Active Learning through Reinforcement Learning

John Chen
Department of Computer Science
University of Toronto
Toronto, ON
johnn.chen@mail.utoronto.ca

Jieyu Lin
Department of Electrical and Computer Engineering
University of Toronto
Toronto, ON
jieyu.lin@mail.utoronto.ca

Abstract

A thorny and universal open problem that continues to plague deep learning is the scarcity of labelled data. Despite copious amounts of unlabelled data (e.g. images on the Internet), labelling is a procedure that is costly, labour-intensive, time-consuming, and not scalable. Pool-based active learning thus deals with picking *good* data points from the unlabelled dataset to label. In *batch-mode active learning*, this discrete choice of *k* unlabelled data points is called a *query*. Recent work such as Variational Adversarial Active Learning (VAAL) has improved performance by learning good representations for the active learning problem. However, we found that both VAAL and simple heuristic query functions such as *uncertainty sampling* are still unequipped to deal with the interaction effects between utility of data points within a batch query. In this report, we investigate the correlated query problem in batch mode active learning, and furthermore propose *learning* the heuristic query selection function using reinforcement learning. Specifically, we explore adding class conditional labels to VAAL, and propose two novel architectures (deep Q network and policy gradient network) for determining a good querying policy. We show that addressing these fundamental issues can improve performance compared to existing work and baselines.

Keywords: active learning, correlated batch query problem, reinforcement learning, uncertainty sampling

1 Introduction

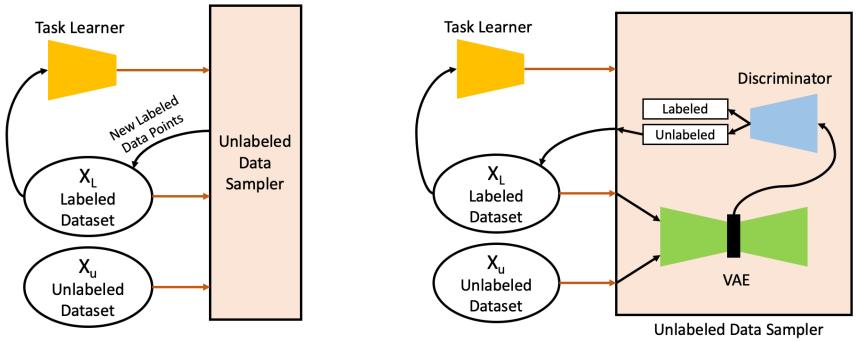
Data scarcity is a significant obstacle to supervised deep learning algorithms in many domains, such as healthcare and language processing. In these contexts, labelling can represent a costly action as it is a time-consuming task performed by highly qualified individuals, such as doctors or linguists. Given a restricted labeling budget, it is often infeasible to query labels of the entire dataset using an oracle.

Active learning, also called *query* learning, is broadly the field of machine learning that deals with improving label efficiency by allowing the learning algorithm (neural network, support vector machine, etc.) to *choose* the datapoints it would like to learn from [13]. Given a task learner T, training on a labelled training dataset X_L, the goal of *pool-based* active learning is to best transfer datapoints from the unlabelled dataset (pool) X_U to X_L to best improve the test performance metric P of T on some testing-dataset. For instance, if the task is image classification, then our task learner T could be a convolutional neural network, and our pool X_U could be ImageNet (masking the labels). In this case, we would like to find a minimal number of examples to add to X_L in order to achieve say 60% F1 score on the held out testing dataset X_T.

The goal of an active learning algorithm then, is to find an optimal querying strategy to best improve T. Unfortunately, finding an optimal querying strategy is difficult with few theoretical guarantees due to myriad factors of variation including randomness on the samples in the training dataset, sampling randomness in the testing dataset, sensitivity of decision boundary to outliers, choice of inductive bias, and choice of test performance metric [1].

Recent work in active learning, such as VAAL [14], is able to achieve state-of-the-art performance through improving data representation using variational auto encoder and adversarial learning techniques. Although these methods are able to achieve improved performance via good data representation, we found that they are sub-optimal in batch query and scarce data environment. On the other hand, the query problem in active learning can also be thought of as a combinatorial optimization problem or discrete search problem where you try to find the optimal strategy for labeling data that maximizes your performance metric. Since we are looking at discrete optimization, reinforcement learning is a popular technique that would *learn to search*.

Preprint. Under review.



(a) Basic Pool-Based Active Learning Setup (b) VAAL

Figure 1: Main architectures of active learning

In this project, we explore combining the latest representation techniques with deep reinforcement learning for improving active learning. More specifically, we improved the data representation method introduced by VAAL, and propose using DQN[11] and policy gradient for improving the search optimization problem in active learning.

Our novel contributions are thus:

- Combining reinforcement learning with representation learning in the active learning domain
- Novel policy gradient method for active learning in a low-dimension representation space
- Proposing DQN architecture for reinforcement learning in active learning using a learnt representation

2 Background & Motivation

2.1 Pool-based Active Learning

In pool-based active learning, the goal is to find a query algorithm that selects the most informative unlabeled data points (the pool) to be labeled. Figure 1a shows a general pool-based active learning setup. In general, the sampler takes the labeled and unlabeled data points and the task learner network as the inputs, and select the unlabeled data to query.

