CS230

Learning To Learn: A Dual Network Approach To Multi-Class Active Learning

Sameer Khanna [1]

Abstract

Active Learning dramatically reduces the numbers of vectors one needs to label for a given problem space, a boon for industries where labeling can be time consuming and expensive such as medicine. Traditional Active Learning's biggest downside is its volatility in performance, in some cases being worse than randomly selecting vectors to label. Deep Learning Based Active Learning, a dual network approach that uses one network to select vectors to label via a ranking system and the other to pseudo-label vectors, solves both of these downsides as we show on a variety of real world problem spaces.

1. Introduction

Supervised Learning (SL) is the holy grail of Machine Learning, however it typically requires large swaths of labeled data. This can hamper its usage in fields such as medicine, where labeling data is prohibitively expensive. Active Learning (AL) is an approach that aims to reduce the number of labels required for algorithms by actively selecting queries for an oracle to label.

While empirical results suggest that AL does work for a variety of different problem spaces [2]–[6], there is no guarantee that AL will work for the problem space one is considering. Using AL can actually require more labelled data than randomly sampling (RS) data to label [7], [8].

The issue is that traditional AL focuses on a singular, model specific strategy. While this works for many spaces, each strategy has potential roadblocks, such as uncertainty sampling's susceptibility to choosing outliers [9], and query-by-committee approaches focusing on inconsequential regions of the problem space [10].

We propose approaching AL as a regression problem, with popular and novel strategies restructured as features that are pulled from a variety of different model types. We show that our Deep Learning Based Active Learning (DLAL) achieves optimal performance when compared to alternative AL strategies.

[1]Stanford University, Palo Alto, California, USA. Correspondence to: Sameer Khanna <sameerk@stanford.edu>.

2. Identifying Optimal Seed Vectors

In order to start querying vectors, we need an initial set of data with which we can train a preliminary model. While traditional AL algorithms create this set via random sampling [11], various studies have shown that utilizing pre-clustering techniques to determine our initial set of labeled vectors can lead to improvements in final model performance [12], [13]. Pre-clustering has only been shown to work in low dimensional, binary classification tasks. Here, we propose an expansion to enable seed identification even in high-dimensional multi-class problem spaces. The entire process is illustrated in Figure 1.

Our initial seed vectors are identified by clustering via Gaussian Mixture Modeling (GMM) [14], using the cluster medoids as our seeds. The optimal clustering, assessed by both number of clusters and distribution of points within clusters, is determined using the average silhouette approach [15].

Clustering techniques have performance issues when utilized in high dimensional spaces, due to higher data sparsity and increased irrelevance of notions of distance [16]. To solve this issue we propose the utilization of the manifold learning technique t-Distributed Stochastic Neighbor Embedding (t-SNE) [17] for dimensionality reduction prior to applying clustering.

There is a strong argument that t-SNE is uniquely suited to our goals. While other variants of manifold learning are catered towards unfolding a single continuous low dimensional manifold, t-SNE focuses instead on structures at the local level. Additionally, t-SNE is better suited at separating data comprised of numerous manifolds at once as is often the case in the real world [18].

t-SNE also compares well for our use case compared to other dimensionality reduction techniques that are not manifold learning algorithms. Such transformations typically leave data in lower dimensional representations projected on top of each other as dimensions disappear [19]. This results in the output of these alternatives being hard to decipher and restricts the ability to make clear statements about separability in higher dimensional spaces when compared to t-SNE.

Using t-SNE allows us to better highlight intraclass disparities. For example, as shown in Figure 1, applying our novel seed identification process enables us to identify all 3 hand-drawn versions of the number 1 found in the dataset; applying GMM directly would only supply us one of these examples.

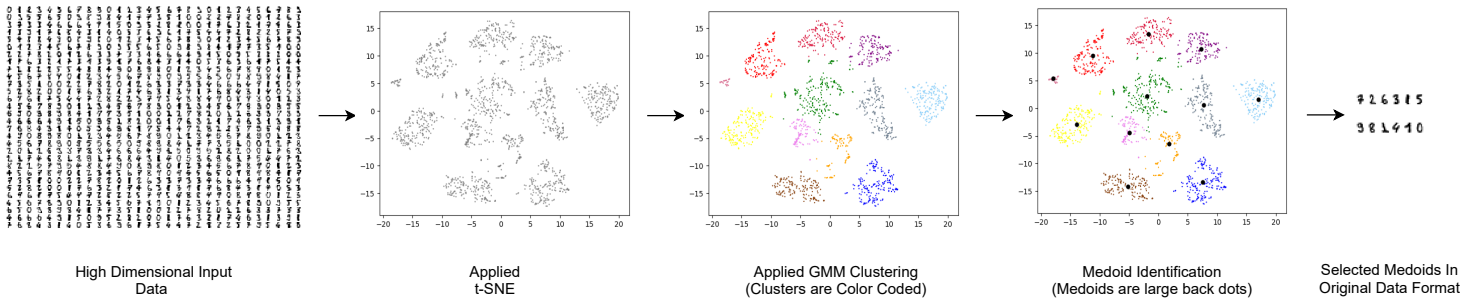

Figure 1. Process of Seed Identification on the Optical Recognition of Handwritten Digits Dataset [1]

3. Vector Selection Strategies

Once we have some labeled data, our preliminary model is trained on the given data and aims to classify the remaining points. In order to identify the optimal vector to query the oracle about, we compute heuristics for each unlabeled vector in our problem space.

