IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 23, 2022, accepted March 22, 2022, date of publication March 25, 2022, date of current version April 4, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3162253

How Much a Model Be Trained by Passive Learning Before Active Learning?

DAE UNG JO[1,2], SANGDOO YUN[3], AND JIN YOUNG CHOI[1], (Member, IEEE)

[1]Department of Electrical and Computer Engineering, Automation and Systems Research Institute, Seoul National University, Seoul 08826, South Korea
[2]Samsung Advanced Institute of Technology, Samsung Electronics, Suwon 16678, South Korea
[3]NAVER AI Laboratory, Seongnam-si 13561, South Korea

Corresponding author: Jin Young Choi (jychoi@snu.ac.kr)

This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea Government [Ministry of Science and Information and Communications Technology (MSIT)], Development of High Performance Visual Bigdata Discovery Platform for Large-Scale Realtime Data Analysis, under Grant 2014-3-00123.

ABSTRACT Most pool-based active learning studies have focused on query strategy for active learning. In this paper, via empirical analysis on the effect of passive learning before starting active learning, we reveal that the amount of data acquired by passive learning significantly affects the performance of active learning algorithms. In addition, we confirm that the best amount of data that should be acquired by passive learning depends on the given settings: network complexity, query strategy, and datasets. Inspired by these observations, we propose a method to automatically determine the starting point of active learning for the given settings. To this end, we suggest entropy of sample-uncertainty to measure the training degree of a target model and develop three empirical formulas to determine an appropriate entropy of sample-uncertainty that should be obtained by passive learning before starting active learning. The effectiveness of the proposed method is validated by extensive experiments on popular image classification benchmarks and query strategies.

INDEX TERMS Active learning, deep learning, labeling, machine learning.

I. INTRODUCTION

In recent years, deep learning-based algorithms have made a breakthrough in many areas of machine learning tasks such as image classification [1]. The success of deep learning is attributed to the huge amount of supervised data. Recent studies have reported that increasing amounts of data can improve the performance of a model [2]. However, in actual applications, obtaining a tremendous amount of supervised data is impractical due to the limitation of time and cost.

Recently in the deep learning field, active learning has emerged as one way to efficiently collect supervised data. The key idea of active learning is to efficiently improve the performance of the target model by actively selecting the data to be labeled through an algorithm, rather than randomly selecting the data to be labeled. There are many scenarios for active learning [3], but recent studies have mainly considered a pool-based active learning scenario. The pool-based scenario consists of two learning phases [3]. First, the model is trained with a small amount of supervised data which are randomly selected from an unlabeled data pool and are labeled by a human. This process is referred to as *Passive Learning* (PL). Next, a *query strategy* suggests worthy samples in the unlabeled data pool. The suggested samples are queried to a human and labeled manually. This process is referred to as *Active Learning* (AL). The pool-based scenario is processed over several iterations. PL is performed for the first few iterations (usually 1 iteration), and then AL is performed during the remaining iterations.

Recent deep AL studies have focused on designing effective query strategies. Thus, most previous works overlook PL in the pool-based scenario. But it should be pointed out that lots of query strategies are dependent on the initial state of a model because most query strategies utilize the model prediction outputs [4]–[7], or features extracted from a certain layer of the model [8]. Thus, the amount of PL used to initialize a model before applying AL can have a significant impact on the performance of an AL algorithm. For this reason, it is important to analyze the effect of PL on the performance of AL and to find an appropriate amount of PL before starting AL in the pool-based scenario.

The associate editor coordinating the review of this manuscript and approving it for publication was Yonghong Peng.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

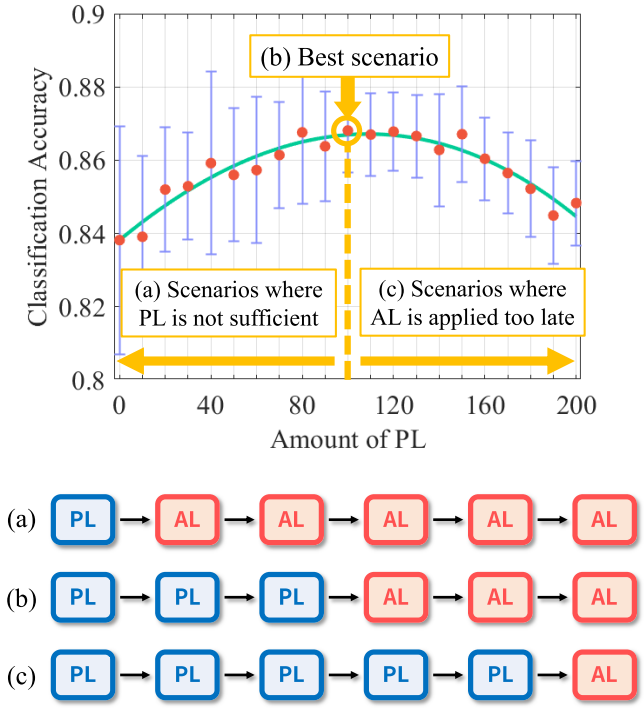


FIGURE 1. An example of PL effects on AL performance. The performance of the AL algorithm can be degraded when the amount of PL is (a) too small or (c) too large before starting AL. The amount of PL refers to the number of data acquired by PL. In this case, we can see the (b) best performance can be achieved when the model is trained by 100 data points acquired by PL before starting AL.

In this paper, we empirically reveal the effect of PL on the performance of an AL algorithm. In particular, we attempt to find the amount of PL before applying AL as shown in Fig. 1. To this end, we conduct extensive experiments that perform pool-based AL scenarios by increasing the amount of PL. Based on the analysis of the results, we present a meaningful trend for an appropriate amount of PL before starting AL. Then, by utilizing the trend, we propose empirical formulas to recommend an appropriate amount of PL without rigorous experiments. For the proposed method, first, we suggest a measure to estimate the training degree of a model, referred to as Entropy of sample-uncertainty (EoU). We utilize EoU as a criterion to determine whether the current amount of PL is sufficient to start AL. Then we analyze the trend of EoU depending on three important settings of AL: network complexity, AL query strategy, and dataset complexity. Based on the trend, we develop empirical formulas to determine an appropriate EoU that should be achieved by PL before AL. Through experiments on MNIST, CINIC, and CIFAR datasets, we validate the effectiveness of our method by applying it to the popular AL algorithms.

Our contributions are summarized as follows.

1) We reveal how the amount of PL affects the performance of an AL algorithm through extensive experiments.
2) We suggest a metric called Entropy of sample-Uncertainty (EoU) to measure the training degree of a model achieved by PL.
3) We develop empirical formulas to automatically determinate an EoU value representing an appropriate amount of PL for a given model, query strategy, and dataset.

II. RELATED WORKS

Recently, the active learning [3] has been applied to various deep learning tasks such as object detection [4], [9], person re-identification [10], multi-task learning [11], named entity recognition [12], human pose estimation [13], action localization [14], and biomedical image analysis [15], [16]. Recent active learning methods can be categorized into two types [17]. The one is the uncertainty-sampling methods [4]–[7], [9], [10], [12], [15] which select the most uncertain (also referred to as informative) samples for the target model. The other is representative subset methods [8], [18], which select the representative subset from the unlabeled data pool.

The core of the uncertainty-sampling methods is to estimate the uncertainty information for the unlabeled sample. In case the target model infers a probability distribution (e.g., image classification), classical methods such as entropy [19] or variational ratio [20] of a probability distribution can be used as uncertainty estimators. Despite its simplicity, classical methods are still utilized in many deep learning applications and show prominent performance [7], [9], [12]–[15], [21].

However, most deep learning applications, such as object detection [22] or human-pose estimation [23], infer deterministic results, instead of probabilistic results. In addition, the classical uncertainty-based active learning methods have a problem with scalability to high dimensional data and a huge number of model parameters [6].

Therefore, recent studies attempt to efficiently estimate the sample-uncertainty in a deep model. Dropout-based method [6] performs multiple forward passes with dropout layers [24] to predict the sample uncertainty. Ensemble-based method [5] utilizes multiple deep neural networks, which have the same structure but are differently initialized. The method proposed in [4] estimates the sample-uncertainty by predicting the loss value of the sample. To this end, they add the additional network (loss prediction module) to the target model. Since the loss function should be defined in any deep learning task, this method can be applied to any deep learning task.

