Università della Calabria

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

DOCTORAL THESIS

An active learning Approach based on learning models' parameters exploitation

---

Author:

Eng.

Francesco Scala

Supervisor:

Prof.

Sergio Flesca

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in

Information and Communication technologies DIMES

June 2023

To my mum...

Abstract

Artificial Intelligence (AI) techniques and in particular Machine and Deep Learning (ML and DL), have been widely adopted to enhance various aspects of human life. ML algorithms can be categorized into four main types: Supervised Learning, Unsupervised Learning, Semi-supervised Learning, and Reinforcement Learning. A significant challenge in these techniques is the requirement for sufficient labeled data for training. Active Learning (AL) is a machine learning framework that addresses this issue by selecting instances to be labeled in a smart way to optimize model training, i.e., AL reduces labeling time and leads to better-performing models by dynamically selecting the most representative samples to be labeled during the training phase. AL was proven to be effective in different scenarios and its choice of querying a label depends on the cost and gain of obtaining the information. In this thesis, are presented two novel approaches for active learning in meta-learning models. The proposed methods, called *LAL-IGradV* and *LAL-IGradV-VAE*, select instances to be labeled using an estimate of their impact on the current classifier. This is achieved by evaluating the importance of previously labeled instances in training the classification model and training another model that estimates the importance of unlabeled instances. The approaches can be instantiated with any classifier that is trainable through gradient descent optimization, and in this study, is provided a formulation using a deep neural network. These approaches have not been thoroughly investigated in previous learning-to-active-learn methods and experimental results demonstrate its promising performance in scenarios where there are only a limited number of initially labeled instances.

Contents

Chapter 1

Introduction

Nowadays, Artificial Intelligence (AI), and particularly Machine and Deep Learning (ML and DL) techniques, have been exploited to perform a plethora of tasks that help in improving the quality of human life. For instance, AI techniques have been exploited to enhance the photographic quality of cell phone cameras, to devise language translation facilities, to implement systems for supporting people with limited abilities, to implement decision support systems in the medical domain (for example cancer diagnosis systems), and so on.

Yet there are some differences in the way various types of machine learning algorithms are defined, in general they can be categorized according to their purpose, and the main categories are the following:

- *Supervised learning*: algorithms attempt to model the relationships and dependencies between target prediction output and input features in order to predict output values for new data based on relationships learned from previous data sets;

- *Unsupervised Learning*: a type of machine learning algorithm commonly used in pattern recognition and descriptive modeling. However, there are no output categories or labels on which the algorithm attempts to model relationships. These algorithms attempt to use techniques on input data to search for rules, identify patterns, summarize and group data points,

and derive meaningful insights and better describe the data to users;

- *Semi-supervised Learning*: in the previous two types, either there are no labels for all observations in the dataset or there are labels for all observations. Semi-supervised learning falls somewhere in between. Labeling is expensive in many practical situations because it requires skilled human experts. Consequently, semi-supervised algorithms are the best candidates for model building when labels are absent in most observations but present in a few. These methods exploit the fact that while the group membership of unlabeled data is unknown, these data contain critical information about the group parameters.

- *Reinforcement Learning*: it allows machines and software agents to automatically determine the optimal behavior in a uncertain context in order to maximize their performance. The agents incrementally learn the optimal behavior by observing a simple reward feedback, known as a reinforcement signal.

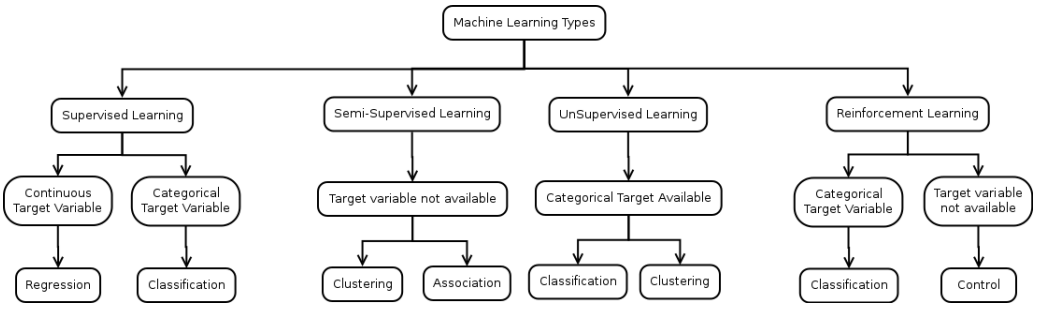


Figure 1.1: The types of Machine Learning Algorithms divided into categories according to their purpose and the main categories

Each ML/DL model must be trained to learn its application domain, and to do this, sufficient data must be available for training. We currently live in an data-driven era, where we are inundated of data, but there is the problem that much of it is not labeled, and not all ML activities are just Clustering or Association that do not require labels. Thus, labeling this data requires resources that can be very expensive in some cases (both in terms of money and

time). To overcome these problems, Active Learning (AL) paradigm was proposed. AL is a machine learning framework, very close to the semi-supervised learning, where models are trained using both labeled and unlabeled data and learning algorithms can interactively query an oracle to label new data points. AL is all about labeling data dynamically and incrementally during the training phase so that the algorithm can identify what label would be the most beneficial for it to learn from. With AL, remarkable results can be achieved, such as reducing labeling time and obtaining better performing models, because the most representative samples are labeled with AL, which is the best solution for training ML/DL models.

A major challenge in active learning is to select the most informative instances to be labeled by an annotation oracle at each step. In this respect, one effective paradigm is to learn the active learning strategy that best suits the performance of a meta-learning model. This strategy first measures the quality of the instances selected in the previous steps and then trains a machine learning model that is used to predict the quality of instances to be labeled in the current step.

For example given a model M, and a dataset containing only 1000 labeled samples (LS) and 10000 unlabeled samples (US), M difficultly can reach a remarkable accuracy by training it only on the 1000 labeled samples. In order to increase its accuracy there are two possibilities:

- give a label to all samples in US and train the model;
- select iteratively n samples form US and train the model at each iteration.

The second approach is the smartest because, most likely, we will only give a label to a portion of US because with AL we select the most representative samples that are most relevant during training and thus we will get the information from the "correct" data and the model will learn with less data in less time.

AL is effective in a variety of situations. Essentially, the choice of whether or not to query each specific label is determined by whether the gain from querying the label outweighs the cost of obtaining the information. In practice, depending on the budget available to the data scientist and other factors, this decision-making process can take several forms.

1.1 Main Contributions

The research underlying this thesis concerns the systematic investigation, analysis, and development of models and techniques in the field of AL. The primary objective is to identify the most challenging problems in AL and to propose appropriate solutions.

More specifically, this thesis introduces two new approaches of learning-to-active-learn that selects the instances to be labeled as the ones producing the maximum change to the current classifier. The key idea is to select such instances according to their importance reflecting variations in the learning gradient of the classification model. The proposed approach can be instantiated with any classifier trainable via gradient descent optimization, and here we provide a formulation based on a deep neural network model, which has not deeply been investigated in existing learning-to-active-learn approaches. The experimental validation of our approach has shown promising results in scenarios characterized by relatively few initially labeled instances.

1.2 Organization

This thesis is conceptually organized into two parts. The first one contains an introduction on AL describing its principles and main existing techniques. The second part presents the algorithmic contribution of this thesis, i.e. the techniques to improve the AL data selection approach *ad hoc* for artificial neural networks. The two parts are articulated around the following chapters.

Chapter 2 is focused on AL, initially describing three different frameworks (Pool based, Stream based and Membership Query Synthesis), and then describing a fundamental topic which is Stopping Criteria, followed by different frameworks of query strategies. Next, an extensive excursus is made on the applications of AL and how it can be combined with other ML/DL techniques to greatly increase its performance.

Chapter 3 presents an AL framework that is able to select the best samples to train a neural network (NN) by exploiting the variation of gradients within it; using these gradients, a relevance score can be estimated for each unlabeled sample, then the calculated score is used to train a simple ML model (*Selector*) that is able to estimate this score for unlabeled samples. This score is subsequently calculated on each US sample using the *Selector* and then the most representative ones are selected to be labeled by the human expert.

Chapter 4 describes an extension of the previous approach (presented in Chapter 3) that, with the support of a Variational AutoEncoder (VAE), the training of the ML selector model is simplified, resulting in a significant improvement in the performance (both in terms of time and accuracy) of the original framework.

Chapter 2

Active Learning

Nowadays, a continuously increasing amount of data is becoming available to data scientists to be analyzed. However, in most cases this data are raw, unlabeled data which cannot be used for training machine learning models and, in particular, deep learning models. In several context, providing the actual labels for this large amount of data requires a significant effort by human experts, which are asked to provide class labels for the available data. Active learning (AL) [75] aims at significantly reducing this effort. It is a machine learning framework where the learning algorithm can interactively query a user (referred to as teacher or oracle) to label new data points with actual labels. The active learning process is also referred to as optimal experimental design. Recently, a lot of effort has been devoted by the research community for the design of effective AL strategies, and in particular AL strategies suitable to be used in training deep learning (DL) models, such as convolutional neural networks (CNN) and long-short term memory networks (LSTMs). In [88], Yanyao Shen et al. proved experimentally that the amount of labeled data used for training DL models is drastically reduced when DL is combined with AL.

More in detail, assume that we want to train a binary image classification model. There can be billions of (unlabeled) images that could be used for training the model. However, before using these images, they must be annotated with their actual class label. As not all of these images are necessary (or even

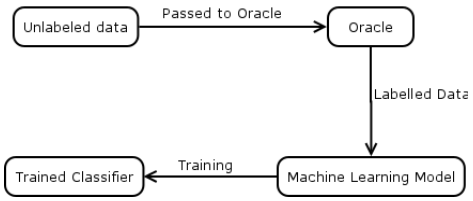


Figure 2.1: Passive learning framework is the standard in which a large amount of labeled data is passed to the algorithm, requires significant effort in labeling the entire dataset

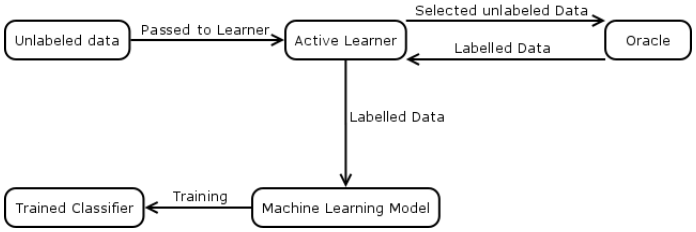


Figure 2.2: Active learning framework where the learning algorithm can interactively query a user to label new data points with real labels

useful) to train a good model, a lot of effort can be spared by labeling useful images only. Indeed, some images may be more profitably exploited for learning a classification model than others. Using active learning, we can select a subset of the set of unlabeled images and exploit crowd-sourcing platforms, to ask human experts to provide labels for this (smaller) set of unlabeled images. In different terms, an active learning algorithm iteratively selects the most informative (unlabeled) images and submits them to an oracle for labeling. This process is iterated until a predetermined number of AL iterations is performed or an halting condition is satisfied. The promise of the Active learning is that, if we are smart enough, we can label only a fraction of the available (unlabeled) examples and use them to train a ML model whose performances are similar or even better [38] than the performances of the model trained by labeling the entire set of unlabeled examples. Usually the quality of an AL approach is measured by comparing the "smart" selection of examples to be labeled with a "random" selection. In Figure 2.3 an example of the improvement obtained by an active learning approach over random selection is shown. The entire set of data points, red triangles and green circles, is not linearly separable. Summa-

rizing, active learning arises exploits the intuition that labeled examples may have different importance in order to train a model. In fact, in many cases the most informative points are those on which the model is most uncertain. With uniform random sampling across all examples, the model may not fully represent the boundaries between classes, but with active learning you select examples near the class boundary to be labeled, thus achieving a more effective classifier. It has been shown that has also shown that active learning allow to obtain better classifier over standard random selection for contexts such as multi-class image classification [40, 70, 13, 43].

2.1 Frameworks of active learning

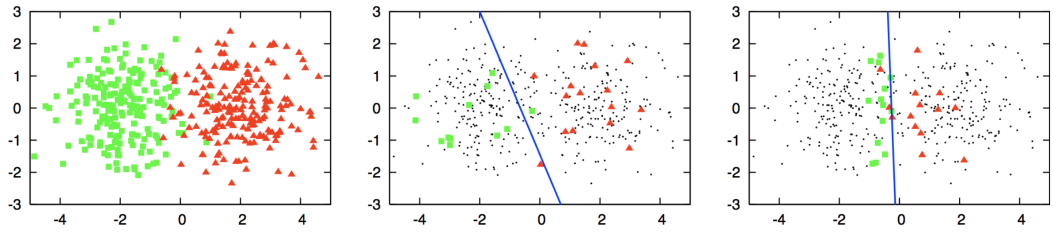
Active learning can be considered a semi-supervised learning method, thus a hybrid between unsupervised learning that uses 0% of the examples and fully supervised learning that uses 100%. By iteratively increasing the size of our labeled training set, we can achieve better performance with a fraction of the cost or time of training models using all the data [25].