2.2 Batch Mode Active Learning

The query algorithm in active learning can query one or more data points at a time. With batch query, performance can degrade:

$$f(\sum_{i=1}^{K} x_i) \leq \sum_{i=1}^{K} f(x_i)$$

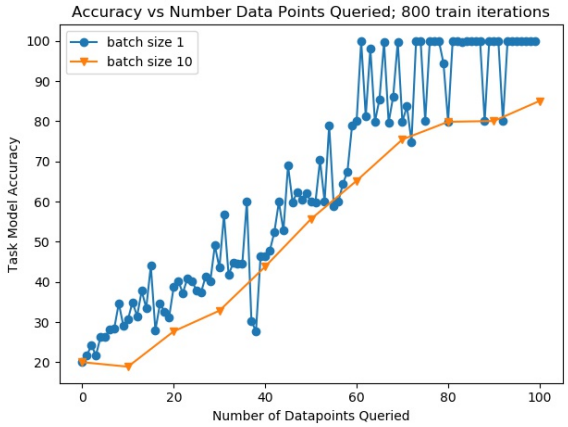
where f is the improvement in P after making a query consisting of x_i and $K > 1$ is the batch size.

2.3 Variational Adversarial Active Learning (VAAL)

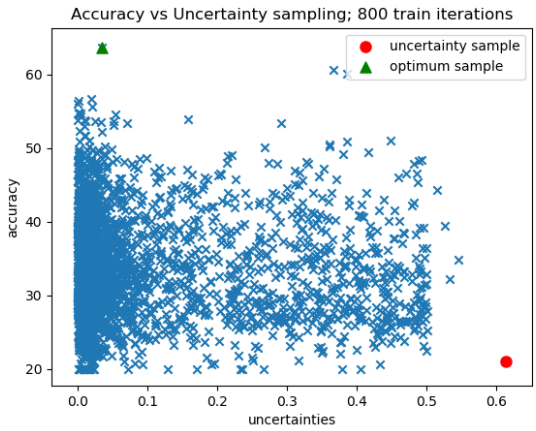
Variation Adversarial Active Learning [14] is a pool based, batch mode active learning method that combines VAE and a discriminator in an adversarial way to help label selection. Figure 1b shows the VAAL method in the format of the general pool-based active learning framework shown in Figure 1a. The VAE transforms the dataset to a latent space representation while making it hard for the discriminator to distinguish, and the discriminator tries to distinguish labeled and unlabeled datapoints. To achieve this goal, the VAE loss is formulated as:

$$\mathcal{L}_{\text{VAE}} = \lambda_1 \mathcal{L}_{\text{VAE}}^{trd} + \lambda_2 \{-\mathbb{E}[\log(D(q_\phi(z_L|x_L)))] - \mathbb{E}[\log(D(q_\phi(z_U|x_U)))]\} \quad (1)$$

where $\mathcal{L}_{\text{VAE}}^{trd}$ is the traditional VAE loss, q_ϕ is the probability distribution parameterized by the encoder of the VAE (inference network), subscript L and U refer to labeled and unlabeled, and λ_1 and λ_2 are hyperparameters. The discriminator loss is categorical cross entropy. The unlabelled points that are most strongly correctly predicted by the discriminator as unlabelled are selected as the query. In our work, we start by analyzing and improving this work, use it as inspiration to develop novel methods.



(a) Illustration of the correlated batch query problem. For the same number of datapoints queried, letting the active learning algorithm query fewer points at a time generally leads to better performance, as the model has a chance to train on an updated training dataset, and adapt its request for new datapoints. Note the relative instability of the uncertainty sampling algorithm.



(b) The data point selected by uncertainty sampling is not necessarily optimal. Here the optimal data point is defined as that when added to X_L, best improves the test performance of the trained model on this augmented dataset. It is found via exhaustive search over the data points in the unlabelled dataset. Note that in general there is a weak correlation between P improvement and uncertainty of a datapoint.

Figure 2: Two issues with uncertainty sampling as a query selection algorithm. Both figures are generated from a simple 2D toy dataset with a task learner learning a 5-way classification from scratch. Each marker represents the final test performance P of training a model with the additional data point added to its labelled training dataset for 800 train iterations. Full experiment details can be found in the appendix.

2.4 Motivation and Limitations of Existing Methods

Uncertainty sampling is a popular and intuitive pool-based heuristic method for querying that has achieved good empirical performance [17, 13, 1]. At a high level, uncertainty sampling is a heuristic query algorithm that selects the *most informative*[1] data points to label. However, uncertainty sampling is not optimal. As 2b shows, the measure of uncertainty in a task model is not necessarily strongly correlated with the performance of the task model. Furthermore, in a batch query setting, the simplistic choice of taking the top k uncertain data points causes degraded performance. The top k suffers from the *correlated batch* problem, where the expected performance improvement of a batch is much less than the sum of the individual. To this end, our project aims to find more effective query algorithm that can improve the performance of active learning.

3 Methods

The methods of this paper can be organized into two parts: the first part extends the VAAL method by adding label information into the adversarial learning to improve data representation; the second part uses reinforcement learning instead of the adversarial learning method for improving query selection.

