Can I see an Example? Active Learning the Long Tail of Attributes and Relations

Tyler L. Hayes[1*]
tlh6792@rit.edu
Maximilian Nickel[2]
maxn@fb.com
Christopher Kanan[1,3]
ckanan@cs.rochester.edu
Ludovic Denoyer[2†]
ludovic.den@gmail.com
Arthur Szlam[2]
aszlam@fb.com

[1] Rochester Institute of Technology
Rochester, NY, USA
[2] Facebook AI Research
New York, NY, USA
[3] University of Rochester
Rochester, NY, USA

Abstract

There has been significant progress in creating machine learning models that identify objects in scenes along with their associated attributes and relationships; however, there is a large gap between the best models and human capabilities. One of the major reasons for this gap is the difficulty in collecting sufficient amounts of annotated relations and attributes for training these systems. While some attributes and relations are abundant, the distribution in the natural world and existing datasets is long tailed. In this paper, we address this problem by introducing a novel incremental active learning framework that asks for attributes and relations in visual scenes. While conventional active learning methods ask for labels of specific examples, we flip this framing to allow agents to ask for examples from specific categories. Using this framing, we introduce an active sampling method that asks for examples from the tail of the data distribution and show that it outperforms classical active learning methods on Visual Genome.

1 Introduction

In active learning [41, 63, 75, 79, 83], a learning agent is provided with unlabeled samples, from which it selects those it considers critical to improving its performance to be labeled by a teacher (oracle). By selecting the most informative samples to be labeled, it is hoped that the agent can achieve better performance with less data. However, in practice, the extra complexity of active learning approaches often outweigh their benefits. In our view, the power of active learning is tempered by the framing of the classical machine learning problems where it is often applied. Namely, a classification problem with few classes, with loss measured in average classification accuracy on the test distribution, and a repertoire of questions of the

* Work done while interning at Facebook AI Research
† Now at Ubisoft

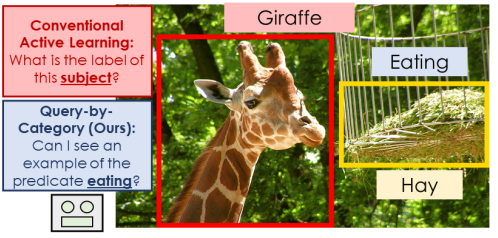


Figure 1: In classical active learning, an agent selects *examples* that it is most uncertain about to be labeled. Conversely, we allow agents to ask for examples at the *category* level instead of the example level. We train agents to predict attributes and relations in scenes. Our framing allows an agent to ask for an example of a rare category "eating" and an oracle provides an example of this predicate.

form "what is the label of example *i*?". This is in contrast to the "active learning" of humans, who have access to extremely rich supervision beyond asking for the label of an example.

In this work, we study the problem of learning to predict objects, attributes, and relations in visual scenes. The set of possible attributes and relations is long-tailed (that is, there are many attributes and relations, with many categories containing very few samples), and its structure suggests more sophisticated questions a learning agent might ask. For example, an agent might ask for an example from a particular *label* it is uncertain about, instead of a particular sample it is uncertain about. By asking for class labels instead of instances, an agent can explicitly sample from more underrepresented classes, which is useful for long-tailed data. We show that several classical active learning methods do not perform significantly better than random sampling; and in particular fail to explore the tail of the distribution. By simply switching the framing to allow the agent to ask for an example for an attribute or relation (rather than ask for the true attribute or relation for object instances in an image), we can significantly improve results on the tail of the distribution, without sacrificing accuracy on the natural distribution (see Fig. 1). This is critical as visual scenes are typically long-tailed and we desire methods capable of performing well on the tail of the distribution.

We make the following major contributions: We introduce a novel incremental active learning framework, coined "Query-by-Category" (QBCat), that allows agents to ask for labels at the *category* level instead of the *example* level. To demonstrate the effectiveness of our framework, we study active learning on a new domain: training agents to predict objects, attributes, and relations in visual scenes. Since the distribution of attributes and relations is naturally long-tailed, we introduce two methods to enable incremental long-tailed learning including class re-balancing and bias correction. Finally, we use our Query-by-Category framework to introduce a new active sampling approach that samples efficiently from the tail of the attribute and relation class distributions. We experimentally validate the effectiveness of our setup, which outperforms classical active learning methods on Visual Genome [38].

Motivation for Query-by-Category Learning. In conventional active learning, the agent computes uncertainty scores over unlabeled examples to select examples to be labeled. While straightforward, this approach has drawbacks: a representative set of unlabeled examples must be collected, the learner must compute uncertainty scores for the examples, and the learner must have robust uncertainty estimates to select examples. These tasks can be time consuming for both annotators and the learner. Moreover, the gamut of queries the learner can ask for is limited, i.e., the model simply asks for the label of an unlabeled example.

We argue that it could be beneficial to allow a model to ask directly for examples from specific classes. This eliminates the need for model inference on unlabeled data. Further, by asking for classes directly, the agent will likely see a wider variety of classes, which is critical in long-tailed settings. Just as in conventional active learning, the primary focus of our framework is on learning efficiency rather than the speed in which an oracle can provide annotations. Moreover, in a practical implementation, an annotator could be augmented with a search engine or lots of thumbnail images to quickly find images of a specific class. Alternatively, the learner could be provided with a search engine or database to gather labeled data for itself. We leave these implementations for future work; here we are first interested in determining if our framing yields performance benefits over conventional active learning.

