TAAL: Target-Aware Active Learning

Kunal Kotian[*]
kkotian@amazon.com

Indranil Bhattacharya[*]
bindrani@amazon.com

Shikhar Gupta[*]
gupshik@amazon.com

Kaushik Pavani
sripava@amazon.com

Naval Bhandari
naval@amazon.com

Sunny Dasgupta
sunnyd@amazon.com

Abstract

Pool-based active learning techniques have had success producing multi-class classifiers that achieve high accuracy with fewer labels compared to random labeling. However, in an industrial setting where we often have class-level business targets to achieve (e.g., 95% recall at 95% precision for each class), active learning techniques continue to acquire labels for classes that have already met their targets, thus consuming unnecessary manual annotations. We address this problem by proposing a framework called Target-Aware Active Learning that converts any active learning query strategy into its target-aware variant by leveraging the gap between each class' current estimated accuracy and its corresponding business target. We show empirically that target-aware variants of state-of-the-art active learning techniques achieve business targets faster on 2 open-source image classification datasets and 2 proprietary product classification datasets.

1 Introduction

Active learning is a popular approach used to reduce the manual labeling effort required to train a classifier. In active learning, we iteratively acquire labels from annotators and use them to (re)-train a classifier. Previous research (Lewis and Gale, 1994; Settles, 2009; Gal et al., 2017; Lin and Parikh, 2017) has demonstrated that choosing a batch of instances with small batch sizes offers a good trade-off between user interactivity and number of labels required to create a classifier. In each active learning iteration, we perform two operations: (i) use a *query strategy* to judiciously select a fixed-sized subset (a batch) of unlabeled instances and send them to the annotators, and, (ii) train a classifier using new and previously labeled instances. To pick the next set of unlabeled instances for annotation, we rank all instances in unlabeled data based on scores such as margin, entropy, expected loss reduction, or sub-modular information measures like mutual information, conditional gain *etc.*, and select the instances which are most likely to benefit the classifier.

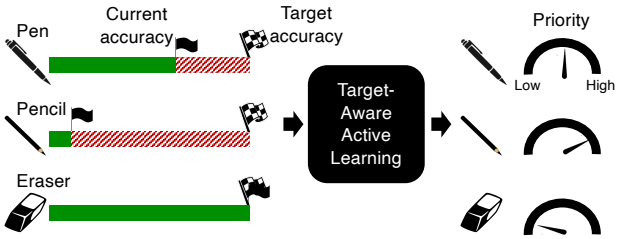

Figure 1: TAAL computes relative priority weights for each class taking into account its current estimated accuracy (solid flag) and target accuracy (checkered flag). Here, the expected priority is *pencil* > *pen* > *eraser*.

Imagine using the active learning paradigm to improve a 3-class classifier with 4 labels: *pens*, *pencils*, *erasers*, and *not-in-k* (*i.e.*, the background class which does not contain pens, pencils, or erasers). In an industrial setting, annotators often work backwards from achieving class-level business-specified accuracy targets, where *accuracy* refers to any measure of a classifier's ability to discriminate between classes, *e.g.*, precision, recall, classification accuracy, false positive rate. *E.g.*, the task could be to achieve 90% recall at 85% precision for each class-of-interest. Vanilla active learning algorithms are sub-optimal for such applications because they ignore business targets. In other words, they continue to select unlabeled data that improves the accuracy of a class even if its estimated accuracy exceeds the business targets; this annotation budget could instead be used to improve other classes that are yet to meet their targets. To this end, we propose a framework **T**arget-**A**ware **A**ctive **L**earning (TAAL) that can be applied on top of any active learning strategy to create its target-aware variant. If the classification task has class-level business targets defined, TAAL increases the likelihood of achieving the targets on all classes-of-

[*]These authors contributed equally to this work

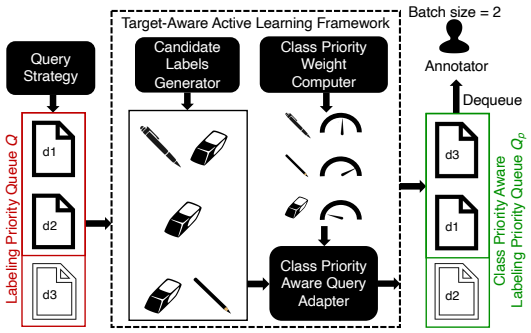

Figure 2: TAAL transforms a queue of unlabeled instances Q (left) generated by an arbitrary query strategy, into queue Q_p (right) by prioritizing classes based on the relative gaps between their estimated accuracy and business target. TAAL uses a *Class Priority Weight Computer* to quantify the attention each class needs. *Candidate Labels Generator* identifies classes that are likely to see accuracy improvement if an unlabeled instance is labeled. *Class Priority Aware Query Adapter* reorders Q into Q_p by prioritizing instances that are likely to improve accuracy over classes that need attention. With a labeling budget of 2 documents, vanilla query strategies would have surfaced documents d_1 and d_2 to the annotator. TAAL prioritizes d_3 in Q_p as d_3 will likely improve the *pencil* class which requires the most attention.

interest given a fixed labeling budget, by leveraging the gap between each class' target and estimated accuracy. Note that TAAL will not help achieve global business targets (*e.g.* micro-recall) faster than baseline query strategies.

