning. An important note is that all of this work involved only binary labels of target activity.

Fingerprints encode each molecule structure into fixed-length bit-vectors where each bit represents one substructure. Besides this, SMILES can be used to represent the sequential atom orders, and therefore fed in as the model input. [14] makes model comparisons based on input features, including Recurrent Neural Network Language Model and Convolutional Neural Networks with SMILES, and shows that CNN is best when evaluated on the log-loss. [10] proposes a different structure called Atomic Convolutional Networks (ACNN), very similar to CNN, but it contains the 3D information. [18] solves the chemical-chemical interaction with CNNs by concatenating two SMILES strings.

Understanding how deep neural networks work is not a trivial task. [17] decodes such black-box prediction using influence function, while [30] explains it from a more bio-informatics aspect.

Another alternative solution is what has been a recently emerging method called [11] generative adversarial nets (GAN) model. GAN models contain a discriminator and a generator. People feed the generator some "fake" or noisy data points, so as to trick the discriminator; and the goal of generator is to automatically generate molecules that a well trained discriminator cannot distinguish. Finally when people feed in random data, it can magically produce new molecules that can be highly possible active against targets. [15] implements such similar framework, and it can produce molecules that are highly possible to be true against target.

7. SUMMARY AND FUTURE WORK

To sum up, we argue that ensemble methods, especially ensembling on both classification and regression model, can lead to better performance. This finding can be even more valuable since in the real setting, it is hard to apply an explicit threshold to separate actives and inactives, like what previous work has been doing; while the actual condition with secondary screening is shown in Table 1. Furthermore, the **Simple Ensemble** shows most promising results here, but more complicated ensemble strategies are worth trial in the future.

Recall that these models are pretrained, and most of them can generalize quickly when applied to a large set of new compounds. Supervised learning models can rank millions of compounds in minutes, and our next step will test our newly proposed ensemble methods on larger compounds and verify predictions by high-throughput screening experiments.

Besides, in Figure 7 and Figure 5 tree-like models can outperform neural networks. But what is well know that one of the biggest advantages of deep network is its ability to learn latent representation from input features. Here is fixed with

1024 fingerprints, which is far away from the complete and raw representation of one compound. In Merck challenge, it is well acknowledged that multi-task learning can outperform random forest. One highly possible reason for that is the Merck compounds' feature contains around 15k bits of values, both the number of bits and type of information are more comprehensive than the 1024 fingerprints. The cost is adding more complexity to model. Many other representations can be considered besides descriptors: 3d fingerprints, graphs, etc.

Below are some other machine learning-model related work that can be interesting in the future.

- [38] proposes adaptive methods like Adam, may not perform as well as SGD, from the respective of accuracy.
- Apply more advanced embedding functions for recurrent network model.
- Explore the temporal relation and cross-validation decisions.
- There is a huge generalization gap among the performance on training set, validation set, and test set, and this might be explained in the latest neural network generalization work.
- Fingerprints have the potential benefit of representing structures in molecule, but have the drawbacks not being able to revert it back to molecules, thus its interpretation is harder. More interpretable and direct featurization can be applied here.

Acknowledgements

The authors acknowledge GPU hardware from NVIDIA and funding from the Center for Predictive Computational Phenotyping NIH/NIAID U54AI117924, the University of Wisconsin-Madison Office of the Vice Chancellor for Research and Graduate Education, and the Morgridge Institute for Research.

References

[1] Rdkit: Open-source cheminformatics.

[2] Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. Low data drug discovery with one-shot learning. *ACS Central Science*, 3(4):283–293, 2017.

[3] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[4] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[5] George Dahl. Deep learning how i did it: Merck 1st place interview. *Online article available from http://blog. kaggle. com/2012/11/01/deep-learning-how-i-did-it-merck-1st-place-interview*, 2012.

[6] George E. Dahl, Navdeep Jaitly, and Ruslan Salakhutdinov. Multi-task neural networks for QSAR predictions. *CoRR*, abs/1406.1231, 2014.

[7] George E Dahl, Navdeep Jaitly, and Ruslan Salakhutdinov. Multi-task neural networks for QSAR predictions. *arXiv preprint arXiv:1406.1231*, 2014.

[8] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[9] Spencer S Ericksen, Haozhen Wu, Huikun Zhang, Lauren A Michael, Michael A Newton, F Michael Hoffmann, and Scott A Wildman. Machine learning consensus scoring improves performance across targets in structure-based virtual screening. *Journal of Chemical Information and Modeling*, 57(7):1579–1590, 2017.

[10] Joseph Gomes, Bharath Ramsundar, Evan N Feinberg, and Vijay S Pande. Atomic convolutional networks for predicting protein-ligand binding affinity. *arXiv preprint arXiv:1703.10603*, 2017.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[12] Kelly A Hoadley, Yutong Xue, Chen Ling, Minoru Takata, Weidong Wang, and James L Keck. Defining the molecular interface that connects the fanconi anemia protein fancm to the bloom syndrome dissolvasome. *Proceedings of the National Academy of Sciences*, 109(12):4437–4442, 2012.

[13] Y. Hochberg and A. C. Tamhane. Multiple comparison procedures. 1987.

[14] Stanisław Jastrzębski, Damian Leśniak, and Wojciech Marian Czarnecki. Learning to smile (s). *arXiv preprint arXiv:1602.06289*, 2016.

[15] Artur Kadurin, Alexander Aliper, Andrey Kazennov, Polina Mamoshina, Quentin Vanhaelen, Kuzma Khrabrov, and Alex Zhavoronkov. The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology. *Oncotarget*, 8(7):10883, 2017.

[16] Steven Kearnes, Brian Goldman, and Vijay Pande. Modeling industrial admet data with multitask networks. *arXiv preprint arXiv:1606.08793*, 2016.

[17] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017.

[18] Sunyoung Kwon and Sungroh Yoon. Deepcci: End-to-end deep learning for chemical-chemical interaction prediction. *arXiv preprint arXiv:1704.08432*, 2017.

[19] Alessandro Lusci, David Fooshee, Michael Browning, Joshua Swamidass, and Pierre Baldi. Accurate and efficient target prediction using a potency-sensitive influence-relevance voter. *Journal of cheminformatics*, 7(1):63, 2015.

[20] Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 55(2):263–274, 2015.

[21] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. DeepTox: Toxicity Prediction using Deep Learning. *Frontiers in Environmental Science*, 3(80), 2016.

[22] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. Deeptox: toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80, 2016.

[23] Merck. Merck molecular activity challenge. *https://www.kaggle.com/c/MerckActivity*, 2012.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] Janaina Cruz Pereira, Ernesto Raul Caffarena, and Cicero Nogueira dos Santos. Boosting docking-based virtual screening with deep learning. *Journal of Chemical Information and Modeling*, 2016.