2 Methods

We operate on *(subject, predicate, object)* triples in images. The subject will be a bounding box marking a region of pixels (e.g., an object) in an image. The object might be another bounding box if the predicate is a spatial relationship, like "to the left of"; or a string like "blue" (which could correspond to a predicate like "has color"). If the object is a string, we use the notation *(subject, predicate, attribute)*. We do not consider higher arity relations or relations that cross multiple images. The learner will be tasked with predicting the missing element in an incomplete triple. We denote the triple with two known elements as the *question* and the correct missing element as the *target*. We use s, o, p, and a to denote subjects, objects, predicates, and attributes respectively. This yields six unique question types: $(?, p, o)$ and $(?, p, a)$ where the target is a subject, $(s, ?, o)$, and $(s, ?, a)$ where the target is a predicate, $(s, p, o?)$ where the target is an object, and $(s, p, a?)$ where the target is an attribute.

Rather than assuming the labeled data is fixed at the start of learning, we perform incremental active learning, i.e., the agent asks for a fixed number of annotations over a sequence of increments. However, we will see that classical active learning techniques do not significantly improve on random selection, in part due to the long tail of attributes and relations. Our main result is that asking for examples of a triple (described in Sec. 2.3) is more effective at exploring the tail of possibilities than asking for triple completions. However, this active sampling strategy biases the model to the tail and reduces performance on the natural distribution. We will see that combined with proper re-biasing techniques, we get the best of both worlds: improved tail performance without sacrificing accuracy on the natural distribution.

2.1 Incremental Training Procedure

Let $\mathcal{D}_L$ and $\mathcal{D}_U$ denote labeled and unlabeled subsets of the dataset $\mathcal{D} = \mathcal{D}_L \cup \mathcal{D}_U$, respectively, where $\mathcal{D}$ is a dataset of triples. The "unlabeled" triples correspond to the "questions" from above; whereas "labeled" examples are the complete triple (question and target together). For the rest of the paper, we will use "natural distribution" to refer to either the base distribution on $\mathcal{D}$ or to targets in $\mathcal{D}$; and the "head" to refer to the elements in $\mathcal{D}$ with the most common targets (or to those targets themselves), and the "tail" to refer to other targets.

In the real world, an agent would receive a stream of inputs for which it could incrementally ask questions to improve its performance. Motivated by this, we perform incremental active learning as follows. We first pre-train the model on $\mathcal{D}_L$ drawn from the head of the training data. We wish to simulate the setting where we first collect some seed data (which, following the natural distribution, would mostly come from the head), and then the agent

actively learns starting from a model pre-trained on the seed data. After pre-training, we initialize a replay buffer with all pre-training data. Then, an active sampling strategy (described in Sec. 4.1) selects B samples from the unlabeled dataset ($\mathcal{D}_U$) to be labeled by an oracle.

After an oracle labels the B samples, training happens in two stages: cross-validation and full training. Cross-validation determines the number of epochs for full training to prevent over-fitting on new samples (see Sec. S2.1). After cross-validation, we reset model parameters to the beginning of the increment, retrain with the validated hyper-parameters, and optionally re-bias to the natural distribution (see Sec. 2.2). We then evaluate the model (see Sec. 4) and put all newly labeled examples in the replay buffer and repeat. The process of adding new data, cross-validation, training, and optional re-biasing is called an "increment."

2.2 Methods to Handle Class Imbalance

Because of the long-tailed data distribution, naive training on increments leads to over-fitting on the head classes and hinders learning after the first increment (see Sec. S4.4). To address this, we re-balance mini-batches during increments such that there are equal amounts of old (frequently represented) data and new (possibly less frequently represented) data (see Sec. S2.2). This allows learning past the first increment; but results in the model learning a distribution different from the natural distribution. To remedy this, we perform post-hoc bias correction [28, 49, 57, 72, 73, 80, 84] to adjust model outputs for the natural distribution.

Bias Correction. We perform bias correction in two stages. After training has finished, we save a copy of all model parameter weights. We then fine-tune the model for a small fixed number of epochs on all data in the replay buffer using standard mini-batches (i.e., from the natural distribution). After fine-tuning, we perform class-specific bias correction on predicates for $(s,?,o)$ questions and on attributes for $(s,p,a?)$ questions since they require a class as the answer. Correction is performed on these two question types independently.

The class-specific bias correction stage learns to adjust class-specific distances to the natural distribution. Our setup is reminiscent of Platt scaling [56], which has been effective for model calibration [22]. Specifically, we compute network predictions for all $(s,?,o)$ or $(s,p,a?)$ questions. We then compute target embeddings for all predicates or attributes in the class dictionary. We then compute the negative Euclidean distance of each predicted embedding to all target embeddings, which yields a score for each class. Finally, we train two parameters per class, α and β, to correct each class score: $s \leftarrow \alpha s + \beta$. These parameters are trained by minimizing a cross-entropy loss between the corrected score and the true label. After these stages, we evaluate the model and reset the model parameters back to their values from before bias correction. Resetting the parameters allows the model to perform better on balanced distributions and only use bias correction for evaluation in imbalanced settings.