2 Target-Aware Active Learning

Consider an active learning setup employing an arbitrary query strategy that generates a query Q, *i.e.*, a queue of instances to be labeled. We make this query strategy *target-aware* under the TAAL framework shown in Fig. 2. We first quantify the relative attention each class requires to achieve its accuracy targets; this is handled by the *Class Priority Weight Computer* (Sec. 2.1). We then use a *Class Priority Aware Query Adapter* (Sec. 2.3) to select a subset of instances from Q and reorder them into a queue Q_p prioritizing classes that require the most attention. This process necessitates identification of the subset of classes whose accuracy is likely to improve if a given instance is labeled. This is accomplished by the *Candidate Labels Generator* (Sec. 2.2). Thus, the TAAL framework can be applied to an arbitrary query strategy to convert its output query Q into a target-aware version Q_p. The following sections discuss the three components of TAAL.

2.1 Class Priority Weight Computer

Let $Y = \{y_1, y_2, ..., y_k\}$ be the set of classes in a multi-class classifier. In TAAL, at any active learning iteration ℓ, a Class Priority Weight Computer quantifies the 'priority weights' (*i.e.*, relative attention) that each class needs to attain its accuracy target. We update these class priority weights at every active learning iteration. Fig. 1 shows an example to motivate class priority weights where the 'pencil' class gets the highest priority because its estimated accuracy is farthest away from its target. For a class y_i at iteration ℓ, let $\rho_{i,\ell}$ represent the target value of an accuracy metric of interest and $\hat{\rho}_{i,\ell}$ be its estimated value. Then, any realization of the Class Priority Weight Computer must generate a weight $w_{i,\ell}$ for class y_i at iteration ℓ such that

$$w_{i,\ell} \propto \max(\rho_{i,\ell} - \hat{\rho}_{i,\ell}, 0) \tag{1}$$

If the goal is to decrease a target metric *e.g.*, false positive rate, it should be expressed as its negative in Eq. (1) *e.g.*, negative false positive rate. In this paper, we implement a realization of Class Priority Weight Computer based on a class-level accuracy metric ρ such that, at each active learning iteration ℓ, it computes a priority weight $w_{i,\ell}$ for class y_i as

$$w_{i,\ell} = \begin{cases} \max(\delta_{i,\ell}, 0) & \text{if } \sum_{j=1}^{k} \max(\delta_{j,\ell}, 0) > 0, \\ 1 & \text{otherwise.} \end{cases} \tag{2}$$

where $\delta_{i,\ell} = \rho_{i,\ell} - \widehat{\rho}_{i,\ell}$, with $\rho_{i,\ell}$ and $\widehat{\rho}_{i,\ell}$ being the target and estimated values of the accuracy metric respectively.

2.2 Candidate Labels Generator

The Candidate Labels Generator identifies the subset of classes whose accuracy is likely to improve if a particular instance from the unlabeled data pool is labeled. Let Y_x be the subset of classes with a high likelihood of containing the true class of an unlabeled instance x. We construct Y_x using any information available on x; *e.g.*, it can leverage the classifier's scores across classes to identify candidate classes. We hypothesize that retraining the classifier after labeling x is likely to improve its accuracy over all classes in Y_x. Any realization of the Candidate Label Generator should generate a candidate labels set Y_x for x such that the true class of x has a high probability of being a member of Y_x, while keeping $|Y_x|$ as small as possible. In this

paper we implement a realization of the Candidate Labels Generator that produces a candidate class labels set for an input instance x given by

$$Y_x = \left\{ y \in Y \;\middle|\; \frac{P(y \mid x)}{\max_{z \in Y} P(z \mid x)} \geq t_c \right\} \quad (3)$$

where $P(y \mid x) \in [0, 1]$ is the probability score assigned to class y by the active learning classifier given instance x, and t_c is a configurable cut-off threshold. We set t_c to 0.5. This means all classes with score $\geq 50\%$ of the maximum score are treated as candidate class labels for each instance.

2.3 Class Priority Aware Query Adapter

Class Priority Aware Query Adapter reorders the input queue of unlabeled instances to reflect class priority weights. Consider the example in Fig. 2 in which instance d_3 has rank 3 in the queue of instances Q from an active learning query strategy (non-target-aware). One of d_3's candidate labels, *Pencil*, has the highest priority weight. So, the Class Priority Aware Query Adapter pushes d_3 ahead of d_1 and d_2 in its reordered output queue Q_p. Our implementation of the *Class Priority Aware Query Adapter* is in Algorithm 1. In each active learning iteration, we start with a ranked queue of unlabeled instances Q from the underlying query strategy. Then, we invoke *Class Priority Weight Computer* to update priority weights W. *Class Priority Aware Query Adapter* starts with sampling a class $y_{selected}$ from Y with priority weights W as sampling probability (Line 3). Then it traverses Q from top to bottom, selecting the first instance d which has $y_{selected}$ in its candidate labels set (Lines 5 to 16), and enqueues d in Q_p if it is not present in Q_p already (Line 8). Finally, it removes class $y_{selected}$ from the candidate labels set of d so that instance d is not picked for $y_{selected}$ again (Line 9). If a class is not a candidate label for any instance in Q, its priority weight is reset to 0 since it cannot be prioritized in the current learning iteration. The process is repeated until $|Q_p|$ reaches N_q or no classes can be prioritized (Lines 2 to 17). The computational complexity of *Class Priority Weight Computer* (Eq. 2) and *Candidate Labels Generator* (Eq. 3) is $\mathcal{O}(k)$. Hence, the computational complexity of the *Class Priority Aware Query Adapter* (Algorithm 1) is $\mathcal{O}(|Q| \cdot |Q_p| \cdot k)$. In practice, $k << |Q|$ and $|Q_p| << |Q|$; hence, the overall computational complexity of a TAAL-enabled query strategy is $\mathcal{O}(|Q|)$.