3.1. Shannon's Entropy

Shannon's entropy (SE), a metric that represents the total amount of information stored in a distribution, is typically thought of as a measure of uncertainty in the field of machine learning [20]. SE is defined by $\arg\max_x -\sum_i p(y_i|x;\theta)\log(p(y_i|x;\theta))$.

The more uniform a distribution is, the larger the entropy. A model with a high confidence or probability score for a particular class will have low entropy, whereas a model that is not confident in deciding between classes will have high entropy, making the metric ideal for modeling uncertainty.

3.2. Confidence Based Strategies

Entropy takes into account uncertainty across all available classes, but a model may have a hard time deciding between two classes. Margin of Confidence (MC), defined by $1 - \left(p(y^*_{(1)}|x;\theta) - p(y^*_{(2)}|x;\theta)\right)$, and Ratio of Confidence (RC), determined via $\frac{p(y^*_{(2)}|x;\theta)}{p(y^*_{(1)}|x;\theta)}$, are metrics that aim to better identify such scenarios [21]. Here, $y^*_{(n)}$ denotes the nth most likely class based on the model's prediction probabilities. MC is the difference between the top two most confident predictions, while RC is their ratio.

An alternative approach is simply choosing the point whose classification the model has the Lowest Confidence (LC) in, as is shown in its formula $\arg\min_x p(y^*_{(1)}|x)$. Despite its simplicity, LC has been shown to work well with conditional random fields [22] as well as for active learning in information extraction tasks [6], [23].

3.3. Distance from Hyperplane

One potential strategy for labeling points is to choose points we expect to maximally narrow the existing margins. The location of a vector with respect to a decision boundary determines the magnitude its labeling changes decision boundary position, with closer vectors having a greater affect. Schohn et.al. has described the approach as a simple and effective form of divide and conquer [24].

Different problem spaces will have differing dimensions, and varying separation between classes. In order to utilize metrics across problem spaces, we scale a vector's boundary distance by the average distance for all points in the problem space.

3.4. Orthogonality to Labeled Points

When performing AL in high dimensional spaces, it is easy for algorithms to ignore particular dimensions or pockets within a problem space due to the nature of having dimensions orders of magnitude larger than the number of examples. This can lead to a major disconnect between the decision boundaries of our model and the true underlying class separation. By searching for examples that are orthogonal to the space spanned by the labeled set, we give the learner information about dimensions it has not yet explored. In order to utilize these principles even in problem spaces of lower dimensionality or with higher space coverage, we relax this constraint to allow for vectors with large angles to be selected. Our orthogonality metric (OM), $\min_{l\in L} \cos^{-1}\frac{<x_i,l>}{|x_i||l|}$, finds the smallest angle between the unlabeled vector x_i in question and the vectors in the labeled set L.

3.5. Information Density

Many AL algorithms aim to query vectors our given model is most uncertain of, leading to a proclivity to query outliers whose labeling will have little to no affect on model performance. This motivating factor led to the development of the information density framework (IDF), $(\arg\max_x \phi_A(x))\left(\frac{1}{U}\sum_u sim(x,x^{(u)})\right)^\beta$ [6]. Manipulating IDF to serve our purpose, we coin the information density metric (IDM), $\frac{1}{U}\sum_u sim(x,x^{(u)})$. IDM aims to scale the strategy by weighing it against the average similarity to all other instances in the input distribution. sim refers to a similarity function such as cosine similarity, the dot product between normalized vectors, [25] or Euclidean similarity, which is the reciprocal of Euclidean distance [26]. The higher the information density, the more similar the given instance is to the rest of the data. While Cosine IDM defines the centermost cluster as most important, Euclidean IDM prefers the center of clusters.

3.6. Perturbation

Inspired by Adversarial Attacks and its implications for deep learning models [27], [28], we aim to extend its usefulness to AL for all model types by identifying the maximal shift in model confidence incurred by adding perturbation to each unlabeled vector. Let $\epsilon \sim \mathcal{N}(0,1)$, then we aim to calculate $D_{KL}(p(y|x)||p(y|x+\epsilon)$; in other words we wish to calculate the Kullback–Leibler divergence (D_{KL}) [29] of the model's prediction probabilities for a given vector before and after adding perturbation. The larger the divergence after adding ϵ, the more crucial a label is to improve model performance.

3.7. Expected Gradient Length

Discriminative models are typically trained using gradient-based optimization; the amount a model will be changed at a given time can be quantified by the expected gradient length (EGL) [30]. In order to make the largest updates to the model possible, it will be optimal to choose a vector x that leads to the largest change in our objective function ℓ, as determined via $\arg\max_x \sum_i p(y_i|x_i;\theta)\,\|\nabla\ell(\mathcal{L}\cup(x,y_i);\theta)\|$. The vector's gradient for a possible class is scaled by its prediction probability as output by the current model.