2.3 Query-by-Category Framework

A challenge in using conventional active sampling methods is that they rely on the following assumptions: they are being applied on a balanced set of classes, performance is measured using accuracy on the test data, and they ask for the label of a specific example. We argue that long-tailed distributions are more common in the real-world and agents should have the ability to ask questions beyond asking for just a label. To this end, we propose the Query-by-Category (QBCat) active learning framework that allows a learner to ask for examples from specific classes (Fig. 2). To query an oracle for P samples, the protocol is: 1) provide a dictionary of attribute and predicate classes to the learner; 2) the learner computes an uncertainty

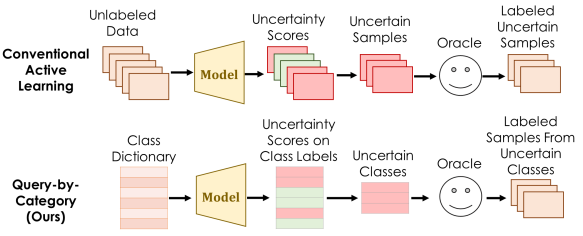


Figure 2: In classical active learning, a model computes uncertainty scores for unlabeled examples and an oracle labels uncertain examples. In our framework, a model computes uncertainty scores for *classes* and an oracle provides examples from uncertain classes.

score for each class; 3) the learner uses weighted random sampling with class uncertainty scores as weights to select the class distribution for the P samples; 4) the learner queries an oracle for P samples using the class distribution from 3); and 5) the provided samples are combined with replay data and the model is updated. See Alg. S1 for an overview.

Using this framework, we propose a simple active sampling method called **QBCat-Tail** that prioritizes rare classes. This method assumes that a pre-training phase occurred on head classes of the dataset. During active learning, it assigns class uncertainty scores to tail classes *uniformly at random*. An oracle then provides the learner with samples from the selected classes uniformly at random. While simple, classical active learning methods often do not consider the label distribution of the data to be learned, which can hinder their performance.

2.4 Model Architecture

Our model architecture takes an incomplete triple (question) as input and outputs a prediction for the missing element in the triple (see Fig. 3). Each subject, object, or predicate is represented as a vector. For subjects and objects that are bounding boxes, we ROI pool ground truth box features from a Faster R-CNN object detector [58] pre-trained on MS-COCO [40] (see Sec. S2.4). For predicates and objects that are strings, we use a lookup-table embedding.

Each triple contains two known elements and one missing element. The elements are either subject/object vectors $s, o \in \mathbb{R}^{d_o}$, a predicate label $p \in \mathbb{R}$, or an attribute label $a \in \mathbb{R}$. We define a feed-forward network $F_O : \mathbb{R}^{d_o} \rightarrow \mathbb{R}^d$ and two embedding layers, F_A and F_P, to embed subjects/objects, attributes, or predicates in d-dimensional space respectively. We define one trainable $\mathtt{N} \in \mathbb{R}^d$ ("Null") per data type to represent the missing triple element, i.e., $\mathtt{N}_A$, $\mathtt{N}_P$, and $\mathtt{N}_O$ represent a missing attribute, predicate, or subject/object respectively.

The two known elements in the triple are embedded in d-dimensional space using the corresponding embedding network/layers, i.e., $h_s = F_O(s)$, $h_o = F_O(o)$, $h_p = F_P(p)$, or $h_a = F_A(a)$. Then, the two embedded representations are concatenated with the appropriate trainable vector $\mathtt{N}$ to yield an embedded vector $h \in R^{3d}$, where $\mathtt{N}$ is concatenated in the exact location of the missing triple element, e.g., the question $q = (s, ?, o)$ maps to $h = [h_s; \mathtt{N}_P; h_o]$. This vector h is processed by another feed-forward network $G : \mathbb{R}^{3d} \rightarrow \mathbb{R}^d$, which outputs a final predicted representation for the missing triple element. Similarly, we define a target embedding feed-forward network F_{OT} and two target embedding layers F_{AT} and F_{PT} to embed target subject/objects, attributes, and predicates in d-dimensional space, respectively.

Metric Learning Loss. Given a mini-batch of M questions, a predicted embedding h_q, its associated target embedding t_q, and all pairs between h_u and t_v in the mini-batch, we

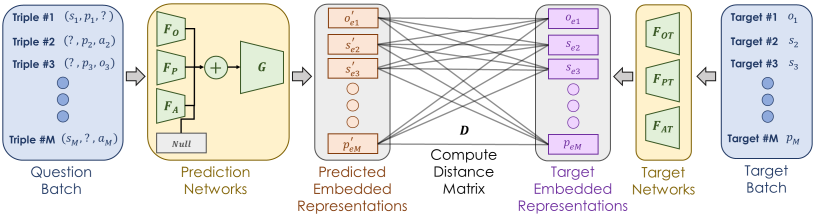


Figure 3: Our architecture for a mini-batch with M questions (triples) and targets. Question elements are processed by their respective embedding network/layer (F_O, F_P, F_A), concatenated with a trainable `Null`, and processed by G, which outputs *predicted* embeddings. Targets are processed by their respective embedding network/layer (F_{OT}, F_{PT}, F_{AT}) to yield target embeddings. A cross-entropy loss is then computed between a pairwise distance matrix of all predicted and target embedding pairs (D) and a target identity matrix via Eq. 1.