Algorithm 1: Class Priority Aware Query Adapter

Inputs : (i) Queue Q of instances from a query strategy, (ii) Class priority weights array W as $[w_1, w_2, ..., w_k]$ where w_j is the priority weight for class y_j from Eq. (1), (iii) No. of output instances $N_q \leq |Q|$

Output : Class priority aware queue Q_p of instances.

1 **Initialize:** (i) Empty queue Q_p, (ii) Candidate labels map $\mathcal{F}$ mapping each instance x in Q to $\mathcal{F}[x]$, with $\mathcal{F}[x] \leftarrow Y_x$ computed using Eq. (3).
2 **while** $|Q_p| < N_q$ ***and*** $\max(W) \neq 0$ **do**
3 Randomly sample 1 class $y_{selected} \in Y$ per weights W.
4 $i \leftarrow 1$
5 **while** $i \leq |Q|$ **do**
6 $d \leftarrow$ instance at index i in Q
7 **if** $y_{selected} \in \mathcal{F}[d]$ **then**
8 **if** $d \notin Q_p$ **then** Q_p.enqueue(d) ;
9 $\mathcal{F}[d] \leftarrow \mathcal{F}[d] \setminus \{y_{selected}\}$
10 Exit inner while loop.
11 **else**
12 $i \leftarrow i + 1$
13 **if** $i > |Q|$ **then** reset weight for $y_{selected}$ in W to 0 ;
14 **end**
15 **end**
16 **end**
17 **return** Q_p

3 Experiments

We use pool-based active learning setup (Lewis and Gale, 1994; Settles, 2009; Gal et al., 2017; Lin and Parikh, 2017) consisting of a model $\mathcal{M}$, a seed set of randomly sampled labeled instances $(x_i, y_i) \in \mathcal{D}_{\text{seed}}$ ($|\mathcal{D}_{\text{seed}}| = 500$) to initialize $\mathcal{M}$ in the first iteration, an unlabeled data pool $\mathcal{D}_{\text{pool}}$, and a query strategy $\mathcal{R}$. We run active learning over a series of iterations until a budget of L ($L = 6000$) labeled instances. At each iteration ℓ, we acquire a batch of B ($B = 500$) new instances from $\mathcal{D}_{\text{pool}}$ governed by $\mathcal{R}$, and re-train model $\mathcal{M}$. We conduct

experiments on 2 image-based and 2 text-based datasets (Sec. 3.2) which are fully-labeled (including "unlabeled" data $\mathcal{D}_{\text{pool}}$) and multi-class. We use a labeling bot to simulate a human-in-the-loop active learning setup (Gal et al., 2017; Lin and Parikh, 2017; Siddhant and Lipton, 2018) using the fully-labeled datasets. The bot provides labels for the data instances selected by the query strategy $\mathcal{R}$. We compare the baseline query strategies listed in Sec. 3.1 against their Target-Aware (TA-) variants created by applying TAAL on top of each query strategy. For image datasets we use ENTROPY, BADGE, CORESET, and SIMILAR as baseline query strategies because they have produced SOTA results on several benchmarks (Citovsky et al., 2021). SIMILAR is an interesting baseline because it gives high accuracy for rare classes in the dataset. For product classification datasets, we use E3G as the baseline query strategy to mimic our internal active learning setup.

3.1 Baseline Active Learning Query Strategies

We discuss active learning strategies in detail under related work (section 5). Below we describe the specific query strategies used in experiments.

