RALF: A Reinforced Active Learning Formulation for Object Class Recognition

Sandra Ebert Mario Fritz Bernt Schiele
Max Planck Institute for Informatics, Saarbrücken, Germany

Abstract

Active learning aims to reduce the amount of labels required for classification. The main difficulty is to find a good trade-off between exploration and exploitation of the labeling process that depends – among other things – on the classification task, the distribution of the data and the employed classification scheme. In this paper, we analyze different sampling criteria including a novel density-based criteria and demonstrate the importance to combine exploration and exploitation sampling criteria. We also show that a time-varying combination of sampling criteria often improves performance. Finally, by formulating the criteria selection as a Markov decision process, we propose a novel feedback-driven framework based on reinforcement learning. Our method does not require prior information on the dataset or the sampling criteria but rather is able to adapt the sampling strategy during the learning process by experience. We evaluate our approach on three challenging object recognition datasets and show superior performance to previous active learning methods.

1. Introduction

Many computer vision algorithms require large quantities of training data to achieve good performance. For example, in object class recognition a significant number of training samples is necessary to capture the intra-class variability and often a strong correlation is postulated between the number of training samples and the final recognition performance. Given these requirements and the complex structure of our target classes, the traditional approach to randomly request labels during training appears insufficient as there is an increasing risk that many draws will result in redundant rather than complementary information. Active learning is a promising research direction to tackle this problem by proposing strategies which label to request next and in turn reducing the total labeling effort.

A popular approach in this area is pool-based active learning [17] that considers the entire data as a pool from which most informative examples given a certain criteria are selected for labeling. Usually, a single criteria is used that limits significantly the performance of active learning also known as the exploitation-exploration dilemma. A pure exploitative (uncertainty-based) criteria leads often to a serious sampling bias as it only focuses on regions that are difficult to learn. This problem is more prominent in a multi-class scenario when some classes are more often requested while other classes are completely overlooked, or on challenging datasets when one class consists of many spatially separated dense regions (e.g., front and side view of a car). In contrast, a pure explorative (density-based) criteria covers the entire data space but needs too many iterations before a good decision boundary is found.

Consequently, methods have been proposed that address this problem by combining different criteria [3, 2, 15]. However, we argue that the proposed combinations are unsatisfactory as they usually combine only two criteria and in practice these criteria are difficult to balance. Additionally, the combination is a fixed one [3, 2] instead of time-varying and it is not obvious how to generalize them to multi-class scenarios. Finally, as we will show empirically in the paper, there is no single, pre-determined combination scheme that does work well across all datasets with differing properties such as varying number of classes, different training set size, and data clustering structure.

However, there is valuable information which has gone largely overlooked in previous active learning approaches: Feedback from the classifier can be used to learn from previous active learning rounds and guide the next label request. In this fashion we learn to trade-off not only between exploration and exploitation but also between different criteria in a time-varying manner. Since we get feedback in each round from the classifier about the label uncertainty it seems a natural choice to use the entire sequence in a reinforcement learning framework to learn the right strategy *during* the labeling process. To the best of our knowledge there is few prior work [1, 15] that has touched upon the idea of incorporating this type of feedback. In addition, previous approaches are limited in the range of possible strategies they can achieve and also face computational challenges. In this work, we consider the active learning process as a Markov decision process (MDP) that gives us the flexibility to handle more than two criteria while simultaneously achieving an adaptive and time-varying trade-off.

To appear in the Proc. of the *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Providence, Rhode Island, June 2012.

Outline. First, we propose a new sampling criteria (Sec. 3) to find representative samples and empirically analyze different sampling criteria (Sec. 5). In Sec. 6, we discuss a common active learning framework that addresses the exploitation-exploration-dilemma and suggest various enhancements for more generality. We explore different combinations of two criteria with fixed and time-varying trade-offs, and show that time-varying trade-offs often work better than fixed trade-offs. Finally, we introduce a reinforced active learning formulation (RALF) in Sec. 7 that deals with different number of criteria as well as different trade-offs and shows improved results across different datasets.

2. Related work

Sampling strategies can be divided into exploitative (uncertainty-based), explorative (density-based), or a combination of both. Most frequently used are exploitative criteria that query for the least certain data point [17]. There are various techniques exploring this direction in the context of SVMs [16, 19], with graph-based label propagation [25, 23], or for object detection [20]. Recent progress was achieved towards multi-class active learning [12, 11, 21]. Despite the success of these methods, they can run into problems by not providing enough coverage of the whole domain or focusing on outliers or inherently ambiguous parts of the data due to their discriminative nature.

In contrast, exploration-driven methods consider the underlying data distribution of the unlabeled data and find more representative samples. There are several algorithms that use clustering to ensure a minimum distance between two requested samples [14, 5]. Another approach uses locally linear reconstructions to find samples that best reconstruct the entire dataset [22]. The drawback of using exploration criteria alone is the missing feedback during the labeling process since their main goal is to sample evenly the data space without looking at the classification uncertainty. Consequently, many label requests are required to converge to a good solution.

There are several approaches that combine exploitation and exploration. In [3], the authors propose a weighted combination of two criteria, [18] switches randomly between uncertainty sampling and random sampling, and [6] construct a bound to modulate a mixing strategy. More recently we have seen a trend towards dynamically balancing this tradeoff by estimating the gain for the next iteration [2, 10] or by using a feedback-driven method [1, 15]. In [1], they use a multi-armed bandit (MAB) formulation to switch among three single criteria and [15] reformulates this work into a simpler but less flexible one. Despite the strong progress of more holistic models, these approaches often come with high computational costs, difficult configurable parameters, unbalanced terms, and missing flexibility in terms of more criteria or time-varying trade-offs.