3.8. Consensus Based Strategies

Consensus based strategies strategies utilize multiple models in various combinations in order to identify vectors of interest. Query-by-committee (QBC) consensus has a committee composed of multiple models trained on our set of labeled data with each model having a unique initialization [31]. Co-Training [32] and Co-Learning [33] approach consensus through different lenses, using differing subsets of features and using different model types altogether respectively. No matter the consensus strategy, they all function in a similar way. The vectors that models disagree the most over have the most potential information to give; these vectors are the most optimal to label.

4. Choosing a Vector

Every strategy detailed in Section 3 approaches the problem space in a unique manner. By reclassifying each strategy into a feature to be fed into a network, we will be able to consider all possible strategies when determining the optimal vector to label. Unlike most AL strategies, each of the features are compiled from up to eight unique models and not solely the model to be trained. Each feature and the relevant ML model is detailed in Table 1.

Table 1. Compiled Features

Heuristic	Relevant ML Model
Shannon's Entropy	Model To Be Trained
Margin Of Confidence	Model To Be Trained
Ratio Of Confidence	Model To Be Trained
Least Confidence	Model To Be Trained
Decision Boundary Ratio	Linear SVM
Decision Boundary Ratio	Sigmoid SVM
Decision Boundary Ratio	RBF SVM
Decision Boundary Ratio	Polynomial SVM
Angle From Labeled Set	N/A
Cosine Density	N/A
Euclidean Density	N/A
QBC	Model To Be Trained
Co-Training	Model To Be Trained
Co-Learning	Perceptron, Random Forest, Softmax
Expected Gradient Length	Softmax Regression
Perturbation	Model To Be Trained
Number of Features	N/A
Number of Classes	N/A

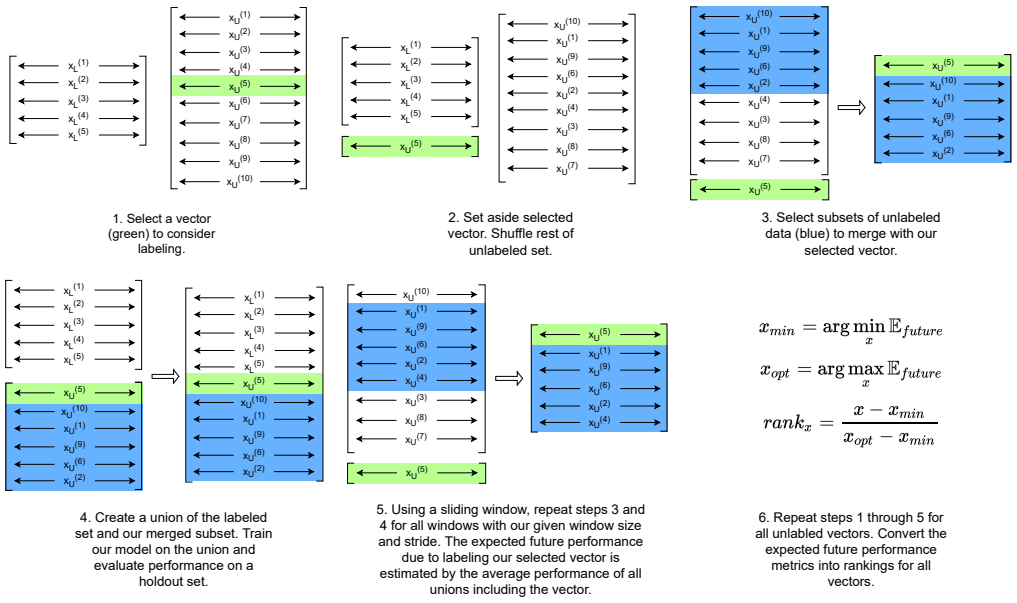

Figure 2. Random Sub-sampling Based Vector Ranking

4.1. Training Set Generation

Our models, both custom instances of Dense Residual Networks, are trained on synthetic datasets based on the Madelon dataset [34], [35] with varying numbers of dimensions, size, and number of clusters. For each set, we alternate between identifying starting values by random selection, or via the seed identification process detailed in Section 2. From here we iterate through all unlabeled vectors one after another, computing all heuristics listed in Table 1.

4.2. Querying The Oracle For Labels

After our training data is compiled, a vector querying neural network (VQNN) is trained to predict from the features detailed in Table 1 the relative ranking of vectors most optimal to label. The larger the ranking, the more confident the model is that the vector is an optimal point to label. We send the vector with the largest ranking to the oracle for labeling.

While ranking vectors based on their direct affect to model performance due to their addition to the labeled set of data would be straightforward, Dasgupta et.al. found that such greedy approaches are not optimal because they don't take into account the way in which a query reshapes the search space [36]. This leads to increasingly subpar performance for greedy AL algorithms as they continue trying to select vectors to query.

Instead, we estimate the potential benefit labeling the given vector will have by taking the average performance increase of labeling multiple random subsets of unlabeled data that include the vector in question. The vector with the largest average performance increase corresponds to the vector that is both most informative as well as best sets up the rest of the query space for optimal further search. The full process used for ranking vectors is shown in Figure 2.

