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.

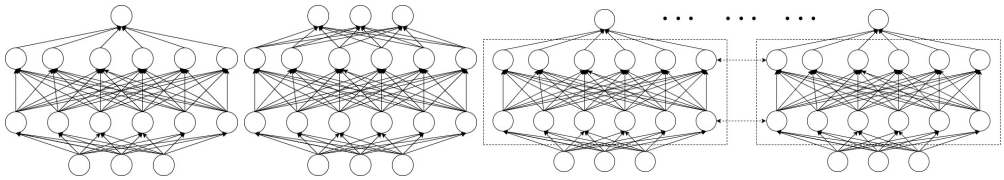
3 PROBLEM BACKGROUND

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

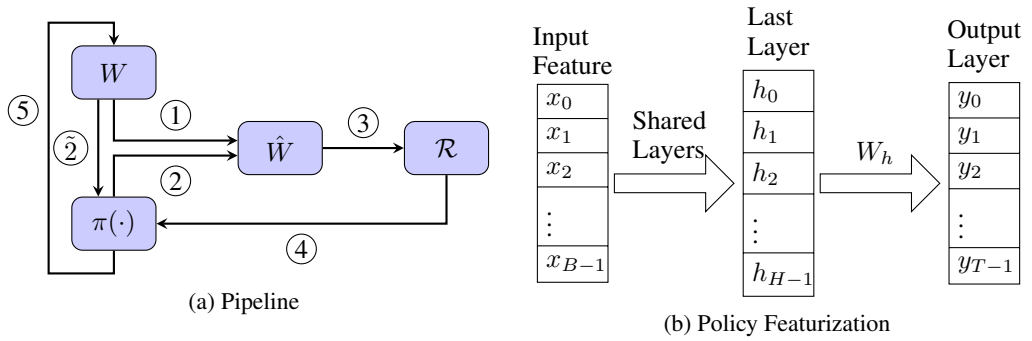
Figure 1: (a) contains one unit in the output layer representing a single task. (b) shares all parameters among tasks except at the output layer, where multiple predictions are made. (c) is a soft-sharing model. Each task has an identical layer structure with parameter similarity constraints.

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 DMTRL policy maps the multi-task network state to an action ($\pi(\cdot) : \mathcal{S} \rightarrow \mathcal{A}$, 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. (1) update $\hat{W}$ by taking current state W_t and (2) the sampled policy $\pi(\cdot)$. (3) get the rewards and (4) use them as feedback to select the best policy $\pi(\cdot)$. (5) use this optimal policy to update the gradient. For the policy-based method parameterized by Θ, the extra step $(\tilde{2})$ maps from state W_t to an action. In step (4) $\pi(\cdot)$ will be updated once receiving rewards. Following in figure 2b, we highlight the featurization part (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

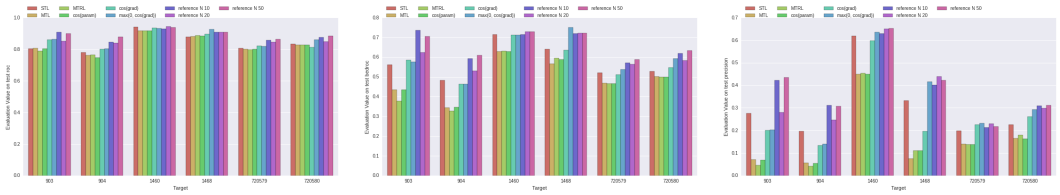
We selected 3 pairs of tasks that can be highly related based on domain knowledge. For evaluation metrics, AUC[ROC], AUC[BEDROC] and AUC[PR] are most widely used.

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 π_θ
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