compute the categorical cross-entropy loss as:

$$\mathcal{L} = -\log\left(\frac{\exp\left(-\|h_q - t_q\|^2\right)}{\sum_{u,v}\exp\left(-\|h_u - t_v\|^2\right)}\right) \quad , \tag{1}$$

which encourages positive pairs to be embedded close to one another in feature space. This is equivalent to Neighborhood Component Analysis loss [20]. Any distance function could be used, but Euclidean distance worked best in early experiments, so we use it here.

Model Inference. We evaluate the model in three ways depending on if the answer is a subject/object, a predicate, or an attribute. First, we compute a predicted embedding for the question. When the target is a subject/object, we exploit the fact that the answer is limited to objects in the same image and compute target embeddings for each object in the image. For predicate/attribute targets, we compute target embeddings for all predicates/attributes. We then compute the negative Euclidean distance between the predicted embedding and all question-specific target embeddings to create a score vector for evaluation.

3 Related Work

Link Prediction. In link prediction, a neural network is provided with two objects and it predicts the relationship ("link") between them. In our setting, a network performs link prediction between objects and node prediction of objects and attributes. Early link predictors were shallow networks that modeled relationships using simple algebraic operations [4, 42, 52, 54, 70, 78, 81] (see [53] for a review). Recently, deep networks have demonstrated more success for modeling relationships due to their expressive power. Knowledge Vault uses a multi-layer perceptron to learn from concatenated subject, relationship, and object embeddings [15]. It has been applied to textual data. We extend Knowledge Vault to visual scenes due to its simplicity, which allows us to focus on active sampling.

Long-Tailed Learning. In long-tailed learning, methods tend to overfit to frequent classes and not generalize to rare classes. There are three main ways to train with imbalanced data (see [85] for a survey): 1) re-balancing classes (e.g., via re-sampling [7, 16, 30, 48, 66], loss adjustment [6, 11, 37, 71], or logit adjustment [28, 49, 57, 72, 73, 80, 84]), 2) transferring information from a pre-training stage or from more to less frequent classes [44, 77], or 3)

improving model performance via classifier design [35, 43, 80] or ensembling [23, 76, 86]. Re-balancing strategies are simplest to implement and usually achieve comparable performance to more complex methods, so we use them here. Only a few long-tailed active learning approaches have been proposed [1, 3, 9, 21]. Similar to [1], we modify the training setup to make methods more amenable to imbalanced data.

Active Learning. Active sampling methods attempt to select the fewest informative examples to be labeled by an oracle [10, 41, 63, 75, 79, 83]. Active learning strategies fall into three categories: model uncertainty [12, 13, 14, 39, 61], diversity-based sampling [21, 24, 51, 62], and expected model change [18, 34, 65, 74]. Uncertainty can be quantified using entropy [27, 45, 64], discriminator scores [68], margins between class probabilities [32, 59], ensembling [2, 19], or model loss [83]. Uncertainty sampling is simple and easy to use with neural networks, so we use it here. Beyond this, there have been active sampling methods designed to discover rare classes and evaluated on standard classification tasks [26, 29, 33]. While active learning has been widely explored for classification [2, 62, 75], detection [17, 25, 60, 69], and semantic segmentation [36, 46, 67, 82], its exploration for node/link prediction has been limited [5, 8, 47, 55]. Most similar to our work are [47, 55], which meta-learn an uncertainty-based active sampler. Conversely, we are the first to perform active learning for node/link prediction on visual scenes.

4 Experimental Setup

4.1 Baselines

Active sampling is performed per question type, i.e., we select an equal number of samples from each question type from Sec. 2. We compare four strategies that assign a weight of how likely each unlabeled sample is to be chosen: **Random** is simplest and uses uniform random weights. **Least Confident** computes the highest class score for an example, which is then inverted so examples with the smallest top scores are prioritized. **Minimum Margin** computes the margin between the highest and second highest score for an example [32, 59]. We invert this score such that samples with smaller margins receive larger weights. **Maximum Entropy** computes the entropy of softmaxed scores across classes [45, 64, 83].

We first assign a weight to each sample in the unlabeled dataset using one of these strategies. We then shift each weight, w_i, such that the smallest weight across samples is 1: $w_i \leftarrow w_i + (1 - \min_j w_j)$. Probabilities are then defined as $p_i = w_i / \sum_j w_j$ and weighted random sampling is used to select samples to be labeled by an oracle. We also compare two baselines. The pre-train baseline is trained on only pre-training data and evaluated immediately after. Pre-train is used to initialize the model for each active learner and is a lower bound. We also train an offline model on all training data, which is an upper bound. Both models are trained with standard mini-batches without bias correction.