[27] Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande. Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072*, 2015.

[28] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[29] Skipper Seabold and Josef Perktold. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

[30] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*, 2016.

[31] S Joshua Swamidass, Chloé-Agathe Azencott, Kenny Daily, and Pierre Baldi. A croc stronger than roc: measuring, visualizing and optimizing early retrieval. *Bioinformatics*, 26(10):1348–1356, 2010.

[32] S Joshua Swamidass, Chloé-Agathe Azencott, Ting-Wan Lin, Hugo Gramajo, Shiou-Chuan Tsai, and Pierre Baldi. Influence relevance voting: an accurate and interpretable virtual high throughput screening method. *Journal of chemical information and modeling*, 49(4):756–766, 2009.

[33] Thomas Unterthiner, Andreas Mayr, Günter Klambauer, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, and Sepp Hochreiter. Deep learning as an opportunity in virtual screening. *Advances in neural information processing systems*, 27, 2014.

[34] Andrew F Voter, Michael P Killoran, Gene E Ananiev, Scott A Wildman, F Michael Hoffmann, and James L Keck. A high-throughput screening strategy to identify inhibitors of ssb protein–protein interactions in an academic screening facility. *SLAS DISCOVERY: Advancing Life Sciences R&D*, page 2472555217712001, 2017.

[35] Jason Wang and Luis Perez. The effectiveness of data augmentation in image classification using deep learning.

[36] Yanli Wang, Stephen H. Bryant, Tiejun Cheng, Jiyao Wang, Asta Gindulyte, Benjamin A. Shoemaker, Paul A. Thiessen, Sigian He, and Jian Zhang. Pubchem bioassay: 2017 update. *Nucleic Acids Research*, 45(D):D955–D963, 2017.

[37] David Weininger, Arthur Weininger, and Joseph L Weininger. Smiles. 2. algorithm for generation of unique smiles notation. *Journal of Chemical Information and Computer Sciences*, 29(2):97–101, 1989.

[38] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *arXiv preprint arXiv:1705.08292*, 2017.

[39] Haozhen Wu. Calibrated boosting-forest. *arXiv preprint arXiv:1710.05476*, 2017.

APPENDIX

A. DATA PROPROCESSING

A.1 Complex Matrix Composition

Each target dataset consists of compounds as rows, and for each compound it details the bio-chemical features such as structure, SMILE info, interaction score results, and most importantly the activity outcome (binary or continuous). For our purposes of featurization, we are only interested in the SMILES and activity outcome of the compounds. The first step was to extract these two properties (SMILES and activity outcome) for each target and construct the data matrix for training. We used RDKit [1], a ChemInformatics python library, for navigating and extracting from these datasets and for generating the fingerprints.

The second step was to merge the target matrices together into one consolidated matrix. We simply used an outer-join operation with the SMILES as the key. That is, given two matrices A and B with two columns: SMILES and target-activity, an outer-join operation will merge rows of A and B that have the same SMILES value into a new matrix M. If there is a row in A with SMILES value s and no corresponding row in B with SMILES value s, then the merge would yield a row in M with an empty target-activity for B. In the resulting matrix, each row is a compound and the columns are: SMILES, 1024-bit fingerprints, and a column for the activity outcome of each target (a total of 5 columns for SSB-PriA and RMI-FANCM and 128 for PCBA). As result, we have two data matrices: SSB-PriA and RMI-FANCM and PCBA, on which we can train either single task or multitask learning methods. Note that merging all the targets introduces many empty cells for the activity outcome columns. For the distribution of active, inactive, and missing for each target, refer to Appendix E. We can observe severe data imbalance; the ratio of positive to negative is very small, ranging from 0.00009 to 0.48324 .

A.2 Fold Splitting

The whole data set was split into 5 fixed folds for cross validation. Label imbalance and the limited number of known active molecules is one of biggest challenges in virtual screening and must be accounted for during modeling. Stratified split is a way to divide data into sub-folds while keeping the same ratio for each homogeneous label.

For single-target task, stratified split can be implemented as combining folds after sampling each class of labels. But this procedure will become more complicated when goes to the multi-task condition. All molecules will appear only in a subset of 131 targets, and merging all molecules into one big matrix, each row represents one molecule, and each column represents one target. For each column (target), molecules can be missing, inactive or active. Similarly for each row (molecule), this molecule can be missing, inactive, or active against the 131 targets. We divide this big matrix into 5 folds, while keeping the same data distribution at the same time.

```
Algorithm 1: Multi-task Data Splitting
   Input: Initial pre-split molecule-target matrix $M$, number of desired folds $k$
   Output: $k$ folds F[1], F[2], ..., F[k] containing stratified splits of $M$
1  shuffle rows of $M$ randomly
2  create k folds F[1], F[2], ..., F[k] which contain the row indexes only
3  indexList ← argsort columns of $M$ from smallest active counts to largest
4  for $i$ in indexList do
5      currColumn ← $M[:, i]$
6      split active indexes of currColumn into the $k$ folds
7      split inactive indexes of currColumn into the $k$ folds
8      split missing indexes of currColumn into the $k$ folds
9      uniquify each fold to remove duplicate row indexes
10     greedily remove overlapping indexes from each fold (fold-by-fold manner)
11 uniquify each fold to remove duplicate row indexes
12 return F[1], F[2], ..., F[k]
```

A.3 Label Imbalance

SSB-PriA and RMI-FANCM has three binary targets PriA-SSB AS , PriA-SSB FP , RMI-FANCM with only 79, 24, and 230 actives, respectively. To alleviate this class imbalance, one solution is to use a weighted schema. For single-target models, we apply Eq (6).

$$weight_{(negative)} = 1, \quad weight_{(positive)} = \frac{n}{p} \tag{6}$$

where $weight_{positive}$ and $weight_{negative}$ are weight scalars for positive (active) and negative (inactive), respectively, and p and n represent the number of positive and negative samples on this target.

Similarly, we apply the weighted schema to multi-task models, defined as Eq (7).

$$weight_{(negative,i)} = s_i, \quad weight_{(positive,i)} = s_i \cdot \frac{n_i}{p_i} \tag{7}$$

where $weight_{(positive,i)}$ and $weight_{(negative,i)}$ are weight scalar for positive and negative for i^{th} target, and p_i and n_i represent the number of positive and negative samples for i^{th} target. t_i defined as Eq (8)

$$t_i = \begin{cases} \frac{\sum_i p_i}{p_i}, & i^{th} \text{ target is in PCBA} \\ \alpha \cdot \frac{\sum_i p_i}{p_i}, & i^{th} \text{ target is in SSB-PriA and RMI-FANCM} \end{cases} \tag{8}$$

In the multi-task setting, we hope to give different weights to each target, and focusing more on the SSB-PriA and RMI-FANCM targets and the PCBA targets that have fewer positive samples. We highlight SSB-PriA and RMI-FANCM by setting $\alpha = 100$, and alleviate the data skewness among targets by the term $\frac{\sum_i p_i}{p_i}$.