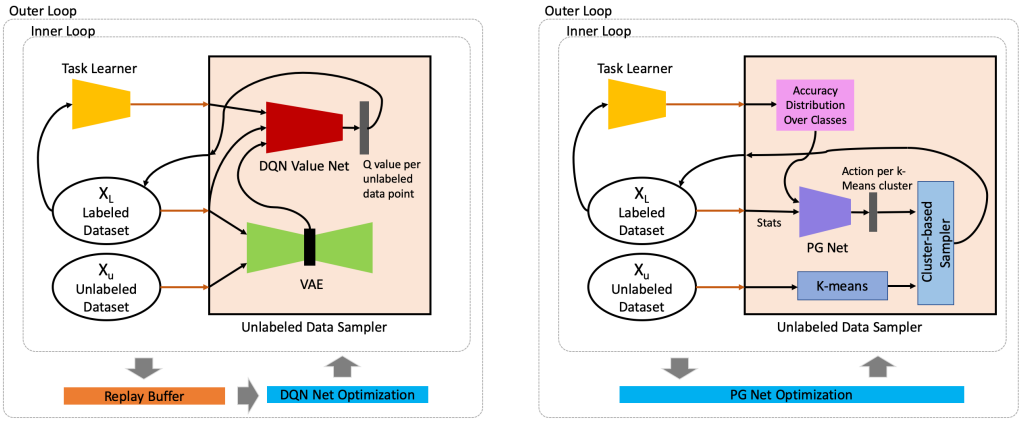
3.1 VAAL with label information (l-VAAL)

As we mentioned in the introduction, one of the key issues in active learning is batch correlation. VAAL has a discriminator that discriminates between labeled and unlabeled data in the latent space. However, the best candidate data points selected by the discriminator can be from the same class, and the discriminator is not able to distinguish the difference between classes. Thus it is important to take data label into consideration during the VAAL adversarial learning. To incorporate the label information into the adversarial network, we created a method call l-VAAL, and we reformulate the VAE loss as follows:

$$\mathcal{L}_{\text{VAE}} = \lambda_1 \mathcal{L}_{\text{VAE}}^{trd} + \lambda_2 \{-\mathbb{E}[\log(\sum_{k \in C} 1 - D(q_\phi(z_{L_k}|x_{L_c})))] - \mathbb{E}[\log(D(q_\phi(z_U|x_U)))]\} \tag{2}$$

where C is the set of all classes. Discriminator loss is the same as a multi-class classifier. Here we have the number of classes plus 1 class for unlabeled data. The intuition for the adversarial loss VAE is to increase the difficulty for the discriminator to distinguish between the the class labels and the unlabeled. The most confident unlabeled data points are selected to be labeled.

[1] See Appendix for full background



(a) DQN-based Active Learning Architecture (b) Policy gradient based Active Learning Architecture

Figure 3: Main reinforcement learning based architectures of active learning

3.2 Reinforcement learning active learning

While adding class label information to VAAL improve the data representation, the query batch correlation issue still remain. Thus we explored using reinforcement learning to learn the query function. The intuition is that if we are able to train a RL agent that is capable of effectively selecting a single sample to be labeled augmented with knowledge of the history (embeded in the agent's RNN core), we can potentially improve the batch correlation by rolling out the RL agent in multiple steps. Now, the RL agent becomes the query model, and is responsible for selecting data points to be labelled. We explored using two RL methods for unlabeled data sampling: Deep Q Network (DQN) and Policy Gradient (PG). Next, we will discuss each method individually:

3.2.1 DQN active learning

Figure 3a shows the high level architecture of DQN active learning. This method replaces the discriminator in l-VAAL with a DQN network. The RL agent consists of a RNN based value network that predicts the Q value of each action. The input state of the RL agent is comprised of: 1. unlabeled data points; 2.task learner prediction of unlabeled data points, and 3. VAE latent embedding of the unlabeled data points. The action space of the RL agent is the number o f unlabeled data points. In each step of a RL episode, we train the task learner until convergence, then the observation is constructed, and the RL agent selects the best unlabeled data point to be labeled (i.e. best action). The reward is the accuracy difference between steps (i.e. $Acc_{t+1} - Acc_t$).

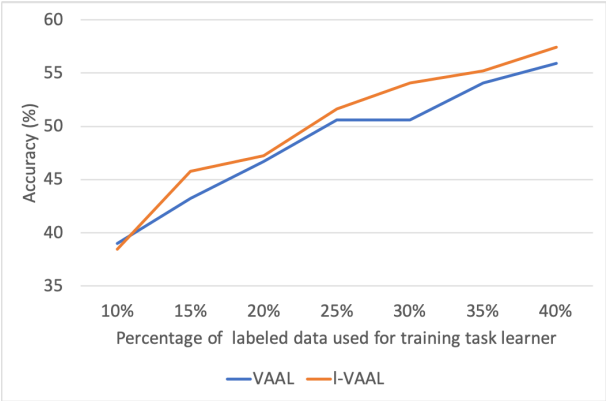
DQN training: To train the DQN network, we have a outer loop that run many episodes, store them in a replay buffer, and train the RL value network using the sample of episode history. We use epsilon-greedy exploration method for training.

To handle potential data reordering, the state input to the RL agent is partitioned into individual data points. That is, each unlabeled data point and its related features are fed individually through the agent Q network, and one Q value is output. This way, we remove the dependency between data and thus is resilient against data reordering.

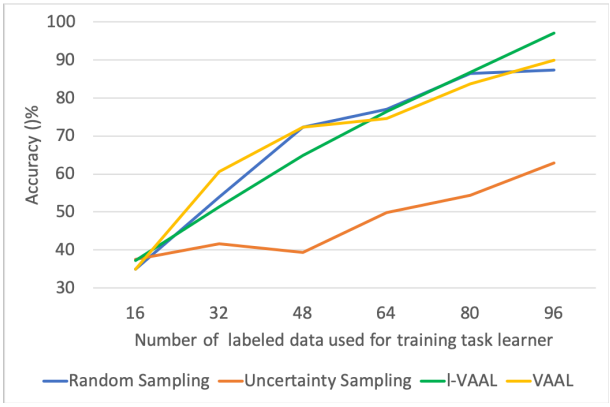
3.2.2 Policy gradient active learning

We propose a novel policy gradient method with epsilon-greedy exploration and k-means clustering to try to learn the optimal query function. To our knowledge, there is no other work that explores the use of policy gradient to find an optimal query in active learning. The high level diagram can be seen in 3b.