Note that the proposed ranking algorithm cannot be used directly to select vectors to query. It assumes knowledge of the true labels of all unlabeled data, which defeats the purpose of AL. By using our VQNN to predict the random sub-sampling based vector ranking, we can optimally select vectors for labeling without requiring access to nonviable information.

While being able to rank all vectors correctly is ideal, we want to provide extra care and attention to the set of vectors whose rankings are close to the optimal; this helps ensure we can always choose the best vector at a given time. To this end, we use the following loss function $\frac{1}{N}\sum_i \exp(-\frac{(y_i-1)^2}{2\tau^2})(y_i - \hat{y_i})^2$, where y_i is the true ranking, $\hat{y_i}$ is the predicted ranking, N is the number of vectors, and τ is a hyperparameter that controls how quickly weight falloff occurs.

4.3. Incorporating Semi-Supervised Techniques

While AL focuses on identifying vectors of concern in a problem space, Semi-Supervised Learning (SSL) aims to take advantage of vectors our given model is confident of. The two sides of the same coin relationship between AL and SSL has allowed their concerted use in previous research with results greater than using one or the other alone [5], [37]–[39].

Focusing on pseudolabeling, we rephrase SSL as a binary classification problem. Our second network, the vector pseudolabeling neural network (VPLNN), aims to predict whether the given vector has been correctly labeled by the model.

A major issue with pseudolabeling is that is prone to producing incorrect results when the model produces unhelpful targets for unlabeled data [40]. To mitigate this issue, a vector is only pseudolabeled if it meets these criteria: the model to be trained is confident in its classification, the VPLNN marks it as valid, and the vector satisfies the smoothness constraint. The smoothness constraint assumes that vectors close together will be of the same class, and there is large separation between vectors of different classes. Requiring fulfillment of all criteria helps ensure an extremely low likelihood chance of incorrect pseudolabels occurring.

5. Architecture

5.1. Dense Residual Unit

Taking inspiration from the ResNet and VGG-16 architectures, our VQNN and VPLNN models are residual networks that are designed to be highly modular. A module, which we call the Dense Residual Unit (DRU), is shown in Figure 3. Unlike in the original ResNet architecture, we use Dense layers in lieu of Convolution layers as the weight sharing assumption does not hold for our compiled feature set. We then scale the residual mapping via a Batch Normalization layer and pass the resulting output to our activation function of choice.

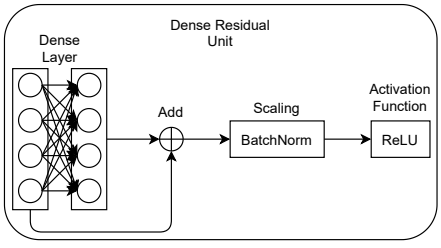

Figure 3. Dense Residual Unit

5.2. Vector Querying Neural Network

The architecture for VQNN is shown in Figure 4. For our hidden layers, we use the Tanh activation with later layers being slightly larger than those in the beginning. We found during testing that this setup helps better identify the correct ordering of high ranked vectors. Since rankings must be non-negative in value, we use a ReLU output activation.

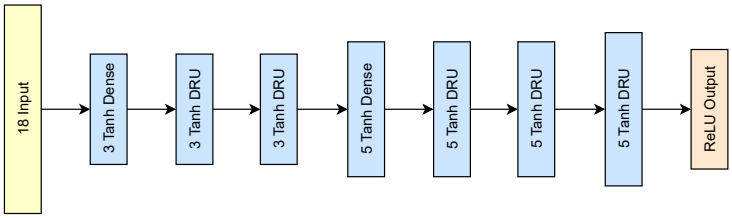

Figure 4. VQNN Architecture

5.3. Vector Pseudo-labeling Neural Network

The architecture for VPLNN is shown in Figure 5. We found that a standard ReLU focused architecture performed the best in improving model accuracy. Since VPLNN performs binary classification, our output activation is the standard sigmoid function.

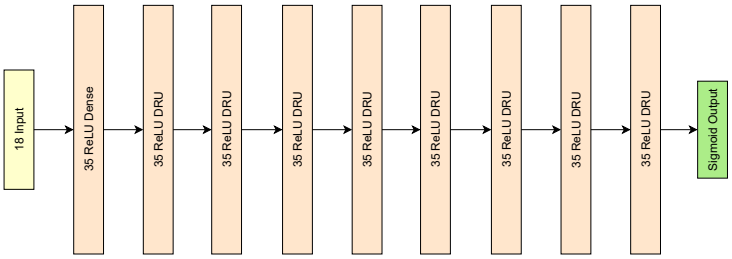

Figure 5. VPLNN Architecture

5.4. Hyperparameter Tuning

For training, we utilize synthetic data rather than real world datasets for training set generation due to the relatively low costs involved in obtaining additional problem spaces to incorporate into our training set. This enables us to create larger and more powerful models than we otherwise would have been able to, as we need to worry about over-fitting less as the amount of data increases. Indeed, we saw no substantial difference in performance for our models on our training set when compared to our training-development set, indicating that model variance was not a significant issue.