B. PCBA QUERY

Download from the PubChem BioAssay database here using the following query: TotalSidCount from 10000, ActiveSidCount from 30, Chemical, Confirmatory, Dose-Response, Target: Single, NCGC. These limits correspond to the search query: (10000[TotalSidCount] : 1000000000[TotalSidCount]) AND (30[ActiveSidCount] : 1000000000[ActiveSidCount]) AND “small molecule"[filt] AND “doseresponse"[filt] AND 1[TargetCount] AND “NCGC"[SourceName].

Cited from [27].

C. SOFTWARE FRAMEWORK

D. SSB-PRIA AS DATA DISTRIBUTION

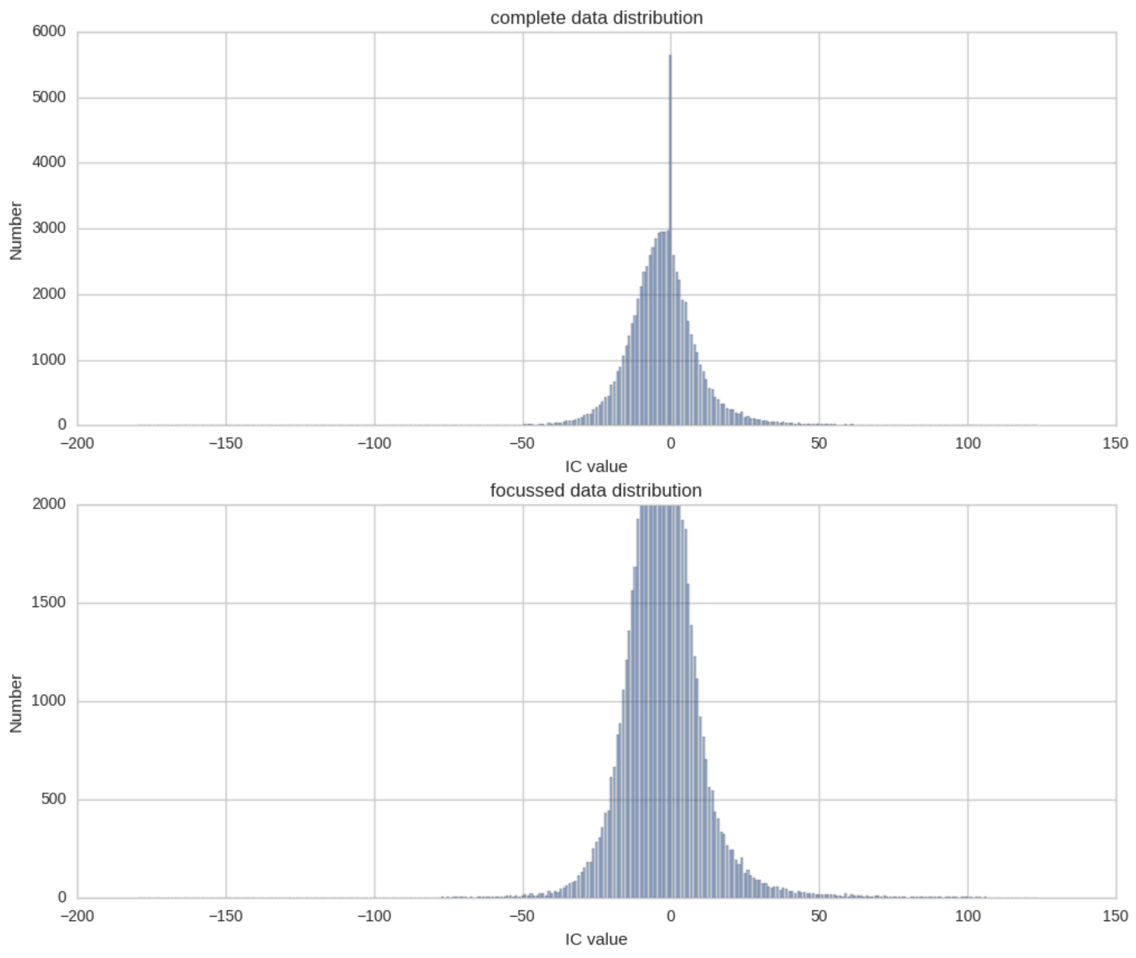


Figure 10. PriA-SSB AS % inhibition complete data distribution.

E. PCBA AND SSB-PRIA AS DATA DISTRIBUTION

task name	positive molecule number	negative molecule number	missing molecule number	$ratio = \frac{\text{pos number}}{\text{neg number}}$
pcba-aid1030	15932	145369	335063	10.95970%
pcba-aid1379	561	196368	314806	0.28569%
pcba-aid1452	178	149367	362573	0.11917%
pcba-aid1454	513	115335	395935	0.44479%
pcba-aid1457	720	202110	308746	0.35624%
pcba-aid1458	5778	188852	311888	3.05954%
pcba-aid1460	5650	217010	283986	2.60357%
pcba-aid1461	2305	206016	301670	1.11885%
pcba-aid1468	1038	251148	259072	0.41330%
pcba-aid1469	170	272533	239423	0.06238%
pcba-aid1471	293	218258	293452	0.13424%
pcba-aid1479	793	269530	241180	0.29422%
pcba-aid1631	892	259030	251482	0.34436%