Our work addresses these issues by proposing a reinforced active learning formulation (RALF) that considers the entire active learning sequence as a process. Our approach can deal with multiple criteria, is able to have time-varying trade-offs between exploration and exploitation, and is fast and efficient due to a compact parameterization of the model without dataset-specific tuning.

3. Sampling criteria

A central part of active learning are the sampling criteria. These criteria rank unlabeled data and request a label for the sample with highest rank. In this section, we discuss two exploitation and three exploration criteria including our new density-based criteria.

3.1. Exploitation

Given $n = l + u$ data points with l labeled examples $L = \{(x_1, \hat{y}_1), ..., (x_l, \hat{y}_l)\}$ and u unlabeled examples $U = \{x_{l+1}, ..., x_n\}$ with $x_i \in \mathbb{R}^d$. We assume c classes and denote $\hat{y} \in \mathcal{L} = \{1, ..., c\}$ the labels.

Entropy (Ent) over the class posterior is the most common criterion for exploitation [1, 12, 15]:

$$Ent(x_i) = -\sum_{j=1}^{c} P(y_{ij}|x_i) \log P(y_{ij}|x_i) \tag{1}$$

where $\sum_j P(y_{ij}|x_i) = 1$ are predictions of a classifier.

Margin (Mar) measures the difference between best versus second best class prediction [12, 17]:

$$Mar(x_i) = P(y_{ik_1}|x_i) - P(y_{ik_2}|x_i) \tag{2}$$

such that $P(y_{ik_1}|x_i) \geq P(y_{ik_2}|x_i) \geq ... \geq P(y_{ik_c}|x_i)$. In each iteration, label $x^* = \text{argmin}_{x_i \in U} Mar(x_i)$ is requested.

Entropy captures the overall uncertainty of one example belonging to a class. In practice, it tends to also sample uninformative ones [12]. In contrast, *margin* focus more on the decision boundaries between two overlapping classes so that the samples are more representative.

3.2. Exploration

Criteria for exploration are mostly used in combination with an exploitation criteria. They usually consider only the overall data distribution and do not get feedback from the classifier during the labeling process. These criteria are computed once at the beginning while exploitation criteria are recomputed after each label. Although this feedback for exploitation is limited as it only considers the current label situation and not the entire sequence it leads often to a slightly better performance when using these criteria alone.

Node potential (Nod) finds dense regions based on a Gaussian weighting function [3]:

$$Nod(x_i) = \sum_{j=1}^{n} e^{-\alpha d^2(x_i, x_j)} \tag{3}$$

with $\alpha = \frac{4}{r_a^2}$, r_a the neighborhood of a node, and Euclidean distance d. After choosing a sample x_i the neighborhood of this image is downweighted with $Nod(x_j) = Nod(x_j) - Nod(x_i)e^{-\beta d(x_i,x_j)^2}$, $\beta = \frac{4}{r_b^2}$. In principle this measure requests samples from dense regions. However, adjusting the parameters can be difficult. When the neighborhood is too small multiple samples are drawn from the same dense region, whereas when the neighborhood is too large the approach tends to sample outliers. We use the suggested setting of [3], e.g., $r_a = 0.4$ and $r_b = 1.25r_a$.

Kernel farthest first (Ker) searches for most unexplored regions given the set of already labeled data [1, 15]. First, it computes the minimum distance of an unlabeled sample to all labeled data

$$Ker(x_i) = \min_{x_j \in L} d(x_i, x_j), \tag{4}$$

with Euclidean distance d. Then, it requests the label for the farthest sample $x^* = \text{argmax}_i Ker(x_i)$. This criteria works well for datasets with a smooth manifold structure as it samples evenly the entire data space. As soon as there are more complex datasets, this measure selects many outliers.

Graph density (Gra) is our novel sampling criteria that uses a graph structure to identify highly connected nodes. We build a k-nearest neighbor graph with $\hat{P}_{ij} = 1$ if $d(x_i, x_j)$ is one of the k smallest distances of x_i with Manhattan distance d and $k = 10$ for the number of nearest neighbors. This graph is symmetric, i.e., $P_{ij} = \max(\hat{P}_{ij}, \hat{P}_{ji})$, and weighted with a Gaussian kernel

$$W_{ij} = P_{ij} \exp\left(\frac{-d(x_i, x_j)}{2\sigma^2}\right). \tag{5}$$

This weight matrix is used to rank all data points according to their representativeness. The intuition behind this is that representative data points for one class are usually well embedded in this graph structure and thus have many edges ($\gg k$) with high weights. To distinguish among data points with many small weighted neighbors, we normalize these weights by the number of edges:

$$Gra(x_i) = \frac{\sum_i W_{ij}}{\sum_i P_{ij}}. \tag{6}$$

Similar to the *node potential*, we reduce the weights of direct neighbors of the currently selected node x_i with $Gra(x_j) = Gra(x_j) - Gra(x_i)P_{ij}$. This avoids that the same dense region is selected multiple times. Our criteria focuses on representative regions due to the normalization factor, it avoids sampling of outliers, and is more robust than *node potential* due to the underlying k-NN graph structure.