ENTROPY (Settles, 2009) sampling queries the labels of instances for which the prediction of the classifier is maximally uncertain in terms of entropy of confidence scores. It selects a batch of B instances from $\mathcal{D}_{\text{pool}}$ with highest entropy in model's output, where the entropy $\mathcal{H}_x$ for each instance x is measured over say k classes as $-\sum_{i=1}^{k} P(y_i|x) \log P(y_i|x)$. **BADGE** (Ash et al., 2019a) samples a batch of B instances from $\mathcal{D}_{\text{pool}}$ that are disparate and high magnitude when represented in a hallucinated gradient space, a strategy designed to incorporate both predictive uncertainty and sample diversity into every selected batch. BADGE uses the k-Means++ seeding algorithm on $\{g_x : x \in \mathcal{D}_{\text{pool}}\}$ where g_x is the gradient embedding of an instance x using current model weights w. **CORESET** (Sener and Savarese, 2017) addresses the problem of selecting diverse examples in a batch by posing active learning as coreset selection, *i.e.* choosing a set of points such that a model learned over the selected subset is competitive for the remaining data points. It samples a batch of B instances from $\mathcal{D}_{\text{pool}}$ by solving the k-center problem on $\{z_x : x \in \mathcal{D}_{\text{pool}}\}$, where z_x is the embedding of an instance x derived from the penultimate layer of the model. **SIMILAR** (Kothawade et al., 2021) is a unified active learning framework that employs different sub-modular information measures, namely, Submodular Mutual Information, Submodular Conditional Gain, Submodular Conditional Mutual Information as query acquisition functions. SIMILAR not only works in standard active learning setup but also easily extends to more realistic settings such as having class imbalance in the data distribution, and provides the SOTA solution for active learning. **E3G** (Slivkins, 2019) (**E**xplore-**E**xploit-**EG**reedy) uses the ϵ-greedy algorithm to choose between Exploration and Exploitation. The exploration strategy performs k-means clustering-based stratified random sampling, whereas the exploitation strategy is entropy-based uncertainty sampling. To form a batch of B instances per active learning iteration, each instance $x \in \mathcal{D}_{\text{pool}}$ is chosen independently by E3G with probability ϵ from the exploration strategy, and with probability $1 - \epsilon$ from the exploitation strategy. During the early iterations of active learning, E3G focuses on the exploration strategy, whereas later it emphasizes the exploitation strategy. It does this by varying ϵ across iterations following $f(N_\ell) = \max(e^{-\gamma N_\ell}, \epsilon_{\text{min}})$, where γ is the decay rate, ϵ_{min} is the minimum probability of choosing an instance per exploration strategy, and N_ℓ is the total number of labeled instances at iteration ℓ.

3.2 Datasets

We use 2 public **image-based** datasets, CIFAR-10 (Krizhevsky et al., 2009) and SVHN (Netzer et al., 2011). CIFAR-10 has 60k images uniformly across 10 classes. The training subset of the dataset has 50k images, and the remaining 10k images form the test set. SVHN has images spread across 10 classes, with 73257 images for training, and 26032 images for testing. To evaluate TAAL, we induce class imbalance and form a long-tailed distribution over class sizes (see D_c in Table 2). For both datasets, we carve out a 20% class-stratified random subset from the imbalanced training set to create a held-out validation set. We also use 2 proprietary fully-labeled **text-based** product classification datasets *HS* and *FEE* derived from 2 different e-commerce product taxonomies. HS has 9 classes-of-interest and 1.15M products, while FEE has 7 classes-of-interest and 740k products. Additionally, both datasets have one special class called 'not-in-k' representing instances in the dataset that do not

belong to any classes-of-interest. Both datasets have an intrinsic long-tailed distribution in terms of class sizes (see D_c in Table 2).

3.3 Active Learning Classifier

We use a common training procedure and hyper-parameters to ensure that all query strategies are given fair treatment across all experiments. For image classification datasets, we follow the setup described in (Kothawade et al., 2021). We use ResNet-18 (He et al., 2016) as our classifier $\mathcal{M}$ and train it using a stochastic gradient descent optimizer with an initial learning rate of 0.01 and momentum of 0.9. During inference, we take the class corresponding to the highest score from the final layer of the trained model as the prediction for an instance. For product classification datasets, we choose fastText (Joulin et al., 2016) as our active learning classifier $\mathcal{M}$ since it provides an attractive trade-off between accuracy and latency. We tune thresholds for each class to achieve the desired precision target of 90% using *out-of-fold* prediction scores in a one vs. rest setting (number of folds is 10). During inference, we treat an instance as member of a class if its output class probability is higher than its corresponding threshold.

3.4 Accuracy Estimation During Active Learning

For **image datasets**, we compute class priority weights using class-level F_1 score as the accuracy metric in Eq. (2), with the per-class F_1 score target being 90%. At every active learning iteration, we estimate class-level F_1 score on the held-out validation set which is an unbiased sample from the entire dataset. For **product classification datasets**, we compute class priority weights using class-level 'recall at precision' (R@P) as the accuracy metric in Eq. (2), with the per-class R@P target being 90R@90P. At every active learning iteration, we estimate class-level R@P using out-of-fold prediction scores over *exploration*-sourced labeled data from the E3G query strategy. This provides an unbiased performance estimate since it avoids the sampling bias induced by the *exploitation* strategy. We also set the priority weight for the 'not-in-k' class to 0 for all active learning iterations because our use case necessitates improving accuracy over the k classes-of-interest only.