task name	positive molecule number	negative molecule number	missing molecule number	$ratio = \frac{\text{pos number}}{\text{neg number}}$
pcba-aid1634	154	261988	250000	0.05878%
pcba-aid1688	2375	201910	305636	1.17627%
pcba-aid1721	1087	289651	220471	0.37528%
pcba-aid2100	1157	291855	218127	0.39643%
pcba-aid2101	288	309907	201813	0.09293%
pcba-aid2147	3473	188764	316586	1.83986%
pcba-aid2242	715	183374	327492	0.38991%
pcba-aid2326	1065	259688	250478	0.41011%
pcba-aid2451	2005	271718	236568	0.73790%
pcba-aid2517	1138	332123	177897	0.34264%
pcba-aid2528	652	340938	170054	0.19124%
pcba-aid2546	10556	267886	223298	3.94048%
pcba-aid2549	1211	230450	279424	0.52549%
pcba-aid2551	16671	253653	225301	6.57236%
pcba-aid2662	110	285240	226836	0.03856%
pcba-aid2675	99	248789	263309	0.03979%
pcba-aid2676	1081	357341	152793	0.30251%
pcba-aid411	1563	69057	440113	2.26335%
pcba-aid463254	41	329171	183043	0.01246%
pcba-aid485281	253	314347	197443	0.08048%
pcba-aid485290	938	335859	174561	0.27928%
pcba-aid485294	148	309649	202351	0.04780%
pcba-aid485297	9128	301294	192746	3.02960%
pcba-aid485313	7569	304194	192964	2.48821%
pcba-aid485314	4493	312590	190720	1.43735%
pcba-aid485341	1729	325703	183135	0.53085%
pcba-aid485349	618	319466	191594	0.19345%
pcba-aid485353	603	322454	188636	0.18700%
pcba-aid485360	1485	216997	292329	0.68434%
pcba-aid485364	10698	331470	159430	3.22744%
pcba-aid485367	557	325598	185584	0.17107%
pcba-aid492947	80	329301	182835	0.02429%
pcba-aid493208	342	41294	470318	0.82821%
pcba-aid504327	766	370995	139769	0.20647%
pcba-aid504332	30264	263754	188014	11.47433%
pcba-aid504333	15673	310114	170836	5.05395%
pcba-aid504339	16859	338757	139821	4.97672%
pcba-aid504444	7388	282993	214527	2.61067%
pcba-aid504466	4169	306751	197207	1.35908%
pcba-aid504467	7648	235607	261393	3.24608%
pcba-aid504706	201	302548	209346	0.06644%
pcba-aid504842	101	324570	187524	0.03112%
pcba-aid504845	100	372270	139826	0.02686%
pcba-aid504847	3509	376531	128747	0.93193%
pcba-aid504891	34	361224	151004	0.00941%
pcba-aid540276	4393	192748	310762	2.27914%
pcba-aid540317	2129	367917	140121	0.57866%
pcba-aid588342	25036	301746	160478	8.29704%
pcba-aid588453	3904	365862	138626	1.06707%
pcba-aid588456	51	384356	127838	0.01327%
pcba-aid588579	1980	384213	124123	0.51534%
pcba-aid588590	3931	352947	151487	1.11376%
pcba-aid588591	4700	367981	134915	1.27724%
pcba-aid588795	1307	376247	133435	0.34738%

task name	positive molecule number	negative molecule number	missing molecule number	$ratio = \frac{\text{pos number}}{\text{neg number}}$
pcba-aid588855	4897	347556	154946	1.40898%
pcba-aid602179	364	384856	126712	0.09458%
pcba-aid602233	165	379055	132911	0.04353%
pcba-aid602310	310	393819	117857	0.07872%
pcba-aid602313	762	372273	138499	0.20469%
pcba-aid602332	69	408322	103836	0.01690%
pcba-aid624170	838	397756	112864	0.21068%
pcba-aid624171	1239	394674	115144	0.31393%
pcba-aid624173	487	399643	111679	0.12186%
pcba-aid624202	3968	362543	141817	1.09449%
pcba-aid624246	101	364511	147583	0.02771%
pcba-aid624287	423	302226	209224	0.13996%
pcba-aid624288	1356	323051	186533	0.41975%
pcba-aid624291	222	331803	180049	0.06691%
pcba-aid624296	9840	282428	210188	3.48407%
pcba-aid624297	6213	301951	197919	2.05762%
pcba-aid624417	6389	319289	180229	2.00101%
pcba-aid651635	3784	343160	161568	1.10269%
pcba-aid651644	748	353982	156818	0.21131%
pcba-aid651768	1677	355992	152950	0.47108%
pcba-aid651965	6346	318038	181566	1.99536%
pcba-aid652025	238	364167	147653	0.06535%
pcba-aid652104	7126	368557	129487	1.93349%
pcba-aid652105	4072	318365	185787	1.27904%
pcba-aid652106	497	362334	148968	0.13717%
pcba-aid686970	5948	331060	169340	1.79665%
pcba-aid686978	62375	236628	150918	26.35994%
pcba-aid686979	48532	257279	157953	18.86357%
pcba-aid720504	10170	340357	151599	2.98804%
pcba-aid720532	976	11815	498529	8.26069%
pcba-aid720542	733	356204	154626	0.20578%
pcba-aid720551	1265	341660	168106	0.37025%
pcba-aid720553	3259	336029	169749	0.96986%
pcba-aid720579	1908	280991	227489	0.67903%
pcba-aid720580	1508	304454	204826	0.49531%
pcba-aid720707	268	363257	148503	0.07378%
pcba-aid720708	661	356743	154231	0.18529%
pcba-aid720709	516	352850	158414	0.14624%
pcba-aid720711	290	363245	148471	0.07984%
pcba-aid743255	901	366915	143579	0.24556%
pcba-aid743266	306	398728	112956	0.07674%
pcba-aid875	34	73821	438407	0.04606%
pcba-aid881	590	103808	407308	0.56836%
pcba-aid883	1217	6647	503215	18.30901%
pcba-aid884	3396	6983	498521	48.63239%
pcba-aid885	160	12683	499293	1.26153%
pcba-aid887	1017	68423	441839	1.48634%
pcba-aid891	1564	6012	503156	26.01464%
pcba-aid899	1773	6141	502609	28.87152%
pcba-aid902	1865	117072	391494	1.59304%
pcba-aid903	338	52451	459169	0.64441%
pcba-aid904	528	50430	460810	1.04700%
pcba-aid912	453	56178	455212	0.80637%
pcba-aid914	221	7524	504330	2.93727%

task name	positive molecule number	negative molecule number	missing molecule number	$ratio = \frac{\text{pos number}}{\text{neg number}}$
pcba-aid915	421	7524	503930	5.59543%
pcba-aid924	1144	118813	391195	0.96286%
pcba-aid925	39	64140	448078	0.06080%
pcba-aid926	345	56230	455376	0.61355%
pcba-aid927	60	58565	453611	0.10245%
pcba-aid938	1781	60720	448014	2.93314%
pcba-aid995	699	65056	445842	1.07446%
PriA-SSB AS	79	72344	439794	0.10920%
PriA-SSB FP	24	72399	439849	0.03315%
RMI-FANCM	230	49566	462270	0.46403%

F. DATASETS ON MODELS

model	PriA-SSB AS Binary	PriA-SSB FP Binary	PriA-SSB AS Continuous	RMI-FANCM Binary	RMI-FANCM Continuous	PCBA
Random Forest	✓					
CBF	✓		✓			
IRV	✓					
STNN-C	✓					
STNN-R			✓			
MTNN	✓					✓
Vanilla LSTM	✓					
Ensemble NN	✓		✓			

Table 9. On target PriA-SSB AS , different datasets that each model is using.

model	PriA-SSB AS Binary	PriA-SSB FP Binary	PriA-SSB AS Continuous	RMI-FANCM Binary	RMI-FANCM Continuous	PCBA
Random Forest		✓				
CBF		✓	✓			
IRV		✓				
STNN-C		✓				
STNN-R			✓			
MTNN		✓				✓
Vanilla LSTM		✓				
Ensemble NN		✓	✓			

Table 10. On target PriA-SSB FP , different datasets that each model is using.

model	PriA-SSB AS Binary	PriA-SSB FP Binary	PriA-SSB AS Continuous	RMI-FANCM Binary	RMI-FANCM Continuous	PCBA
Random Forest				✓		
CBF				✓	✓	
IRV				✓		
STNN-C				✓		
STNN-R					✓	
MTNN				✓		✓
Vanilla LSTM				✓		
Ensemble NN				✓	✓	

Table 11. On target RMI-FANCM , different datasets that each model is using.