2.1.1 Active Learning Pool Based

Pool-based active learning assumes that one has a large pool of unlabeled data samples and selects the most informative points iteratively until the model reaches a certain level of performance, such as accuracy [16]. There are many interesting works in this area, one of which was introduced by Contardo et al. [17] who explored the learning challenge when collecting training labels is prohibitively expensive, which is typically addressed in the literature by employing AL strategies. These offer methods for selecting instances to label before or during training. These tactics are typically based on heuristics or even theoretical metrics. The researchers created a model that attempts to teach AL skills in a meta-learning environment. Specifically, they analyzed a pool-based scenario in which the system looks at all the samples in a problem dataset and has to select a portion of them to be ranked in one shot.

2.1.2 Active Learning Stream Based

In Stream Based Active Learning, it is assumed that obtaining an unlabeled instance is free; on this basis, unlabeled instances are selected one at a time and allow the learner to determine whether he or she wants to query (explained in 2.3.6) the label of the instance or reject it based on its informativeness. To determine the informativeness of the instance, a query strategy is used. Thus, by taking a sample from the set of unlabeled, it is determined whether it should be labeled or discarded, then repeated with the next sample.



(a) (b) (c)

Figure 2.3: An example of pool-based active learning [81]. A data set of 400 instances, sampled into two class Gaussians (a). A logistic regression model trained with thirty labeled instances randomly drawn from the problem domain. The line represents the decision boundary of the classifier (70% accuracy) (b). (third graph) A logistic regression model trained with thirty actively queried instances using uncertainty sampling (90%) (c).

2.1.3 Membership Query Synthesis

In Membership Query Synthesis [2], we assume that the active learner produces an example that he or she would like to be labeled. This scenario requires that the model be able to capture the distribution of data well enough to create reasonable, clearly labeled examples. There is not much interest in this scenario because, for example, if we are classifying images and the learner produces an image that is pure noise, we will not be able to label it. However, some innovative and promising real-world applications of the membership query scenario have been described in recent years Query synthesis may

be a promising direction for automated scientific discovery. However, King et al. [47] described an innovative and promising application of the membership query scenario, an instance is a mixture of chemical solutions that constitute a culture medium as well as a particular yeast mutant. All experiments are synthesized independently using an active learning approach based on inductive logic programming and performed physically with a laboratory robot. This active method results in a three-fold reduction in the cost of experimental materials compared to running the least expensive experiment and a 100-fold reduction in cost compared to randomly generated experiments.

2.2 Stopping Criteria

In active learning, a learning process can be stopped at a predetermined iteration or the time at which learning should be stopped must be determined, which is a critical issue. Zhu et al. [101] introduced four simple stopping criteria based on a measure of confidence estimation on the unlabeled data pool, including:

1. **Maximum Uncertainty** (MU): the learner selects the most uncertain unlabeled example as the most informative one. The uncertainty value of this example is an excellent proxy for the current classifier's confidence on all the unlabeled samples. If the uncertainty value of the selected example is small enough, it can be assumed that the current classifier has enough confidence in its classification of the remaining unlabeled data. As a result, the active learning process is stopped. The first strategy, known as the Maximum Uncertainty (MU) method, is based on this assumption:

$$SC_{MU} = \begin{cases} 1 & \forall x \in U, UM(x) \leq \theta_{MU} \\ 0 & otherwise \end{cases} \tag{2.1}$$

where $UM(.)$ denotes the uncertainty measurement, θ_{MU} is a user predefined uncertainty threshold, and U denotes the set of unlabeled examples;

2. **Overall Uncertainty** (OU): at each learning cycle, the maximum uncertainty criteria only evaluates the most uncertain example. The overall uncertainty technique takes into account the total uncertainty across all unlabeled cases. It can be assumed that the present classifier has sufficient confidence in its classification of the remaining unlabeled data if the aggregate uncertainty of all unlabeled cases becomes very tiny. The OU method's strategy is to consider if the average uncertainty value of all remaining unlabeled instances is less than a very tiny preset threshold. The stopping criterion SC_{OU} is defined as follows:

$$SC_{OU} = \begin{cases} 1 & \frac{\sum_{\forall x \in U} UM(x))}{|U|} \leq \theta_{OU} \\ 0 & otherwise \end{cases} \tag{2.2}$$

where θ_{OU} is a user-predefined uncertainty threshold, and $|U|$ denotes the size of the unlabeled data pool U;

3. **Selected Accuracy** (SA): the classification accuracy on the top-m selected examples at each learning cycle in batch mode active learning settings would be a helpful signal to indicate the current classifier's confidence on remaining unlabeled examples. It is thus simple to estimate this accuracy based on the oracle's feedback when an active learner requests true labels for these selected unlabeled samples. At the end of each learning cycle, the current classifier should have the least confidence in its classifications of these selected unlabeled samples. If the current classifier properly classifies these selected unlabeled instances, it can be assumed that the classifier has sufficient confidence in classifying the remaining unlabeled data. This technique considers if the present classifier is capable of correctly predicting the labels of the top-m unlabeled samples. As a result, the stopping criterion SC_{SA} can be defined as follows:

$$SC_{SA} = \begin{cases} 1 & ACC_m(C) \geq \theta_{SA} \\ 0 & otherwise \end{cases} \tag{2.3}$$

where θ_{SA} is a user-predefined accuracy threshold and function $ACCm(C)$ evaluates the accuracy performance on the top-m selected unlabeled examples through feedback of the Oracle;

4. **Minimum Expected Error Methods** (MEE): this stopping criterion is based on an estimate of the current classifier's expected error on all future unlabeled samples. The justification for MEE is that in the learning process, a classifier C with maximum effectiveness results in the lowest predicted error on the entire test set. The SC_{MEE} stopping criterion can be defined as:

$$SC_{SA} = \begin{cases} 1 & Error(C) \leq \theta_E \\ 0 & otherwise \end{cases} \tag{2.4}$$

where $Error(C)$ evaluates the expected error of classifier C that closely reflects the classifier effectiveness. θ_E is a user-predefined error threshold.

2.3 Query Strategy Framework

The difference between an active and a passive learner is the ability to query instances and their labels. All AL scenarios require some sort of measure of informativeness of unlabeled instances. This section will explain different approaches to querying instances.

2.3.1 Uncertainty Sampling

An active learner differs from a passive learner in that the former has the ability to query instances based on past queries and the answers (labels) of those queries. All active learning scenarios assume that each unlabeled sample is assigned a relevance score. In this section, various approaches to instances will be explored under the common theme called uncertainty sampling because of the use of probabilities.

2.3.1.1 Smallest Margin Uncertainty

Smallest Margin Uncertainty (SMU) is a comparison of the uncertainty between the best and second best. It is the probability of ranking the most likely class minus the probability of ranking the second most likely class (equation 2.5).

$$\Phi_{SM}(x) = P_\theta(y_1^*|x) - P_\theta(y_2^*|x) \tag{2.5}$$

The intuition behind this metric is that if the probability of the most probable class is significantly greater than the probability of the second most probable class, the classifier is more confident of belonging to the example class. Similarly, if the probability of the most likely class is not significantly greater than the probability of the second most likely class, the classifier is less confident of belonging to the example class. The active learning algorithm will select the example with the lowest SMU value.

2.3.1.2 Least Confidence Uncertainty

Least Confidence Uncertainty (LCU) measure selects the example for which the classifier is least confident of the selected class. LCU selection examines only the most likely class and selects the example with the lowest probability assigned to that class (equation 2.6).

$$\Phi_{LC}(x) = 1 - P_\theta(y_1^*|x) \tag{2.6}$$

2.3.1.3 Entropy Reduction

Entropy reduction is a measure of the uncertainty of a random variable. We can consider Shannon's Entropy, which has several basic properties, including:

1. uniform distributions have the greatest uncertainty;

2. uncertainty is additive for independent events;

3. adding a zero-probability result does not has no effect;

4. events with a certain outcome have zero effect [93].

By considering class predictions as outcomes, we can measure the Shannon entropy of the probabilities of the predicted classes. Higher values of entropy indicate greater uncertainty in the probability distribution [40]. In each active learning step, for each unlabeled example in the training set, the active learning algorithm calculates the entropy on the predicted class probabilities and selects the example with the highest entropy. The example with the highest entropy is the one for which the classifier is least confident about its class membership (equation 2.7).

$$\Phi_{ENT}(x) = \sum_{y} P_\theta(y|x) log P_\theta(y|x) \tag{2.7}$$

2.3.1.4 Largest Margin Uncertainty

Largest Margin Uncertainty (LMU) is a comparison of the best and worst uncertainty. This measure is given by the probability of classification of the most likely class minus the probability of classification of the least likely class. The intuition behind this metric is that if the likelihood of the most likely class is significantly greater than the likelihood of the least likely class, the classifier is more certain to belong to the example class. Similarly, if the probability of the most likely class is not significantly greater than the probability of the least likely class, the classifier is less confident of belonging to the example class. The active learning algorithm will select the example with the minimum value of LMU (equation 2.8).

$$\Phi_{LM}(x) = P_\theta(y_1^*|x) - P_\theta(y_{min}|x) \tag{2.8}$$

2.3.2 Query By Committee

The idea is similar to that of uncertainty sampling, but extended to ensembles: instead of examining the uncertainty of a single model on the unla-

beled ensemble, we train several models on the labeled ensemble and observe their disagreement on examples from the unlabeled ensemble. If we consider our ensemble as a set of models from different vital parts of the hypothesis class, then choosing the example with the largest disagreement corresponds to making the current version space as small as possible. Just as we can define different measures of "uncertainty," we can also define different measures of "disagreement." An example of such a measure, which corresponds to the maximum entropy measure of uncertainty sampling, is the "entropy of votes." In this case, we count the votes for each possible label, normalize so that it is a distribution, and observe the entropy. If M is the size of our set, the decision rule we obtain is:

$$\underset{argmax}{x} -\sum_{y \in Y} \frac{Votes(y)}{M} log(\frac{Votes(y)}{M}) \tag{2.9}$$

2.3.3 Expected Model Change

Expected Model Change is an active learning framework that uses a decision-theoretic approach, selecting the instance x that makes the model change more than the others if its label is known, in other words if labeled and added to ζ (the training set), it would result in the new training gradient of the largest magnitude. Probabilistic discriminative models are typically trained using gradient-based optimization, which considers the "change" made to the model by measuring the length of the training gradient (this can be applied to any learning problem where gradient-based training is used).

The target of this framework is to select the instances that are likely to most influence for the model, regardless of the label of the resulting query. This approach has been shown to work well, but it is necessary that both the feature space and the set of labels are not very large, because it can be computationally expensive under these conditions.

These approaches may suffer from a problem, that is, the informativeness of a given instance may be overestimated simply because one of its feature values is unusually large, or the corresponding parameter estimate is larger,

both resulting in a high gradient. To solve this problem it is desirable to resize the features, one approach may be to regularize the parameters [15] which may help to control this effect somewhat, actually in practice it does not seem to be a significant problem.

A query strategy in this framework is the "Expected Gradient Length" (EGL) approach for discriminative probabilistic model classes [84]. Given $\bigtriangledown l_\theta(\zeta)$ the gradient of the objective function l with respect to the model parameters θ and given $\bigtriangledown l_\theta(\zeta \cup \langle x, y\rangle)$ the new gradient that would be obtained by adding the training tuple $\langle x, y\rangle$ to ζ. Since the query algorithm does not know the true label y in advance, should be calculated the length as an expectation over the possible labelings:

$$x^*_{EGL} = \underset{x}{argmax} \sum_i P_\theta(y_i|x)|| \bigtriangledown l_\theta(\zeta \cup \langle x, y_i\rangle)|| \tag{2.10}$$

Where $||\cdot||$ is the Euclidean norm of each resulting gradient vector. At query time, $|| \bigtriangledown l_\theta(\zeta)||$ should be nearly zero since converged at the previous round of training. Thus, we can approximate $\bigtriangledown l_\theta(\zeta \cup \langle x, y_i\rangle) \approx \bigtriangledown l_\theta(\langle x, y_i\rangle)$ for computational efficiency, because training instances are usually assumed to be independent.