The representative subset methods select the representative subset from the unlabeled data pool. Core-set approach [8] formulated the active learning problem as k-Center problem [25]. The goal of the k-Center problem is to select the k points that maximize the minimum distance among the selected points and their nearest centers. Then the Core-set method solves the problem via integer programming. Variational adversarial active learning method [18]

selects unlabeled data that are not similar to labeled data by training VAE [26] and discriminator adversarially. k-centered clustering algorithms such as k-medoid clustering [27] can be utilized for selecting representative samples by choosing cluster-center [28].

Some recent studies [29], [30] have attempted to combine the two kinds of active learning strategies mentioned before. The method proposed in [29] selects samples that have high uncertainty while preserving the distribution of the dataset. The method proposed in [30] improves its original one [29] by considering the easiness of a sample.

III. ANALYSIS SETTINGS

In this section, first, we introduce pool-based active learning (AL) and define notations. Then, we describe our evaluation scenarios and settings of datasets and models to analyze the effect of passive learning (PL) on the performance of the AL algorithm.

A. POOL-BASED ACTIVE LEARNING

Pool-based AL is an iterative process. At each iteration, the following steps are performed given the labeled dataset D_l, unlabeled dataset D_u, and model $\mathcal{M}$.

First, $\mathcal{M}$ is trained by D_l, and a pooling subset P_u is randomly chosen from D_u. A query strategy will be applied to P_u rather than D_u to mitigate overlapping problem [4], [5]. Next, necessary information to perform a query strategy is collected, such as output map or layer activation map (feature map) of the trained model for samples in D_l and P_u. Then the query strategy suggests queried samples Q_u from P_u, and Q_u are labeled by human experts. Note that if the query strategy is a random selection, the process is referred to as PL. Finally, newly labeled samples are added to D_l and subtracted from D_u. The aforementioned process is repeated until the T-th iteration. In general, the cardinality of Q_u is referred to as budget b which is set to a fixed value for every iteration.

To provide a clear explanation, we define additional notations. Let $\mathbf{x}_i$ be the i-th sample in a dataset. Then p_i^c denotes the predicted probability that $\mathbf{x}_i$ belongs to the $c-$th class, where $c \in \{1, \ldots, C\}$ in a C-class classification problem.

B. EVALUATION SCENARIOS ON THE EFFECT OF PL

The goal of this paper is to reveal the effect of PL on the performance of AL, especially in terms of the amount of PL (i.e., the number of data acquired by PL). In our evaluation scenarios, we have trained a model during T iterations, where in the k-th scenario, the first k iterations adopt PL and the last $(T - k)$ iterations adopt AL. From $k = 0$ to $k = T, T + 1$ scenarios are conducted. Thus, in the k-th scenario, the passively labeled samples and actively labeled samples become $k \times b$ and $(T - k) \times b$, respectively. The total training samples for every scenario are the same as $T \times b$. Each scenario is evaluated by the test accuracy of the trained model.

TABLE 1. For each dataset, the corresponding deep model, b (the number of samples to be labeled at each iteration), T (the number of iterations to be performed during one pool-based AL scenario), and $|P_u|$ (the cardinality of the pooling subset) are given.

Dataset	Model	b	T	$\lvert P_u \rvert$
MNIST	Keras Imple.	10	20	6000
CIFAR-10	ResNet-18	250	20	5000
CINIC-10	ResNet-18	250	20	5000
CIFAR-100	ResNet-18	250	20	5000

C. EVALUATION SETTINGS

1) DATASETS

Image classification is the most popular application for AL studies [4]–[6], [8], [15]. Thus, we performed experiments on four widely used image classification datasets: MNIST [31], CINIC-10 [32], CIFAR-10, and 100 [33]. The details for datasets are provided in Appendix A-B. Table 1 presents AL settings for each dataset. To observe the effect of PL in a fine-grained manner, we set a small b that represents the resolution for a fixed amount of total training samples. T is set to 20 due to a computational limitation: note that the computation burden to perform one evaluation scenario is roughly equivalent to that for learning tons of samples as many as $b \times (2T^3 + 6T^2 + 3T)/6$ samples. For $|P_u|$, we empirically found that when $|P_u|$ was 10% of the full dataset size, we achieved the best performances.

2) MODEL

We employed Keras implementation [34] as a deep model for MNIST dataset. To validate the effect of network complexity on AL, we further experimented on four variations of the original implementation: M1, M2, M3, M4 (original), and M5. The higher numbered network has the higher complexity. For CIFAR and CINIC-10, we employed ResNet-18 [35] as a deep model. A drop-out layer was added before the last fully connected layer with a dropout ratio of 0.5. The first convolution layer of ResNet-18 was modified as '3×3 kernel, stride of 1 and padding of 1' to fit CIFAR and CINIC-10. Detailed implementations and training schemes are provided in Appendix A.

3) QUERY STRATEGY

For analysis, we mainly considered three popular query strategies: Entropy-based Sampling (Entropy) [19], Variational Ratio (VarR) [20], and Deep Bayesian Active Learning (DBAL) [6].

IV. EMPIRICAL ANALYSIS AND METHODOLOGY

In this section, based on the evaluation results of the proposed scenarios, we present our observations on the effect of PL on the performance of AL. From the observations, we found the best amount of PL depends on the target settings: model complexity, query strategy, and datasets, which is related to the training degree of a model. To estimate the training degree of a model, we propose a metric based on the entropy of

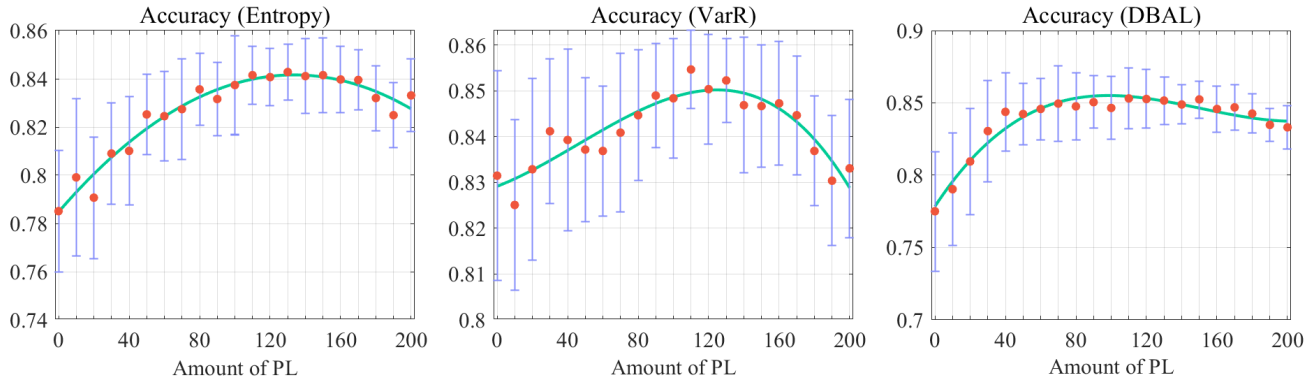


FIGURE 2. The results of our evaluation scenarios on MNIST with three AL methods (Entropy, VarR, and DBAL) and a M1 network model. In each graph, the *x* axis denotes the number of labeled data acquired by PL, and the *y* axis denotes the test performance of the model. The mean and standard deviation of the repeated results are represented as a red point and a blue bar, respectively. A cubic polynomial fitted curve (green curve) of the results is also presented to clearly show the tendency.

sample-uncertainty. Using the metric, we suggest empirical formulas to determine the best amount of PL for AL.

A. EVALUATION RESULTS FOR THE SCENARIOS

Fig. 2 shows the results of the proposed evaluation scenarios for MNIST dataset. See Appendix B for more results on the other datasets and network models. In each graph, the x axis denotes the amount of PL (the number of labeled data acquired by PL) and y axis denotes the test accuracy of the model after completing the pool-based AL scenario. To obtain robust results, the experiments were repeated by 20/5/5 times for MNIST / CINIC-10 / CIFAR, respectively. The mean and standard deviation of the repeated results are presented as red points and blue bars in the graphs. A cubic polynomial fitted curve (green curve) is also illustrated to clearly show the tendency of the results.