G. HYPERPARAMETER GRID SEARCH

Hyperparameters	Candidate Values
learning rate	0.00003, 0.0001, 0.003
weighted schema	no_weight, weighted_sample
epoch patience	[epoch_size: 200, patience: 50], [epoch_size: 1000, patience: 200]
activations	[ReLU, Sigmoid, Sigmoid], [ReLU, ReLU, Sigmoid]

Table 12. Hyperparameter Sweeping for Deep Classification Neural Network, including Single-task and Multi-task.

Hyperparameters	Candidate Values
learning rate	0.00003, 0.0001, 0.003
weighted schema	no_weight
epoch patience	[epoch_size: 200, patience: 50], [epoch_size: 1000, patience: 200]
activations	[ReLU, Sigmoid, Sigmoid], [ReLU, ReLU, Sigmoid]

Table 13. Hyperparameter Sweeping for Deep Regression Neural Network.

Hyperparameters	Candidate Values
learning rate	0.00003, 0.0001, 0.003
epoch patience	[epoch_size: 200, patience: 50]
embedding size	30, 50, 100
hidden size	[50], [100], [100, 10], [100, 50], [50, 10]
drop out	0.2, 0.5

Table 14. Hyperparameter Sweeping for Vanilla Recurrent Neural Network.

Hyperparameters	Candidate Values
n_estimators	4000, 8000, 16000
max_features	None, sqrt, log2
min_samples_leaf	1, 10, 100, 1000
class_weight	None, balanced_subsample, balanced

Table 15. Hyperparameter Sweeping for Random Forest.

Hyperparameters	Candidate Values
number of neighbors	5, 10, 20, 40, 80
epoch patience	[epoch_size: 1000, patience: 20]
batch size	8192
learning rate	0.01
penalty	0.05

Table 16. Hyperparameter Sweeping for IRV.

In deep neural networks, 80% of the 4 folds were used for training and 20% for validation. For random forest, the first 3 folds were used for training and the 4th fold for validation to prune 108 models down to 7 models. They are both ad-hoc with the goal of pruning model search space. IRV has one parameter for the number of neighbors which we assumed would not improve drastically past 100 neighbors, and so, we saw no need to sweep.