4.2 Visual Genome Dataset

We conduct experiments on Visual Genome 1.4 [38]. We pre-process the dataset following [31], i.e., we partition the data into train (80%), val (10%), and test (10%) sets, and filter objects based on size, number of occurrences, and number of relationships. This yields 62,565 train, 5,062 val, and 5,096 test images. We then form three questions from each relationship triple and each attribute triple in an image. This yields 3,474,969 train, 279,273

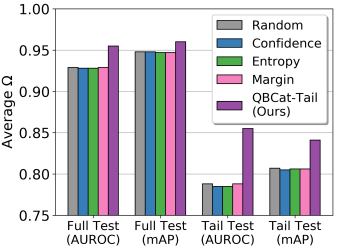


Figure 4: Average Ω performance of active learners over all 10 increments and five question types evaluated on the full and tail test sets.

val, and 281,739 test questions (triples). The final dataset has 253 unique attribute classes (which includes object classes) and 46 unique predicate classes. All (s, p, a) triples use the same predicate ("has attribute"). Histograms of attributes and predicates are in Fig. S1.

4.3 Evaluation Protocol and Metrics

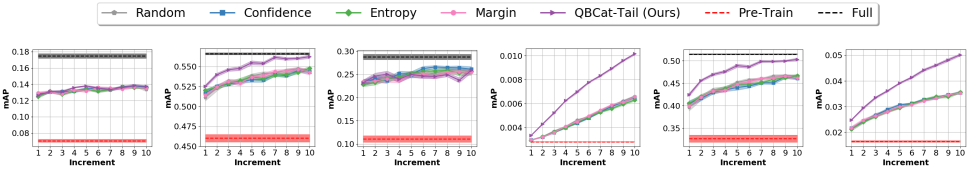
We compute performance on the full test set, as well as a test set consisting of samples from the tail of the attribute and predicate distributions. We define head classes as those containing more samples than the mean across counts over all classes for predicates and attributes separately. This yields 66 head and 187 tail classes for attributes and 9 head and 37 tail classes for predicates. More details and class lists are in Sec. S3.

Given score vectors (from Sec. 2.4) and one-hot encodings of the answer, we compute two metrics for each question type: area under the receiver operating characteristic (AUROC) curve and mean average precision (mAP). When a subject/object is the target, we use sample-wise averaging across questions since images do not contain a uniform number of objects, i.e., score vectors are different lengths. When a predicate or attribute is the target, we use micro averaging. mAP emphasizes positive classes, making it more ideal for long-tailed datasets. It is also useful to summarize performance over increments. To do this, we define performance of an offline upper bound as γ_{offline}. Given the performance of a learner at increment t as γ_t, overall performance is given by: $\Omega = \frac{1}{T}\sum_{t=1}^{T}\left[1-(\gamma_{\text{offline}}-\gamma_t)\right]$ over T increments. If the learner performed as well as the offline method, then $\Omega = 1$. Higher Ω values are better. Ω makes comparisons across question types, metrics, and test sets easier.

5 Results

Each experiment was conducted with 10 random network initializations and we report the average over runs. After pre-training, all methods performed incremental active learning over 10 increments, where each active sampling method chose 100 samples from each of the six question types at each increment (i.e., 600 selected samples total per increment). Additional implementation details and hyper-parameters are in Sec. S2. While $(s, ?, a)$ questions are used during training, we do not report performance on them since it is uninteresting, i.e., it only requires the model to output the predicate "has attribute." Our QBCat-Tail method uses the class breakdown from Fig. S1 for determining which classes belong to the tail.

In Fig. 4, we plot the Ω scores of each method averaged over all five question types



(a) Full Test $(s,p,a?)$ (b) Full Test $(?,p,o)$ (c) Full Test $(s,?,o)$ (d) Tail Test $(s,p,a?)$ (e) Tail Test $(?,p,o)$ (f) Tail Test $(s,?,o)$

Figure 5: Incremental performance on the full and tail test sets for various question types. Each curve is the average over 10 runs with shaded standard error. For clarity, the offline upper bound has been removed from the tail plots for $(s,p,a?)$ and $(s,?,o)$, where the offline baseline achieved an average mAP of 0.312 and 0.263 respectively. More plots are in Sec. S4.

on each test set using AUROC and mAP. Raw Ω scores are in Table S2. When evaluated on the tail test set, our QBCat-Tail method outperforms baselines by a large margin. For example, it outperforms the closest baselines by 6.7% in average Ω AUROC and 3.4% in average Ω mAP. While performance differences are smaller on the full test set, our QBCat-Tail method outperforms the closest baselines by 2.6% and 1.2% in average Ω AUROC and mAP, respectively. The main advantage of our method is its ability to achieve strong performance on tail data without sacrificing performance on the natural data distribution. This strong performance could be a result of training on a balanced distribution of head and tail data, which could reduce over-fitting on head data. By reducing over-fitting, the model could be learning more generalized representations that improve overall performance.