What was a substantial hurdle to proceed through was the data-mismatch problem; our networks are trained on data distributions different from the distributions they will be evaluated on. The data mismatch problem is only exacerbated by the fact that we wish for DLAL to work on a variety of different real world problem spaces; our target distribution is extremely complex and protean in nature.

To tune hyperparameters to mitigate data mismatch, we compiled development and test sets via the process detailed in Section 4.1 on the evaluation datasets listed in Table 2. Via manual coarse-to-fine random search, we identified that the hyperparameters τ, activation functions, layer size and number, and learning rate had the greatest affect to performance during validation.

This list of hyperparameters was too large to effectively search through manually. Once we found a list of parameters to tune, we utilized the sequential model-based optimzation approach Tree Parzen Estimation (TPE) [41]. Since TPE tracks previous evaluation results in order to map hyperparameter sets to probabilistic models, this enabled us to tune hyperparameters faster and has empirically shown can lead to better results than alternative approaches to hyperparameter tuning.

5.5. DLAL In Its Entirety

Figure 6 shows the entire process of DLAL in action. We first select an initial set of vectors for labeling. After training a model on this set, we enter a cycle, where we alternate between querying vectors as described in Section 4.2 and pseudo-labeling vectors as detailed in Section 4.3. This cycle continues until we are confident in our labeled set of data and our target model performance is acceptable.

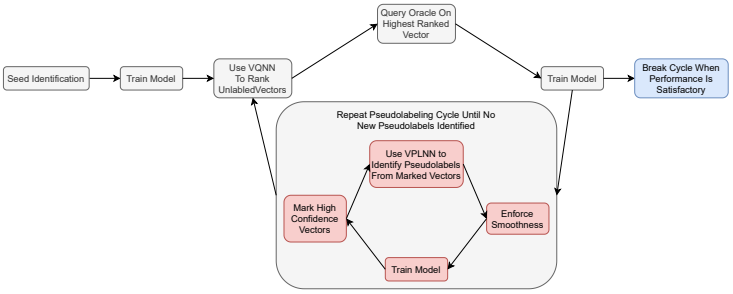

Figure 6. DLAL Process

6. Experiments

6.1. Evaluation Datasets

In order to ascertain the effectiveness of DLAL, we use a variety of real world datasets. From forensic analysis to disease identification to network security vulnerability detection, we evaluate effectiveness not only by varying number of classes, features, and input data format, but also by picking real world problem spaces from a wide range of topics. Table 2 gives a full list of datasets as well as the fields they hail from.

Table 2. Evaluation Datasets

Dataset	Dataset Codename	Problem Topic
Optical Recognition of Handwritten Digits [1]	Digits	Image Classification
Iris Flower [42]	Iris	Pattern Recognition
Wine Quality [43]	Wine	Cultivar Chemical Analysis
Balance Scale [44]	Balance	Cognitive Development
Car Evaluation [45]	Car	Constructive Induction
Agaricus and Lepiota Gilled Mushrooms [46]	Mushrooms	Toxicology
Pen-Based Handwritten Digit Recognition [47]	Pen	Image Classification
Cleveland Heart Disease [48]	Heart	Disease Diagnosis
Multi-spectral Landsat Satellite Statlog of Neighborhoods [49]	SatImage	Satellite Remote Sensing
Glass Type Identification [50]	Glass	Forensic Analysis
Vision Group Outdoor Image Segmentation [51]	Segmentation	Image Segmentation
Connectionist Bench Vowel Recognition [52]	Vowel	Connectionist Analysis
Eryhemato-Squamous Diseases [53]	Dermatology	Disease Diagnosis
IoT Device Identification On Corporate Networks [54]	IoT	Network Security

6.2. Baseline Algorithms

In order to evaluate the performance of DLAL, we compare its performance to the baselines listed in Table 3. RS was chosen to act as minimum threshold; if an AL algorithm performs worse than RS, then we consider the algorithm to have failed on the given dataset. Algorithms EGLAL, LDAL, EAL, EIDAL, MCAL, RCAL, and LCAL were selected due to their ubiquity in AL publications as well as in industry. MVAL was chosen due to being showcased in its original publication as one of the best retraining based AL algorithms [55], a subclass of AL where the model in question is retrained for every unlabeled vector for every possible class the vector could be. MVAL can achieve great performance albeit at immense computational cost, thus acting as a high bar to compare DLAL against.

Table 3. Baseline Algorithms

AL algorithm	AL Codename
Expected Gradient Length Active Learning	EGLAL
Linear Decision Boundary Active Learning	LDAL
Entropy Active Learning	EAL
Entropy - Euclidean Information Density Active Learning	EIDAL
Margin Confidence Active Learning	MCAL
Ratio Confidence Active Learning	RCAL
Least Confidence Active Learning	LCAL
Passive Learning / Random Selection	RS
Maximizing Variance Active Learning	MVAL