Our method works as follows. We begin by performing a k-means clustering of the unlabelled datapoints; this k will also determine the size of our action space. Our state space is constructed from the performance of the task learner on the testing dataset. Next, we create a policy network which maps from states to actions, which is a choice of k-means cluster. To get the actual query, we randomly select a point from this chosen cluster. After every query, we simulate the "environment": we retrain a task learner using the augmented dataset $X_L + x_i$, the newly labelled query datapoint. After getting the next state, we compute the reward, which is also related to the performance of T on P. Note the special property of this problem formulation: we receive a reward after every action/query. To encourage stability, we introduce batching then use the REINFORCE gradient estimator [15] to update our policy. Finally, we also employ the use of epsilon greedy exploration to encourage diversity in sampling. For full explicit details, please refer to the appendix.



(a) l-VAAL v.s. VAAL on 10% CIFAR 10 Dataset



(b) l-VAAL v.s. other methods on toy dataset (Rings)

Figure 4: l-VAAL performance comparison with other methods

4 Experiment and Results

Experiment Setup: In this section, we discuss our experiement setup and results. We experiment with two different datasets: 10% CIFAR 10 and a toy data set (called rings) for fast experimentation and analysis purpose. This dataset is shown in Figure 7 in the appendix. It contains 2500 two dimensional data points evenly distributed over five concentric rings which form the five classes.

Results: The result section contains two sections. The first part presents the results for our VAAL method with class label. The second section examines reinforcement learning. Code for the experiment and results can be found on our github [2]

4.1 Results for representation improvement

We compare our l-VAAL method to the original VAAL, and other methods using the 10% CIFAR 10 dataset and our toy dataset. Figure 4a shows the accuracy of VAAL and l-VAAL method as we increase the labeled dataset from 10% to 40% on the 10% CIFAR 10 dataset (i.e. 1% to 4% of the whole CIFAR 10 dataset). In this experiment, the model select 5% of data to be labeled in each step. The results are averaged over 3 runs. As we can see from the result the performance of the l-VAAL method is slightly better than the VAAL. We also compared l-VAAL, VAAL, uncertainty sampling and random sampling using the Rings dataset. The results are shown in Figure 4b. In this experiment, the model selects 16 data points to be sampled in each step. The results are average over 5 random runs. Interestingly, the performance of VAAL is quite similar to random sampling, whereas l-VAAL outperforms both VAAL and random sampling. Uncertainty sampling method, on the other hand, uncertainty sampling performs the worst among all the method with the same labeling budget. This again demonstrates the importance of adding class label information to the adversarial training of VAAL.

Although l-VAAL has good performance, we want to explore the relationship between accuracy and label batch size. From our experiment, we found, surprisingly, that l-VAAL with labeling batch size 16 perform worst than uncertainty sampling with batch size 1 (shown in Figure 2a). This shows again shows that the batch correlation is still an issue in active learning, even with the state of the art methods. Comparing this with the results from Figure 4a, it also shows that when running active learning with smaller amount of data points, the batch correlative issue is more severe. This result motivated us to investigate the possibility of using reinforcement learning for active learning to improve the batch correlation issue.

4.2 Learning the query function

We develop two novel reinforcement learning approaches to learn the query function. Furthermore, our experiments in this section investigate the *cold start*, extreme low-data problem where our initial training dataset X_L is small, and our query size is small (on the order of tens of datapoints). Note that the experiments in this section are not directly comparable with one another; rather we present exploratory results using reinforcement learning.

4.2.1 Results for DQN active learning

To facilitate analysis and reduce the amount of computing required, we use a subset of the Rings dataset for evaluate the DQN method. In particular, we use a dataset with 25 points (5 points in each class) sampled from the Rings dataset. Although this

[2]For l-VAAl, DQN: https://github.com/jyericlin/vaal ; For Policy gradient:https://github.com/johntiger1/vaal_querying

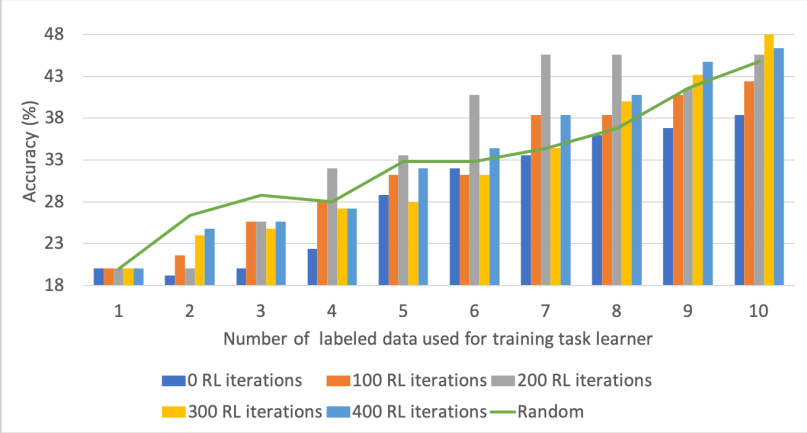


Figure 5: DQN querying method accuracy performance with different number of labeled data at different RL training iteration. X-axis represents the number of labeled points used for training the task learner; y-axis is the accuracy of the task learner model. The colors of the bars in each cluster of bars represent different RL training iterations, ranging from 0 to 400. The green line shows the performance of the random policy.