To explore performance differences further, we show mAP learning curves for each active learning method in Fig. 5. Additional learning curves are in Sec. S4 and show similar trends. Across all increments, our QBCat-Tail method either rivals or outperforms alternative methods across question types and test sets. When evaluated on the full test set, our QBCat-Tail method outperforms all methods by a large margin on box-based questions (i.e., $(?,p,a)$, $(?,p,o)$, and $(s,p,o?)$). For questions where the agent must predict a predicate $(s,?,o)$ or an attribute $(s,p,a?)$, performance of the QBCat-Tail method on the full dataset is similar to other active learning strategies. QBCat-Tail achieves significant performance improvements across question types on the tail test set. Overall, box-based questions appear to be easiest, yielding the highest performance values, while the $(s,?,o)$ and $(s,p,a?)$ questions are harder, yielding the lowest performance values. This implies that it is more difficult for learners to predict specific attribute or predicate classes. Performance differences among baseline active learners across test sets are minimal. This is because baseline methods do not explore the tail of the distribution and oversample head data (see histograms in Sec. 5.1). Additional studies of active learning baselines on only tail data are in Sec. 5.1.

5.1 Additional Studies

Next, we study several components of our training procedure to identify which yield the most improvement. In Table 1, we study the Ω performance of QBCat-Tail in four settings: **Standard Mini-Batches** uses standard mini-batch construction, i.e., batches are sampled uniformly at random without replacement. **Without Bias Correction** uses re-balanced mini-batches without bias correction. **Main Setup** uses re-balanced mini-batches, bias correction, and selects tail classes uniformly at random. **Frequency Probabilities** assigns each tail class

Table 1: Ω mAP performance for various versions of our QBCat-Tail method. Main Setup denotes the main model that uses re-balanced mini-batches, bias correction, and uniform class probabilities. Each result is the average over 10 runs.

STUDY	$(?,p,a)$	$(s,p,a?)$	$(?,p,o)$	$(s,?,o)$	$(s,p,o?)$
Full Test Set					
Without Bias Correction	0.864	0.871	0.941	0.780	0.941
Standard Mini-Batches	0.870	0.952	0.968	0.919	0.955
Main Setup	0.919	0.959	0.985	0.958	0.979
Frequency Probabilities	**0.921**	**0.962**	**0.990**	**0.973**	**0.981**
Tail Test Set					
Without Bias Correction	0.770	0.692	0.948	**0.796**	0.916
Standard Mini-Batches	0.669	0.691	0.921	0.756	0.894
Main Setup	0.819	0.695	0.965	0.776	0.949
Frequency Probabilities	**0.828**	**0.696**	**0.971**	0.783	**0.954**

a probability equal to $\frac{num\_samples\_in\_class}{num\_samples\_in\_dataset}$ with re-balanced mini-batches and bias correction.

On the full test set, performing bias correction is critical to model performance. Without bias correction, performance on $(s,?,o)$ and $(s,p,a?)$ questions drops by 17.8% and 8.8% respectively. On both test sets, using standard mini-batches yields slightly worse performance than using re-balanced batches, with performance on $(?,p,a)$ questions being the most negatively impacted. Surprisingly, performing bias correction improves performance on the tail classes for all question types except $(s,?,o)$. Frequency-based probabilities yield slightly better performance than uniform probabilities with a maximum performance difference of 1.5% across questions and test sets. Future work could explore more methods for assigning tail probabilities (e.g., using class counters, pseudo-labeling, etc.).

6 Discussion and Conclusion

In this work, we introduced a new active learning framework and demonstrated its effectiveness on a new problem domain (active learning for visual triple completion). Specifically, we proposed new methods and training paradigms for incremental active learning of long-tailed attributes and relationships. We then introduced the Query-by-Category (QBCat) active learning setup which changes the framing of how agents ask oracles for more training data. It allows models to ask for an example of a particular class instead of asking for the label of an uncertain example. We then proposed the QBCat-Tail method and showed that when combined with suitable re-biasing, it performs comparably to existing active learning methods on the natural long-tailed distribution, and demonstrates significant performance improvements on tail classes. These contributions provide future researchers with a new active learning protocol, as well as a framework for studying how agents should ask for labels.

In the future, it would be interesting to extend to settings where more than one element in a triple (or a higher order relation) is missing, or to relations spanning multiple images. While we focused on uncertainty-based active learning methods due to their simplicity, it would be interesting to explore diversity-based methods that could exploit the metric space learned by our model. We demonstrated that asking for specific classes improved performance over standard active sampling methods. It would be interesting to develop methods that exploit this finding to improve tail performance further. Overall, our framing encourages future studies on how agents should pose questions to an oracle to best improve performance.

Acknowledgements. TH and CK were supported in part by the DARPA/SRI Lifelong Learning Machines program [HR0011-18-C-0051], NSF award #1909696, and NSF award #2047556. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies or endorsements of any sponsor. We thank Robik Shrestha, James Arnold, and Manoj Acharya for their comments and useful discussions.

References

[1] Umang Aggarwal, Adrian Popescu, and Céline Hudelot. Active learning for imbalanced datasets. In *WACV*, pages 1428–1437, 2020.

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

[3] Aditya R Bhattacharya, Ji Liu, and Shayok Chakraborty. A generic active learning framework for class imbalance applications. In *BMVC*, page 121, 2019.

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *NeurIPS*, 26, 2013.