This strategy was introduced for active learning in the multiple-instance setting (or "multi-instance learning", these terms are used interchangeably in the literature and they both convey the crucial point of difference with traditional single-instance learning). In the single-instance learning where each observation or learning object is described by a number of feature values and, possibly, an associated outcome, differently in the multiple-instance setting (MIL) [32] is different, in fact the data structure is more complex: a learning sample or object is called a bag, associated with multiple instances or descriptions. Each instance is described by a feature vector, at the same way of single-instance learning, but an associated outcome is never reported. The only information available for each instance, aside from its feature values, is its membership relationship to a bag. Formally, an instance x (in the single-instance setting) corresponds to a point in the instance space $\mathbb{X}$ (in the

multiple-instance setting). It is commonly assumed that $\mathbb{X} \subseteq R_d$, that is, each instance is described by a vector of d real-valued numbers, its feature values. Typically the datasets often contain mixed types of features, in order to model these situations, $\mathbb{X}$ can be generalized to $\mathbb{X} \subseteq A^d = A_1 \times \cdots \times A_d$, such that each instance is described by a d-dimensional vector, where each attribute $A_i (i = 1, \cdots, d)$ takes on values from a finite or infinite set V_i. In this way, we can deal with mixed feature sets in which some of the features are categorical and others are numeric. A bag $\mathbb{X}$ is a collection of n instances, where each instance x_i is drawn from the instance space $\mathbb{X}$. Many authors define a bag as $X \in N^{\mathbb{X}}$, that is, a multi-set containing elements from $\mathbb{X}$ such that duplicates can occur, because each bag can have a different size, (so the value n vary among the bags in the dataset), because multiple copies of the same instance can be included in a bag and/or the bags are allowed to overlap and contain copies of the same instance. This forms an indication of the higher level of complexity of MIL compared to single-instance learning.

Coming back to the EGL query strategy, it aims to identify the instance that would impart the greatest change to the current model if is known its label. Since we train MILR (Multiple Instance Logistic Regression in which is trained a probabilistic model for multiple-instance tasks using a generalization of the Diverse Density (DD) framework [60, 21] that provides a way to learn those ambiguous features by maxmising the DD estimator, and the maximum of DD estimator is called a concept) with gradient descent, this involves querying the instance which, if $\langle B_{ij}, Yij \rangle$ is added to the training set, would create the greatest change in the gradient of the objective function. Let $E(\theta)$ be the gradient of E with respect to θ, which is a vector whose components are the partial derivatives of E with respect to each model parameter:

$$E(\theta) = \left[\frac{\partial E}{\partial \theta_1}, \frac{\partial E}{\partial \theta_2}, \cdots \frac{\partial E}{\partial \theta_m} \right] \tag{2.11}$$

Now let $E_{ij}^{+}(\theta)$ be the new gradient obtained by adding the positive tuple $\langle \{B_{ij}\}, 1 \rangle$ to the training set, and likewise let $E_{ij}^{-}(\theta)$ be the new gradient if a query results in the negative tuple $\langle \{B_{ij}\}, 0 \rangle$ being added. Since we do not

know in advance what label the oracle will provide, we instead calculate the expected length of the gradient based on the learner's current belief o_{ij} about each outcome. More precisely, we define the Expected Gradient Length (EGL) to be:

$$EGL(B_{ij}) = o_{ij} \left|\left|E_{ij}^{+}(\theta)\right|\right| + (1 - o_{ij}) \left|\left|E_{ij}^{-}(\theta)\right|\right| \tag{2.12}$$

This selection strategy does not explicitly encode MI bias, but employs class probabilities to determine the expected label for candidate queries, with the goal of maximizing parameter changes in what is an MI learning algorithm. This strategy can be generalized to query other properties in non- MI active learning as well. For example, Zhu et al. [103] use a related approach to determine the expected label of candidate query instances when combining active learning with graph-based semi-supervised learning. However, instead of trying to maximize the expected change in the learning model, they select the expected reduction in the estimated error on unlabeled instances.

2.3.4 Expected Error Reduction

Expected Error Reduction is an active learning framework that aims to estimate future expected error of a model trained using $\zeta \cup \langle x, y \rangle$ on unlabeled versions left in U and query that version with expected error minimum in the future. One approach is to minimize the expected $0/1 - loss$:

$$x_{0/1}^{*} = \underset{x}{argmin} \sum_{i} P_{\theta}(y_i|x) \left(\sum_{U}^{u=1} 1 - P_{\theta^{+\langle x, y_i \rangle}} \left(\hat{y}|x^{(u)} \right) \right) \tag{2.13}$$

where $\theta^{+\langle x, y_i \rangle}$ refers to the the new model after it has been re-trained with the training tuple $\langle x, y_i \rangle$ added to ζ. With this approach the true label for each query instance isn't known, so is approximated using expectation over all possible labels under the current model θ. The target here is to reduce the expected total number of incorrect predictions. Another, less stringent goal is to minimize the expected log-loss:

$$x^*_{log} = \underset{x}{argmin} \sum_i P_\theta(y_i|x) \left(- \sum_{u=1}^{U} \sum_j P_{\theta^{+\langle x,y_i \rangle}}(y_i|x^{(u)}) log P_{\theta^{+\langle x,y_i \rangle}}(y_i)|x^{(u)} \right) \tag{2.14}$$

which is equivalent to reducing the expected entropy over U. Another interpretation of this strategy is maximizing the expected information gain of the query x, or the mutual information of the output variables over x and U. Typically this technique is also the most computationally expensive query framework, indeed not only does this require an estimate of the expected future error over U for each query, but a new model must be gradually retrained for each possible query designation, which in turn iterates over the entire pool. A classification task with three or more labels using the $MaxEnt$ [7] model would require a time complexity of $O(M^2)ULG$, where M is the number of class labels, U is the size of the unlabeled pool υ, L is the size of the current training set ζ, and G is the number of gradient computations required by the by optimization procedure until convergence. In a CRF sequencing task, the complexity explodes to $O(TM^{T+2})ULG$, where T is the length of the input sequence. For this reason, the application of the expected error reduction framework usually considered only simple binary classification tasks. Roy and McCallum [77] first proposed the expected error reduction framework for text classification using naive Bayes. Guo and Greiner [30] employ an "optimistic" variant that redirects the expectation to the most probable label for computational convenience, using uncertainty sampling as a fallback strategy when the oracle provides an unexpected labeling. All that is required is a suitable objective function and a way of estimating subsequent label probabilities. For example, strategies in this framework have been used successfully with a variety of models, including pure Bayesian, Gaussian random fields, logistic regression, and SVMs.

2.3.5 Variance Reduction

We can reduce the generalization error indirectly by minimizing the variance of the output, which sometimes has a solution in closed form. Minimizing the expectation of a loss function directly is expensive and in general cannot be done in closed form. We can take advantage of Geman et al. [27], showing that the expected future error of a learner can be decomposed as follows.

$$E_T[(\hat{y} - y)|x] = E[(y - E[y|x])^2] + (E_\zeta[\hat{y}] - E[y|x])^2 + E_\zeta[(\hat{y} - E_\zeta[\hat{y}])^2] \quad (2.15)$$

where:

- $E_\zeta[\cdot]$: is an expectation over the labeled set ζ;
- $E[\cdot]$: is an expectation over the conditional density $P(y|x)$;
- E_T: is an expectation over both;
- x: indicates an instance;
- y: indicates the label for an instance;
- $\hat{y}$: indicates the label for an instance given by the model.

More informely we can individuate in this equation three terms, where each one has its meaning, in details:

- $E[(y - E[y|x])^2]$: this is the noise, which is the variance of the true label y given only x and is independent of the model or training data. Such noise may be caused by stochastic effects of the method used to acquire the labels, or because the feature representation is inadequate, where minimizing this is guaranteed to minimize the future generalization error of the model;
- $(E_\zeta[\hat{y}] - E[y|x])^2$: this is the bias, is the component of the overall error caused by a given model class being used to learn a particular type of algorithm. The bias term represents the inaccuracy caused by the model class itself, rather than the shape of the computer's underlying data set;

- $E_\zeta[(\hat{y} - E_\zeta[\hat{y}])^2]$: this is the variance of the model, which is the remainder of the learner's squared-loss with respect to the target function.

Variance reduction strategies have some practical disadvantages in terms of processing costs. For each new instance, estimating the variance of the output requires inversion of a K matrix, where K is the number of model parameters, with a time complexity of $O(UK^3)$, where U is the size of the U query pool. This quickly becomes intractable for large K, which is common in work such as natural language processing. Some approaches exist to reduce this complexity, e.g., Hoi et al. [34] use principal component analysis to reduce the dimensionality of the parameter space, but generally these methods remain significantly slower than simpler query algorithms such as uncertainty sampling.

2.3.6 Density-Weighted Methods

The central idea of error estimation and variance reduction systems is that they focus on the entire input space rather than individual cases. The example is certainly on a taxonomic boundary, but it is not "representative" of other individuals in the distribution, so knowing its label is unlikely to improve accuracy over the data set. The main idea is that instances provide information not only as instances of uncertainty, but as instances that are "representative" of the underlying distribution. Therefore, we want to query instances, as proposed by Settles and Craven [82] (but not only those), as follows:

$$x^*_{ID} = \underset{x}{argmax}\phi_A(x) \times \left(\frac{1}{U}\sum_{u=1}^{U} sim(x, x^U)\right)^\beta \tag{2.16}$$

where $\phi A(x)$ represents the informativeness of x according to some query strategy A, while the second part of the equation weights the informativeness of x according to its average similarity to all other instances in the input distribution, subject to a parameter β that controls the relative importance of the density term. Clustering and representativeness measures are critical to the

way the data are queried for interactive active learning with real-time oracles. McCallum and Nigam [62] developed a density-weighted Query-By-Committee approach for text classification with naive Bayes, which is a special case of information density, whereas Xu et al. [98] use clustering to construct query sets for active learning in batch mode with SVM. Settles and Craven [82] have shown that if densities can be efficiently precomputed and cached for later use, the time required to select the next query is essentially no different from the basic information measure (e.g., uncertainty sampling).

2.3.7 Balance Exploration and Exploitation

The selection of instances to label is viewed as a conflict between exploration and exploitation of the data space representation. This tradeoff is managed by modeling the active learning issue as a contextual bandit problem. For example, Bouneffouf et al. [11] present a sequential algorithm called Active Thompson Sampling (ATS), which, in each round, assigns a sampling distribution on the pool, samples a point from this distribution, and queries the oracle to obtain the label of the sampling point.

2.3.8 Exponentiated Gradient Exploration for Active Learning

Many traditional active learning algorithms focus on refining the decision border rather than exploring new, more informative regions. In this context, Bouneffouf [10] introduces EG-Active, a sequential algorithm that can improve any Active learning algorithm through optimal random exploration. The method performs a sampling procedure in each iteration to select a new ϵ from a ϵ finite set of possibilities. The Exponentiated Gradient (EG) [49] (this algorithm uses the components of the gradient in the exponents of factors that are used in updating the weight vector multiplicatively) is used to uniformly establish and update the probabilities associated with the candidates. This updating rule raises the likelihood of a candidate ϵ if it results in a user

click. First, is assumed that is available a finite number of ϵ candidate values, indicated as $(\epsilon_1, ..., \epsilon_T)$, and it aims to find the best ϵ from this collection. To that aim, the EG-Active introduces $p = (p_1, ..., p_T)$, where p_i represents the likelihood of utilizing ϵ_i in the $\epsilon - Active$ algorithm. These probabilities are initially set to $\frac{1}{T}$ and then iteratively adjusted over iterations. The algorithm uses a set of weights $w = (w_1, ..., w_T)$ to keep track of each ϵ_i's performance and updates them using the EG algorithm. If the algorithm obtains a click from utilizing ϵ_i, the idea is to increase w_i. Finally, the procedure computes p by smoothing w and normalizing it. The indicator function is $I[z]$, k is a regularization factor used to deal with singular w_i, and the smoothing factors in weights updating are *tau* and *beta*.

Algorithm 1 EG-Active

Input: $(\epsilon_1, ..., \epsilon_T)$: candidate values for ϵ

β, τ, k: parameters for EG

N: number of iterations

$p_k \Leftarrow \frac{1}{T}$ and $w_k \Leftarrow 1, K = 1, ..., T$

for i = 0 **to** N **do**

Sample d from discrete $(p_1, ..., p_T)$

Run the $\epsilon - Active$ with ϵ_d

Receive the feedback r_t

$w_k \Leftarrow w_k exp(\frac{\tau[r_i I(k=d)+\beta]}{p_k}), K = 1, ..., T$

$p_k \Leftarrow (1-k)(\frac{w_k}{\sum_{j=1}^{T} w_j} + \frac{k}{T}), K = 1, ..., T$

end for

2.3.9 Conformal Predictors

Conformal prediction (CP) [86] is a set of algorithms designed to measure the uncertainty of machine learning model predictions, which are made for an online section in its purest form. After predicting a label, its actual label is known before the next prediction; consequently, the underlying model can be retrained with this additional data point and the next prediction will

be based on a calibration set that includes $n + 1$ data points, whereas the previous model contained n data points. CP methods achieve this by calculating and comparing the nonconformity measures, typically called *alpha* values, of the training set instances with the measures calculated for the test set instances. Conforming predictors are classified in two ways (these differ primarily in terms of computational complexity and suitability for regression or classification applications):

- **Inductive algorithms**: that train one or more machine learning models that are re-used for future test items and can be used for both classification and regression tasks;
- **Transductive techniques**: that re-train the model for each test object and are only suitable for classification tasks.