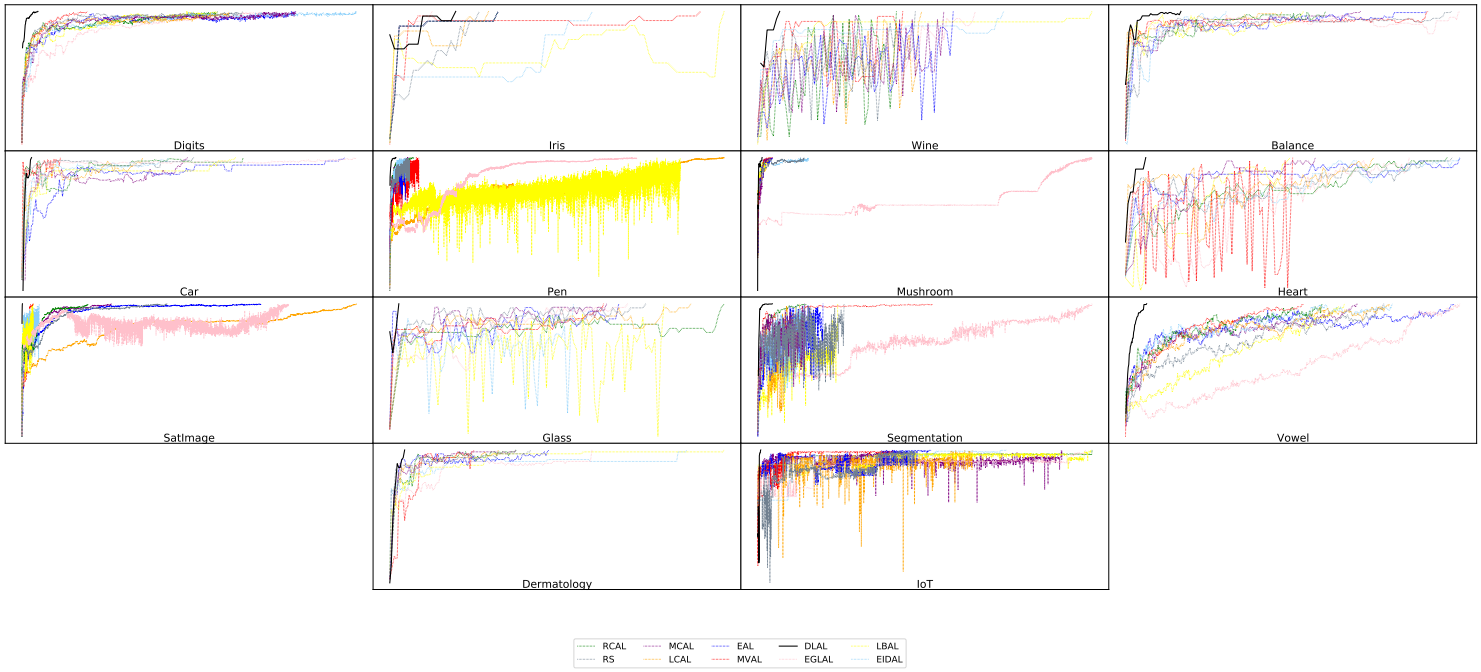

Figure 7. DLAL vs Baselines on all evaluation datasets. X-axis is number of vectors labeled. Y-axis is the scaled model performance.

6.3. Evaluation Methodology

To identify how well an AL algorithm performs on a given dataset, we first split the given dataset into a training and a holdout set via a 70%-30% split. We identify the optimal performance of our model to be trained by evaluating its performance on the holdout set after being trained on the entire training set, a score we call $score_{train}$. For each vector labeled, we will evaluate its relative performance on the holdout set via the evaluation metric $\frac{\min(score_{AL}, score_{train})}{score_{train}}$, where $score_{AL}$ is the performance of the model on the set of labeled vectors the AL algorithm has queried so far. The AL algorithms will continue to query points until its labeled set of vectors reaches the same performance as the entire training set; in other words, when the evaluation metric is 1.0. AL algorithms will be assessed based upon the number of labeled points needed as well as how volatile the model's performance is as the algorithm aims to reach the maximum performance. In order to ensure all algorithms start labeling with the same number of initial points, the baseline models will have an initial set of vectors whose number is determined via the number of medoids selected by DLAL's seed identification; this initial set will be identified via random selection as is customary for AL [11].

For EGLAL, we will use softmax regression as our target model as it is the natural extension of logistic regression, the target model used in the original paper, to multi-class problem spaces. For the rest of the AL algorithms, we will use a Linear Support Vector Machine in conjunction with Platt scaling [56] as the target model due to its common usage for AL.

Each experiment was repeated 5 times, with the median outcome of each experiment reported.

7. Results

Plots showcasing the performance of the AL algorithms on each dataset are shown in Figure 7. While all AL algorithms appear to behave nicely with relatively minor dips and plateaus for simpler problem spaces like Digits, the issues with using single heuristic strategies becomes very clear when seeing performance on more complex problem spaces like IoT, Wine, etc. In AL, we are trying to label the minimum number of points possible in order to get an accurate representation of the entire problem space. With such small sets of labeled vectors, labeling a sub-optimal vector can lead to a dramatic shift in our representation of the problem space, and thus cause a reduction in the trained model performance. This phenomenon is clearly occurring in our baseline models, which show high volatility in performance as we label additional vectors. DLAL in comparison shows significantly lower volatility, a clear showcase of the vigilance and care DLAL takes in selecting vectors for labeling that most optimally carves the search space.