[5] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. Active learning for graph embedding. *arXiv preprint arXiv:1705.05085*, 2017.

[6] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *NeurIPS*, 2019.

[7] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[8] Ke-Jia Chen, Jingyu Han, and Yun Li. Hallp: A hybrid active learning approach to link prediction task. *J. Comput.*, 9(3):551–556, 2014.

[9] Jongwon Choi, Kwang Moo Yi, Jihoon Kim, Jinho Choo, Byoungjip Kim, Jinyeop Chang, Youngjune Gwon, and Hyung Jin Chang. Vab-al: Incorporating class imbalance and difficulty with variational bayes for active learning. In *CVPR*, pages 6749–6758, 2021.

[10] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.

[11] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, pages 9268–9277, 2019.

[12] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pages 746–751, 2005.

[13] Ido Dagan and Sean P Engelson. Committee-based sampling for training probabilistic classifiers. In *Machine Learning Proceedings 1995*, pages 150–157. Elsevier, 1995.

[14] Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *ICML*, pages 208–215. ACM, 2008.

[15] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–610, 2014.

[16] Chris Drumnond. Class imbalance and cost sensitivity: Why undersampling beats oversampling. In *ICML-KDD Workshop*, volume 3, 2003.

[17] Di Feng, Xiao Wei, Lars Rosenbaum, Atsuto Maki, and Klaus Dietmayer. Deep active learning for efficient training of a lidar 3d object detector. In *Intelligent Vehicles Symposium*, pages 667–674. IEEE, 2019.

[18] Alexander Freytag, Erik Rodner, and Joachim Denzler. Selecting influential examples: Active learning with expected model output changes. In *ECCV*, pages 562–577. Springer, 2014.

[19] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *ICML*, pages 1183–1192. PMLR, 2017.

[20] Jacob Goldberger, Geoffrey E Hinton, Sam Roweis, and Russ R Salakhutdinov. Neighbourhood components analysis. *NeurIPS*, 17, 2004.

[21] Denis Gudovskiy, Alec Hodgkinson, Takuya Yamaguchi, and Sotaro Tsukizawa. Deep active learning for biased datasets via fisher kernel self-supervision. In *CVPR*, pages 9041–9049, 2020.

[22] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, pages 1321–1330, 2017.

[23] Hao Guo and Song Wang. Long-tailed multi-label visual recognition by collaborative training on uniform and re-balanced samplings. In *CVPR*, pages 15089–15098, 2021.

[24] Yuhong Guo. Active instance sampling via matrix partition. In *NeurIPS*, pages 802–810, 2010.

[25] Elmar Haussmann, Michele Fenzi, Kashyap Chitta, Jan Ivanecky, Hanson Xu, Donna Roy, Akshita Mittel, Nicolas Koumchatzky, Clement Farabet, and Jose M Alvarez. Scalable active learning for object detection. In *Intelligent Vehicles Symposium*, pages 1430–1435. IEEE, 2020.

[26] Jingrui He and Jaime G Carbonell. Rare class discovery based on active learning. 2008.

[27] Alex Holub, Pietro Perona, and Michael C Burl. Entropy-based active learning for object recognition. In *CVPRW*, pages 1–8. IEEE, 2008.

[28] Youngkyu Hong, Seungju Han, Kwanghee Choi, Seokjun Seo, Beomsu Kim, and Buru Chang. Disentangling label distribution for long-tailed visual recognition. In *CVPR*, pages 6626–6636, 2021.

[29] Timothy M Hospedales, Shaogang Gong, and Tao Xiang. Finding rare classes: Active learning with generative and discriminative models. *IEEE Transactions on Knowledge and Data Engineering*, 25(2):374–386, 2011.

[30] Xinting Hu, Yi Jiang, Kaihua Tang, Jingyuan Chen, Chunyan Miao, and Hanwang Zhang. Learning to segment the tail. In *CVPR*, pages 14045–14054, 2020.

[31] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *CVPR*, 2018.

[32] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *CVPR*, pages 2372–2379. IEEE, 2009.

[33] Christoph Käding, Alexander Freytag, Erik Rodner, Paul Bodesheim, and Joachim Denzler. Active learning and discovery of object categories in the presence of unnameable instances. In *CVPR*, pages 4343–4352, 2015.

[34] Christoph Käding, Erik Rodner, Alexander Freytag, and Joachim Denzler. Active and continuous exploration with deep neural networks and expected model output changes. *arXiv preprint arXiv:1612.06129*, 2016.

[35] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *ICLR*, 2020.

[36] Tejaswi Kasarla, Gattigorla Nagendar, Guruprasad M Hegde, Vineeth Balasubramanian, and CV Jawahar. Region-based active learning for efficient labeling in semantic segmentation. In *WACV*, pages 1109–1117. IEEE, 2019.

[37] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8): 3573–3587, 2017.

[38] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalanditis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 123(1):32–73, 2017.

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

[40] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014.

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

[42] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2015.

[43] Jialun Liu, Yifan Sun, Chuchu Han, Zhaopeng Dou, and Wenhui Li. Deep representation learning on long-tailed data: A learnable embedding augmentation perspective. In *CVPR*, pages 2970–2979, 2020.