dataset may appear to be small, the possible RL trajectory space is very large. From example, sequentially selecting 10 points from 25 points yields 1.18×10^{13} possibles trajectories

During training, we train the task learner for 800 iterations, and we train the DQN network in the outer loop for 400 iterations. We use a exploration factor ϵ starting from 1 and decay by a factor of 0.9 in every 100 outer loop iterations. The DQN network evaluation is executed every 100 iterations for 5 rollouts with no exploration (i.e. $\epsilon = 0$). The result is shown in Figure 5. We can observe that the DQN policy get better as we train it with more iterations, and the DQN policy outperforms random policy after approximately 200 RL training iterations. We are comparing with random policy as it has relatively good performance when the number of data points is small. Another observation we found is that the random policy may have better performance in the first couple samples, because our model has seen less number of data points than the number of classes. The result shows that DQN policy can be used for efficient selecting unlabeled data points to be labeled.

4.2.2 Results for policy gradient active learning

We experimented with multiple different configurations and hypotheses for our policy network formulation, but present our best results on our final model for space purposes. [3] As Figure 6 shows, our method generally outperforms uncertainty sampling and the random baseline. One key benefit of our method is the speed, and we are able to train our small policy network on the full 2500 point Rings dataset.

Our policy gradient network with epsilon-greedy exploration combines the strong early performance of uncertainty sampling (before it converges and gets stuck in a local minima), but also give it the relative stability and well-behavedness of the random baseline. During experimentation, we noticed that our policy gradient method was susceptible to falling into situations where it would achieve 100% accuracy on one class, but 0% accuracy on other classes. This behaviour is also widely observed in uncertainty sampling, as we note in the appendix. Hence, we added epsilon-greedy exploration in order to break out of these local minima. Not only does exploration make our policy gradient query algorithm more robust to noise and less likely to converge to a bad solution early, randomness also improves the overall performance.

5 Limitations & Future Extensions

5.1 l-VAAL

Both the l-VAAL and VAAL methods are still not effective in selecting AL batches that have low correlation. Further integration of VAAL's representation learning with RL method can be a direction of improving the batch correlation issue.

5.2 DQN

Although the current DQN method is able to work for a small dataset, it still has a number of important limitations:

[3] See Appendix for a full list of experiments run, and analysis on uncertainty sampling

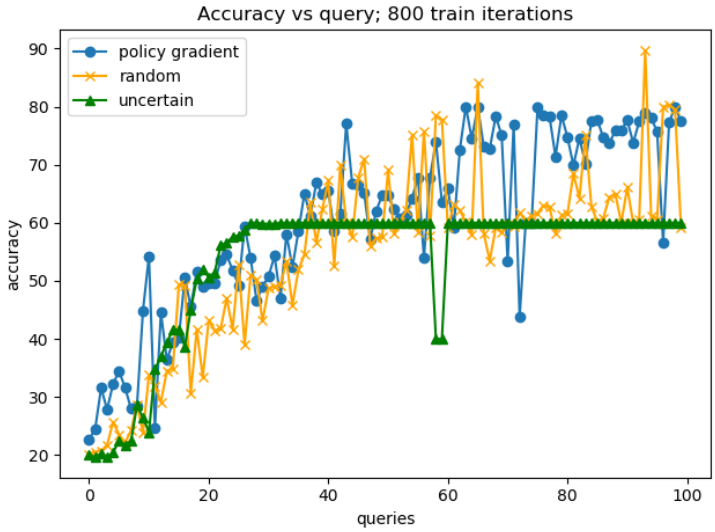


Figure 6: Performance of policy gradient active learning algorithm in comparison to random baseline and least confidence uncertainty sampling. We run our experiments on the toy dataset, where the task model begins with a single datapoint training dataset and queries a single data point each time. We run with $k = 5$; i.e. the number of clusters is the same as the number of classes. Results are generated with exploration parameter ϵ set to 0.25, batch update steps set to 10 (meaning 10 rewards are computed to produce a single update to the policy network parameters), 16 batch size for the task learner.

1. **Large action space**: Since the current proof of concept method uses the number of unlabeled data points as the action space, it does not scale well to large dataset. The possible extension to this to use hierarchical reinforcement learning [4, 3], where the high level RL agent selects a subset of unlableled points based on some representation, and the low level RL agent selects one or more data points to be label. The low level RL can also be modeled using a multi-agent reinforcement learning problem [5, 7], where we can exponentially reduce the action space.
2. **Stochasticity of reward and state transition**: We observe that the the accuracy improvement (i.e. the reward) when adding a certain labeled point is quite stochastic. This is due to the fact that the task learning can converge to different points during training, and thus have different accuracy. This also leads to the stochasticity of state transition. We observe that the DQN training is not very stable in a stochastic environment, and could exhibit divergence behavior.
3. **Batch active learning query**: The RNN memory core of a RL agent can serve as a context or history for the query model to select decorrelated datapoints. However, we will need to remove the dependency on the task learner output, which will allow for more effective multi-step rollout.
4. **Generalization**: We did not try experimenting to see whether our trained RL agent could generalize over similar types of datasets and task learners. This is an interesting research area for exploration both for active learning and reinforcement learning.

5.3 Policy network

Our policy network approach suffers from similar issues as the DQN, including stochasticity, correlated batch and generalization. Additionally, our policy network also has unique limitations, described here.