4. Experimental setup

In this section, we briefly describe the datasets we used in this work. After that, we explain the classifier that are applied in this active learning framework. We use a semi-supervised label propagation algorithm as well as a supervised SVM.

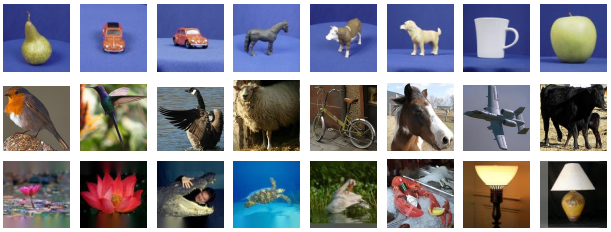

Figure 1. ETH (first row), C-PASCAL (second row), and Caltech 101 (third row)

Datasets and representation We analyze three datasets with increasing number of object classes and different difficulty. Example images are shown in Fig. 1.

ETH-80 (ETH) [13] contains 3,280 images divided in 8 object classes with 10 instances per class. Each instance is photographed from 41 viewpoints in front of a uniform background.

We introduced Cropped PASCAL (C-PASCAL) in [7]. Bounding box annotations of the PASCAL VOC challenge 2008 training set [8] are used to extract the objects such that classification can be evaluated in a multi-class setting. The resulting data set contains 4,450 images of aligned objects from 20 classes but with varying object poses, challenging appearances, background clutter, and truncation.

Caltech 101 [9] is a dataset with 9,144 images and 102 object classes. Objects are mostly centered but there is still background clutter and a large intra-class variability. For the data representation of all datasets, we use a HOG [4] representation.

Label propagation (LP). We follow the method by [24]. Based on a k nearest neighbor graph (Eq. 5) a normalized graph Laplacian $S = D^{-1/2}WD^{-1/2}$ with degree matrix $D_{ii} = \sum_j W_{ij}$ is computed. For learning, we split our multi-class problem into c binary problems and get a prediction matrix $Y^* \in \mathbb{R}^{n \times c}$ that is composed of c vectors one for each class computed by an iterative procedure,

$$Y_m^{(t+1)} = \alpha S Y_m^{(t)} + (1 - \alpha) Y_m^{(0)}, \tag{7}$$

with $1 \leq m \leq c$ and Y_m^* the limit of this sequence.

The initial label vector is set as follows $Y_m^{(0)} = (y_1^m, ..., y_l^m, 0, ..., 0)$ with $y_i^m \in \{-1, 1\}$. Parameter $\alpha \in (0, 1]$ controls the influence of the original labels. Finally, the prediction of the data $\hat{Y} \in \mathcal{L}$ is obtained by $\hat{Y} = \text{argmax}_{1 \leq m \leq c} Y_m^*$.

SVM. We use libSVM to calculate results for a SVM with RBF kernel. Parameters are empirically determined ($C = 100$, and $\gamma = 1$). Finally, we use probability estimate to make the predictions comparable in our multi-class setting similar to [12].

	ETH		C-PASCAL		Caltech 101		mean	
criteria	SVM	LP	SVM	LP	SVM	LP	SVM	LP
random	74.3	74.8	26.5	27.7	30.4	33.4	43.7	45.3
exploitation								
Ent	79.8	81.6	26.7	28.1	28.7	33.5	45.1	47.8
Mar	78.8	**81.7**	27.6	**30.2**	29.5	34.4	45.3	48.8
exploration								
Nod	57.3	66.2	17.1	18.0	17.8	23.8	30.7	36.0
Ker	74.6	71.0	21.2	22.6	27.3	29.3	41.0	41.0
Gra	66.8	71.8	28.7	29.9	35.5	**38.9**	43.6	46.9

Table 1. Overall accuracy for all single criteria and random sampling after $\max(5c, 100)$ iterations.

5. Analysis of single criteria

In this section, we show results of single criteria. We compare to a random baseline where samples are drawn with a uniform distribution. Overall accuracy after $\max(5c, 100)$ iterations with c number of classes are shown in Tab. 1 for all three datasets. The lower bound of 100 ensures for each dataset a minimum of labels. In all experiments, we start with one randomly drawn label per class. We document results for label propagation (LP) as well as SVM. Finally, we average over all datasets shown in the last two columns.

We make the following observations: (1) LP is always better than SVM due to the small amount of labels. (2) The ordering of criteria according to the performance is the same for both SVM and LP.[1] When comparing the different criteria *Gra* works best for Caltech 101 with 35.5% (SVM) and 38.9% (LP). In average, *Mar* has best performance with 45.3% (SVM) and 48.8% (LP). (3) Differences between SVM and LP are larger the better the underlying criteria. For example, the difference for random sampling between SVM with 43.7% and LP with 45.3% is only 1.6% while for the best criteria *Mar* this difference increase to 3.5% between SVM and LP. This illustrates the potential of SSL if the commonly used random sampling is replaced by a more appropriate choice.

(4) In average, exploitation criteria work better than exploration criteria due to the local feedback after each labeling iteration. But given this drawback, our *Gra* works remarkably well. The more difficult the dataset the larger the benefit from this criteria. For Caltech 101, our criteria is with 38.9% even better than *Mar* with 34.4%.

(5) Finally, there is no criteria that works best across all datasets. *Mar* that has in average best performance over all datasets shows indeed best performance for ETH and C-PASCAL but it lose almost 5% for Caltech 101. To conclude this section, all of the criteria have their strengths and weaknesses and it is hard to choose one criteria that works consistently best for all datasets.