B. EFFECT OF PL ON PERFORMANCE OF AL

Based on the experimental results, we found a common tendency. The finally achieved performance is degraded when the amount of PL is too large or too small. First, in the case that the amount of PL is too large, performance degradation is reasonable because the pool-based AL scenario is terminated with little benefits from AL.

Interestingly, the performance can also be degraded when the amount of PL is too small. We analyzed this phenomenon in relation to the training degree of the model, which is achieved by PL. The fundamental goal of AL is to select the most helpful samples for the performance improvement of the *current model*. Most query strategies must be dependent on the training degree of a model [4]–[8]. For example, in the case of uncertainty sampling methods [3], such as Entropy-based sampling [19], unlabeled samples are feed-forwarded into a model, and the model outputs are utilized by a query strategy. Another example is Core-set method [8]. The Core-set does not utilize predicted results, but it also depends on a model since it utilizes the responses of a specific layer of the model as features for a sample. Therefore, if the model does not achieve a certain training degree by a sufficient amount of PL, a query strategy could not choose proper samples for training the model. Poor sample selection can lead to performance degradation of an AL algorithm. According to the above investigations, useful observations are summarized as below.

Observation 1: The amount of PL before AL affects the performance of AL.

By investigating further experimental results (see Appendix B), we discovered additional phenomenons. As shown on graphs given in Appendix B, the best amount of PL is different depending on the target settings: model complexity, query strategy, and datasets. This result means that the sufficient training degree of a model for AL starting can vary depending on the target AL settings. From these phenomenons, we suggest the following observation.

Observation 2: The best amount of PL depends on the target settings: model complexity, query strategy, and datasets.

Based on **Observation 1** and **2**, our objective is to find a method to determine the best amount of PL before AL depending on the target settings. Note that performing extensive experiments on $T + 1$ scenarios to determine the best amount of PL is impractical in actual applications. Therefore, we aim to propose formulas to automatically determine the best amount of PL without conducting the evaluation experiments via multiple scenarios.

In our observation, the amount of PL is highly correlated with the training degree of the target model trained by PL. Hence, to efficiently search for the best amount of PL, we should find a metric that can measure the training degree of a model for given settings. As the metric, we suggest the **entropy of sample-uncertainty (EoU)** in the following section. Using EoU, we can automatically determine the amount of PL before the starting AL in one scenario without an investigation of $T + 1$ scenarios.

C. EoU: TRAINING DEGREE OF A MODEL

To estimate the training degree of a model, we have designed a formula that considers both the trained model $\mathcal{M}$ and the pooling subset P_u. To this end, we have focused on the

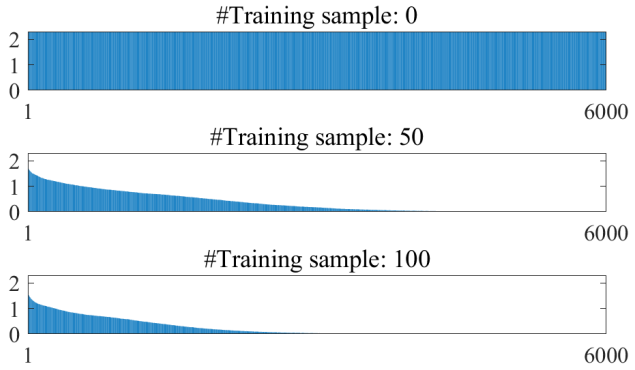


FIGURE 3. Sample-uncertainty (entropy) distribution for unlabeled samples in a pooling subset of MNIST with models of various training degrees. For a clear visualization, the sample-uncertainty values are sorted in descending order.

uncertainty of model responses for an unlabeled sample, which is referred to as *sample-uncertainty* for simplicity.

Sample-uncertainty becomes low for a familiar sample similar to the samples learned already by the model, whereas it becomes high for an unfamiliar sample rarely presented to the model. When the model has not learned any data, it is unfamiliar with all samples in an unlabeled dataset. Therefore, for all unlabeled data, sample-uncertainties are estimated to be high. Then, the sample-uncertainties for uncertain samples in the unlabeled dataset will be uniformly distributed. The more a model learns labeled data, the more unlabeled samples will have low sample-uncertainty. Thus, the distribution of the sample-uncertainties for samples in the unlabeled dataset tends to be concentrated as model-training proceeds.

Fig. 3 shows sample-uncertainties estimated for all unlabeled samples in P_u of MNIST. We trained models with three different sizes of the randomly chosen training dataset. In Fig. 3, as aforementioned, the sample-uncertainties are uniformly distributed when the model is not trained (top graph). And the distribution of sample-uncertainties becomes unbalanced as the model is trained with more training samples (middle and bottom graph). Therefore, using the distribution of sample-uncertainties, we can quantify the training degree of a model.

Entropy [19] can be utilized as a metric to measure the uncertainty of a distribution. To apply entropy, the sample index i is defined as a discrete random variable. Then the probability that the i-th sample is unfamiliar (uncertain) to the model, is defined by normalized sample-uncertainty. Let u_i be the sample-uncertainty of the i-th sample. Then the uncertain probability q_i for the i-th sample is given by

$$q_i = \frac{u_i}{\sum_{j=1}^{M} u_j} \quad i \in 1, \cdots, M, \tag{1}$$

where $M = |P_u|$. Then a normalized entropy for a distribution of sample-uncertainty is defined by

$$H = -\frac{\sum_{i=1}^{M} q_i \log q_i}{\log M}, \tag{2}$$

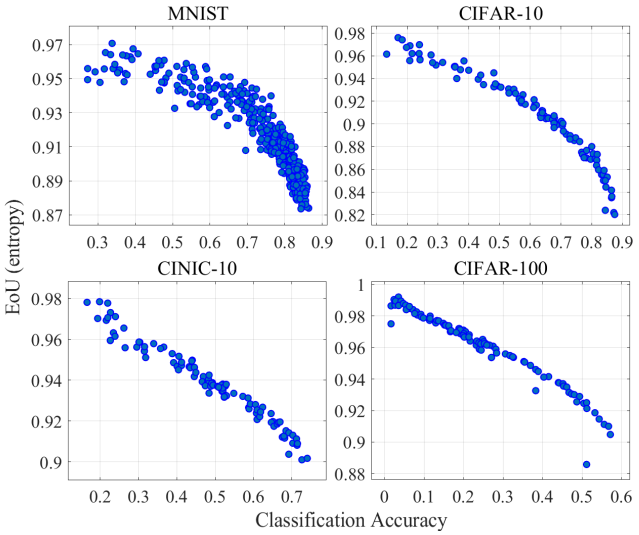


FIGURE 4. Correlation between EoU and the test performance of the model. Models (points in graph) are trained with various amounts of randomly chosen training data.

where $\log M$ is introduced to normalize the entropy to $[0, 1]$ since the maximum entropy depends on the number of unlabeled samples varying during AL process.

In conclusion, we suggest H, referred to as Entropy of sample-Uncertainty (EoU), as a measure to estimate the training degree of a model. Fig. 4 illustrates the correlation between the performance (classification accuracy on the testing dataset) and EoU of the models. The result shows that EoU is negatively correlated with the performance (training degree) of the model. Note that the validation performance can be utilized to measure the training degree of the model. However, the validation performance varies largely depending on the experiment settings and tasks. In contrast, EoU is experimentally normalized to a scalar value between $[0.80, 1.00]$. Thus, EoU provides a more consistent criterion when judging the training degree of a model.

D. DETERMINATION OF THE BEST AMOUNT OF PL

Utilizing EoU, we propose a method to automatically determine the best amount of PL. The proposed method is based on two of our investigations: (1) For an effective AL, a model should achieve a certain training degree, and (2) EoU decreases as the model trained. The concept of the proposed method is simple. Start AL when EoU of the model is less than some scalar value τ. To this end, at each iteration, estimate EoU of the model and compare it to the τ. If EoU becomes less than τ, start AL. The amount of PL at this iteration is regarded as the best amount of PL. Algorithm 1 describes the detailed process of the proposed method. Though Algorithm 1 assumed sample-uncertainty as the Entropy method, other types of sample-uncertainty can be also applied.