5.3.1 Policy Collapse

The policy network suffers from policy collapse, where the action distribution converges to unit probability on one cluster. There are several possible reasons for this. We operate in the cold-start problem setting, where the initial training dataset is just one point. Hence, the first few data points we add to X_L are especially important; the task learner cannot be penalized during training if it has no points of that class in its training dataset. Furthermore, since our reward is computed without differentiating the individual classes, there is no additional penalty for focusing on exclusively one class.

Analysis of the training mechanism can help us determine why this might be the case. We use the REINFORCE gradient estimator to directly train the policy:

$$\nabla_\theta \mathbb{E}_\tau[r(\tau)] = \mathbb{E}_\tau[r(\tau)\nabla_\theta \log p_\theta(\tau)]$$

As we skew X_L further and further, the network learns to optimize the reward by focusing on the probability of a rollout $\log p_\theta(\tau)$, and essentially ignoring the reward signal. Essentially, rather than learn to maximize the reward (as specified by the

0/1 accuracy performance) the network learns to minimize the chance of incorrectly predicting a rollout. We hypothesize that additional reward engineering may alleviate this problem.

5.3.2 Learning a good action representation

K-means may not be an appropriate representation, especially once the data becomes higher dimensional. Even in the toy dataset, K-means is not ideal, since our classes are essentially defined by their radial distance from the origin; k-means cannot correctly capture this behaviour. This could also explain why the network becomes obsessed with maximizing log probability; it takes too long for the network to discover a useful mapping between k-means clusters and the original classes. Given the well-studied limitations of clustering and distance metrics in higher dimensions [19], this is the most promising and substantial line of work for improving our PG method.

6 Related work

6.1 Reinforcement Learning in Active Learning

There are a few previous works that also examine the application of reinforcement learning in active learning. However, they explore reinforcement learning in a specific setting of active learning, such as multi-task active learning or stream-based active learning. [2] also examine learning the query function, but their approach involves using matching networks. Additionally, they explore in the one-shot metalearning setting; they metalearn the query function in a multitask environment, with the assumption that the query function is transferrable across tasks. We explore the more challenging and fundamental problem where we have just the task at hand. [6] explore the use of reinforcement learning for active learning in the natural language processing domain. They use reward shaping to guide a deep Q network; similarly to our policy gradient formulation, the reward is immediately computed after every time step as $Acc_t - Acc_{t-1}$. They consider the more simplistic stream-based active learning setup, while we tackle the more general pool-based active learning.

6.2 Learning the query function

Reinforcement learning is not the only approach to learning the query function. [10] propose learning a regressor to predict the expected error reduction. They also run experiments on a 2D toy dataset, and in general are closest in spirit to our policy-gradient approach, but frame it as a strictly supervised problem, using monte carlo estimates of the error improvement. [16] similarly pose the problem as a contextual bandit problem. They explore the related domain of stream-based *online* active learning, where unlabelled examples arrive in a stream, and the query algorithm must decide immediately whether to pay a cost to label it, or pass.

6.3 Decorrelating the batch query

There are existing works that examine reducing the degradation of performance in batch mode active learning. [12] proposes a principled Bayesian batch active learning algorithm to produce decorrelated batch queries. The key is to formulate the batch active learning problem as sparse subset approximation of the posterior distribution of the full dataset. [18] looks at using an explicit, hand-crafted, heuristic decorrelation technique to increase diversity within a batch. They pose the problem as the NP-hard Facility Location problem, and use k-means as an approximation, picking unlabelled datasets that are closest to the cluster centers.

6.4 Representation Learning using VAEs

Using VAEs to learn good representations, for semi-supervised, low-data settings can be found in [9]. In Kingma's work, they simply treat the missing label as part of a specialized missing data imputation task, which a generative model can answer. Similarly, [8] follows and [14] are both extensions on [9]. In [8] they perform inference of the labels using a Bayesian Neural Network, which is claimed to be better at modelling uncertainty than the recognition (encoder) network. In [14] they combine an adversarial discriminator into the process to learn a useful representation for querying by uncertainty sampling.

7 Conclusion and Future Work

In this paper, we identify two key problems in the active learning literature: the use of heuristic query functions and the batch correlation problem. We show that by enhancing VAAL with labels, we can improve performance; this provides research direction for addressing the batch query correlation problem. Furthermore, we propose using reinforcement learning as a solution to hard-coded heuristics, and demonstrate empirical performance of two novel architectures on a toy dataset. Future research directions involve combining our two methods more thoroughly. This paper takes a step towards achieving decorrelated learnt batch query functions, which will enable the flourishing of deep learning in label-scarce domains.

References

[1] C. C. Aggarwal, X. Kong, Q. Gu, J. Han, and S. Y. Philip. Active learning: A survey. In *Data Classification*, pages 599–634. Chapman and Hall/CRC, 2014.

[2] P. Bachman, A. Sordoni, and A. Trischler. Learning algorithms for active learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 301–310. JMLR. org, 2017.

[3] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[4] P. Dayan and G. E. Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pages 271–278, 1993.

[5] S. De Jong, K. Tuyls, and K. Verbeeck. Fairness in multi-agent systems. *The Knowledge Engineering Review*, 23(2):153–180, 2008.

[6] M. Fang, Y. Li, and T. Cohn. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, 2017.

[7] J. N. Foerster, C. A. S. de Witt, G. Farquhar, P. H. Torr, W. Boehmer, and S. Whiteson. Multi-agent common knowledge reinforcement learning. *arXiv preprint arXiv:1810.11702*, 2018.