CP provides high levels of accuracy and confidence, but is computationally inefficient; for example, when CP is combined with a technology that requires long training durations, such as neural networks, the problem of computational inefficiency becomes huge. Papadopoulos et al. [67] proposed a conformal inductive prediction (ICP) technique, which allows a confidence predictor to be built on neural network without the high computational costs of CP. This technique includes practice-relevant confidence metrics alongside its predictions while maintaining the computational efficiency of the underlying neural network. The algorithm is given below:

Algorithm 2 Conformal Prediction with Neural Networks

Input: $z_{m+t} = (x_{m+t}, y_{m+t}), t = 1, ..., q$: samples in calibration set

$x_{l+g}, g = 1, ..., r$: test pattern

Split the training set into the proper training set with $m < l$ examples

Split the calibration set with $q := l - m$ examples

Use the proper training set to train the Neural Network

for z **in** calibration set **do**

$o_1^{m+t}, ..., o_c^{m+t}$ = output obtained by x_{m+t} to the neural network

α_{m+t} = non-conformity score of (x_{m+t}, y_{m+t}) obtained by (2.17) (2.18)

end for

for x **in** test pattern **do**

$o_1^{l+g}, ..., o_c^{l+g}$ = output obtained by x_{l+g} to the neural network

for $Y_u, u = 1, ..., c$ **in** possible classification **do**

compute $\alpha_{l+g} = \alpha_{l+g}^{(Y_u)}$ of the pair (x_{l+g}, Y_u) by applying to the neural network (2.17) (2.18)

calculate the p-value $p(Y_u)$ of the pair (x_{l+g}, Y_u) by applying (2.19) to the nonconformity scores of the calibration examples and $\alpha_{l+g}^{(Y_u)}$

end for

Predict the classification with the smallest non-conformity score

Output as confidence to this prediction one minus the second largest p-value

Output as credibility the p-value of the output prediction i.e. the largest p-value

end for

Where:

- z: an example;
- l: is the number of training examples;
- m: a limit split size of l;
- x: input pattern;

- r: is the number of test patterns;
- c: is the number of possible classifications;
- α: non conformity score;
- y: a possible label;
- o: output.

With the non-conformity measures (the second with an hyperparameter γ useful to gain control over which category of outputs will be more important in determining the resulting non-conformity scores):

$$\alpha_i = \max_{j=1,...,c:j\neq u} o_j^i - o_u^i \tag{2.17}$$

$$\alpha_i = \frac{\max_{j=1,...,c:j\neq u} o_j^i}{o_u^i + \gamma} \tag{2.18}$$

And $p - value$ function:

$$p(Y_j) = \frac{\{i = m+1, ..., m+q, l+g : \alpha_i \geq \alpha_{l+g}^{(Y_j)}\}}{q+1} \tag{2.19}$$

2.3.10 User Centered Labeling Strategies

The central idea of this family of techniques is driven by the fact that learning occurs by reducing the dimensionality of graphs and figures such as scatter plots, in which the user is asked to classify the data collected as categorical, numerical, relevance ratings, and relations between two occurrences.

In this context, Jürgen et al. [8] proposed a technique called visual-interactive labeling (VIAL) that provides users with an active role in the labeling process, with the goal of combining the potential of humans and machines to make labeling more efficient. This technique contributes to a systematic quantitative analysis of the strategies of users, who employ different tactics when marking occurrences with interactive visual interfaces. In this work,

they discovered the building blocks of users' tactics, coded them, and examined their potential for various machine learning tasks. These experiments led to the following key results:

- Guessed that specific user tactics can significantly reduce the bootstrap (cold start) problem during the early labeling phases;
- Guessed that in later stages, they have the potential to surpass conventional active learning systems;
- Examined the discovered fundamental building elements, which may be used to develop new selection techniques.

2.3.11 Meta Learning Strategies

AL strategies are usually based on heuristics or even theoretical measures, but they are not learned because they are used directly during training. Described below are some models that aim to use a meta-learning approach (this is a subfield of ML in which machine learning algorithms are applied to metadata about machine learning experiments; the main goal is to use metadata to understand how machine learning can become flexible in solving learning problems, thus improving the performance of existing learning algorithms or learning - inducing - the learning algorithm itself, thus learning to learn).

2.3.11.1 RALF

Ebert et al. [22] analyzed several sampling criteria, including a new density-based criterion, and demonstrated the importance of combining exploration and exploitation sampling criteria. They also showed that combining time-varying sampling criteria often improves performance. They proposed a new feedback-driven framework based on reinforcement learning (this is a method with reward values assigned to the different stages addressed by the algorithm; the goal of the model is then to accumulate as many reward points as possible and, eventually, achieve a final goal) by modeling criterion selection

as a Markov decision process (this is a random process in which the transition probability that determines the transition to a system state depends only on the immediately preceding system state "Markov property" and not on the way that state was arrived at). Their method does not require prior knowledge of the data set or sampling criteria, but can adapt the sampling strategy through experience during the learning process. They addressed this problem by formulating active learning as a Markov decision process (MDP). Their model has two parameters for Q-Learning, the number of states and actions that must be kept as small as possible to speed up initialization. All parameters are the same for all datasets. There is no tuning for a specific dataset. The challenge they faced is the initialization of the method, since they start from an empty Q table. Ideally, they visit each state-action-pair once or twice but this is intractable for a large state and action space. The number of iterations are limited and they would try out many transitions that are harmful for our learning process. Therefore, they proposed a guided initialization phase. They computed the expected entropy reduction $\hat{r}_i^{(t)}$ for all actions a_i. Each action a_i requests a label for sample x_i. As they do not know the label for this sample, they applied their classifier for each class and calculate the overall entropy. These entropies are weighted by their current prediction probability $p(y_{ij}|x_i)$:

$$\hat{r}_i^{(t)} = \sum_{j=1}^{c} p(y_{ij}|x_i) \sum_{k=1}^{n} Ent_j(x_k) \tag{2.20}$$

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

2.3.11.2 COMB

Baram et al. [5] proposed a framework for combining an ensemble of active learners online to accelerate learning progress in pool-based active learning. Based on a well-known competitive algorithm for the multi-armed bandit

(MAB) problem, they created an active learning algorithm. The MAB is a problem in which a finite set of resources must be allocated among alternative choices in a way that maximizes expected gain, when the properties of each choice are only partially known at the time of allocation and can be better understood with the passage of time or by allocating resources to the choice. One of the most difficult aspects of selecting the best performing active learners online is accurately estimating their progress during the learning session. For this purpose, they proposed a simple maximum entropy criterion that provides effective estimates in realistic contexts. The proposed combination algorithm, called COMB, is based on the EXP4 MAB algorithm. The EXP4 proposed by Auer et al. [3] stands for Exponential weighting for Exploration and Explotation with Experts.The concept of the algorithm is simple: since exponential weighting worked so well in the standard bandit problem, we should apply it to the problem at hand. However, since the goal is now to compete with the best expert a posteriori, we should score the experts rather than the shares. Consequently, the algorithm will maintain a probability distribution, denoted Q_t, on the experts and use it to determine the next action. Once the action is selected, we can use our preferred reward estimation procedure to estimate the rewards for all actions, which can be used to estimate the total reward that individual experts would have earned up to that point, which can be used to update Q_t.

COMB algorithm utilizes an ensemble of active-learning algorithms and tracks online the best algorithm in the ensemble. Steps 1 and 2 compute the active learners' advice probability vectors, as required by the original EXP4 algorithm. In Step 1, each algorithm in the ensemble provides a scoring vector that "rates" each point in the pool U. In practice, the three algorithms under consideration provide such ratings naturally:

- SIMPLE [94] makes use of the kernel distance from the decision hyperplane;
- SELF-CONF [78] makes use of the expected loss;

- KFF makes use of the kernel distance from the current training set.

In Step 2, they scale these scoring vectors using a scaling parameter β and the Gibbs probability function $exp\{-\beta x\}$. After creating the advice probability vectors for the active learners (in Step 2), they project the pool U over high probability candidate instances in Step 3. U_e represents the projected pool. The parameter α controls this projection, and an instance x in U remains in U_e if at least one active learner assigns x a probability mass greater than α. In Step 6, the learner chooses one (unlabeled) point x_q from U_e as the next query. According to EXP4, this choice should be random according to the distribution computed in Step 5. In Step 10 is calculated the "utility" of the last query, that is defined using the (convex) function e^x on the entropic reward calculated in Step 9.

Algorithm 3 COMB

Input: A Pool $U = x_1, ..., x_n$; (ii); An ensemble $ALG_{jj}^{k} = 1$ of k active learners; An initial training set L_0

Parameters: A probability threshold α; A probability scaling factor β; A bound g_{max} on the maximal reward

Initialize expert weights: $w_j = 1, j = 1, ..., k$

for $t = 1$ **to** N **do**

Receive advice scoring vectors from $ALG_j, j = 1, ..., k : e^j(t) = (e_1^j(t), ..., e_n^j(t))$. $e_i^j(t)$ is the score of the i-th point in the pool. The scores are normalized to lie within [0,1].

For each $ALG_j, j = 1, ..., k$, compute an advice probability vector $b^j(t) = (b_1^j(t), ..., b_n^j(t))$ by scaling the advice scoring vectors. For each $b^j(t), j = 1, ..., k$, for each $b_i^j(t), i = 1, ..., n$: $b_i^j(t) = (exp\beta(1e_i^j(t)))/Z$, where Z normalizes $b^j(t)$ to be a probability vector.

Extract from U an "effective pool" U_e by thresholding low probability points: For each point $x_i \in U$ leave x_i in U_e, iff $max_j b_i^j \geq \alpha$. If $|U_e| = 0$, reconstruct with $\alpha/2$, etc. Set $n_e = |U_e|$.

set $\gamma = \sqrt{\frac{n_e ln\, k}{(e-1)g_{max}}}$

Set $W = \sum_{j=1}^{k} w_j$ and for $i = 1, ..., n_e$, set $p_i = (1\gamma) \sum_{j=1}^{k} w_j b_i^j(t)/W + \gamma/n_e$.

Randomly draw a point x_q from U_e according to $p_1, ..., p_{n_e}$.

Receive the label y_q of x_q from the teacher and update the training set and the pool: $L_t = L_{t1} \cup (x_q, y_q); U_{t+1} = U_t x_q$.

Train a classifier C_t using L_t.

Use C to classify all points in U and calculate $H_t = H_t(\frac{|C^{+1}(U)|}{|U|})$, the entropy of the resulting partition $C^{+1}(U), C^1(U)$.

Calculate the "reward utility" of $x_q : r(x_q) = ((e^{H_t} e^{H_{t1}})(1e))/(2e2)$.

For $i = 1, ..., n$, set $\hat{r}_i(t) = r(x_q)/p_q$ if $i = q$ and $\hat{r}_i(t) = 0$ otherwise.

Reward/punish experts: $w_j(t+1) = w_j(t) exp(b^j(t)\hat{r}(t)\gamma/n_e)$.

end for

The algorithm successfully combines elements of statistical and online (adversarial) learning. Extensive empirical results show that this algorithm can keep up with the best algorithm in the ensemble on real-world problems.

2.3.11.3 LAL

Ksenia et al. [51] proposed a data-driven approach to AL. The main idea is to train a regressor that predicts expected error reduction for a candidate sample in a specific learning state. They are not limited to working with existing AL heuristics because they formulate the query selection procedure as a regression problem; instead, they learn strategies based on experience from previous AL outcomes.

Assume the representative dataset is made up of a training set D and a testing set D'. Let f be a classifier with a predefined training procedure. They begin gathering data for the regressor by dividing D into a labeled set L_t of size τ and an unlabeled set U_τ of the remaining points (Algorithm 4). They then train a classifier f on L_τ, yielding a function f_τ that they can use to predict class labels for elements x' in the test set D' and estimate the test classification loss l_τ. We define the classifier state using K parameters $\phi_\tau = \{\phi_\tau^1, ..., \phi_\tau^K\}$, which are specific to the classifier type and are sensitive to changes in the training set while remaining relatively invariant to stochasticity of the optimization procedure. They then choose a new datapoint x at random from U_τ, which is defined by R parameters $\phi_x = \phi_x^1, ..., \phi_x^R$. They create a new labeled set $L_x = L_\tau \cup \{x\}$ and train f. The test set loss l_x is produced by the new classifier f_x. Finally, they keep track of the difference between the previous and new loss $\delta_x = l_\tau - l_x$, which is associated with the learning state in which it was received. The learning state is characterized by a vector $\xi_\tau^x \leftarrow [\phi_\tau^1 \cdots \phi_\tau^K \psi_x^1, ..., \psi_x^R] \in \mathbb{R}^{K+R}$ whose elements depend both on the state of the current classifier f_τ and on the datapoint x. To build an AL strategy "LAL-Independent" they repeat the "Data Monte Carlo" procedure for Q different initializations $L_\tau^1, L_\tau^2, ..., L_\tau^Q$, and T various labeled subset sizes $\tau = 2, ..., T + 1$ (Algorithm 5). For each initialization q and iteration τ, they sample M

different datapoints x each of which yields classifier/datapoint state pairs with an associated reduction in error. This results in a matrix $\Xi \in \mathbb{R}^{(QMT)\times(K+R)}$ of observations ξ and a vector $\Delta \in \mathbb{R}^{QMT}$ of labels δ.