H. CROSS-VALIDATION: TUKEY UNIVERSAL CONFIDENCE INTERVALS

I. CROSS-VALIDATION: MODEL COMPARISON RESULTS

J. CROSS-VALIDATION: METRIC COMPARISON RESULTS

K. CROSS-VALIDATION: SCATTER PLOT RESULTS

L. PROSPECTIVE SCREENING: TEST SCORE PLOTS

M. PROSPECTIVE SCREENING: MODEL COMPARISON RESULTS

N. PROSPECTIVE SCREENING: METRIC COMPARISON RESULTS

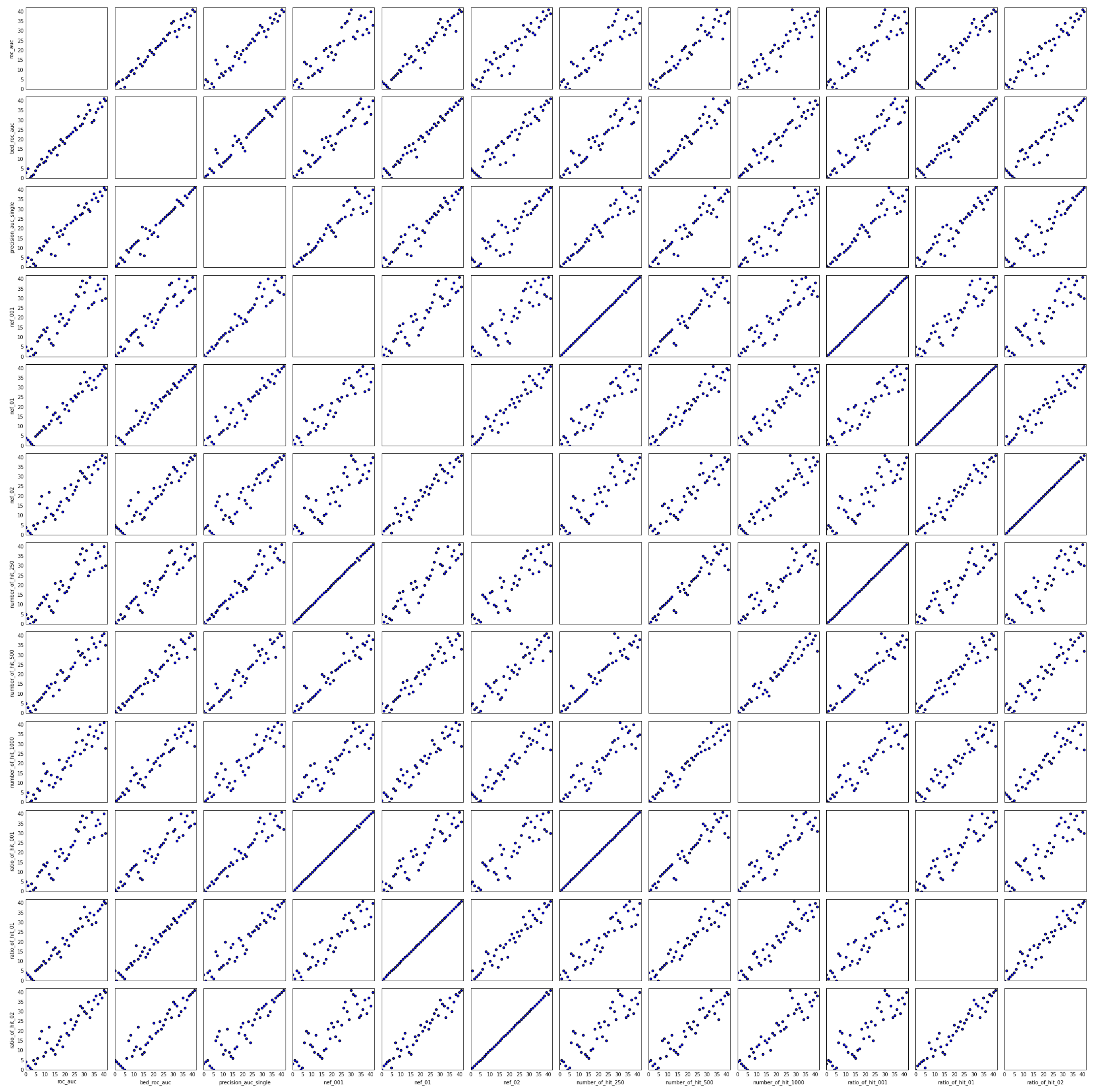


Figure 11. Complete Prospective Screening Metric Comparison

O. PROSPECTIVE SCREENING: SCATTER PLOT RESULTS

P. CV VS PS: MODEL ORDERING COMPARISON

Q. CV VS PS: METRIC ORDERING COMPARISON

R. NUMBER OF HITS IN TOP 250 PREDICTIONS

Table 17. Number of active hits in top 250 predictions. The last two columns correspond to two clustering methods, and we use this to show how diverse molecules our models can find. Both algorithms have 40 clusters in all. SIM was identified by Wards clustering based on Tanimoto from ECFP4 fingerprints. MSC identifies a maximum common substructure which will be further used to group compounds.

model name	number of hits	SIM Cluster	MCS Cluster
Baseline	33	16	18
CBF_a	47	24	24
CBF_b	50	24	25
CBF_c	48	24	25
CBF_d	45	23	23
CBF_e	47	23	23
CBF_f	48	24	24
ConsensusDocking_efr1_opt	0	0	0
ConsensusDocking_max	2	2	2
ConsensusDocking_mean	1	1	1
ConsensusDocking_median	2	2	1
ConsensusDocking_rocauc_opt	0	0	0
Docking_ad4	6	5	5
Docking_dock6	3	3	2
Docking_fred	2	2	2
Docking_hybrid	2	2	2
Docking_plants	1	1	1
Docking_rdockint	2	2	2
Docking_rdocktot	2	2	2
Docking_smina	0	0	0
Docking_surflex	1	1	1
IRV_a	17	9	12
IRV_b	25	13	16
IRV_c	30	16	19
IRV_d	30	16	19
IRV_e	30	16	19
LSTM_a	1	1	1
LSTM_b	1	1	1
MultiClassification_a	26	13	16
MultiClassification_b	31	16	20
RandomForest_a	39	19	22
RandomForest_b	39	19	22
RandomForest_c	39	19	22
RandomForest_d	38	20	22
RandomForest_e	39	20	23
RandomForest_f	27	16	19
RandomForest_g	40	21	24
RandomForest_h	41	21	25
SingleClassification_a	25	12	15
SingleClassification_b	34	18	20
SingleRegression_a	35	16	20
SingleRegression_b	35	20	20
Simple Ensemble	49	25	26

S. VENN DIAGRAM IN TOP 250 PREDICTIONS

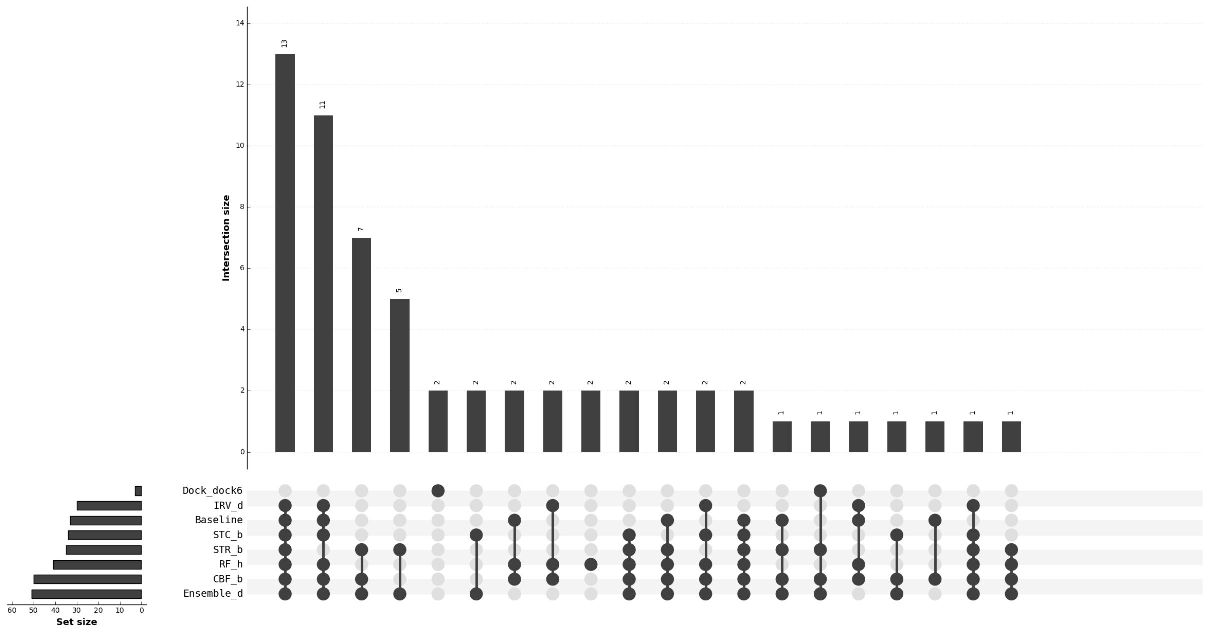


Figure 12. Venn Diagram on 6 selected models.