[1] As this holds true for this and all subsequent experiments of this paper we will report results on LP only for the remainder of this paper.

6. Fixed and time-varying combination

In this section, we analyze different combinations of exploration and exploitation criteria. We first explain our framework that allows a fixed as well as a time-varying trade-off between exploration and exploitation. Finally, we show in the experiments that there is a consistent improvement compared to the previous section.

Method. Our framework is inspired by [3] that combines exploration and exploitation with a parameter β:

$$H(x_i) = \beta U(x_i) + (1-\beta)D(x_i) \tag{8}$$

where $U \in \{Ent, Mar\}$, $D \in \{Nod, Ker, Gra\}$, and $\beta \in [0, 1]$. In addition, we introduce two new improvements. First, we use a ranking function $r : \mathbb{R} \to \{1, ..., u\}$

$$r(F(x_i)) = m_i, \text{ where } F(x_i) \leq F(x_j) \Leftrightarrow m_i \geq m_j \tag{9}$$

with $m \in \{1, ..., u\}$ for all $l+1 \leq i, j \leq n$ and $F \in \{-Ent, -Mar, Nod, Ker, Gra\}$. This maps the continuous values of D and U to a natural number and makes both terms more comparable among each other and across all datasets. Otherwise the range of values of D and U is strongly dependent on the given dataset and requires a non-trivial adjustment of β.

Our second improvement replaces the fixed β by a sequence over time, i.e., $\beta(t) : \{1, ...T\} \to [0, 1]$ with T the maximal number of queried labels. This allows us to have a constant trade-off as well as a time-varying trade-off. The main idea is that some datasets might need more exploration at the beginning and more exploitation at the end while other datasets might need a constant trade-off.

The final active learning framework is of the following form:

$$H(x_i) = \beta(t) r(U(x_i)) + (1-\beta(t)) r(D(x_i)) \tag{10}$$

In each iteration we request the label for the sample with the minimal score $\text{argmin}_{x_i \in U} H(x_i)$.

Experiments. We investigate different forms of $\beta(t)$ ranging from constant to concave and convex shapes. Similar to the previous section we document overall accuracy for $\max(5c, 100)$ samples per dataset. Fig. 2 shows all combinations with different constant functions from $\beta(t) = 0$ (pure exploration) to $\beta(t) = 1$ (pure exploitation). For the time-varying function, we use $log(t)$ and t that represents the strategy of exploration at the beginning followed by more exploitation, and $-log(t)$ and $-t$ as the complement. These values are rescaled such that $\beta(t) \in [0, 1]$. Tab. 2 contains all results. As fixed combination, we show results for $\beta(t) = 0.5$ that in average is among the best-performing fixed combinations (see Fig. 2). Finally, we compute the average over all datasets shown in Tab. 3.

We observe: (1) Combinations consistently improve single criteria. This can be seen in Fig. 2 where best values of

	ETH					C-PASCAL					Caltech 101				
	fixed	time-varying				fixed	time-varying				fixed	time-varying			
combination	0.5	log(t)	t	-log(t)	-t	0.5	log(t)	t	-log(t)	-t	0.5	log(t)	t	-log(t)	-t
Mar+Nod	81.5	**82.9**	82.5	78.5	81.0	27.0	**29.3**	28.7	25.3	27.9	29.5	**32.0**	28.8	25.8	29.1
Mar+KFF	78.9	**81.0**	80.6	74.2	77.3	26.9	**29.8**	26.3	26.5	28.6	34.4	**35.2**	34.9	30.6	33.6
Mar+Gra	**83.9**	83.1	83.4	77.6	82.5	30.1	33.6	34.5	34.6	**36.2**	39.5	37.9	**39.7**	39.5	39.1
Ent+Nod	**82.5**	81.5	82.0	79.0	81.0	26.3	**31.5**	30.8	26.7	30.3	33.5	34.2	**35.0**	26.7	32.8
Ent+Ker	78.7	**82.3**	81.6	75.1	78.0	23.6	**27.3**	25.9	24.8	25.3	33.2	32.4	**33.3**	30.4	32.2
Ent+Gra	**83.0**	81.8	82.3	78.2	82.3	31.4	34.9	35.5	34.4	**36.6**	39.8	36.0	**39.9**	39.7	39.1

Table 2. Overall accuracy for $\beta(t) = 0.5$ and four different time-varying combinations.

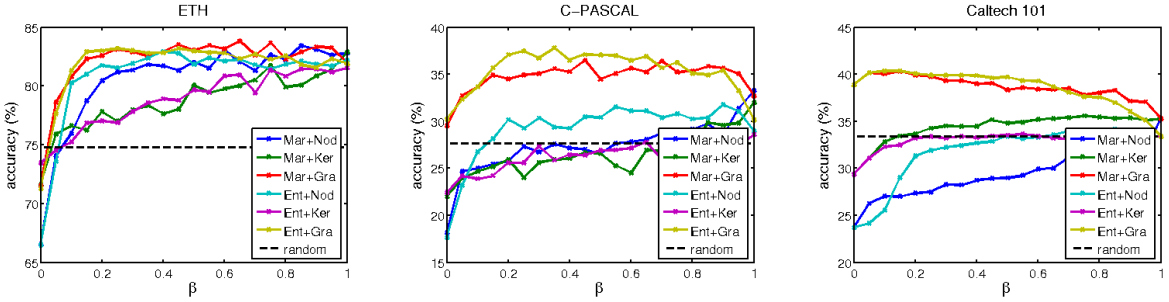

Figure 2. Simple mixtures with different constant β for all datasets and the comparison to random sampling.