Algorithm 4 Data Monte Carlo

Input: training set D and test set D', classification procedure f, partitioning function $Split$, size τ

$L_\tau, U_\tau \leftarrow Split(D, \tau)$

train a classifier f_τ

estimate the test set loss l_τ

compute the classification state parameter $\phi \leftarrow \{\phi_\tau^1, ..., \phi_\tau^K\}$

for $m = 1$ **to** M **do**

select $x \in U_\tau$ at random

form a new labeled dataset $L_x \leftarrow L_\tau \cup \{x\}$

compute the datapoint parameters $\psi \leftarrow \{\psi_x^1, ..., \psi_x^R\}$

train a classifier f_x

estimate the new test loss l_x

compute the loss reduction $\sigma_x \leftarrow l_\tau - l_x$

$\xi \leftarrow [\phi_\tau^1 \cdots \phi_\tau^K \; \psi_x^1, ..., \psi_x^R], \sigma_m \leftarrow \sigma_x$

end for

$\Xi \leftarrow \{\xi_m\}, \Delta \leftarrow \{\sigma_m\}_1 \leq m \leq M$

Return: matrix of learning states $\Xi \in \mathbb{R}^{M\times(K+R)}$, vector of reductions in error $\Delta \in \mathbb{R}^M$

Their insight is that observations ξ should lie on a smooth manifold and that similar states of the classifier result in similar behaviors when annotating similar samples. From this, a regression function can predict the potential error reduction of annotating a specific sample in a given classifier state. In algorithm 5 looks for a mapping $g : \xi \mapsto \delta$. This mapping is not specific to the dataset D, and thus can be used to detect samples that promise the greatest increase in classifier performance in other target domains Z. The resulting "LAL Independendent" strategy greedily selects a datapoint with the highest

Algorithm 5 Build LAL Independent

Input: iteration range $\{\tau_{min}, ..., \tau_{max}\}$, classification procedure f

$Split \leftarrow$ random partitioning function

generate train set D and test D'

for τ **in** $\{\tau_{min}, ..., \tau_{max}\}$ **do**

  for $q = 1$ **to** Q **do**

    $\Xi_{rq}, \Delta_{rq} \leftarrow DataMonteCarlo(D, D', f, Split, \tau)$

  end for

end for

$\Xi, \Delta \leftarrow \{\Xi_{rq}\}, \{\Delta_{rq}\}$

train a regressor $g : \xi \mapsto \delta$ on data Ξ, Δ

construct "LAL Independendent" $A(g) : x^* = argmax_{x \in U_t} g[\xi_{t,x}]$

Return: "LAL Independendent"

potential error reduction at iteration t by taking the maximum of the value predicted by the regressor g.

For any AL strategy at iteration $t > 0$, the labeled set L_t consists of samples selected at previous iterations, which is clearly not random. However, the dataset D is split into L_τ and U_tau randomly no matter how many labeled samples τ are available. To account for this, they modified the original approach "Build LAL Iterative". Instead of partitioning the dataset D into L_τ and U_τ randomly, they suggested simulating the AL procedure which selects datapoints according to the strategy learnt on the previously collected data (Algorithm 6). It first learns a strategy $A(g_2)$ based on a regression function g_2 which selects the most promising 3rd datapoint when 2 random points are available. In the next iteration, it learns a strategy $A(g_3)$ that selects 4th datapoint given 2 random points and 1 selected by $A(g_2)$ etc. In this way, samples at each iteration depend on the samples at the previous iteration and the sampling bias of AL is represented in the data Ξ, Δ from which the final strategy "LAL Iterative" is learnt. The resulting strategies "LAL Independendent" and "LAL Iterative" are both reasonably fast during the online steps of AL:

Algorithm 6 Build LAL Iterative

Input: iteration range $\{\tau_{min}, ..., \tau_{max}\}$, classification procedure f

$Split \leftarrow$ random partitioning function

generate train set D and test D'

for τ **in** $\{\tau_{min}, ..., \tau_{max}\}$ **do**

  for $q = 1$ **to** Q **do**

    $\Xi_{rq}, \Delta_{rq} \leftarrow DataMonteCarlo(D, D', f, Split, \tau)$

  end for

  $\Xi_\tau, \Delta_\tau \leftarrow \{\Xi_{rq}\}, \{\Delta_{rq}\}$

  train a regressor $g_\tau : \xi \mapsto \delta$ on data Ξ_τ, Δ_τ

  $Split \leftarrow A(g_\tau)$

end for

$\Xi, \Delta \leftarrow \{\Xi_\tau, \Delta_\tau\}$

train a regressor $g : \xi \mapsto \delta$ on Ξ, Δ

construct "LAL Iterative" $A(g)$

Return: "LAL Iterative"

they just require evaluating the RF regressor. The offline part, generating a datasets to learn a regression function, can induce a significant computational cost depending on the parameters of the algorithm. For this reason, "LAL Independendent" is preferred to "LAL Iterative" when an application-specific strategy is needed.

2.4 Applications

Research on Active Learning is mainly concerned with image processing. Active learning is gradually making its way into NLP as well. Active learning deals with how to efficiently handle high-dimensional data query samples and reduce labeling costs when it comes to computer vision work. Active learning allows pseudo-labels to be assigned to samples with high confidence and added to the set of highly uncertain samples queried with the uncertainty-based active learning approach before training the active learning model image classifier using the larger training set. The following are some areas where active learning is helpful.

2.4.1 Gene Expression

Gene expression is the process by which information from a gene is used for the synthesis of a functional gene product, enabling it to produce end products such as proteins or non-coding RNAs and, consequently, to affect a phenotype. These products are often proteins, but the product of non-protein-coding genes, such as transfer RNA (tRNA) and small nuclear RNA (snRNA), is functional non-coding RNA. The fundamental principle of molecular biology, first established by Francis Crick [92], and expanded by subsequent discoveries and RNA replication. Gene expression is the most fundamental level of genetics in which the genotype gives rise to the phenotype, or observable trait. The genotype is the genetic information encoded in DNA, while the phenotype is the consequence of the "interpretation" of that information. Phenotypes often manifest themselves through the creation of proteins that regulate the

structure and development of the organism or act as enzymes catalyzing certain metabolic processes.

Liu et al. [58] developed an approach for active learning with SVM and applied it to gene expression patterns of colon, lung, and prostate cancer tissues. After the training set was constructed, either actively (the most informative m instances were labeled) or passively (examples were labeled randomly), a classifier created a model from the training set, and the model was used to classify the cancer examples (lung cancer, prostate cancer, or colon cancer). SVM proved to be an excellent solution for the active learning system. In addition, SVM has been successfully used to classify cancer using gene expression data. An SVM chooses a small number of crucial boundary samples for each category, known as support vectors, and creates a linear discrimination function that separates them as much as possible. If linear separation is impractical, a "kernel" technique is used to automatically inject the training samples into a higher-dimensional space and develop a separator in that space. SVM produces a hyperplane that divides two different categories of feature vectors with a maximum margin, the distance between the separating hyperplane and the nearest training vector, into linearly separable instances. The support vectors are the training instances closest to the hyperplane. The hyperplane was created by identifying another vector w and a parameter b that minimizes $||w||^2$ and satisfies the conditions listed below:

$$w \cdot x - i + b \geq +1,\ for\ y_i = +1\ Category\ 1\ (pos.) \tag{2.21}$$

$$w \cdot x - i + b \leq -1,\ for\ y_i = -1\ Category\ 2\ (neg.) \tag{2.22}$$

where:

- y_i is the category index (i.e., active, inactive);
- w is a vector normal to the hyperplane;
- $|b|/||w||$ is the perpendicular distance from the hyperplane to the origin;

- $||w||^2$ is the Euclidean norm of w.

After determining w and b, a given vector x can be categorized by the sign $[(w \cdot x) + b]$. The researchers compared the categorization performance of active learning with that of passive learning. The results showed that using the active learning method can achieve good accuracy while reducing the demand for labeled training examples. To achieve 96% overall positives for lung cancer categorization. Active learning resulted in a reduction of more than 82%. The areas under receiver operating characteristic (ROC) curves were above 0.81 in active learning and below 0.50 in passive learning. The researchers demonstrated that active learning with support vector machines can accurately classify tumors based on expression data from DNA microarray hybridization assays and provided some theoretical foundations for the effectiveness of active learning.

Singh et al. [90] addressed the sampling problem for such experiments by calculating which time points should be chosen to reduce data collection costs. They focused on a nascent type of studies that evaluate different signals at each time point and in which raw materials/observations are initially archived and then selectively evaluated later, with analysis being the most expensive stage. They introduced an active learning approach to iteratively select the time points to be sampled, with the objective function being the uncertainty in the quality of the currently estimated time-dependent curve. They demonstrated that their algorithm works effectively and can greatly reduce experimental costs without sacrificing information, using both simulated data and gene expression data. The core of this work is the Choose-Next-Point algorithm, which, given data from j previously sampled time points, uses them to choose the $(j+1) - th$ sampling point:

- Generate a smoothing function for the j time-points:

  Evaluate all possible smoothing parameters for the spline model using cross-validation;

- Use the smooth hing function to choose the next time-point that should be sampled:

  Compute locally-sensitive confidence intervals over the continuous function at all of the sampled time-points;

  Use active learning to suggest the next time-point to sample, based on the confidence intervals.

With each iteration, at step 1, the above algorithm computes an error estimate in order to solve *errThresh*, they choose time-points until this error falls below the error threshold C_e; to solve *costThresh*, they continue until K time-points have been chosen. In details:

- **costThresh**: their method performs better (in terms of e_f^T) than both random and uniform sampling, especially when using more difficult cost targets and harder datasets. they performed the comparison for different cost thresholds and they accepted the estimate to be a good fit with the original if $e_f^T < 1.15$;

- **errThresh**: defining a measure of error e_f over the estimated functions. In an ideal world, such an error measure would react identically to the true error, ef. For simulated datasets, the true error can be calculated and used to assess performance in *errThresh* issue situations. They demonstrated that their online method outperforms the random sampling approach, requiring less samples to obtain the same inaccuracy. The distinction is especially noticeable in "hard" problem scenarios.

They measured the *true error* e_f^T, of $\hat{f}$ as the average difference between the two curves over the time-range of interest:

$$e_f^T = \frac{1}{(t_N^n - t_1^n)} \int_{t_1^n}^{t_N^n} |f^*(t) - \hat{f}(t)| dt \tag{2.23}$$

where:

- f is the true function;

- $\hat{f}$ is the estimated function;
- t is a time-point;
- N quantity of time points.

They introduced an online system using active learning to establish an optimal sampling strategy for time series experiments with high data collection costs. They proposed an efficiently calculated objective function to estimate the uncertainty of estimated smooth splines and demonstrated how this function can be used together with active learning to suggest the next sampling point. This algorithm can be used to sample and estimate any continuous function with a single independent variable with no variance. In a large sensor network, for example, obtaining continuous readings from all sensors incurs prohibitive communication/energy costs. Many of these readings may be redundant at the same time. Given some k, this technique can help identify the ideal selection of k sensors whose combined readings produce the least overall uncertainty in the overall observation.

Begum et al. [6] developed an active learning (AL) model for cancer prediction using SVM together with a feature selection (FS) technique called Symmetrical Uncertainty (SU). The efficiency of the proposed AL and SU combination was demonstrated, and biomarkers or cancer genes discovered by the proposed approach were reported on four gene expression datasets. In addition, studies were carried out on the biological significance of cancer biomarkers obtained from the datasets. With a small labeled dataset and a large number of unlabeled multidimensional data, the whole dataset is divided into training-test partitions of 50-50%. Feature selection is performed only on the 50% training set. Three feature selection methods were used to extract features from the training data:

- **CBAE**: Correlation Based Attribute Evaluation [89] is a method for selecting optimal attribute subset based on correlation using Genetic algorithm (GA), where GA is used as optimal search tool for selecting subset of attributes;

- **SU**: Symmetric Uncertainty [1] is a pure filter based feature subset selection technique which incurs less computational cost and highly efficient in terms of classification accuracy;

- **GRAE**: Gain ratio attribute evaluation [44] is a method to illustrate the significance of feature subset selection for classifying Pima Indian diabetic database.