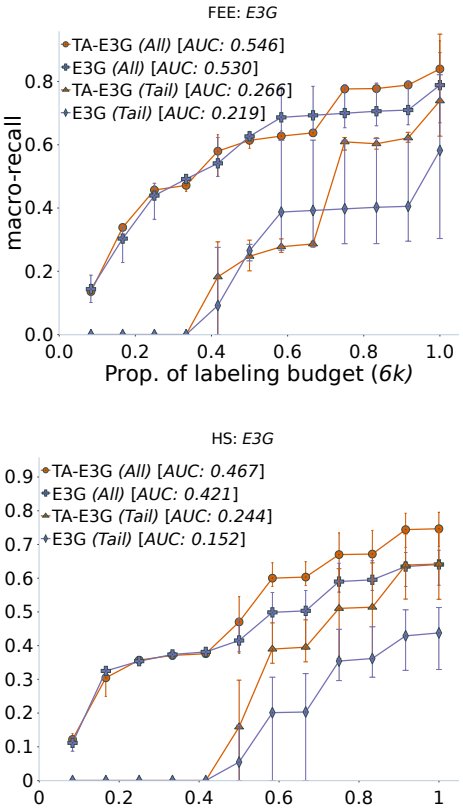

Figure 3: Progression of *macro-recall* across all classes and tail classes for **E3G** and its target-aware variant **TA-E3G** on HS and FEE datasets. **Axes' labels are identical for both plots**. Metrics are averaged across 3 experiments, with error bars showing min and max values. Legends report area under each macro-recall curve (AUC) summarizing macro-recall across label counts. TA-E3G has greater AUC for all-class macro-recall and tail-class macro-recall than E3G.

3.5 Metrics for Comparison of Query Strategies

For experiments on **image datasets**, we report class-level F_1 on a held-out test set to replicate the setup in (Kothawade et al., 2021) to compare each query strategy against its target-aware variant. For experiments on **product classification datasets**, we report class-level recall computed on the full dataset to mimic our internal product classification setup where customers want to achieve target accuracy for each class-of-interest on the entire domain. We summarize class-level metrics by reporting *macro*-recall and *macro*-F_1 score. We choose *macro* over *micro* averaging because accuracy of each class is equally important for our use-case.

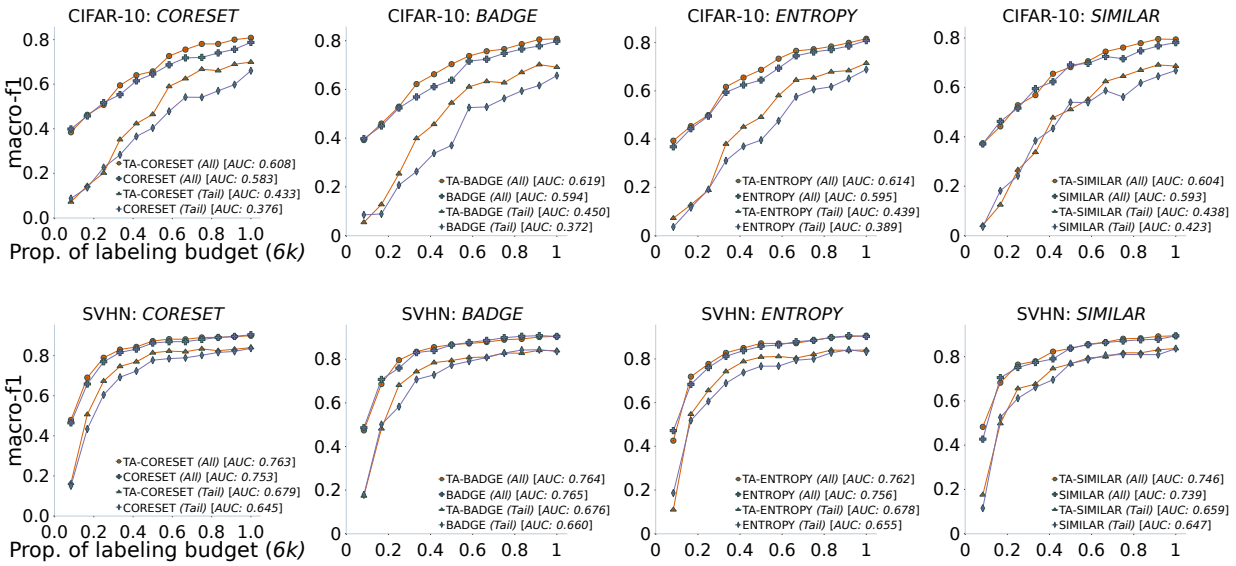

Figure 4: Progression of *macro*-F_1 across all classes and tail classes for 4 query strategies and their target-aware variants (TA-*) on CIFAR-10 test set (top) and SVHN test set (bottom). **Axes' labels are identical for all plots**. Legends report area under each macro-F_1 curve (AUC) summarizing macro-F_1 across all label counts. Each query strategy's target-aware variant provides greater or similar AUC for all-class macro-F_1 and greater AUC for tail-class macro-F_1.

4 Results

Figs. 3 and 4 show the progression of macro-F_1 and macro-recall with increasing number of training examples over product classification and image classification datasets, respectively. Plot legends report Area Under the Curve (AUC) for each query strategy, summarizing its accuracy metric values across all label counts (*higher the better*). Averaged across all 10 *dataset* + *query strategy* combinations, TAAL increases AUC by 1.6 percentage points (from 63.3% to 64.9%). For 8 out of the 10 *dataset* + *query strategy* combinations, we see an increase in AUC with TAAL, indicating that TAAL performs better than the baseline strategies measured across all label counts. For SVHN dataset, the target-aware variants of BADGE and ENTROPY have the same AUC as their baseline strategies indicating that TAAL offers no improvement. We hypothesize that this is because SVHN is easy to classify as we get high (> 0.8) macro-F_1 scores across tail classes with baseline query strategies.