Table 4. Baseline Comparison: Number Of Vectors Labeled

Dataset	Dataset Size	RS	EAL	RCAL	LBAL	EGLAL	EIDAL	LCAL	MCAL	MVAL	DLAL
Digits	1797	753	931	669	668	869	1137	709	933	230	**64**
Iris	150	22	26	26	74	**12**	46	24	26	69	17
Wine	178	53	65	47	109	72	81	55	62	49	**10**
Balance	625	198	183	123	110	203	64	97	144	184	**37**
Car	1728	165	306	211	204	317	141	117	159	40	**13**
Pen	10992	439	356	536	6305	5362	371	7257	415	638	**142**
Mushroom	8124	511	166	144	125	3672	557	134	149	171	**51**
Heart	297	132	135	130	68	115	135	101	78	69	**20**
SatImage	6435	1799	2954	821	145	3295	220	4131	1113	145	**12**
Glass	214	91	82	117	97	85	67	106	78	77	**9**
Segmentation	2310	355	270	202	329	1353	194	176	196	710	**66**
Vowel	990	487	616	373	415	627	503	497	438	339	**50**
Dermatology	358	101	113	91	231	142	206	79	91	63	**16**
IoT	2032	662	726	1396	1391	1370	1041	899	1271	566	**19**

DLAL not only is less volatile in its approach to the optimum, it reaches there faster than baseline measures. Table 4 lists the number of labeled vectors each AL algorithm needed to reach optimal performance on the target model. As can be seen, DLAL consistently outperforms its peers, in some cases by extremely large margins. DLAL also does not fall victim to pitfalls that affect other algorithms, as evidenced by its great performance when RS outperformed baseline AL algorithms.

Our seed identification gives DLAL an incredible headstart in initial performance over its peers, its affects clearly shown in the datasets Digits, SatImage, Pen, and Glass. However, while our seed identification exacerbates the difference between a single heuristic approach and DLAL, it is not the sole reason behind DLAL's performance. As can be seen in the Car and Dermatology datasets, seed identification did not give DLAL an advantage over the baseline models, due to the datasets having poor separation. Despite this, DLAL reached optimal performance using far fewer vectors than the baseline models.

8. Conclusion

The results show the power of DLAL's multi-heuristic, dual-network approach. DLAL utilizes information from multiple different heuristics with each giving a unique view of the problem space, allowing for it to not succumb to issues that may befall AL algorithms using single heuristics. Note that the baseline algorithms are primarily composed of algorithms represented by the features we feed into our neural network, yet DLAL still shined in scenarios where they failed. These heuristics are clearly more powerful together than they are apart.

References

[1] E. Alpaydin and C. Kaynak, "Optical recognition of handwritten digits data set," *UCI Machine Learning Repository*, 1998.

[2] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine learning*, vol. 15, no. 2, pp. 201–221, 1994.

[3] C. A. Thompson, M. E. Califf, and R. J. Mooney, "Active learning for natural language parsing and information extraction," in *ICML*, Citeseer, 1999, pp. 406–414.

[4] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of machine learning research*, vol. 2, no. Nov, pp. 45–66, 2001.

[5] G. Tur, D. Hakkani-Tür, and R. E. Schapire, "Combining active and semi-supervised learning for spoken language understanding," *Speech Communication*, vol. 45, no. 2, pp. 171–186, 2005.

[6] B. Settles and M. Craven, "An analysis of active learning strategies for sequence labeling tasks," in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 2008, pp. 1070–1079.

[7] A. I. Schein and L. H. Ungar, "Active learning for logistic regression: An evaluation," *Machine Learning*, vol. 68, no. 3, pp. 235–265, 2007.

[8] Y. Guo and D. Schuurmans, "Discriminative batch mode active learning," in *Advances in neural information processing systems*, 2008, pp. 593–600.

[9] S. Ebrahimi, W. Gan, K. Salahi, and T. Darrell, "Minimax active learning," *arXiv preprint arXiv:2012.10467*, 2020.

[10] N. Roy and A. McCallum, "Toward optimal active learning through monte carlo estimation of error reduction," *ICML, Williamstown*, pp. 441–448, 2001.

[11] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[12] H. T. Nguyen and A. Smeulders, "Active learning using pre-clustering," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 79.

[13] W. Yuan, Y. Han, D. Guan, and S. Lee, "Initial training data selection for active learning," Jan. 2011, p. 5. DOI: 10.1145/1968613.1968619.

[14] G. J. McLachlan and K. E. Basford, *Mixture models: Inference and applications to clustering*. M. Dekker New York, 1988, vol. 38.

[15] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[16] M. Steinbach, L. Ertöz, and V. Kumar, "The challenges of clustering high dimensional data," in *New directions in statistical physics*, Springer, 2004, pp. 273–309.

[17] L. Van Der Maaten, "Learning a parametric embedding by preserving local structure," in *Artificial Intelligence and Statistics*, 2009, pp. 384–391.

[18] 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*, vol. 12, pp. 2825–2830, 2011.

[19] A. Akalin, *Computational Genomics with R*. CRC Press, 2020.

[20] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE mobile computing and communications review*, vol. 5, no. 1, pp. 3–55, 2001.