	mean over all datasets				
	fixed	time-varying			
combination	0.5	log(t)	t	-log(t)	-t
Mar+Nod	46.0	**48.0**	46.7	43.2	46.0
Mar+KFF	46.7	**48.7**	47.2	43.8	46.5
Mar+Gra	51.1	51.5	52.5	50.5	**52.6**
Ent+Nod	47.4	49.1	**49.3**	44.2	48.0
Ent+Ker	45.2	**47.4**	46.9	43.4	45.2
Ent+Gra	51.4	50.9	52.6	50.8	**52.7**

Table 3. Overall accuracy for $\beta(t) = 0.5$ and four different time-varying combinations.

solid lines are usually in the range of $0 < \beta(t) < 1$. (2) Almost all combinations are better than random sampling. For C-PASCAL, we improve LP with random sampling from 27.7% to 36.6% with *Ent+Gra*. (3) Our *Gra* criteria works always best for all datasets in combination either with *Ent* (red curve) or with *Mar* (yellow curve). This improvement is more pronounced the more difficult the dataset. (4) A time-varying scheme is across all datasets (Tab. 3) better than a fixed combination. For *Mar+Gra*, we improve the fixed combination of $\beta(t) = 0.5$ from 51.1% to 52.5% when using $\beta(t) = t$. (5) Different datasets need a different trade-off (see Fig. 2). While all curves for ETH have a tendency to increase toward $\beta(t) = 1$, the two top-curves of Caltech 101 are decreasing, i.e., need more exploration than exploitation. In average over all datasets *Ent+Gra* with $\beta(t) = -t$ work best with 52.7%. But when we look at each specific dataset this only holds true for C-PASCAL. ETH shows best performance with *Mar+Gra* and $\beta(t) = 0.5$ and Caltech 101 with *Ent+Gra* and $\beta(t) = t$.

Again, there is no single strategy that works best across all datasets. Consequently, we propose a new dataset-specific method in the next section that allows time-varying combinations as well as different combinations of criteria.

7. Adaptive strategies for active learning

A combination of two criteria as well as a time-varying trade-off between exploration and exploitation are key ingredients to improve active learning. But there is no single strategy that works best across all datasets. The question arises how can we find a good trade-off and combination of criteria without prior knowledge on the dataset and its interplay with the criteria? To address this challenge, we consider the entire active learning sequence as a process that is optimized by learning a strategy from feedback "on the fly". This constitutes a challenging meta-learning problem only allowing for indirect and approximative observations. In the following, we investigate different proxies for the true gain in performance in each stage and how they can be used to guide the next label query. A crucial difference to the previous methods is that we now aim at modeling the progress of the learnt classifier and exploit this information to control the trade-off between different individual criteria. In the second part we propose a model to aggregate feedback over time and learn an effective strategy from experience over multiple active learning rounds. Hence, we phrase the problem as learning to perform active learning. The integration of multiple criteria which are combined in this flexible and adaptive fashion shows excellent performance across 3 challenging datasets without any need to inform the method about the specifics of the dataset or the available criteria.

7.1. Feedback by approximative performance

Inspired by [15], we explore feedback after each labeling iteration to update a probability

$$p = \max(\min(p\lambda \exp(r^{(t)}), 1-\epsilon), \epsilon) \tag{11}$$

that is used to switch randomly between exploration and exploitation with reward $r^{(t)}$ and $p \in [\epsilon, 1-\epsilon]$. ϵ ensures minimal exploration (resp. exploitation). The higher p the higher the probability that exploration is selected for the next iteration. Parameter λ is the learning rate that controls the influence of the reward. Feedback $r^{(t)}$ is given by the change of the previous hypothesis $Y^{(t-1)}$ (Eq. 7) to the current hypothesis $Y^{(t)}$

$$r^{(t)} = \frac{\langle Y^{(t-1)}, Y^{(t)} \rangle}{\|Y^{(t-1)}\| \|Y^{(t)}\|}. \tag{12}$$

This feedback is rescaled with $r^{(t)} \leftarrow 3 - 4r^{(t)}$ otherwise exploration dominates exploitation.

Our first improvement integrates p in our active learning framework that means $\beta(t) = 1-p$ so that there is always a mixture of two criteria. Second, we propose a more general rescaling function $s : \mathbb{R} \mapsto I = [-1, 1]$ as the previous mentioned rescaling turns out to not generalize well to new datasets. Instead, we consider all observed rewards until iteration t to map these values into an interval I,

$$s(r^t) = \tilde{r}^{(t)} \frac{\max(I) - \min(I)}{\max_i r^{(i)} - \min_i r^{(i)}} - \min(I) \tag{13}$$

with $1 \leq i \leq t$ and $\tilde{r}^{(t)} = r^{(t)} - \min_i r^{(i)}$.

Third, we propose a new reward function $r^{(t)}$ that is more closely related to the actual performance of the classifier compared to the mere change in the prediction. While classification accuracy on the whole dataset is obviously not available during learning, we propose to use the difference in the overall entropy of the class posteriors between two time steps as a proxy for measuring the learning progress,

$$r^{(t)}_{Ent} = \sum_{i=1}^{u} Ent^{(t-1)}(x_i) - \sum_{i=1}^{u} Ent^{(t)}(x_i). \tag{14}$$

with $Ent^{(t)}(x_i)$ the entropy of unlabeled sample x_i at iteration t. This reward is rescaled with our function s from Eq. 13 to get positive as well as negative feedback.