Impact on *tail* classes: *Tail* classes are the smallest classes-of-interest comprising $\leq 5\%$ of the full dataset. Achieving business targets on tail classes is important because they represent rare, but strategically important classes-of-interest. Averaged across all 10 *dataset* + *query strategy* combinations, TAAL increases tail-class AUC by 4.2 percentage points (from 45.4% to 49.6%), explained by a corresponding increase in the proportion of tail class labels by 11.3 percentage points (from 14.2% to 25.5%) at 6k labels. Tail classes have a lower likelihood of getting labeled compared to *head* classes. With fewer training instances, the classifier performs poorly over them. This is seen in Figs. 3 and 4, where tail-class AUC is lower than all-class AUC. As labeling progresses, all baseline strategies except SIMILAR continue to focus on all classes equally. In contrast, TAAL assigns greater importance to tail classes through weights generated by *Class Priority Weight Computer* (Sec. 2.1). As depicted in Fig. 2, TAAL then prioritizes examples for labeling deemed optimal per the underlying query strategy, but at the same time relatively more likely to benefit tail classes. This is reflected in Table 2 where the class distribution of training instances L_c is less skewed for TAAL compared to the corresponding baseline strategies. *E.g.*, at 6k labels, HS-4 (a tail class) constitutes 6.2% of total labels with TA-E3G vs. 1.8% with E3G, an increase by a factor of 3.4.

Impact on *head* classes: *Head* classes are the largest classes-of-interest comprising $\geq 95\%$ of the full dataset. Averaged over all 10 *dataset* + *query strategy* combinations, TAAL reduces the propor-

Table 1: Area under the accuracy metric curve (AUC) for all *dataset + query strategy* combinations at 6k labels. The accuracy metric summarized by AUC is F_1 score for image datasets and recall for product classification datasets. We report AUC for each query strategy ('Baseline' columns) against its target-aware variant ('TAAL' columns) over all classes, head classes, and tail classes.

Dataset	Query Strategy	Baseline AUC (%)			TAAL AUC (%)		
		All	Head	Tail	All	Head	Tail
SVHN	CORESET	75.25	82.43	64.48	76.30	81.94	67.86
	SIMILAR	73.87	79.99	64.68	74.61	80.40	65.92
	ENTROPY	75.60	82.34	65.49	76.16	81.76	67.78
	BADGE	76.48	83.49	65.96	76.39	82.27	67.57
CIFAR-10	CORESET	58.35	72.16	37.63	60.84	72.50	43.34
	SIMILAR	59.31	70.62	42.34	60.38	71.40	43.84
	ENTROPY	59.54	73.29	38.91	61.41	73.07	43.93
	BADGE	59.36	74.13	37.22	61.91	73.19	44.98
FEE	E3G	53.03	76.35	21.95	54.64	75.63	26.65
HS	E3G	42.06	75.64	15.19	46.69	74.54	24.40

tion of head class labels by 11.3 percentage points (from 84.5% to 73.2%, Table 2) at 6k labels, while only reducing head-class AUC by 0.4 percentage points (from 77.0% to 76.7%, Table 1).

Comparison of SIMILAR and TA-SIMILAR: Experiments show that TA-SIMILAR gives only a minor improvement over SIMILAR (+0.7 percentage points all-class AUC, +1.2 percentage points tail-class AUC). This is expected because SIMILAR is designed to focus on tail classes, and the same tail classes also get prioritized by TAAL since they lag in class-level accuracy during active learning. SIMILAR is different from TAAL in that (a) SIMILAR needs specification of the explicit set of rare classes in the dataset (unlike TAAL), which is not always known *a priori* and, (b) SIMILAR, like any other non-TAAL query strategy, will continue to improve the accuracy of a class even if it has already met its target, thus consuming unnecessary manual annotations.

5 RELATED WORK

We refer readers to survey papers (Settles, 2009; Ren et al., 2020) for a comprehensive overview of active learning. In this section, we discuss the core ideas we have borrowed from prior-art in Active Learning.

Active learning query strategies determine what unlabeled data should be annotated next. The most popular query strategies are uncertainty-based (Settles, 2009; Holub et al., 2008; Beluch et al., 2018; Wu et al., 2020; Lewis and Gale, 1994), diversity-based (Bilgic and Getoor, 2009; Guo, 2010; Dasgupta and Hsu, 2008; Sener and Savarese, 2017; Jiang and Qing-Yu, 2015), and expected-model-change based (Freytag et al., 2014; Roy and Mccallum, 2001). Sec. 3.1 describes the specific query strategies we used in experiments.

We use batch-mode active learning in this paper. Previous research (Lewis and Gale, 1994; Settles, 2009; Gal et al., 2017; Lin and Parikh, 2017) has demonstrated that choosing a batch of instances with small batch sizes offers a good trade-off between user interactivity and number of labels required to create a classifier.