[8] J. Gordon and J. M. Hernández-Lobato. Bayesian semisupervised learning with deep generative models. *arXiv preprint arXiv:1706.09751*, 2017.

[9] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

[10] K. Konyushkova, R. Sznitman, and P. Fua. Learning active learning from data. In *Advances in Neural Information Processing Systems*, pages 4225–4235, 2017.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[12] R. Pinsler, J. Gordon, E. Nalisnick, and J. M. Hernández-Lobato. Bayesian batch active learning as sparse subset approximation. *arXiv preprint arXiv:1908.02144*, 2019.

[13] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[14] S. Sinha, S. Ebrahimi, and T. Darrell. Variational adversarial active learning. *arXiv preprint arXiv:1904.00370*, 2019.

[15] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[16] S. Wassermann, T. Cuvelier, and P. Casas. Ral-improving stream-based active learning by reinforcement learning. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD) Workshop on Interactive Adaptive Learning (IAL)*, 2019.

[17] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 113(2):113–127, 2015.

[18] F. Zhdanov. Diverse mini-batch active learning. *arXiv preprint arXiv:1901.05954*, 2019.

[19] A. Zimek, E. Schubert, and H.-P. Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining*, 5(5):363–387, 2012.

Appendices

A Extended Background

A.1 Uncertainty Sampling

Uncertainty sampling refers to a family of query selection algorithms where the query is directly chosen according to what makes the task learner T the most "uncertain". For classifiers where the output of T has some probabilistic interpretation (for

instance, the vector of class probabilities assigned by a neural network with a softmax output layer), this corresponds to looking at the probabilities assigned to the output of T on a pass through the entire unlabelled dataset and applying some heuristic decision function. There are several heuristics for choosing the specific datapoints based on T's outputs, including least-confident, maximum-margin and maximum entropy. For instance, the least-confident uncertainty sampling method selects the following points:

$$\arg\max_{x_i \in D_u} [1 - \max P_\theta(y|x_i)] \text{ , where } \theta \text{ are the task learner model parameters}$$

There are two important points to note here. First, note that since the argmax is over the unlabelled dataset, the true label is unknown. Hence, we necessitate the use of the max; this is what the task learner *would* classify the point as. Second, note that regardless of the heuristic used, uncertainty sampling will simply take the top k datapoints according to the heuristic, with no regard to the interaction effects between the correlation of the datapoints in the query.

A.2 Learning to Search: Direct Discrete Optimization

Hence, there are two major difficulties in utilizing expected error reduction as our query algorithm: computational difficulty and correlated queries in the batch. Because a query is necessarily discrete (we either choose to reveal a label or not) we choose to apply discrete optimization techniques to directly optimize for the expected error improvement. For these purposes, we utilize the REINFORCE gradient estimator in our policy network approach, and Q-learning methods developed by Sutton for the DQN.

A.3 Expected error improvement

As an alternative to heuristic based uncertainty sampling, maximizing expected error reduction is a more principled approach to querying that tries to directly optimize the desired quantity or task performance metric P on testing dataset D_T. This also has an interpretation of maximizing the expected information gain of the query with respect to the unlabelled dataset [13]. In this approach, query points are chosen according to the expected future performance of a model trained on a dataset augmented with these query points. Mathematically:

$$\arg\max_{x_i \in D_U} P_{\{D_L + x_i\}}(D_T) - P_{\{D_L\}}(D_T)$$

where $P_{\{D_L+x_i\}}$ is the performance metric of a classifier trained on labelled dataset D_L with the addition of x_i fron D_U. Since P cannot be computed without the ground truth label, we must instead use an expectation over all labels for x_i. Unfortunately, this technique is very computationally expensive to run (requiring posterior inference over the labels) and retraining a classifier for each datapoint, and so in practice must be estimated.

B Dataset

Figure 7 shows the Rings dataset that we have generated for experiment purposes.

C Policy Gradient

To make things explicit:

- States: the states are a 1x10 vector formed from concatenating the current 0/1 accuracy of the task learner in each of the 5 classes (1x5 vector) in the **testing dataset** X_T with the class-based counts of the labelled datapoints in X_L (1x5 vector)
- Actions: the actions are a 1xK vector, where K is the number of clusters chosen in the initial K-means clustering algorithm.
- Reward: the reward is the 0/1 accuracy **on the testing dataset** of the task learner in each of the 5 classes (1x5 vector); i.e. the first half of the state vector. We subtract a baseline in order to make the policy more performant.

8 Experiments run on Policy Gradient Network

We experimented with epsilon decay schedule, different forms of reward engineering, varying the number of rewards to batch for an update to the policy network, gradient clipping, different levels of epsilon, and different formulations of the state and action spaces. As a gold standard, we also experimented with a PG network where we were allowed to examine the unlabelled dataset