The retrieved features are employed in the experiment, which is carried out using the proposed method. Following that, the data sets are divided into:

- training dataset;

- pooled dataset;

- test dataset.

On the training data, the classifier is trained. The classifier selects the most informative samples from the set of clustered data in each iteration. Human experts classify the informative samples and add them to the original training data. This updated training set is used to predict the test data. Ten samples are considered training data among the 50% training data, while the remaining samples are considered pooled data for each dataset. The goal is to reduce the error and improve the accuracy of the ASVM framework (their proposed Active SVM) by choosing the most uncertain samples from the pooled data collection. It should be noted that unlabeled clustered data are formed by removing the label vector of the original dataset. The following steps summarize the algorithm:

- **1.** the data set is preprocessed to reduce the dimension of the data by using three feature selection methods introduced previously;

- **2.** the traditional ISVM is trained with the training data. The function, $\Phi(X_p, Y)$ estimates the class label, Y for the training samples, X_p;

- **3.** the classifier queries for the samples from the pooled data. Let X_pbe the set of training data. Let assume that $x_i^p \in R^s$ is the i^{th} vector in X_p. If U is a set of test data, then $\underset{u\ j}{\rightarrow} \in R^S$ is the j^{th} vector in U. In ASVM the selection of the uncertain samples from the set of pooled data P are based on the following criteria:

  train the ISVM using X_P and predict the test set U;

  compute the decision values of the positive and negative uncertain samples within the margin band;

  select uncertain samples p_Φ from P that are closest to the decision boundaries within the margin b;

  the new training set $X_P = X_P \cup p_\Phi$ and the pooled data set $P^1 = P - P_\Phi$ where P_Φ is the set of samples closest to the decision boundaries at each iteration;

  the chosen new samples are provided to the human annotator to annotate the samples by predicting the true classes;

- **4.** retrain the classifier using the updated training set;

- **5.** algorithm terminates after a finite number of Iterations.

The algorithm works in a simple way. The classifier is initially trained with a small number of training samples and then classifies test sets. The ASVM proposed in the AL framework starts by choosing the uncertain samples closest to the decision boundaries from the collected data set. Next, the samples are given to a human expert who assigns the right label to those samples. As the original datasets are available, they provide the correct label to the searched samples in the same way a human annotator would. The original training set is then updated by including the uncertain samples, and the pooled data set is updated by removing the uncertain samples. This process is repeated until the n^{th} iteration, at which point the best results of the n iterations are considered. The results of the proposed model would provide the physician with an unbiased alternative model for predicting cancer subtypes, unlike existing methods

that rely on clinical or histopathological data, which do not always reveal the true outcome. Consequently, for its potential in the clinical management of cancer, it is critical to focus on class prediction. In addition, observation of incorrect genes results in poor performance in typical supervised approaches. Consequently, selected features are critical in classifying data.

2.4.2 Robotics

In order to navigate autonomously in outdoor contexts, complex classification challenges must be solved, as the following (these are examples of classification tasks that have been effectively tackled utilizing supervised machine learning approaches):

- Obstacle detection;
- Road following;
- Terrain categorization;
- etc.

In order to obtain adequate generalization, large volumes of training data are frequently required. In such instances, manually classifying data becomes a costly and time-consuming operation, and AL comes useful.

Dima et al. [19] proposed a strategy to reduce the amount of data to be presented to a human trainer. To identify "interesting" scenes in a dataset, the approach employs kernel density estimation. Their system requires no interaction with a human expert for image selection, and only minor adjustments are needed. They used data acquired with two different vehicles to illustrate its utility in multiple studies (using distance data from laser rangefinders, color, infrared, and texture information acquired):

- **CMU autonomous tractor**: [91] this is a tractor automation system. A task is programmed by a human by driving the relevant routes. The assignment is broken down into subtasks and allocated to a fleet of tractors,

which drive portions of the routes. Each tractor has on-board sensors that identify people, animals, and other vehicles in its route, causing it to stop until it receives instructions from a supervisor via a wireless link;

- **GDRS XUV**: this is being done to investigate the effects of introducing semi-autonomous systems to the Army After Next Scout Platoon. This program is intended to progress and demonstrate the technology needed to construct future unmanned ground combat vehicles through three key thrusts:

  coordinated technology development;

  modeling, simulation, and experimentation;

  technology integration and user evaluation.

  Demo III focuses on the demonstration of technologies that will allow for the construction of small, extremely nimble unmanned vehicles capable of off-road, semi-autonomous operation at speeds of up to 32 km/hr during daylight and 16 km/hr at night by the fourth quarter of fiscal year 2001.

The sensors of the two autonomous vehicles (color cameras, infrared cameras, and laser rangefinders) are calibrated against each other, which means that they can obtain color and infrared information for each three-dimensional (3D) point returned by the laser rangefinder that is in the field of view of our cameras, with very mild assumptions about the geometry of the scene. Similarly, 3D laser dots can be projected into any photograph. The assumption is that the image and distance sensors are not too far apart relative to the distance of the scene being imaged. Even if this assumption is violated, problems arise only in the case of occlusions. Both vehicles are capable of collecting huge amounts of laser and image data. The most recently recorded laser data is projected into each acquired image. The image is divided into a grid of rectangular patches, and numerous features are retrieved from each data mode. Each data register contains multiple images, and each image contains many patches, from which several features are retrieved. Once the features are retrieved, the

image patches can be classified as obstacles/non-obstacles or roads/non-roads using any conventional learning method, such as neural networks, support vector machines, decision trees and so on. The standard data labeling technique involves manually scanning the entire data log, selecting photographs that a human expert considers interesting, and categorizing portions of images as belonging to various classes. Their method is a non-interactive strategy that evaluates the attributes associated with each image in the dataset to identify photographs that are "surprising" given the probability distribution of the rest of the data. Importantly, this is done before the data are labeled by a human expert. Consequently, the author refers to his technique as "unlabeled data filtering". This differs from better known active learning strategies, such as trust-based query selection or vote-based query selection, which start with a small amount of labeled data and then interactively offer other data to a human expert for classification. However, using their method does not preclude the use of another interactive active learning strategy. On the contrary, their method can be used to produce a good small data set to initiate the other interactive methods. This method reduces the amount of data labeling needed for outdoor categorization problems. The results obtained on datasets from very different environments show that the method can be used to automatically "filter" huge datasets and retrieve salient photos, achieving strong coverage of the feature space. The preliminary classification test revealed that, under some circumstances, the error rates obtained using 5 informative images can be as excellent as those obtained by labeling a complete dataset of hundreds of images. These results represent an important step toward using machine learning to solve large-scale classification challenges in outdoor robotics. This active learning approach can be useful in two ways:

- it minimizes the requirement for expensive labeled data;
- it reduces the amount of data that needs to be processed by field than previously possible.

They are interested in applying ways to increase the speed and resilience with

which probability density functions may be computed.

Pillai et al. [69] investigated active learning methodologies applied to three language problems of varying difficulty to determine which methods are appropriate for improving data efficiency in learning. Their method was designed to analyze the complexity of the data in this combined problem space and to report how the characteristics of the underlying task and design factors affect the results, such as:

- feature selection;
- classification model.

They found that representativeness, along with diversity, is critical in the selection of data samples. They learned the relationships between the RGB-D (color+depth) images of elements in a dataset and the language that defines them using various active learning approaches. The goal then becomes the identification of concepts with grounded meaning, the creation of lexical terms in a formal representation of the underlying meaning, and the learning of visual classifiers that correctly identify the elements referred to in subsequent language interpretation tasks. At a high level, language is grounded in the learning of concept-specific classifiers such as color, shape, and object; the many types of concepts are acquired through human-provided descriptions of selected things. Each concept is linked to a learned classifier, and all selected objects described by that concept are used as training data for that classifier. For the actual grounding, they relied on existing datasets and categorization algorithms. They stated that the evaluations performed in this work are intended to compare the success of different active learning methodologies for the same topic. To mimic the limited training available from human encounters, they limited the training data to a single description of each object. They employed active learning methodologies to perform reproducible experiments in which the objects (and accompanying training and evaluation information, such as identified descriptions and concepts) are extracted from a pre-existing data pool rather than produced de novo through human interaction. They

changed the active learning strategy used to select new object descriptions to add to the training pool in their key experiments. In addition, they experimented with different features and categorization algorithms. Because the challenge focuses on selecting objects for which to collect labels, it is similar to asking a human for a description of a specific object, but it allows them to conduct large-scale and more repeatable tests. Their purpose is to study data selections in limited situations to improve performance in the early stages. They are not about improving absolute learning performance; using a new or complex strategy risks introducing unknown confounding factors. Their system prioritizes the categorization of the most informative and different objects from a pool of unlabeled objects. As active learning strategies, they used probabilistic clustering (especially point process modeling). Because their data are inherently noisy, they found that variations of Gaussian mixture models (GMMs) and deterministic point processes (DPPs) were robust selection methods in their tests. GMMs can handle mixed memberships, and soft cluster assignment allows for uncertainty. They used parametric methods in their learning strategies because they are statistically more stable than nonparametric models. Consequently, they focus on GMM-based and DPP-based techniques applied to visually grounded object attributes to choose the most informative points from a pool of unlabeled instances. They do not explore deep learning algorithms, which often perform well on huge datasets, because they focus on learning from limited data. Five different active learning models, one uncertainty-based method (GMM Log Density) and four pool-based techniques are analyzed in all experiments:

- GMM Max Log Density Based;
- VL-GMM;
- DPP.

This is a novel method to solve the grounded language problem: a structured DPP-based active learning technique (GMM-DPP). They evaluated these variants of active learning strategy against a basic random sample on our three

variables (color, shape and object). By querying all N objects at once, their algorithms identify instances that are informative and different. This method is also known as "batch mode." To allow for more extensive and reproducible testing, they drew from an existing pool of human-provided descriptors rather than explicitly seeking additional labels through interaction. In their work, they thoroughly examined various active learning techniques to support unrestricted natural language with real sensor data. They showed that active learning has the potential to reduce the amount of data needed to base language on objects, a hot topic in NLP, robotics and machine learning from sparse data in general. They also suggested approaches that would be appropriate given the perceptual and verbal complexity of a situation.

Chao et al. [14] instead used nonverbal gestures to ask a human teacher questions about a presentation in the context of a social interaction to perform active learning on the Simon robot (Figure 2.4). The goal was to build a learner robot that can exploit transparency to help a human teacher provide better instruction. Preliminary data analysis reveals that transparency through active learning has the potential to increase the accuracy and efficiency of the teaching process. However, their research seems to show potential negative impacts from the human teacher's perspective in terms of interaction balance. These preliminary results argue for control strategies that strike a balance between leading and following during a social learning engagement.

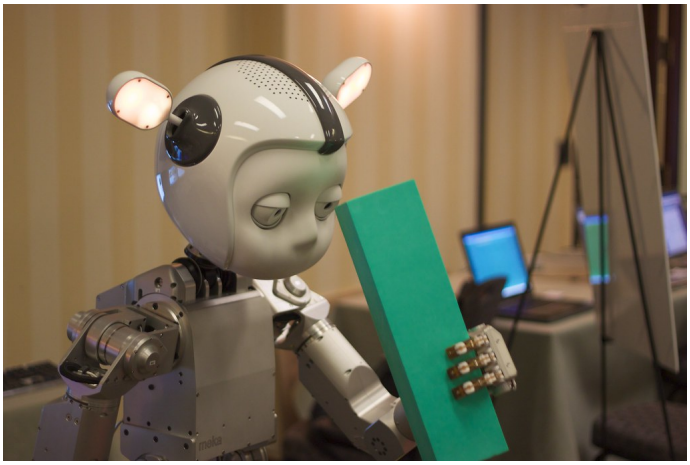


Figure 2.4: Simon is a socially aware robot designed to explore ways of interacting with humans naturally, through sounds, movements, and gestures.