Hybrid query strategies (Ash et al., 2019b; Shui et al., 2019) deal with controlling the classical *exploration vs. exploitation* trade-off. Exploitation aims to find data with highest uncertainty that is likely to help the model learn fast (Bloodgood and Vijay-Shanker, 2009), and exploration aims to find data that is representative of the unlabeled data. One of our baseline strategies E3G (Sec. 3.1)) falls under this umbrella of hybrid approaches.

Stopping the active learning loop involves defining exit criteria that help annotators decide when to stop labeling. The most common stopping method is to use a predefined criterion (Budd et al., 2019; Liu et al., 2016; Schröder and Niekler, 2020) such as maximum number of active learning iterations, maximum annotation budget/time, or minimum classification accuracy to be achieved. Recent papers on stopping criteria have argued for checking if predictions on unlabeled data have stabilized (Bloodgood and Vijay-Shanker, 2014; Vlachos, 2008; Budd et al., 2019; Zhu et al., 2010). We use maximum labeling budget as the sole stopping criterion in this paper for a fair comparison

Table 2: Comparison of relative size of each class in the full dataset (D_c) and in labeled data (L_c) at 6k labels. For L_c, we compare each baseline active learning strategy against its target-aware variant ('TA-'). Asterisk indicates tail classes.

Text-based product taxonomy dataset **FEE** (Data represented in %)										
Class	F-0	F-2	F-3	F-1	F-5*	F-4*	F-6*	F-not-in-k	-	-
D_c	50.6	30.8	10.7	6.4	0.6	0.5	0.3	0.12	-	-
E3G L_c	34.3	31.1	13.3	10.6	3.6	2.3	2.2	2.5	-	-
TA-E3G L_c	28.5	21.5	7.7	4.9	9.3	8.8	17.8	1.4	-	-
Text-based product taxonomy dataset **HS** (Data represented in %)										
Class	HS-2	HS-3	HS-8	HS-7	HS-6*	HS-0*	HS-5*	HS-1*	HS-4*	HS-not-in-k
D_c	41.9	25.8	14.7	11.1	2.4	0.5	0.4	0.4	0.3	2.6
E3G L_c	34.1	16.4	13.4	11.5	3.3	1.8	4.5	2.4	1.8	10.8
TA-E3G L_c	32.9	14.9	10.2	7.9	2.8	2.6	5.7	5.0	6.2	11.9
Image-based dataset **SVHN** (Data represented in %)										
Class	S-1	S-3	S-5	S-6	S-7	S-0	S-2*	S-4*	S-8*	S-9*
D_c	28.9	17.7	14.4	12.0	11.7	10.3	1.3	1.3	1.3	1.3
SIMILAR L_c	16.6	17.1	7.1	10.4	12.5	6.3	7.6	7.6	7.4	7.6
TA-SIMILAR L_c	13.8	17.2	10.4	12.3	11.5	4.7	7.6	7.6	7.4	7.6
ENTROPY L_c	19.4	20.6	13.2	14.0	11.1	8.7	3.2	3.5	3.1	3.3
TA-ENTROPY L_c	14.0	17.1	11.3	11.3	8.3	7.0	7.8	7.7	7.7	7.9
BADGE L_c	19.7	19.1	14.5	13.9	11.1	9.0	3.1	3.3	3.3	3.2
TA-BADGE L_c	15.0	16.9	11.1	11.2	8.2	6.9	7.7	7.7	7.6	7.8
CORESET L_c	22.9	18.1	12.7	12.4	10.9	9.4	3.5	4.0	2.6	3.5
TA-CORESET L_c	17.2	15.7	9.0	10.8	8.8	7.8	7.7	7.7	7.5	7.8
Image-based dataset **CIFAR-10** (Data represented in %)										
Class	C-0	C-1	C-3	C-5	C-6	C-7	C-2	C-4*	C-8*	C-9*
D_c	15.8	15.8	15.8	15.8	15.8	15.8	1.3	1.3	1.3	1.3
SIMILAR L_c	16.2	17.2	10.9	10.4	10.2	18.0	3.7	4.2	4.8	4.6
TA-SIMILAR L_c	13.3	8.7	20.4	18.4	9.6	11.0	4.5	4.6	4.9	4.7
ENTROPY L_c	13.1	9.2	21.5	17.2	12.8	14.6	3.4	3.4	2.3	2.5
TA-ENTROPY L_c	13.8	8.2	20.2	15.7	10.9	12.5	4.5	4.5	4.9	4.7
BADGE L_c	14.2	9.3	21.2	18.3	13.0	13.5	2.9	2.7	2.6	2.5
TA-BADGE L_c	13.3	7.9	20.6	16.6	10.5	12.6	4.5	4.5	4.9	4.7
CORESET L_c	19.8	16.8	15.2	12.6	11.2	13.6	2.3	2.5	3.2	2.9
TA-CORESET L_c	13.5	8.8	20.2	16.4	10.1	12.9	4.3	4.4	4.8	4.7

of baseline query strategies and their target-aware variants.