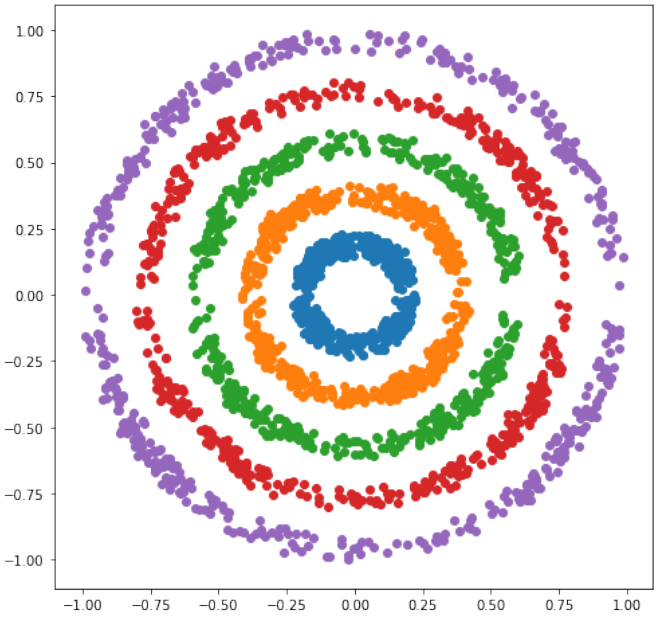


Figure 7: Toy data set: Rings

for elements of a specific class; i.e. we knew the ground truth labels. As expected; the general optimal querying strategy is to ensure that the cumulative training dataset X_T is well distributed amongst the ground truth classes of the prediction task.

Our policy network is a simple 2-layer feedforward neural network with 64 hidden layer neurons. We use Adam optimizer with learning rate set to 5e-3.

9 Analysis of uncertainty sampling

We can see several trends from figure 6. First, uncertainty sampling converges at 60% accuracy, which is the result of achieving 100% accuracy in 3 classes, and 0% accuracy in the remaining 2 classes. Although the given figure is produced from a single run, we observed this type of behaviour across multiple random seeds: sometimes the uncertainty sampling would converge to 40% accuracy, while other times it would successfully converge to near 100% accuracy. Common to all these runs was that the per-class accuracy would be highly skewed, distributed bimodally at either extreme. This is one drawback of the uncertainty sampling algorithm; it cannot pull itself out of local minima.

Indeed, when we examine the composition of the cumulative training dataset X_L queried by the US algorithm, we see extreme skew in the class composition.

We believe this behaviour is due to the fact that the task model essentially serves double duty as the query model in the uncertainty sampling approach. Hence, US has high sensitivity to noise in the training process. Since each marker in the plot is generated from retraining a classifier on an augmented dataset $X_L + x_i$, there is randomness in the initialization of the neural network weights, and in every step of the SGD minibatch-guided training process thereafter: order of the batch, composition of the batches and so forth. This noise means that sub-optimal queries lead to task learners with worse decision boundaries, which in turn leads to even more sub-optimal queries in the future, in a vicious cycle that leads to convergence in a local minima.

During experimentation, we noticed that our policy gradient method was also susceptible to this local minima trap. Hence, we added epsilon-greedy exploration in order to break out of these local minima. Not only does randomness make our policy gradient query algorithm more robust to noise and less likely to get trapped in local minima, randomness also improves the overall performance. In effect then, our policy gradient network with epsilon-greedy exploration is an attempt to combine the best of both worlds. We try to take advantage of the strong performance of US (before it converges and gets stuck in a local minima), but also give it the relative stability and well-behavedness of the random baseline.

The random baseline is quite strong and competitive in this toy data setting, likely because it can avoid getting step in these local minima. This is likely due to the low-dimensional nature of our problem; the random baseline is effective at sampling all the classes.

10 Miscellaneous Figures

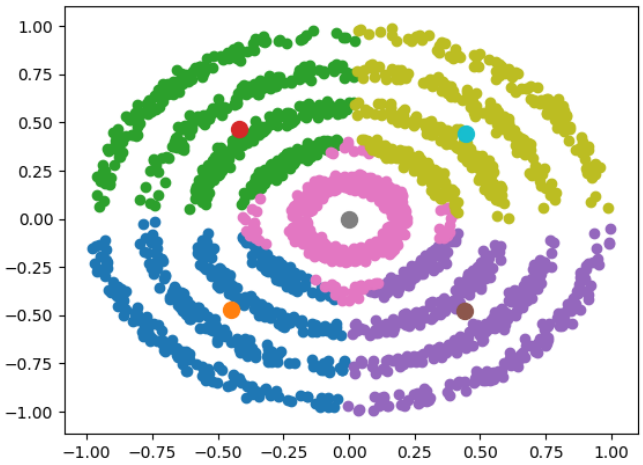


Figure 8: K-means is not necessarily an appropriate dimensionality reduction technique.

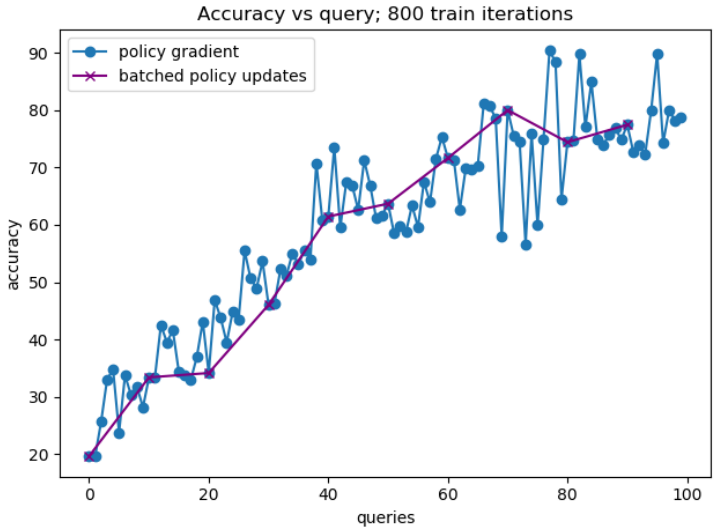


Figure 9: The use of batched updates to the policy network improved performance.