Then determination of τ is the remaining problem. Since τ depends on the settings: query strategy, model

Algorithm 1 Proposed Method

Input: Initial model $\mathcal{M}^0$, Initial labeled dataset D_l^0, Initial unlabeled dataset D_u^0, Budget b, Maximum iteration T, size of pooling subset M.
Functions:
$Train(\mathcal{M}, D_l)$: Train model $\mathcal{M}$ with dataset D_l
$Rand(D_u, M)$: Randomly sample M elements from D_u
$Active(D_u, M)$: Actively sample M elements from D_u
Hyper-parameter: τ; EoU (H) threshold for AL starting
Procedure:

for $t = 0$ to $T-1$ **do**
  $\mathcal{M} \leftarrow train(\mathcal{M}^0, D_l^t)$
  $P_u \leftarrow Rand(D_u^t, M)$
  for $\mathbf{x}_i \in P_u$ **do** ▷ for calculating EoU
    $\{p_i^c\}_{c=1}^C = \mathcal{M}(\mathbf{x}_i)$
    $u_i = -\sum_{c=1}^C p_i^c \log p_i^c$
    $q_i = u_i / \sum_{j=1}^M u_j$
  end for
  $H = -\sum_{i=1}^M q_i \log q_i / \log M$ ▷ calculate EoU
  if $H > \tau$ **then**
    $Q_u \leftarrow Rand(P_u, b)$ ▷ perform PL
  else
    $Q_u \leftarrow Active(P_u, b)$ ▷ perform AL
  end if
  $l_q \leftarrow$ Annotate Q_u by human
  $D_l^{t+1} = D_l^t \cup (Q_u, l_q)$
  $D_u^{t+1} = D_u^t \setminus Q_u$
end for
$\mathcal{M} \leftarrow train(\mathcal{M}^0, D_l^T)$

complexity, and dataset, hence, in the following section, we analyzed the dependency of EoU for the given target settings and provided the empirical formulas to determine τ.

E. TREND OF BEST EoU FOR GIVEN TARGET SETTINGS

In this section, we propose empirical formulas to determine the hyper-parameter τ in Algorithm 1. First, we set a base τ_0 and adjust it by the proposed formulas: $\tau_1 = f_1(\tau_0, \alpha_1)$, $\tau_2 = f_2(\tau_1, \alpha_2)$, and $\tau = f_3(\tau_2, \alpha_3)$, where $\alpha_1 \in [0, 1]$ denotes a guessed reliability of the sample-uncertainty depending on the query strategy, $\alpha_2 \in [0, 1]$ a guessed model complexity, and $\alpha_3 \in [0, 1]$ a guessed dataset complexity.

To design the formulas empirically, we analyzed the dependency of EoU for the given target settings. For a concise explanation, we denote the amount of PL and EoU of the trained model by P and H, respectively. The best cases are denoted by P^* and H^*. To see the dependency of the best cases on the settings, P^* and H^* were chosen by those of the scenario closest to the peak of the fitting curve.

TABLE 2. The best amount of PL (P^*) and EoU (H^*) of various model structures and query strategies for MNIST. Higher numbered model are more complex model.

Network	Entropy		VarR		DBAL	
	P^*	H^*	P^*	H^*	P^*	H^*
M1	130	0.937	130	0.921	100	0.976
M2	130	0.918	100	0.902	90	0.971
M3	110	0.908	100	0.911	80	0.961
M4	110	0.908	80	0.914	80	0.955
M5	70	0.911	50	0.898	70	0.946

TABLE 3. The best amount of PL (P^*) and EoU (H^*) of various datasets.

Network	Method	Dataset	P^*	H^*
ResNet-18	Entropy	CIFAR-10	500	0.949
		CINIC-10	250	0.956
		CIFAR-100	4500	0.963
	VarR	CIFAR-10	750	0.926
		CINIC-10	750	0.939
		CIFAR-100	4500	0.956
	DBAL	CIFAR-10	500	0.965
		CINIC-10	500	0.979
		CIFAR-100	4750	0.980

1) DEPENDENCY OF H^* ON QUERY STRATEGY TO DESIGN $f_1(\cdot)$

As shown in the Table 2, H^* varies depending on the query strategy. First, we analyzed the trend of H^* in correlation with the meaning of EoU.

As explained in the previous section, EoU (H) can be interpreted as a measure of the training degree of a model, which depends on the amount of unfamiliar samples to the model in an unlabeled dataset. For the cases of Entropy and VarR, H^* has been measured around $0.90 \sim 0.93$. This means that the query strategy works well when the amount of sample uncertain to the model is around $90 \sim 93\%$ of the unlabeled dataset, i.e., the model needs to understand around $7 \sim 10\%$ of the unlabeled dataset for PL. For DBAL, the model needs to understand only $3 \sim 5\%$ of the unlabeled dataset to utilize the AL query strategy well. In our opinion, the reason that DBAL needs less EoU than others is that DBAL utilizes multiple predictions for one sample. By aggregating multiple predictions, DBAL can estimate more reliable sample-uncertainties than Entropy and VarR which use only one prediction per sample. Then, for DBAL, query strategy can be effectively utilized for a model with a low training degree. Thus it is reasonable that DBAL requires higher H^* than Entropy and VarR.

Since $\alpha_1 \in [0, 1]$ in $f_1(\cdot)$ denotes the reliability of sample-uncertainty estimated by a query strategy, high α_1 means that the estimated sample-uncertainty is reliable.

TABLE 4. Evaluation results depending on the methods to determine the best amount of PL: existing, rigorous, and our guided searches. The average accuracy and standard deviation of repeated 20 / 5 experiments are reported for MNIST / the other datasets, respectively. Bold numbers denote superior performance between the evaluation results of the existing method and the proposed method.

Method	MNIST			CIFAR-10		
	Existing method	Proposed method	Rigorous search	Existing method	Proposed method	Rigorous search
Entropy	83.91 ± 2.21	**86.76 ± 1.95**	86.81 ± 1.14	70.66 ± 2.10	**71.53 ± 0.67**	71.90 ± 1.13
VarR	86.06 ± 1.64	**87.20 ± 1.07**	87.97 ± 0.98	**71.86 ± 1.07**	**71.86 ± 1.07**	72.52 ± 1.44
DBAL	84.55 ± 3.93	**87.69 ± 0.79**	87.89 ± 1.20	70.69 ± 0.95	**71.66 ± 1.07**	71.93 ± 1.33
K-means	78.19 ± 1.55	**80.80 ± 1.61**	84.78 ± 1.25	69.60 ± 0.59	**69.74 ± 1.16**	70.63 ± 0.76
Coreset	76.87 ± 5.55	**84.30 ± 2.38**	85.60 ± 1.75	**71.04 ± 1.15**	69.43 ± 0.65	71.04 ± 1.15
LLAL	80.73 ± 2.42	**83.62 ± 1.40**	84.87 ± 1.10	66.25 ± 1.70	**69.39 ± 1.06**	69.39 ± 1.06
Method	CINIC-10			CIFAR-100		
	Existing method	Proposed method	Rigorous search	Existing method	Proposed method	Rigorous search
Entropy	52.89 ± 0.98	**53.31 ± 0.65**	53.48 ± 0.86	25.12 ± 0.58	**27.03 ± 1.03**	27.49 ± 1.03
VarR	**53.76 ± 0.92**	**53.76 ± 0.92**	53.88 ± 0.49	25.72 ± 0.88	**26.89 ± 0.89**	27.62 ± 0.75
DBAL	53.36 ± 1.37	**53.51 ± 0.82**	53.81 ± 0.61	24.46 ± 1.19	**27.03 ± 1.03**	27.03 ± 1.03
K-means	52.97 ± 0.11	**53.49 ± 0.77**	53.55 ± 0.52	25.79 ± 1.77	**27.03 ± 1.03**	28.07 ± 0.38
Coreset	52.76 ± 1.35	**52.94 ± 1.04**	53.25 ± 0.50	**27.49 ± 1.37**	27.03 ± 1.03	27.70 ± 1.31
LLAL	46.74 ± 1.39	**51.56 ± 1.22**	51.56 ± 1.22	21.03 ± 0.73	**25.85 ± 1.23**	25.90 ± 0.26

Then considering the results on H^* in Table 2 and 3, we adjust τ_0 according to α_1 for the query strategy, following **Formula 1** designed empirically.