6 Conclusions and Future Work

We introduce a framework called Target-Aware Active Learning (TAAL) that converts any arbitrary active learning query strategy into its target-aware variant by leveraging the gap between each class' current estimated accuracy and its corresponding business target. We perform extensive experiments comparing 5 baseline query strategies and their target-aware variants on 2 image classification and 2 text-based product classification datasets. Our results indicate that TAAL improves the likelihood of achieving business targets on 8 out of 10 *dataset + query strategy* combinations, and matches the baseline performance of BADGE and ENTROPY strategies on the SVHN dataset. In addition, the run-time complexity of TAAL scales linearly with the size of the input query, making it practical for large-scale active learning. In the future, we plan to test TAAL with classifiers like PECOS (Yu et al., 2022) which further improve the accuracy of tail classes, especially for high-cardinality classification problems.

References

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. 2019a. Deep batch active learning by diverse, uncertain gradient lower bounds. *CoRR*, abs/1906.03671.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. 2019b. Deep batch active learning by diverse, uncertain gradient lower bounds. *CoRR*, abs/1906.03671.

William H. Beluch, Tim Genewein, Andreas Nürnberger, and Jan M. Köhler. 2018. The power of ensembles for active learning in image classification. In *CVPR*, pages 9368–9377. IEEE Computer Society.

Mustafa Bilgic and Lise Getoor. 2009. Link-based active learning. In *NIPS Workshop on Analyzing Networks and Learning with Graphs*.

Michael Bloodgood and K. Vijay-Shanker. 2009. Taking into account the differences between actively and passively acquired data: The case of active learning with support vector machines for imbalanced datasets. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, May 31 - June 5, 2009, Boulder, Colorado, USA, Short Papers*, pages 137–140. The Association for Computational Linguistics.

Michael Bloodgood and K. Vijay-Shanker. 2014. A method for stopping active learning based on stabilizing predictions and the need for user-adjustable stopping. *CoRR*, abs/1409.5165.

Samuel Budd, Emma C. Robinson, and Bernhard Kainz. 2019. A survey on active learning and human-in-the-loop deep learning for medical image analysis. *CoRR*, abs/1910.02923.

Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. 2021. Batch active learning at scale.

Sanjoy Dasgupta and Daniel Hsu. 2008. Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning*, pages 208–215.

Alexander Freytag, Erik Rodner, and Joachim Denzler. 2014. Selecting influential examples: Active learning with expected model output changes. In *European conference on computer vision*, pages 562–577. Springer.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR.

Yuhong Guo. 2010. Active instance sampling via matrix partition. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Alex Holub, Pietro Perona, and Michael C Burl. 2008. Entropy-based active learning for object recognition. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE.

S Jiang and OU Qing-Yu. 2015. Batch-mode active learning approach of computer viruses classifier based on information density. *Naval Univ. Eng*, 431(5).

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Suraj Kothawade, Nathan Beck, Krishnateja Killamsetty, and Rishabh Iyer. 2021. Similar: Submodular information measures based active learning in realistic scenarios.

Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer.

Xiao Lin and Devi Parikh. 2017. Active learning for visual question answering: An empirical study. *arXiv preprint arXiv:1711.01732*.

Peng Liu, Hui Zhang, and Kie B. Eom. 2016. Active deep learning for classification of hyperspectral images. *CoRR*, abs/1611.10031.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. 2020. A survey of deep active learning. *CoRR*, abs/2009.00236.

Nicholas Roy and Andrew Mccallum. 2001. Toward optimal active learning through monte carlo estimation of error reduction. In *in Proceedings of the International Conference on Machine Learning*.

Christopher Schröder and Andreas Niekler. 2020. A survey of active learning for text classification using deep neural networks. *CoRR*, abs/2008.07267.

Ozan Sener and Silvio Savarese. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*.

Burr Settles. 2009. Active learning literature survey.

Changjian Shui, Fan Zhou, Christian Gagné, and Boyu Wang. 2019. Deep active learning: Unified and principled method for query and training. *CoRR*, abs/1911.09162.

Aditya Siddhant and Zachary C Lipton. 2018. Deep bayesian active learning for natural language processing: Results of a large-scale empirical study. *arXiv preprint arXiv:1808.05697*.

Aleksandrs Slivkins. 2019. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286.

Andreas Vlachos. 2008. A stopping criterion for active learning. 22(3):295–312.

Jian Wu, Victor S Sheng, Jing Zhang, Hua Li, Tetiana Dadakova, Christine Leon Swisher, Zhiming Cui, and Pengpeng Zhao. 2020. Multi-label active learning algorithms for image classification: Overview and future promise. *ACM Computing Surveys (CSUR)*, 53(2):1–35.

Hsiang-Fu Yu, Kai Zhong, Jiong Zhang, Wei-Cheng Chang, and Inderjit S. Dhillon. 2022. Pecos: Prediction for enormous and correlated output spaces.

Jingbo Zhu, Huizhen Wang, Eduard H. Hovy, and Matthew Y. Ma. 2010. Confidence-based stopping criteria for active learning for data annotation. *ACM Trans. Speech Lang. Process.*, 6:3:1–3:24.