A good teacher maintains a mental model of the learner's understanding in a contextual learning encounter and organizes the learning task appropriately with timely feedback and directions. The learner contributes to the process by communicating his or her own internal condition (e.g., expressing understanding, confusion, attentiveness, etc.) This reciprocal and closely connected relationship enables the learner to use the teacher's instructions to construct the necessary representations and associations. This type of human tutoring is a social and cooperative activity. The human input scenario has received some attention in the machine learning and robotics communities. Much past research has focused on the scenario in which a machine learns by observing human behavior. Many of these learning systems benefit from this work on transparency and active learning. Their specific goal is to create more interactive systems that can learn in real time from ordinary people. Research has been conducted on the computational benefits of teacher/learner pairs, particularly in relation to active learning. AL is a field concerned with the effective use of an external oracle. One system demonstrates trust-based active learning with human (non-social) labelers. Queries can be seen as a type of transparency in the learning process, which is why we are interested in applying active learning in the context of human-robot interaction. The difficulty lies in devising methods that allow the robot to use natural social cues

to solicit acceptable examples from a human partner in the learning activity. Research on active learning focuses on how to select questions that maximize information gain or otherwise make optimal use of the oracle. This work, on the other hand, focuses on the problem of allowing the oracle to be a machine-learning non-expert and the use of active learning for learning robotic tasks. They presented a learning system, an experiment with human teachers and data-based observations. They used query-by-committee for sample selection in Simon's active learning method. This method employs a committee of competing hypotheses to choose the example with the greatest difference between the hypotheses in terms of expected labels. In their implementation, the committee is the version space of the symbol, that is, the set of all hypotheses with the highest score monitored during symbol learning. The committee assigns a value of $+1$ for a positive label and a value of -1 for a negative label, implying that the example with the smallest mean value of the expected label in absolute value in the entire symbol version space will yield the largest information gain. Iterative labeling of the instances chosen by the committee has the effect of eliminating as much of the committee as possible until only one correct hypothesis remains, minimizing committee entropy. When there are multiple instances with the same best value, the active learner can impose additional ordering or select them randomly. The next section on transparency methods describes additional restrictions used by the active learner. Label prediction is done with a hypothesis committee using the committee's majority label, which is also the sign of the average label. The measure of label confidence is the gap between the majority label and the mean label. Therefore, confidence is 0 in the degenerate case of no majority label. They used active learning to teach the Simon robot to query an external entity about areas of uncertainty in its hypothesis space, and then created a series of nonverbal gestures to provide feedback on uncertainty, as well as nonverbal gestures for the robot to ask a human teacher about the feature space in a tangram symbol learning task. In a preliminary study, they found that transparency through active learning has the potential to improve both the accuracy and efficiency of a teaching process;

moreover, they found that people who understood the robot's questions could train a model with perfect accuracy relatively quickly. These people were also confident that the trained model was complete. However, they felt that an encounter driven entirely by active learning might be undesirable from the human teacher's point of view, because both subjects observed in this scenario showed that they wanted to exert more influence on the process. Subjects in the non-transparent condition who did not see the version of the robot with active learning, on the other hand, expressed a desire for the robot to take the initiative or communicate what it knew or did not know.

2.4.3 Wearable Devices

Human activity monitoring has grown in popularity in recent years and is currently employed in a wide range of industries and applications. Most proposed activity tracking algorithms have focused on the use of inertial sensors in smartphone devices or other worn sensors, but wearable inertial sensors are not interactive and cell phones are cumbersome to wear. As wristwatch technology advances, new options emerge for user engagement and highly accurate and personalized identification of activities. Specific behaviors can be learned by interrogating unknown actions with Active Learning, an interactive machine learning approach.

There are several publications in this context, one of which was introduced by F. Shahmohammadi et al. [87], who propose a smartwatch-based active learning system for task identification, which identifies five typical daily tasks. According to the results of this study, this approach has an accuracy of 93.3% on 12 people. The results show that an interactive learning technique based on active learning in smartwatches outperforms smartphones and other devices in task detection tests. This work used twelve people who used an application designed for wristwatch to collect activity data. In the first phase of the experiment, they trained a supervised model on the data set, which served as the basic model for activity recognition, and then evaluated models that used various combinations of inputs from different sensors. In this work, they

employed two active learning algorithms, namely Uncertainty Sampling and Query by Committee, to perform online active learning and query subjects as sensor data were received. They compared the performance of these two methods for activity recognition to determine which method is more appropriate for the application, retrained the classifier for each dataset, and made predictions for the opposite dataset that had not been queried. With this method, the annotation of a sample did not cause the model to have deceptively high accuracy because of prior knowledge of its future classifications. The researchers achieved high accuracy with a small number of training samples, demonstrating that this strategy is very valid; moreover, they showed that the use of software-based wrist sensors can increase the accuracy of activity detection models.

In this context, Jie Xu et al. [97] proposed a new online contextual learning method for activity classification based on data acquired from low-cost inertial sensors and smartphones. The proposed technique is able to address the special problems associated with online, personalized and adaptive activity classification without the need for an individual training period. Another significant problem with activity categorization is that labels can change over time as the data and activity being tracked change, and the actual label is often expensive and difficult to obtain. The proposed algorithm can actively learn when to request the actual label, weighing the benefits and costs of doing so.

Another interesting work, presented by Abhijith Ragav et al. [73], focuses on health, specifically that psychological stress is increasingly present in people and early detection is critical to avoid health problems. In this context, they provided a method to represent model uncertainty in Bayesian neural networks through approximations using Monte-Carlo (MC) Dropout. Researchers process unlabeled data in real time through appropriate ground-truthing (active learning) approaches, which help construct affective states (labels) while simultaneously selecting only the most useful data points to query from an oracle. For active learning, this is coupled with appropriate acquisition features.

Empirical results on a common stress and affective detection dataset (SWELL [50]), explored on a Raspberry Pi 2, show that the proposed framework provides significant performance improvement during inference, with significantly low number of aggregation points acquired across multiple acquisition features. Variation Ratios achieves an accuracy of 90.38%, equivalent to the maximum accuracy obtained during training on about 40% less data.

2.4.4 Data Analysis

Another field in which active learning is receiving much attention is computer vision, since preparing a good set of labeled images for vision data analysis is time-consuming and costly. As an instance selection criterion, most of the known active learning techniques in computer vision use the uncertainty metric. Although most uncertainty instance selection algorithms are successful in many situations, they do not take into account the information present in a significant number of unlabeled instances and are prone to query outliers.

In this field, Suju Rajan et al. [74] have proposed an active learning methodology that uses fewer labeled data points than semisupervised algorithms to update current classifiers. In addition, unlike semi-supervised approaches, this strategy is particularly suitable for learning or adapting classifiers when the spectral signatures of labeled and unlabeled data differ significantly. Consequently, this active learning approach is also useful for identifying a set of spatially/temporally related images with different spectral signatures. The semi-supervised active learning strategy was tested on single, spatially/temporally linked hyperspectral datasets. This active learning technique can be combined with any classifier that establishes the decision boundary by estimating a posteriori class probabilities. They make use of:

- The a posteriori probability distribution function $P(Y|X)$ to guide our active learning process;
- Loss function in that they attempt to increase the information gain between $P_{D_L^+}(Y|X)$ and $P_{DL}(Y|X)$, i.e respectively the a posteriori pdfs

estimated from D_L that is set of random instances and D_L^+ that is a subset for which the true target value.

Maximizing the expected information gain between $P_{D_L^+}(Y|X)$ and $P_{DL}(Y|X)$ is equivalent to selecting the data point $\hat{x}$ (the instance to select) from D_{UL} such that the expected KL (Kullback–Leibler) divergence between $P_{D_L^+}(Y|X)$ and $P_{DL}(Y|X)$ is maximized, by first selecting $\hat{x} \in D_{UL}$ and assuming $\hat{y}$ to be its label. Let $D_L^+ = D_{UL} \backslash \hat{x}$, $D_{L^+} = D_L \cup (\hat{x}, \hat{y})$ and $|D_{UL}^x|$ be the number of data points in the set D_{UL}^x. Estimating via sampling, the proposed KL^{max} function can be written as:

$$KL_{D_L^+}^{max}(\hat{x}, \hat{y}) = \frac{1}{D_{UL}^+} \sum_{x \in D_{UL}^+} KL(P_{D_L^+}(Y|x)||P_{D_L}(Y|x)) \tag{2.24}$$

The KL divergence between the two probability distributions is defined as:

$$KL(P_{D_L^+}(Y|x)||P_{D_L}(Y|x)) = \sum_{x \in D_{UL}^+} P_{D_L^+}(Y|x) log(\frac{P_{D_L^+}(Y|x)}{P_{D_L}(Y|x)}) \tag{2.25}$$

Interesting is that assigning a wrong class label to $\hat{y}$ for $\hat{x}$ can result in a large value of the corresponding $KL_{D_L^+}^{max}$. Then is used expected KL distance from $P_{D_L^+}(Y|x)$ and $P_{D_L}(Y|x)$ with the expectation estimated over $P_{D_L}(Y|x)$, and then select the $\hat{x}$ that maximizes tance as:

$$\hat{x} = \underset{\hat{x} \in D_{UL}}{argmax} \sum_{\hat{y} \in Y} KL_{D_L^+}^{max}(\hat{x}, \hat{y}) P_{D_L}(\hat{y}|\hat{x}) \tag{2.26}$$

The effectiveness of this strategy is highly dependent on the accuracy of posterior probability estimates. The high dimensionality of more than 100 features in hyperspectral data, together with the lack of labeled data, can lead to biased estimates of the probability distribution parameters. The dimensionality of the data is reduced by using feature selection/extraction techniques, and the Expectation Maximization algorithm is used along with the active learning process to improve the estimates.

Goo Jun et al. [41] proposed an approach for more effective knowledge transfer using active learning based on Rajan's previous work, which allows for

faster learning curves by adjusting the distributions of labeled data differently for old and new data, so that the classifier can effectively transfer its learned knowledge from one region to a spatially or temporally separated region whose spectral signature is different. Since it is not practical to obtain the ground truth of all areas at multiple times and training a classifier for land cover characterization based on hyperspectral images usually requires large amounts of labeled data. Obtaining ground truth class labels of a remote sensing image is expensive. Assuming that there are two distinct datasets from temporally or spatially distant regions, which we will denote as areas 1 and 2, respectively. The $KL-max$ technique treats all samples from area 1 and new samples from area 2 the same way for ML estimation, although their distributions may be very different. As a result, since we have only a small percentage of area 2 samples compared to the number of area 1 samples, the expected distribution is significantly closer to the area 1 distribution. Instead of using cumulative updates, they create a new distribution of weights for each classifier. The default technique is to provide lower weights to the misclassified samples in D_L and higher weights to the misclassified samples in D_N, the set of newly obtained labeled samples. This is because more samples in D_N resulted in a more reliable classifier, which reinforces the hypothesis that misclassified samples in D_L are less useful. Consequently, misclassified samples in D_L should be assigned lower weights to obtain more samples in D_N. Another observation is that while highlighting misclassified points in D_N initially speeds up the learning-by-transfer process, it may eventually make the classifier vulnerable to outliers or overfit. Consequently, once there are enough samples in D_N, the algorithm should gradually reduce the weights of the misclassified samples in D_N. Weights for sample points in D_L^*, having as the current assumption $(x_i, y_i) \in D_L^*$ and $h^* : X \to Y$, are calculated as (the rules for updating the weights were determined heuristically):

- if $(x_i, y_i) \in D_L$ and $h^*(x_i) \neq y_i, w_i = (1 + log|D_N|)^{-1}$;
- $(x_i, y_i) \in D_N$ and $h^*(x_i) \neq y_i, w_i = 1 + \frac{\epsilon_N}{1-\epsilon_N} \times log[|D_N| \times (|D_L| - |D_N|)]$

with $(|D_N| < |D_L|)$;

- if $h^*(x_i) = y_i, w_i = 1$.

Where:

- $(|D_N| < |D_L|)$ is an indicator function,;
- D_L^*, that is $D_L^* = D_L \cup D_N$, is an augmented set of labeled data;
- x_i is a data point;
- w_i is the weight associated with data point x_i;
- ϵ_N is the error rate measured on the set D_N.

The mean and the covariance of the class-conditional distribution, $P_{D_L}(x|y)$ are estimated using weighted ML.

Xin Li et al.[56] proposed a new adaptive active learning strategy that combines an information density measure and a maximum uncertainty metric to identify crucial instances to label for image classification. This work demonstrated the effectiveness of the proposed approach by applying it to two key computer vision tasks:

- Object Recognition;
- Scene Recognition.

This combination procedure allows the proposed method to better integrate the strengths of the two measures in different stages and active learning scenarios. This method can effectively use the information contained in the unlabeled data to improve the performance of uncertainty sampling. Tests conducted on image classification problems have shown that the proposed approach can significantly reduce the training set required to learn a successful classifier. The overall active learning algorithm is reported in Algorithm 7:

Algorithm 7 Adaptive Active Learning Algorithm

Input: Labeled set L, Unlabeled set U, $B = [0.1, 0.2, ..., 1]$

while not enough instances are queried **do**

Training a probabilistic classifier θ_L on L

for $i \in U$ **do**

Compute $f(x_i)$ using 2.27

Compute $d(x_i)$ using 2.28

$h_\beta(x_i)$ with different $\beta \in B$ via 2.29

end for

Let $S = \emptyset$

for $\beta \in B$ **do**

Select an instance $x = argmax_{i \in U} h_\beta(x_i)$

Put x into set S, $S = S \cup x$

end for

Select instance x^* from S using 2.30

Remove x^* from the unlabeled set U

Query the true label y^* of x^*, and update L by adding $< x^*, y^* >$ into it

end while

For probabilistic classification models, the uncertainty measure is defined as the conditional entropy of the label variable Y given the candidate instance x_i:

$$f(x_i) = H(Y|x_i, \theta_L) = -\sum_{y \in Y} P(y|x_i, \theta_L) log P(y|x_i, \theta_L) \tag{2.27}$$

The information density definition can be written into the following form:

$$d(x_i) = \frac{1}{2} ln \left(\frac{\sigma_i^2}{\sigma_{i|U_i}^2} \right) \tag{2.28}$$

where U_i denotes the index set of unlabeled instances after removing i from U and θ is the conditional covariance. They indeed proposed to combine the two measures in a general product form of combination framework as below in order to pick the most informative instance for reducing the generalization

error of the classification model without the computationally expensive steps of retraining classification model for each candidate instance):

$$h_\beta = (x_i)^\beta d(x_i)^{1-\beta} \tag{2.29}$$

where:

- $0 \leq \beta \leq 1$ is a tradeoff controlling parameter over the two terms;
- $f(x_i)^\beta$ is an uncertainty term and it is a discriminative measure;
- $d(x_i)^{1-\beta}$ is the information density term and is computed in the input space, it has no direct connection with the target discriminative classification model.