Formula 1: Depending on the query strategy, the hyper-parameter τ_1 is determined according to $\alpha_1 \in [0, 1]$ as

$$\tau_1 = f_1(\tau_0, \alpha_1) = \tau_0 + 0.1 \cdot \alpha_1 - 0.05. \tag{3}$$

Remark 1: In the analysis, we have observed a *positive* correlation between the best EoU H^* and the guessed reliability α_1 on the query strategy and so adopted a linear regression model for $f_1(\tau_0, \alpha_1)$ function. First we can set τ_0 as the baseline for Entropy strategy. Then we set the slope and bias to 0.1 and -0.05 for scaling τ_1 into the range of [0, 1] without adjusting τ_0 for middle reliability $\alpha_1 = 0.5$. We provide examples of how to determine τ_0 and α_1. We adopt Entropy strategy for the baseline. Since H^* for Entropy strategy is less than 0.93, we set $\tau_0 = 0.93$ and $\alpha_1 = 0.5$ (median) as a middle reliability for the baseline. VarR has a similar strategy to Entropy. Thus α_1 for VarR are set to the same as that of Entropy. For DBAL, we increase α_1 to 0.7 since its sample-uncertainty is more reliable than that of Entropy as aforementioned. For LLAL which utilizes an add-on module to calculate sample-uncertainty [4], we decrease α_1 to 0.2 since the add-on module only returns scalar values, which requires lots of training to get reliable predictions.

2) DEPENDENCY OF H^* ON MODEL COMPLEXITY TO DESIGN $f_2(\cdot)$

According to the results of Table 2, P^* and H^* values tend to decrease as the model becomes complex (note that a higher-numbered model has more complexity). In general, the more complex a model is, the better the model can learn the given samples by over-fitting. Over-fitting can be beneficial to AL because an over-fitted model can distinguish between untrained (uncertain) samples and trained samples well. As shown in Table 2, a complex model can achieve a similar EoU (H^*) with smaller PL samples (P^*) than a simple model.

Since $\alpha_2 \in [0, 1]$ in $f_2(\cdot)$ is defined by a guessed model complexity, high α_2 means that the model is complex. Then considering the results on H^* in Table 2, we adjust τ_1 according to α_2, following **Formula 2**.

Formula 2: Depending on the model complexity, the hyper-parameter τ_2 is determined according to $\alpha_2 \in [0, 1]$ as

$$\tau_2 = f_2(\tau_1, \alpha_2) = \tau_1 - 0.1 \cdot \alpha_2 + 0.05. \tag{4}$$

Note that α_2 is subtracted because a complex model have lower H^* than a simple model.

Remark 2: **Formula 2** is also designed as a linear function in a similar way to **Formula 1**. However, the signs of slope and bias are reversed to model a *negative* correlation between H^* and α_2. We set the baseline to M4 (original Keras) network. Thus $\alpha_2 = 0.5$ is given for M4 network. We give $\alpha_2 = 0.7$ for ResNet-18 since ResNet-18 is more complex than M4 network.

3) DEPENDENCY OF H^* ON DATASET TO DESIGN $f_3(\cdot)$

Table 3 shows the experimental results for various datasets. CINIC-10 has intermediate complexity between CIFAR-10 and 100 [32]. Regardless of a query strategy, H^* for a complex dataset is higher than that for an easy dataset. We analyzed this result as follows. Under the same conditions (network, budget, etc.), the complex dataset makes a model hardly understand unlabeled data. Thus, for complex datasets, the overall training degree of a model becomes low. Therefore, H^* will also tend to increase for the complex dataset.

Thus we define $\alpha_3 \in [0, 1]$ to represent the dataset complexity in $f_3(\cdot)$. High α_3 means that a dataset is complex. Considering the results on H^* in Table 3, we adjust τ_2 according to α_3, following **Formula 3**.

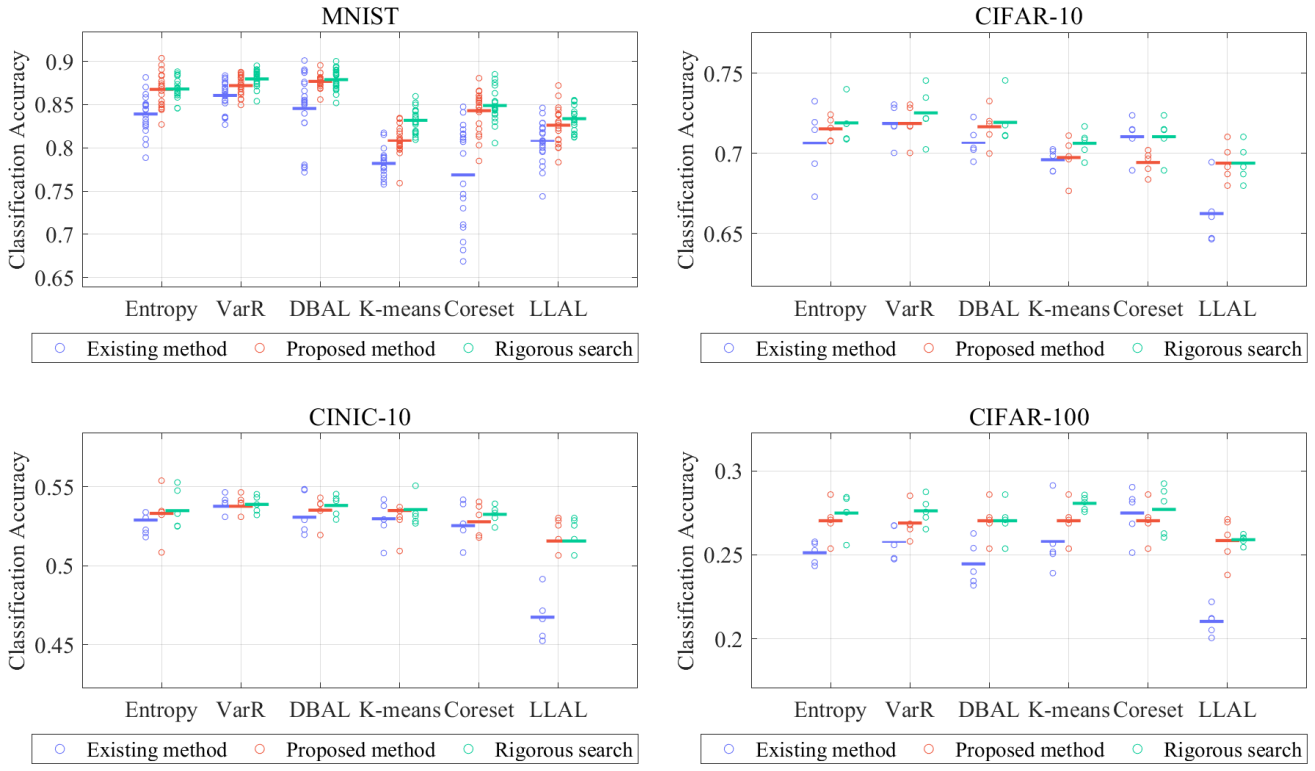


FIGURE 5. Non-parametric analysis for evaluation results. *x* axis denotes AL methods, and *y* axis denotes the test performance of the model. Each circle represents an individual result of repeated experiments, and the horizontal line represents the mean value. For each AL method, the results of the existing method (blue), the proposed method (red), and the rigorous search are presented.

Formula 3: Depending on the dataset complexity, hyper-parameter τ is determined according to $\alpha_3 \in [0, 1]$ as

$$\tau = f_3(\tau_2, \alpha_3) = \tau_2 + 0.1 \cdot \alpha_3 - 0.05. \tag{5}$$

Remark 3: **Formula 3** is also designed as a linear function in a similar way to **Formula 1, 2**. The signs of slope and bias are determined equal to **Formula 1** in order to model a *positive* correlation between H^* and α_3. We set the baseline to MNIST case. Thus $\alpha_3 = 0.5$ is given for MNIST. Then we set $\alpha_3 = 0.8/0.9/1.0$ for CIFAR-10 / CINIC-10 / CIFAR-100, respectively. Finally using **Formulas 1-3**, we can determine the hyper-parameter $\tau \sim H^*$ for Algorithm 1.