7.2. Reinforced active learning formulation (RALF)

The previous method proposes a first way to incorporate feedback. But there is no learning involved yet. Therefore we suggest a method that accumulates feedback over time and is also capable to deal with more than one criteria.

We address this problem by formulating active learning as a Markov decision process (MDP). Fig. 3 shows on the left side a simple MDP for two criteria. In this

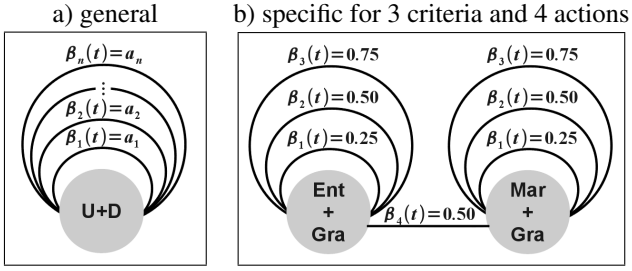

Figure 3. Simple Markov decision process with a) 1 state $S = \{U + D\}$ with $U \in \{Mar, Ent\}$ and $D \in \{Nod, Ker, Gra\}$, and n actions $A = \{\beta_i(t) = a_i\}$ with $a_i \in [0, 1]$; b) 2 states $S = \{Ent + Gra, Mar + Gra\}$ and 4 actions $A = \{0.25(stay), 0.5(stay), 0.75(stay), 0.5(switch)\}$

MDP denoted by a 4-tuple (S, A, Q, R), there is only one state $S = \{U + D\}$ with $U \in \{Mar, Ent\}$ and $D \in \{Nod, Ker, Gra\}$ a mixture of two sampling criteria. Further, there are n actions that represent n different fixed trade-offs, i.e., $A = \{\beta_1(t) = a_1, \beta_2(t) = a_2, ..., \beta_n(t) = a_n\}$ with $a_i \in [0, 1]$. Note, although actions have a fixed $\beta(t)$ this does not contradict our previous assumption of using a time-varying trade-off because we are always able to switch among different actions. R is the reward for executing action a_i in state s_j. We use the overall entropy from Eq. 13. Finally, Q are the transition weights that action a_i is selected in state s_j. Even though each state consists of a mixture of two criteria it is still possible to have single criteria by choosing action $\beta_i(t) = 0$ or $\beta_i(t) = 1$.

This simple MDP can be naturally extended to a larger state space with more sampling criteria. On the right side of Fig. 3 there is an example for three criteria, i.e., $U \in \{Ent, Mar\}$, $D \in \{Gra\}$, and three different mixtures. Additional to the three actions of changing the trade-off there are now m actions for each new state to switch the state with m different trade-offs.

To learn this MDP, we use the *model-free* method Q-Learning as we have no prior knowledge about the underlying model. Q-Learning is a fast and adaptive reinforcement learning algorithm that learns our transition table $Q \in \mathbb{R}^{|S| \times |A|}$ online during the active learning process. After each transition $s^{(t-1)} \rightarrow a \rightarrow s^{(t)}$ entry $Q(s^{(t-1)}, a)$ is updated given the current reward $r^{(t)}$, i.e,

$$\begin{aligned} Q(s^{(t-1)}, a) \leftarrow\ & Q(s^{(t-1)}, a) + \lambda(r^{(t)} + \\ & \gamma \max_{a_i} Q(s^{(t)}, a_i) - Q(s^{(t-1)}, a)). \end{aligned} \tag{15}$$

Parameter λ is the learning rate that controls the influence of the current reward $r^{(t)}$, and $0 \leq \gamma \leq 1$ is the discount factor that weights the future reward. When $\gamma = 0$ only the current reward $r^{(t)}$ is considered for updating and any previous experience with this state-action-pair are ignored. During the active learning process, we decide for action $a = \max_{a_i} Q(s^{(t-1)}, a_i)$ and use mixture of state $s^{(t)}$ with

trade-off a to request the next label.

In summary, our model has two parameters for Q-Learning that are obtained from previous reinforcement learning papers. In addition, there are the number of states and actions that should be kept as small as possible to speed up the initialization. All parameters are the same across all datasets. There is no tuning to one specific dataset.

Initialization. One challenge we face is initialization of the method as we start with an empty Q table. Ideally, we visit each state-action-pair once or twice but this is intractable for a large state and action space. The number of iterations are limited and we would try out many transitions that are harmful for our learning process.

Therefore, we propose a guided initialization phase inspired by [21]. We compute the expected entropy reduction $\hat{r}_i^{(t)}$ for all actions a_i. Each action a_i requests a label for sample x_i. As we do not know the label for this sample, we apply our classifier for each class and calculate the overall entropy. These entropies are weighted by our current prediction probability $p(y_{ij}|x_i)$, i.e.,

$$\hat{r}_i^{(t)} = \sum_{j=1}^{c} p(y_{ij}|x_i) \sum_{k=1}^{n} Ent_j(x_k). \tag{16}$$

Ent_j is the entropy after running our classifier with label j for sample x_i. Finally, we select the next action with $a = \text{argmax}_i \hat{r}_i^{(t)}$. Of course, this is a time-consuming step but we use this only for the first few iterations. Also, we can reduce the number of classes for estimation with threshold $p(y_{ij}|x_i) > 0.01$. Usually, there are only 2 to 4 classes left.