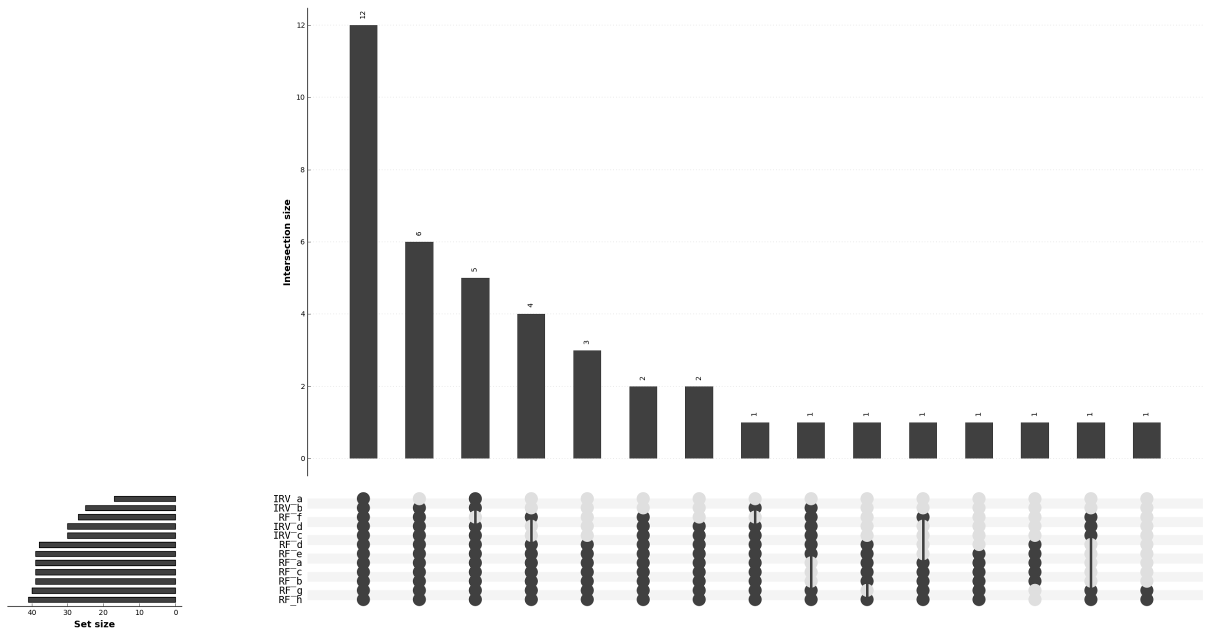


Figure 13. Venn Diagram on IRV and RF.

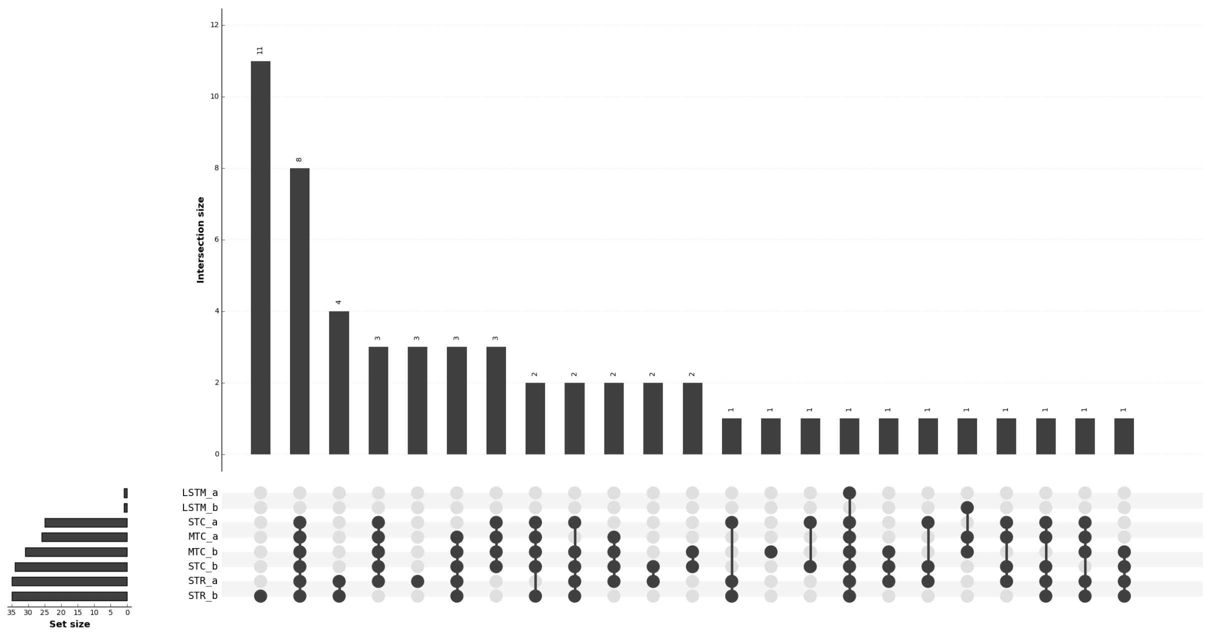


Figure 14. Venn Diagram on NN.

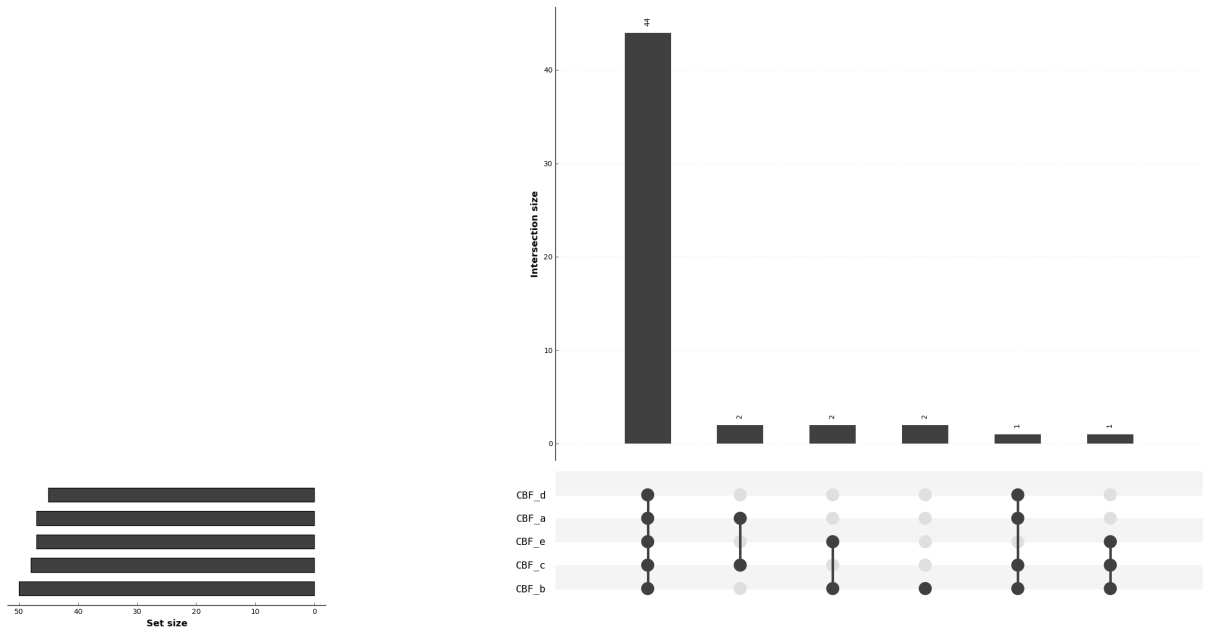


Figure 15. Venn Diagram on CBF.