V. EXPERIMENTS

A. EVALUATION OF PROPOSED METHOD

To verify the effectiveness of the proposed method to determine the best amount of PL, we compared the three methods. (1) **'Existing method'** represents the method adopted in each AL study, where the pre-determined amount of PL performs only in the first iteration and AL algorithm performs in the remaining iterations. (2) **'Rigorous search'** chooses the best case among $T + 1$ scenarios evaluated according to our evaluation scenarios. The result of the rigorous search is the upper bound of the proposed method. (3) **'Proposed method'** follows the proposed Algorithm 1 where the hyper-parameter τ is determined according to **Formula 1-3**.

We evaluated methods to various AL query strategies including Entropy, VarR, DBAL, K-means, Coreset [8], LLAL [4]. Experimental settings are equivalent to our analysis setting.

Table 4 shows the results. In most cases, the proposed method achieved an equal or better performance than the existing search method. However, for Coreset method, the proposed method achieved degraded performance. In our opinion, the reason is that τ was not accurately estimated since Coreset does not utilize sample-uncertainty, whereas the proposed method is based on EoU. We also provide a non-parametric analysis for the repeated experiments in Fig. 5 by presenting all individual results of repeated experiments. Each circle denotes an individual result, and the horizontal line denotes the mean value. For each AL algorithm, the results of the existing method (blue), the proposed method (red), and the rigorous search are presented.

Since the proposed method calculates EoU at every iteration of an AL scenario, the proposed method requires more computation than the existing search method. Calculating EoU requires as much time as applying the AL query strategy to obtain sample uncertainties. Therefore, for the proposed scenario with T iterations, the proposed method requires additional computation equivalent to T times of complexity of a query strategy. In general, the time complexity of a query

TABLE 5. Network specifications for M1 to M5.

	Layer	Output	FLOPs	Parameters
M1	conv	$28 \times 28 \times 16$	112,896	$F_{conv} = [3, 3, 1, 16], s = [1, 1], p = [1, 1]$
	conv	$26 \times 26 \times 32$	3,115,008	$F_{conv} = [3, 3, 1, 32], s = [1, 1], p = [0, 0]$
	max-pool	$13 \times 13 \times 32$	-	$F_{pool} = [2, 2], s = [2, 2], p = [0, 0]$
	drop-out	$13 \times 13 \times 32$	-	$p_{drop} = 0.25$
	fc	64×1	346,112	-
	drop-out	64×1	-	$p_{drop} = 0.50$
	fc	10×1	640	-
	Total FLOPs		**3,574,656**	
	Layer	Output	FLOPs	Parameters
M2	conv	$28 \times 28 \times 16$	112,896	$F_{conv} = [3, 3, 1, 16], s = [1, 1], p = [1, 1]$
	conv	$26 \times 26 \times 32$	3,115,008	$F_{conv} = [3, 3, 1, 32], s = [1, 1], p = [0, 0]$
	max-pool	$13 \times 13 \times 32$	-	$F_{pool} = [2, 2], s = [2, 2], p = [0, 0]$
	drop-out	$13 \times 13 \times 32$	-	$p_{drop} = 0.25$
	fc	128×1	692,224	-
	drop-out	64×1	-	$p_{drop} = 0.50$
	fc	10×1	1,280	-
	Total FLOPs		**3,921,408**	
	Layer	Output	FLOPs	Parameters
M3	conv	$28 \times 28 \times 32$	225,792	$F_{conv} = [3, 3, 1, 32], s = [1, 1], p = [1, 1]$
	conv	$26 \times 26 \times 64$	12,460,032	$F_{conv} = [3, 3, 1, 64], s = [1, 1], p = [0, 0]$
	max-pool	$13 \times 13 \times 64$	-	$F_{pool} = [2, 2], s = [2, 2], p = [0, 0]$
	drop-out	$13 \times 13 \times 64$	-	$p_{drop} = 0.25$
	fc	64×1	692,224	-
	drop-out	64×1	-	$p_{drop} = 0.50$
	fc	10×1	640	-
	Total FLOPs		**13,378,688**	
	Layer	Output	FLOPs	Parameters
M4	conv	$28 \times 28 \times 32$	225,792	$F_{conv} = [3, 3, 1, 32], s = [1, 1], p = [1, 1]$
	conv	$26 \times 26 \times 64$	12,460,032	$F_{conv} = [3, 3, 1, 64], s = [1, 1], p = [0, 0]$
	max-pool	$13 \times 13 \times 64$	-	$F_{pool} = [2, 2], s = [2, 2], p = [0, 0]$
	drop-out	$13 \times 13 \times 64$	-	$p_{drop} = 0.25$
	fc	128×1	1,384,448	-
	drop-out	64×1	-	$p_{drop} = 0.50$
	fc	10×1	1,280	-
	Total FLOPs		**14,071,552**	
	Layer	Output	FLOPs	Parameters
M5	conv	$28 \times 28 \times 32$	225,792	$F_{conv} = [3, 3, 1, 32], s = [1, 1], p = [1, 1]$
	conv	$26 \times 26 \times 64$	12,460,032	$F_{conv} = [3, 3, 1, 64], s = [1, 1], p = [0, 0]$
	max-pool	$13 \times 13 \times 64$	-	$F_{pool} = [2, 2], s = [2, 2], p = [0, 0]$
	drop-out	$13 \times 13 \times 64$	-	$p_{drop} = 0.25$
	conv	$13 \times 13 \times 64$	6,230,016	$F_{conv} = [3, 3, 64, 64], s = [1, 1], p = [1, 1]$
	conv	$11 \times 11 \times 64$	4,460,544	$F_{conv} = [3, 3, 64, 64], s = [1, 1], p = [0, 0]$
	max-pool	$5 \times 5 \times 64$	-	$F_{pool} = [2, 2], s = [2, 2], p = [0, 0]$
	drop-out	$13 \times 13 \times 64$	-	$p_{drop} = 0.25$
	fc	256×1	409,600	-
	drop-out	64×1	-	$p_{drop} = 0.50$
	fc	10×1	2,560	-
	Total FLOPs		**23,788,544**	

strategy is much smaller than the model training. Thus, the time complexity of the proposed method is not too heavy.

VI. SUMMARY AND DISCUSSION

In this paper, we revealed how the amount of PL affects the performance of an AL algorithm through extensive experiments. From the results, we observed that finding the best amount of PL before starting AL is important. In order to utilize our observations practically, we developed a method to automatically determine the best amount of PL without extensive experiments. To this end, we suggested Entropy of sample-uncertainty (EoU) to measure the training degree of a model, and utilize it as a criterion to determine the best amount of PL. By exploring the trends of the best amount of PL and EoU for various AL settings, we suggested empirical formulas to determine the hyper-parameter (τ) of the proposed method. We demonstrated the effectiveness of the proposed method by applying it to various AL algorithms.