We set $\epsilon = 0.05$ and we fix $\gamma = 1$ as we want as much as possible benefit from our previous experience. Finally, we want a time-varying also called *non-stationary* model. So we set $\lambda = 0.5$ as otherwise it converges to a fixed solution and later changes are almost impossible.

	ETH				**C-PASCAL**			
combination	[15]	$s(r^t)$	$r_{Ent}^{(t)}$	RALF	[15]	$s(r^t)$	$r_{Ent}^{(t)}$	RALF
Mar+Nod	81.4	82.9	**83.2**	81.8	31.3	32.5	32.1	31.7
Mar+KFF	81.5	81.2	**82.8**	80.0	31.2	**33.2**	30.5	31.6
Mar+Gra	82.0	83.2	83.6	**83.8**	32.1	32.8	34.2	**36.5**
Ent+Nod	80.9	82.1	81.6	**82.5**	27.6	30.0	29.4	**31.2**
Ent+Ker	81.5	81.9	81.9	82.1	27.8	29.9	**30.1**	29.8
Ent+Gra	81.5	81.8	82.3	**83.6**	28.4	31.9	33.7	**37.3**
	Caltech 101				**mean over all datasets**			
combination	[15]	$s(r^t)$	$r_{Ent}^{(t)}$	RALF	[15]	$s(r^t)$	$r_{Ent}^{(t)}$	RALF
Mar+Nod	35.1	**35.8**	35.4	30.5	49.3	**50.4**	50.2	48.0
Mar+KFF	34.6	**35.8**	35.4	35.1	49.1	**49.9**	49.6	48.9
Mar+Gra	35.0	35.4	35.9	**39.8**	49.7	50.5	51.2	**53.4**
Ent+Nod	33.0	33.1	33.6	**33.9**	47.1	48.4	48.2	**49.2**
Ent+Ker	33.4	33.4	**33.6**	33.1	47.6	48.4	**48.5**	48.3
Ent+Gra	33.6	33.7	33.8	**40.2**	47.9	49.1	49.9	**53.7**

Table 4. Accuracy for [15], our rescaling function (Eq. 13), the entropy-based reward (Eq. 14), and our MDP-based method.

Experiments. In Tab. 4, we show results for all feedback-driven methods with two criteria. The first column of each block shows performance of the method by [15]. The second and third columns contain our improvements for this method from Eq. 13 and 14. The last column shows our MDP method with one state $S = \{U + D\}$ for each combination and three different trade-offs $A = \{\beta(t) = 0.25, \beta(t) = 0.5, \beta(t) = 0.75\}$. As before, we document overall accuracy after $T = \max(100, 5c)$ iterations, and start with one label per class.

All our proposed methods outperform [15]. Moreover, for our best combination *Ent+Gra*, there is a consistent increase in performance from the first column to the last column across dataset. C-PASCAL, e.g., get a performance of 28.4% with [15]. It is then increased to 31.9% with our general scaling function $s(r^{(t)})$, and to 33.7% with our entropy-based reward function. Finally, we improve up to 37.3% when using our MDP-based method that outperforms the best time-varying combination from Tab. 2 $\beta(t) = -t$ with 36.6%. This observation also holds true for the mean over all datasets where we increase *Ent+Gra* from 47.9% to 53.7%.

In the last part of this section, we demonstrate the flexibility of our MDP-based model. In Tab. 5, we add consecutively states to our model starting with 2 states $S = \{Ent + Gra\}$ and ending with 4 states $S = \{Ent + Gra, Mar + Gra, Ent + Ker, Mar + Ker\}$, i.e, 4 criteria. In addition, we compare our results to the baseline of randomly switch between all state-action-pairs to show that our model goes beyond this baseline.

		ETH			C-PASCAL			Caltech 101		
$\|S\|$	criteria	rand	QL	diff	rand	QL	diff	rand	QL	diff
2	Ent, Gra	82.2	83.6	+1.5	36.2	37.3	+1.1	36.0	40.2	+3.7
3	+Mar	81.5	83.2	+1.7	35.7	36.7	+1.0	35.2	38.3	+3.1
4	+Ker	81.7	82.9	+1.2	34.1	36.2	+2.1	34.3	36.3	+2.1

Table 5. Accuracy for our MDP-based approach with 2 to 4 states compared to randomly switching among those action-state-pairs, and the difference to the later one.

We observe a slight decrease in performance due to the larger number of states. Both initialization and the time-varying trade-off are more difficult to learn. Nevertheless, all results are better than the random-state-transition baseline. This illustrates once more that our model benefits from the accumulated knowledge represented by the Q table. After a short initialization, our algorithm makes use of the collected experience so far, and picks a good state-action pair given the current Q table. For the $|S| = 4$ where we use 4 different sampling criteria, our algorithm favors after only a few iterations either *Mar+Gra* or *Ent+Gra* that is in agreement with our results from the previous section.

8. Conclusion

In this work, we model active learning as a feedback-driven Markov decision process that can change over time and find a successful strategy for each individual dataset.