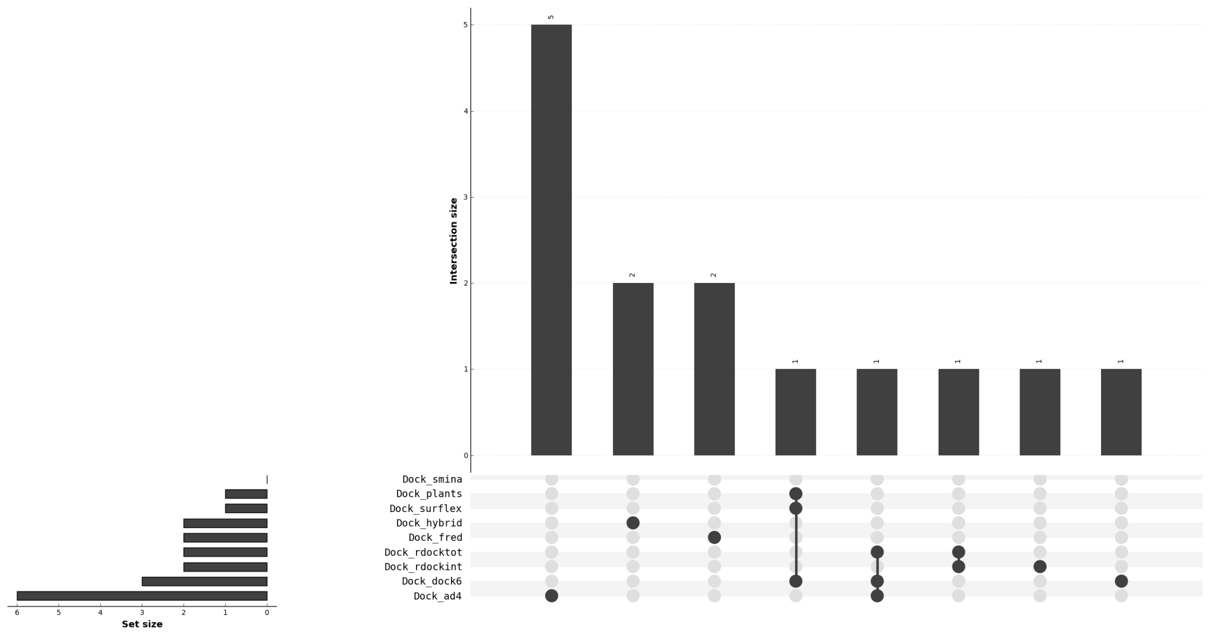


Figure 16. Venn Diagram on Docking.

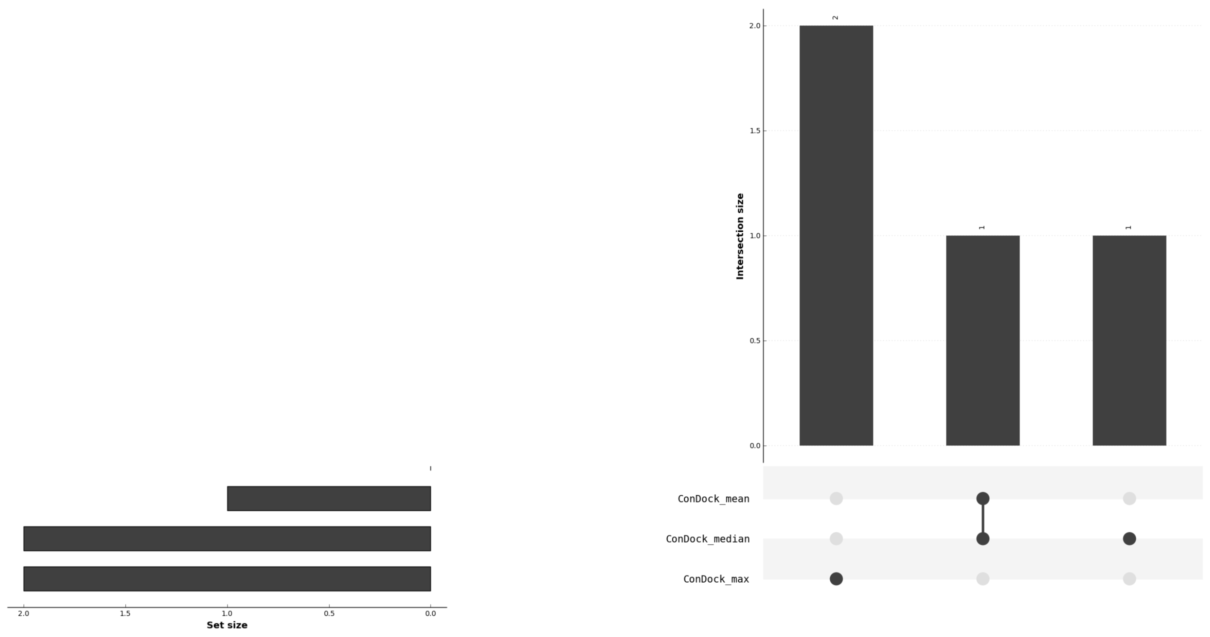


Figure 17. Venn Diagram on Consensus Docking.

T. SOFTWARE

Here we provide an overview of the libraries used. The particular versions and in-depth details can be found at the project Github page:

T.1 Why Not DeepChem

One contribution is to provide a user-friendly framework. All codes are published at `https://github.com/chao1224/virtual-screening`

1. Our implementation is more generalized, can easily switch between Theano and TensorFlow. And comparing to TensorFlow, Keras is easier to start with.
2. At the time, DeepChem was being updated, particularly their deep-learning framework. Furthermore, there was no support for early stopping using a validation set and consecutive iterative runs of the same model. Due to these limitations we decided not to use DeepChem except for IRV since an implementation existed. We modified DeepChem to work with early stopping for IRV.
3. DeepChem is not using the distributed version, which means they didn't benefit any speed up from the TensorFlow framework, so switching to it cannot bring us instant benefit. Because using CHTC pools, we should be able to get as computation performance as DeepChem. Besides, we also have another distributed version in PyTorch.
4. DeepChem still keeps updating and miss some functionality, and customized framework can help us better develop our ideas. Besides, DeepChem group has a lot of computation resources, that's why they don't need validation and early stopping, and we offer a framework that can be fit for both sufficient and constrained computation conditions, and this will be a good optional choice.

T.2 Model Libraries

1. Neural network models use Keras [4] a Python library that works on top of Theano or Tensorflow.
2. Random Forest models use Scikit-Learn [25].
3. IRV uses a modified version of DeepChem [2].
4. Calibrated-Decision-Trees use [39].
5. Docking use .

T.3 Metric Libraries

1. AUC[ROC] calculation uses Scikit-Learn [25].
2. AUC[PR] calculation uses PRROC R package [8].
3. Tukey-HSD uses the `statsmodel` Python library [29].