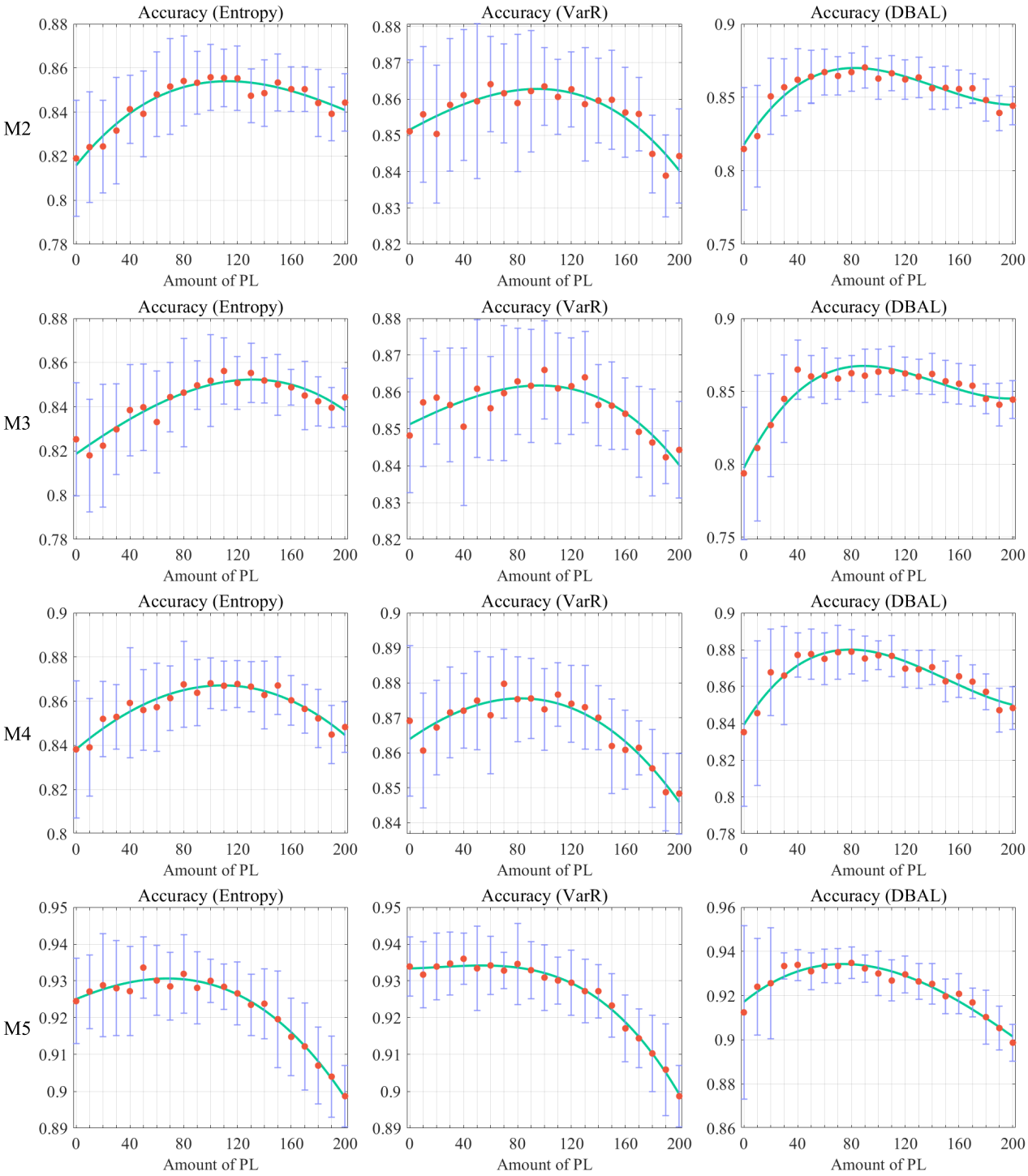


FIGURE 6. Results of our experimental scenarios on MNIST dataset, with 3 AL methods (Entropy, VarR, and DBAL) and 4 network models (M2, M3, M4, and M5). In each graph, *x* axis denotes the number of labeled data acquired by PL and *y* axis denotes the test accuracy of the model. Mean and standard deviation of the repeated results are indicated as a red point and blue bar, respectively. A cubic polynomial fitted curve (green curve) of the results is also presented to clearly show the tendency of results.

APPENDIX A
IMPLEMENTATION DETAILS
A. NETWORK STRUCTURES

In the experiment, we employed Keras implementation [34] for a deep model on MNIST [31]. To analyze the influence of network complexity on AL, we further experimented on four variations of the original implementation; M1, M2, M3, M4 (original), and M5.

The detailed structure of each model for MNIST is described in Table 5. For a concise description, we define

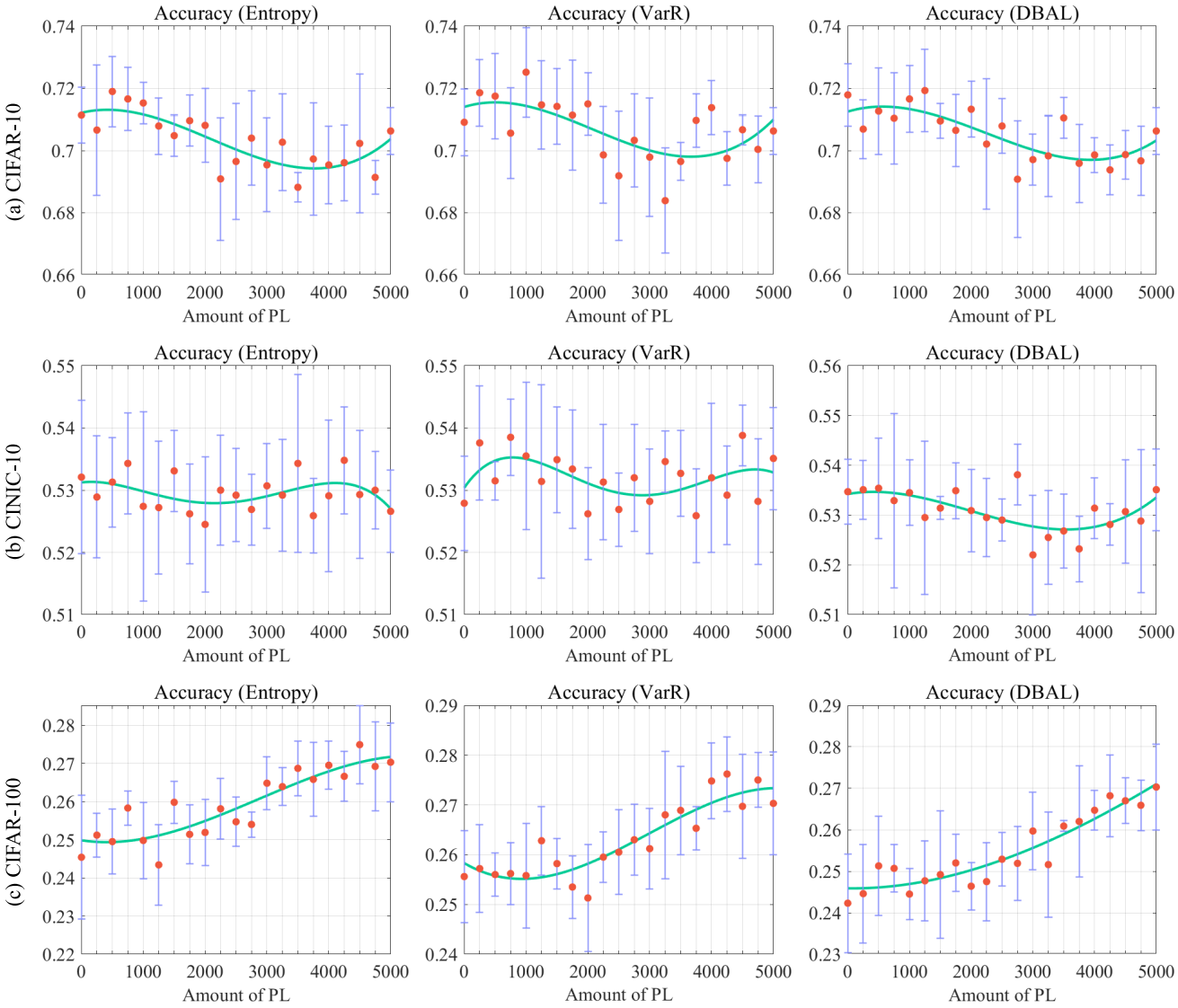


FIGURE 7. Results of our experimental scenarios on CIFAR-10/100 dataset, with three AL methods (Entropy, VarR, and DBAL) and ResNet-18.

several notations. $F_{conv} = [h, w, i, o]$ denotes a shape of a convolutional filter with the filter height h and width w, the number of input-channels i, and the number of output channels o. $F_{pool} = [h, w]$ denotes a shape of a pooling filter with the filter height h and width w. s and p denote the numbers of the stride and zeros-padding on (height, width) side, respectively. p_{drop} denotes a probability of dropout.

For CIFAR and CINIC-10, we employed ResNet-18 [35] for the deep model. A drop-out layer was added before the last fully connected layer with a dropout ratio of 0.5. The first convolution layer of ResNet-18 was modified as '3×3 kernel, stride of 1 and padding of 1' to fit CIFAR and CINIC-10.