	ETH		C-PASCAL		CALTECH		mean	
strategy	LP	diff	LP	diff	LP	diff	LP	diff
random	74.8		27.7		33.4		45.3	
single criteria								
margin	81.7	+6.9	30.2	+2.5	34.4	+1.0	48.8	+3.5
graph density	71.8	-3.0	29.9	+2.2	38.9	+5.5	46.9	+1.6
fixed and time-varying trade-off								
$\beta(t) = 0.5$	83.0	+8.2	31.4	+3.7	39.8	+6.4	51.4	+6.1
$\beta(t) = -t$	82.3	+7.5	36.6	+8.9	39.1	+5.7	52.7	+5.7
$\beta(t) = t$	82.3	+7.5	35.5	+7.8	39.9	+6.5	52.6	+7.3
Feedback-driven								
our RALF	**83.6**	+8.8	**37.3**	+9.6	**40.2**	+6.8	**53.7**	+8.4

Table 6. Summary: Random sampling, best single exploitation and exploration criteria, best combination with fixed and time-varying trade-off, our RALF approach, and differences to random sampling.

This proposed model is based on our findings from the first part of this paper where we analyze different sampling criteria as well as different combinations of exploration and exploitation. We argue that different datasets need different sampling strategies in a time-varying manner.

In Tab. 6, we summarize the main findings of this paper. The first row contains results for random sampling when selecting samples with a uniform distribution. In all following lines, we calculate the difference to these numbers (column *diff*). The next two rows show best single criteria for exploitation, i.e., *margin* and exploration, i.e., *graph density* our novel criteria that works best across all datasets. Almost all these numbers are better than random sampling. In average, exploitation works slightly better than exploration due to the local feedback after each labeling iteration.

Below, we list three different fixed and time-varying trade-offs that work best across all datasets. As can be seen, time-varying strategies are better than fixed strategies. Surprisingly, not the common sense strategy $\beta(t) = t$ with a short exploration at the beginning and a long exploitation at the end is the best time-varying trade-off but rather the opposite strategy with $\beta(t) = -t$ in particular for C-PASCAL. In the last line, we show results of our MDP-based method that outperforms all previous methods and leads to a final improvement of 9.6% for C-PASCAL and in average to 8.4% across all datasets. This underlines the capabilities of our model to adapt to different dataset and learn an effective active learning strategy "on the fly".

For future work, we intend a faster initialization of this model by incorporating domain knowledge or other prior knowledge as we observe even better performance when using a previous learned transition table.

References

[1] Y. Baram, R. El-yaniv, and K. Luz. Online Choice of Active Learning Algorithms. *JMLR*, 5:255–291, 2004. 1, 2, 3

[2] A. Bondu, V. Lemaire, and M. Boullé. Exploration vs. exploitation in active learning: a Bayesian approach. In *IJCNN*, 2010. 1, 2

[3] N. Cebron and M. R. Berthold. Active learning for object classification: from exploration to exploitation. *DMKD*, 18(2):283–299, 2009. 1, 2, 3, 4

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 3

[5] S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *ICML*, 2008. 2

[6] P. Donmez and J. Carbonell. Dual strategy active learning. In *ECML*, 2007. 2

[7] S. Ebert, D. Larlus, and B. Schiele. Extracting Structures in Image Collections for Object Recognition. In *ECCV*, 2010. 3

[8] M. Everingham, L. Van Gool, and C. K. Williams. The PASCAL VOC, 2008. 3

[9] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *TPAMI*, 28(4):594–611, 2006. 3

[10] S. Huang, R. Jin, and Z. Zhou. Active Learning by Querying Informative and Representative Examples. In *NIPS*, 2010. 2

[11] P. Jain and A. Kapoor. Active learning for large multi-class problems. In *CVPR*, 2009. 2

[12] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In *CVPR*, 2009. 2, 3

[13] B. Leibe and B. Schiele. Analyzing Appearance and Contour Based Methods for Object Categorization. In *CVPR*, 2003. 3

[14] H. T. Nguyen and A. Smeulders. Active learning using pre-clustering. In *ICML*, 2004. 2

[15] T. Osugi and S. Scott. Balancing Exploration and Exploitation: A New Algorithm for Active Machine Learning. In *ICDM*, 2005. 1, 2, 3, 6, 7

[16] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. *ICML*, 2000. 2

[17] B. Settles and M. Craven. An analysis of active learning strategies for sequence labeling tasks. In *EMNLP*, 2008. 1, 2

[18] S. B. Thrun and K. Moeller. Active Exploration in Dynamic Environments. In *NIPS*, 1992. 2

[19] S. Tong and D. Koller. Support Vector Machine Active Learning with Applications to Text Classification. *JMLR*, pages 45–66, 2001. 2

[20] S. Vijayanarasimhan and K. Grauman. Large-Scale Live Active Learning : Training Object Detectors with Crawled Data and Crowds. In *CVPR*, 2011. 2

[21] R. Yan, J. Yang, and A. Hauptmann. Automatically labeling video data using multi-class active learning. In *ICCV*, 2003. 2, 7

[22] L. Zhang, C. Chen, J. Bu, D. Cai, X. He, and T. S. Huang. Active Learning based on Locally Linear Reconstruction. *PAMI*, 6(1), 2007. 2

[23] B. Zhao, F. Wang, C. Zhang, and Y. Song. Active model selection for graph-based semi-supervised learning. In *ICASSP*, 2008. 2

[24] D. Zhou, O. Bousquet, and T. Lal. Learning with local and global consistency. In *NIPS*, 2004. 3

[25] X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions. In *ICML, WS*, 2003. 2