[44] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *CVPR*, pages 2537–2546, 2019.

[45] Wenjie Luo, Alex Schwing, and Raquel Urtasun. Latent structured active learning. *NeurIPS*, 26:728–736, 2013.

[46] Radek Mackowiak, Philip Lenz, Omair Ghori, Ferran Diego, Oliver Lange, and Carsten Rother. Cereals-cost-effective region-based active learning for semantic segmentation. In *BMVC*, 2018.

[47] Kaushalya Madhawa and Tsuyoshi Murata. Active learning on graphs via meta learning. In *ICML Workshop on Graph Representation Learning and Beyond, ICML*, 2020.

[48] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, pages 181–196, 2018.

[49] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. In *ICLR*, 2021. URL https://openreview.net/forum?id=37nvvqkCo5.

[50] Diganta Misra. Mish: A self regularized non-monotonic neural activation function. In *BMVC*, 2020.

[51] Hieu T Nguyen and Arnold Smeulders. Active learning using pre-clustering. In *ICML*, page 79, 2004.

[52] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, 2011.

[53] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1): 11–33, 2015.

[54] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *AAAI*, 2016.

[55] Natalia Ostapuk, Jie Yang, and Philippe Cudré-Mauroux. Activelink: deep active learning for link prediction in knowledge graphs. In *The World Wide Web Conference*, pages 1398–1408, 2019.

[56] John C Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, 1999.

[57] Foster Provost. Machine learning from imbalanced data sets 101. In *AAAIW*, pages 1–3. AAAI Press, 2000.

[58] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[59] Dan Roth and Kevin Small. Margin-based active learning for structured output spaces. In *ECML*, pages 413–424. Springer, 2006.

[60] Soumya Roy, Asim Unmesh, and Vinay P Namboodiri. Deep active learning for object detection. In *BMVC*, volume 362, page 91, 2018.

[61] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer, 2001.

[62] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *ICLR*, 2018. URL https://openreview.net/forum?id=H1aIuk-RW.

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

[64] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *EMNLP*, pages 1070–1079, 2008.

[65] Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. *NeurIPS*, 20:1289–1296, 2007.

[66] Li Shen, Zhouchen Lin, and Qingming Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *ECCV*, pages 467–482. Springer, 2016.

[67] Yawar Siddiqui, Julien Valentin, and Matthias Nießner. Viewal: Active learning with viewpoint entropy for semantic segmentation. In *CVPR*, pages 9433–9443, 2020.

[68] Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. In *ICCV*, pages 5972–5981, 2019.

[69] Sayanan Sivaraman and Mohan M Trivedi. Active learning for on-road vehicle detection: A comparative study. *Machine Vision and Applications*, 25(3):599–611, 2014.

[70] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *NeurIPS*, pages 926–934, 2013.

[71] Jingru Tan, Changbao Wang, Buyu Li, Quanquan Li, Wanli Ouyang, Changqing Yin, and Junjie Yan. Equalization loss for long-tailed object recognition. In *CVPR*, pages 11662–11671, 2020.

[72] Kaihua Tang, Jianqiang Huang, and Hanwang Zhang. Long-tailed classification by keeping the good and removing the bad momentum causal effect. In *NeurIPS*, 2020.

[73] Junjiao Tian, Yen-Cheng Liu, Nathan Glaser, Yen-Chang Hsu, and Zsolt Kira. Posterior re-calibration for imbalanced datasets. In *NeurIPS*, 2020.

[74] Alexander Vezhnevets, Joachim M Buhmann, and Vittorio Ferrari. Active learning for semantic segmentation with expected change. In *CVPR*, pages 3162–3169. IEEE, 2012.

[75] Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600, 2016.

[76] Tao Wang, Yu Li, Bingyi Kang, Junnan Li, Junhao Liew, Sheng Tang, Steven Hoi, and Jiashi Feng. The devil is in classification: A simple framework for long-tail instance segmentation. In *ECCV*, pages 728–744. Springer, 2020.

[77] Yiru Wang, Weihao Gan, Jie Yang, Wei Wu, and Junjie Yan. Dynamic curriculum learning for imbalanced data classification. In *ICCV*, pages 5017–5026, 2019.

[78] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.

[79] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *ICML*, pages 1954–1963, 2015.

[80] Tong Wu, Ziwei Liu, Qingqiu Huang, Yu Wang, and Dahua Lin. Adversarial robustness under long-tailed distribution. In *CVPR*, pages 8659–8668, 2021.

[81] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[82] Lin Yang, Yizhe Zhang, Jianxu Chen, Siyuan Zhang, and Danny Z Chen. Suggestive annotation: A deep active learning framework for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 399–407. Springer, 2017.

[83] Donggeun Yoo and In So Kweon. Learning loss for active learning. In *CVPR*, pages 93–102, 2019.

[84] Songyang Zhang, Zeming Li, Shipeng Yan, Xuming He, and Jian Sun. Distribution alignment: A unified framework for long-tail visual recognition. In *CVPR*, pages 2361–2370, 2021.

[85] Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *arXiv preprint arXiv:2110.04596*, 2021.

[86] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *CVPR*, pages 9719–9728, 2020.