B. DATASET

We conducted the experiments on MNIST [31], CIFAR-10, CIFAR-100 [33], and CINIC-10 [32]. **MNIST** consists of 70,000 gray-scale images of size 28 × 28 for handwritten digits from 0 to 9. We utilize 60,000 images for training and 10,000 images for testing. **CIFAR-10** and **CIFAR-100** consist of 60,000 RGB images of size 32 × 32 for 10 / 100 object classes, respectively. For both dataset, we utilize 50,000 images for training and 10,000 images for testing set. **CINIC-10** contains 270,000 images from CIFAR-10 and ImageNet [1], which are down-sampled to 32 × 32. Following [32], we utilize 90,000 images for training and 90,000 images for a testing set.

Pre-processing techniques were also applied to the data during model training. For CIFAR and CINIC-10, we normalized the data by a channel-wise mean (0.4914, 0.4822, 0.4465) and standard deviation (0.2470, 0.2435, 0.2616) and augmented the original data with random cropping from zero-padded 36 × 36 images and random horizontal flipping.

C. TRAINING SCHEME

Models were trained by Adam optimizer [36] with a learning rate of 5×10^{-4} and a batch size of 64 during 100 epochs. For each iteration of pool-based AL, the models were trained from scratch with the updated labeled dataset.

D. COMPUTING ENVIRONMENTS

We implemented the proposed algorithm using PyTorch library [37]. We ran the experiments on Intel Core i7-6700K CPU @ 4.00 GHz and a single NVIDIA GeForce GTX 1080-Ti of 12 GB memory.

APPENDIX B
FULL RESULTS OF EVALUATION SCENARIOS

In this section, we provide the experimental results that are not included in the manuscript. Fig. 6 shows the results for MNIST, with three AL methods (Entropy [19], VarR [20], and DBAL [6]) and 4 networks (M2, M3, M4, and M5). Fig. 7 shows the results for CINIC-10 [32] and CIFAR-10, 100 [33] with three AL methods (Entropy, VarR, and DBAL) and ResNet-18 [35]. Note that we fit results of Entorpy and VarR in CINIC-10 with the quartic polynomial curve since the results are noisy. We provide some explanations for noisy results. For MNIST, we got smooth results since (1) we repeated experiments 20 times, (2) results are reported finely with small b, and (3) images of MNIST are well aligned. However, for CIFAR and CINIC, the results seem to be noisy since (1) experiments are repeated 5 times the same as the existing works, (2) b is larger than b for MNIST, and (3) CIFAR contains hard samples.

Interesting point is that applying AL few iterations shows worse performance than not applying AL. We guess the reason is as follows. Most datasets may contain outlier samples. Outliers are deviated from the distribution of the entire dataset, but have a significant influence on the formation of a decision boundary. From the perspective of query strategies, outliers may seem informative, but they adversely affect performance in reality. Some outliers can be included in Q_u during the first few steps when applying the AL algorithm. In consequence, applying only PL can achieve better performance than applying AL very few steps. Note that results of MNIST show a clear shape since images of MNIST are well aligned (i.e. fewer outliers).

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[2] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 843–852.
[3] B. Settles, "Active learning literature survey," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1648, 2009.
[4] D. Yoo and I. S. Kweon, "Learning loss for active learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Oct. 2019, pp. 93–102.
[5] W. H. Beluch, T. Genewein, A. Nürnberger, and J. M. Köhler, "The power of ensembles for active learning in image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Sep. 2018, pp. 9368–9377.
[6] Y. Gal, R. Islam, and Z. Ghahramani, "Deep Bayesian active learning with image data," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1183–1192.
[7] K. Wang, D. Zhang, Y. Li, R. Zhang, and L. Lin, "Cost-effective active learning for deep image classification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 12, pp. 2591–2600, Oct. 2016.
[8] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–13.
[9] H. H. Aghdam, A. Gonzalez-Garcia, J. V. D. Weijer, and A. M. López, "Active learning for deep detection neural networks," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 3672–3680.
[10] Z. Liu, J. Wang, S. Gong, H. Lu, and D. Tao, "Deep reinforcement active learning for human-in-the-loop person re-identification," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 6122–6131.
[11] K. Murugesan and J. Carbonell, "Active learning from peers," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 7008–7017.
[12] Y. Shen, H. Yun, Z. C. Lipton, Y. Kronrod, and A. Anandkumar, "Deep active learning for named entity recognition," 2017, *arXiv:1707.05928*.
[13] B. Liu and V. Ferrari, "Active learning for human pose estimation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Aug. 2017, pp. 4373–4382.
[14] F. Caba Heilbron, J.-Y. Lee, H. Jin, and B. Ghanem, "What do i annotate next an empirical study of active learning for action localization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 199–216.
[15] Z. Zhou, J. Shin, L. Zhang, S. Gurudu, M. Gotway, and J. Liang, "Fine-tuning convolutional neural networks for biomedical image analysis: Actively and incrementally," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, Hawaii, HI, USA, 2017, pp. 7340–7349.
[16] L. Yang, Y. Zhang, J. Chen, S. Zhang, and D. Z. Chen, "Suggestive annotation: A deep active learning framework for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* New York, NY, USA: Springer, 2017, pp. 399–407.
[17] S. Dasgupta, "Two faces of active learning," *Theor. Comput. Sci.*, vol. 412, no. 19, pp. 1767–1781, 2011.
[18] S. Sinha, S. Ebrahimi, and T. Darrell, "Variational adversarial active learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 5972–5981.
[19] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.
[20] L. C. Freeman, *Elementary Applied Statistics: For Students in Behavioral Science*. Hoboken, NJ, USA: Wiley, 1965.
[21] Y. Siddiqui, J. Valentin, and M. Nießner, "ViewAL: Active learning with viewpoint entropy for semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 9433–9443.
[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[23] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7291–7299.
[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "DropOut: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[25] R. Z. Farahani and M. Hekmatfar, *Facility Location: Concepts, Models, Algorithms Case Studies*. Berlin, Germany: Springer, 2009.
[26] D. P Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.
[27] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for K-medoids clustering," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3336–3341, 2009.
[28] F. Zhdanov, "Diverse mini-batch active learning," 2019, *arXiv:1901.05954*.
[29] Z. Wang and J. Ye, "Querying discriminative and representative samples for batch mode active learning," *ACM Trans. Knowl. Discovery Data*, vol. 9, no. 3, pp. 1–23, Feb. 2015.
[30] Y.-P. Tang and S.-J. Huang, "Self-paced active learning: Query the right thing at the right time," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 5117–5124.
[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

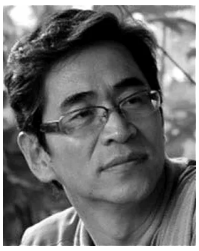
[32] L. N. Darlow, E. J. Crowley, A. Antoniou, and A. J. Storkey, "CINIC-10 is not ImageNet or CIFAR-10," 2018, *arXiv:1810.03505*.
[33] A. Krizhevsky and G. Hinton, *Learning Multiple Layers of Features From Tiny Images*. Princeton, NJ, USA: Citeseer, 2009.
[34] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io
[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Apr. 2016, pp. 770–778.
[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[37] A. Paszke, S. Gross, F. Massa, and A. Lerer, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf



DAE UNG JO received the B.S. and Ph.D. degrees in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2016 and 2022, respectively. He is currently a Staff Engineer with Samsung Electronics, Kyeonggi, South Korea. His research interests include machine learning, computer vision, and the architecture of deep learning.



SANGDOO YUN received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 2010, 2013, and 2017, respectively. He is currently a Research Scientist with NAVER AI Laboratory. His current research interests include computer vision, deep learning, and image classification.



JIN YOUNG CHOI (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in control and instrumentation engineering from Seoul National University, Seoul, South Korea, in 1982, 1984, and 1993, respectively. From 1984 to 1989, he was with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea, where he was involved in the project of switching systems. From 1992 to 1994, he was with the Basic Research Department, ETRI, where he was a Senior Member of Technical Staff involved in the neural information processing system. From 1998 to 1999, he was a Visiting Professor with the University of California at Riverside, Riverside, CA, USA. Since 1994, he has been with Seoul National University, where he is currently a Professor with the School of Electrical Engineering. He is also with the Engineering Research Center for Advanced Control and Instrumentation, Automation and Systems Research Institute, and the Automatic Control Research Center, Seoul National University. His current research interests include adaptive and learning systems, visual surveillance, motion pattern analysis, object detection, object tracking, and pattern recognition.