[21] M. Li and I. K. Sethi, "Confidence-based active learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 8, pp. 1251–1261, 2006.

[22] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[23] A. Culotta and A. McCallum, "Reducing labeling effort for structured prediction tasks," in *AAAI*, vol. 5, 2005, pp. 746–751.

[24] G. Schohn and D. Cohn, "Less is more: Active learning with support vector machines," in *ICML*, Citeseer, vol. 2, 2000, p. 6.

[25] A. Singhal *et al.*, "Modern information retrieval: A brief overview," *IEEE Data Eng. Bull.*, vol. 24, no. 4, pp. 35–43, 2001.

[26] K. L. Elmore and M. B. Richman, "Euclidean distance as a similarity metric for principal component analysis," *Monthly weather review*, vol. 129, no. 3, pp. 540–549, 2001.

[27] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[28] M. Ducoffe and F. Precioso, "Adversarial active learning for deep networks: A margin based approach," *arXiv preprint arXiv:1802.09841*, 2018.

[29] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[30] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning," *Advances in neural information processing systems*, vol. 20, pp. 1289–1296, 2007.

[31] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 287–294.

[32] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*, 1998, pp. 92–100.

[33] Y. Zhou and S. Goldman, "Democratic co-learning," in *16th IEEE International Conference on Tools with Artificial Intelligence*, IEEE, 2004, pp. 594–602.

[34] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the nips 2003 feature selection challenge," *Advances in neural information processing systems*, vol. 17, pp. 545–552, 2004.

[35] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: Experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[36] S. Dasgupta, "Analysis of a greedy active learning strategy," *Advances in neural information processing systems*, vol. 17, pp. 337–344, 2005.

[37] X. Zhu, J. Lafferty, and Z. Ghahramani, "Combining active learning and semi-supervised learning using gaussian fields and harmonic functions," in *ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, vol. 3, 2003.

[38] Y. Leng, X. Xu, and G. Qi, "Combining active learning and semi-supervised learning to construct svm classifier," *Knowledge-Based Systems*, vol. 44, pp. 121–131, 2013.

[39] M. F. A. Hady and F. Schwenker, "Combining committee-based semi-supervised learning and active learning," *Journal of Computer Science and Technology*, vol. 25, no. 4, pp. 681–698, 2010.

[40] A. Oliver, A. Odena, C. Raffel, E. D. Cubuk, and I. J. Goodfellow, "Realistic evaluation of semi-supervised learning algorithms.," in *ICLR (Workshop)*, 2018.

[41] J. Bergstra, D. Yamins, D. D. Cox, *et al.*, "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms," in *Proceedings of the 12th Python in science conference*, Citeseer, vol. 13, 2013, p. 20.

[42] R. Fisher, *Iris flower dataset*, 1936.

[43] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision support systems*, vol. 47, no. 4, pp. 547–553, 2009.

[44] T. R. Shultz, D. Mareschal, and W. C. Schmidt, "Modeling cognitive development on balance scale phenomena," *Machine learning*, vol. 16, no. 1, pp. 57–86, 1994.

[45] M. Bohanec and V. Rajkovic, "Knowledge acquisition and explanation for multi-attribute decision making," in *8th Intl Workshop on Expert Systems and their Applications*, 1988, pp. 59–78.

[46] J. Schlimmer, "Mushroom records drawn from the audubon society field guide to north american mushrooms," *GH Lincoff (Pres), New York*, 1981.

[47] F. Alimoglu, E. Alpaydin, and Y. Denizhan, "Combining multiple classifiers for pen-based handwritten digit recognition," 1996.

[48] R. Detrano, A. Janosi, W. Steinbrunn, M. Pfisterer, J.-J. Schmid, S. Sandhu, K. H. Guppy, S. Lee, and V. Froelicher, "International application of a new probability algorithm for the diagnosis of coronary artery disease," *The American journal of cardiology*, vol. 64, no. 5, pp. 304–310, 1989.

[49] C. J. Tucker, D. M. Grant, and J. D. Dykstra, "Nasa's global orthorectified landsat data set," *Photogrammetric Engineering & Remote Sensing*, vol. 70, no. 3, pp. 313–322, 2004.

[50] I. W. Evett and E. J. Spiehler, "Rule induction in forensic science," in *Knowledge Based Systems*, 1989, pp. 152–160.

[51] D. Dua and C. Graff, *UCI machine learning repository*, 2017.

[52] D. H. Deterding, "Speaker normalisation for automatic speech recognition.," Ph.D. dissertation, University of Cambridge, 1990.

[53] G. Demiroz, H. Govenir, and N. Ilter, "Learning differential diagnosis of eryhemato-squamous diseases using voting feature intervals," *Artificial Intelligence in Medicine*, vol. 13, no. 3, pp. 147–165, 1998.

[54] S. Khanna, X. Liu, and J. Zhang, *Iot device identification on corporate networks via adaptive feature set to balance computational complexity and model bias*, U.S. Patent 17139398, Sept. 2020.

[55] Y. Yang and M. Loog, "A variance maximization criterion for active learning," *Pattern Recognition*, vol. 78, pp. 358–370, 2018.

[56] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.