The expected loss of the candidate instance x can be computed as a weighted sum of the prediction loss obtained using all possible labels y under the distribution $P(y|x, \theta_L)$. This is the equation used to do the instance selection from the set S:

$$x^* = \underset{x \in S}{argmin} \sum_{y \in Y} P(y|x, \theta_L) \left(\sum_{i \in U} (1 - P(\hat{y}_i | x_i, \theta_{L+\langle x,y \rangle})) \right) \tag{2.30}$$

where:

- $\theta_{L+\langle x,y \rangle}$ denotes the new model parameter after retraining on the augmented set $L + \langle x, y \rangle$;
- $\hat{y}_i$ is the predicted label for instance x_i.

This work proposed an adaptive active learning strategy for instance selection that combines a measure of information density with a measure of higher uncertainty adaptively. The proposed method can better incorporate the strengths of the two measures at different stages and circumstances of active learning through the adaptive combination procedure. This method can effectively exploit the information in the unlabeled data to improve the performance of uncertainty sampling. Researchers have shown that the proposed approach can reduce the training set required to learn a good classifier.

2.4.5 Social Networking

The last few decades have seen a huge increase in the popularity and consequently the use of social networks, which has led to the possibility of collecting a large amount of data. This information is used for many purposes, from commercial to research. Most studies on social networks make use of whole-network (sociocentric) or egocentric study designs. Whole-network studies examine relationships between people or agents that are perceived to be limited or closed for analytical purposes, even if in reality the network boundaries are porous and/or ambiguous. When doing whole network research, the goal is to analyze the structural patterns of interaction between individuals within the network and how these patterns explain specific health outcomes. When conducting a whole network analysis, the basic assumption is that individuals who make up a group or social network interact more than a randomly selected group of comparable size. Given the enormous amount of data, most often unlabeled, active learning in this area takes on enormous importance, as it can help retrieve information from social media data.

Mustafa Bilgic et al. [9] proposed an active learning algorithm for network data classification. Training instances are linked together to form a network, labels of connected nodes are associated, and the goal is to exploit these dependencies and label nodes appropriately. This challenge arises in several disciplines, including social and biological network research, document categorization and collective classification of network nodes. They have shown how an active learning system can take advantage of the network structure. To increase the accuracy of learning from fewer labeled instances, their approach efficiently exploits the links between instances and the interaction between the local and collective components of a classifier. Given a graph $G = (\nu, \varepsilon)$ where $\rho \subset \nu$ is the pool of unlabeled examples, a classification model, which will be used to train CC and $C0$, a batch size k and a budget B (the maximum number of iterations). The task is, within the constraints of B, to make a series of selections of k elements from ρ to be labeled by an oracle, so as to maximize the accuracy of CC on unseen data, after training it on the labeled exam-

ples acquired L. Their active learning algorithm for collective classification is presented below:

Algorithm 8 ALFNET: Active Learning for Networked Data

Input: $G = (\nu, \varepsilon)$: the network, $C0$: content-only learner, CC: collective learner, k: the batch size, B: the budget

Output: ζ: the training set

$\zeta \leftarrow \emptyset$

$C \leftarrow$ Cluster the nodes ν of the network F into at least k clusters

$C^k \leftarrow$ Pick k clusters from C

for $C_i \in C^k$ **do**

  $V_j \leftarrow$ Pick an items from C_i

  add V_j to ζ

end for

while $|\zeta| < B$ **do**

  Retrain $C0$ and CC

  for $C_i \in C$ **do**

    $score(C_i) \leftarrow Disagreement(CC, C0, C_i, \zeta)$

  end for

  $C^k \leftarrow$ Pick k clusters based on the $scores$

  for $C_i \in C^k$ **do**

    $V_j \leftarrow$ Pick an item from $C_i \cap \rho$

    Add V_j to ζ

    Remove V_j form ρ

  end for

end while

where:

- ε: edge set;
- ν: node set.

They conducted experiments on two real-world collective reference classification domains (Cora [63] and CiteSeer [28]) and showed that their system can produce accurate results even when only a small percentage of labeled data is available. Their ALFNET agorithm uses the structure of the network in various ways to choose samples for labeling in an informed way. They demonstrated how to adapt standard active learning concepts, such as disagreement and clustering, to a context in which network structure and attribute information is available. In addition, they demonstrated how to greatly improve the basic performance of their active learner by combining dimensionality reduction with semi-supervised learning. They conducted experimental tests and were able to show that the use of structure principle using ALFNET offers significant advantages over other previously implemented approaches.

Yuxiang Ren et al. [76] have studied the network alignment problem in order to merge online social networks. The goal of network alignment is to infer a set of anchor links that correspond to the shared entities in different information networks, which has become a critical step for the effective fusion of heterogeneous information networks. The researchers present a unique network alignment paradigm, ActiveIter (Active Iterative Alignment), to address three social network alignment problems that are extremely difficult to manage for a variety of reasons:

- Lack of training data;
- Network heterogeneity;
- One-to-one constraint.

Existing network alignment efforts often require a large number of training examples, but such a demand is difficult to meet in applications due to the high cost of human anchor link labeling. In contrast to prior homogenous network alignment research, information in online social networks is typically of heterogeneous categories, which is difficult to incorporate into model development. Furthermore, the one-to-one cardinality requirement on anchor connections

makes their inference process inextricably linked. ActiveIter is a model that specifies a set of inter-network meta diagrams for anchor link feature extraction, uses active learning for effective label querying, and employs greedy link selection for anchor link cardinality filtering. For all the potential anchor links in set H ($H = U^{(1)} \times U^{(2)}$, where $U^{(1)}$ and $U^{(2)}$ denote the user sets in $G^{(1)}$ and $G^{(2)}$ respectively, G^n is a social network), a set of features is extracted based on the meta diagrams Ψ. Formally, the feature vector extracted for anchor link $l \in H$ can be represented as vector $x_l \in \mathbb{R}^d$ (parameter d is the feature size). The label of link $l \in \zeta$ can be denoted as $y_l \in Y(Y = 0, +1)$, which denotes the existence of anchor link l between the networks. For the existing anchor links in set $\zeta+$, they will be assigned with $+1$ label; while the labels of anchor links in U are unknown. All the labeled anchor links in set $\zeta+$ can be represented as a tuple set $(x_l, y_l)_{l \in \zeta_+}$. Depending on whether the anchor link instances are linearly separable or not, the extracted anchor link feature vectors can be projected to different feature spaces with various kernel functions $g : \mathbb{R}^d \to \mathbb{R}^k$. For instance, given the feature vector $x_l \in \mathbb{R}^d$ of anchor link l, can be represented its projected feature vector as $g(x_l) \in \mathbb{R}^k$. Here the linear kernel function will be used for simplicity, and therefore they have $g(x_l) = x_l$ for all the links l. In the active network alignment model, the discriminative component can effectively differentiate the positive instances from the non-existing ones, which can be denoted as mapping $f(\cdot; \theta_f) : \mathbb{R} \to +1, 0$ parameterized by θ_f. They use a linear model to fit the link instances, and the discriminative model to be learned can be represented as $f(x_l; w) = w^T x_l + b$ where $\theta_f = [w, b]$. By adding a dummy feature 1 for all the anchor link feature vectors, they incorporate bias term b into the weight vector w and the parameter vector can be denoted as $\theta_f = w$ for simplicity. Based on this descriptions, they introduced the following discriminative loss function on the labeled set ζ_+:

$$L(f, \zeta_+; w) = \sum_{l \in \zeta_+} (f(x_l; w) - y_l)^2 = \sum_{l \in \zeta_+} (w^T x_l - y_l)^2 \tag{2.31}$$

They proposed a study of the alignment problem of heterogeneous networks

based on the active learning setting, technically known as the active alignment problem of heterogeneous networks (ANNA). ANNA allows models to request additional labels for unlabeled anchor links in the learning process, subject to a predefined request budget. To address this challenge, they propose an ActiveIter active learning model based on meta-diagrams. To represent heterogeneous features, meta-diagrams can be retrieved from the network. Initially, their experiments confirmed the usefulness of feature vectors based on meta-diagrams. They presented a query approach in the ActiveIter active learning model selection process to search for optimal unlabeled links. Extensive experiments were conducted on a dataset of real-world aligned networks (Foursquare and Twitter), and the experimental results demonstrated the usefulness of ActiveIter; in fact, it requires only a small training set to be built initially and can outperform other non-active models with a much smaller number of training instances.

Michael Hopwood et al. [35] proposed a study examining different tactics for selecting nodes to use as training data, demonstrating which strategy is better or worse and on what percentages of nodes in the network. Fundamental network characteristics provide an unsupervised method for determining the appropriate active learning sampling direction process. Although new methods for convolutional graph neural networks have introduced new tools for inferring community labels of nodes, they still require the provision of a labeled dataset, the obtaining of which can be costly, and measures to reduce the number of labels needed can both accelerate and reduce costs. Community membership labels are used for nodes in many online networking platforms, whether online social networks or academic citation networks, to generalize overlaps based on shared attributes or affiliations. Two example methods are considered in this work:

- **Descending**: training instances are chosen using descending sampling by gradually acquiring from the most important nodes to the least significant ones;

- **Ascending**: selects training samples gradually from the least significant nodes to the most important.

Three different criteria are used to evaluate a node's importance (centrality) for sampling orders:

- **Degree sampling**: a node is acquired for training based on their corresponding number of directly connected neighbours;
- **PageRank algorithm** [66]: derives a web page (node)'s rank by accumulate its incoming neighbors' ranks proportionally to their total number of outgoing connections;
- **Resulting ranking**: represents the relative importance of pages in the network.

In this work, they applied PageRank to rank all nodes in their graphs and then sampled them according to their rankings. Finally, the VoteRank [99] algorithm iteratively selects a set of important nodes called spreaders using the voting scores given by neighboring nodes. When a node is chosen as a spreader, it is excluded from the next voting round and the voting capabilities of its direct neighbors decrease in the same way. They applied VoteRank to all nodes in the graph and then sampled them according to their rank. Participants in networking platforms, such as social networks, continue to send more material to these platforms. It is a matter of efficiency whether a subset of nodes can be sampled to offer information about other nodes with unknown membership labels. The coefficient of variation of the degree of nodes is the best indicator of whether nodes should be sampled in terms of ascending or descending centrality, according to this study conducted on a set of networks comprising different sources of information. This finding can be intuitively interpreted as related to the sparseness of the network topology. Understanding the overall degree distribution of communities may indicate whether sampling should be done in an ascending or descending direction when trying to infer the labels of network users in an active learning paradigm.

2.4.6 ECG Signal Analysis

The electrocardiogram (ECG) is a graphic representation of the electrical activity of the heart; these signals are a great source of information about the rhythm and function of the heart. As a result, there has been a great deal of interest in recent years in creating approaches for automatic processing of ECG signals. Because of its practical advantages for detecting and monitoring cardiac disorders, automatic categorization of ECG signals has gained much attention in the biomedical engineering community. This is due to the fact that cardiovascular disease (CVD) continues to be a leading cause of death worldwide. Arrhythmia is one of the consequences of CVD. Arrhythmia is a general term for aberrant electrical activity of the heart, manifested by a slow, rapid, or irregular heartbeat. Some arrhythmias are not life-threatening, but others, such as ventricular fibrillation and tachycardia, can lead to stroke, heart failure, rapid death, and hemodynamic collapse during cardiac arrest.

Edoardo Pasolli et al. [68] introduced three active learning algorithms for categorizing ECG signals. These learning algorithms choose a few beat samples (which are manually labeled before being added to the training set) from a huge amount of unlabeled data after starting with a short and unsatisfactory training set. The entire approach is repeated until a final training set is created that is representative of the classification problem under consideration. The proposed methods are based on support vector machine classification and the following principles:

- **Margin Sampling**: based on SVM, this is an active learning algorithm for classification problems. In the case of a simple binary case with linearly separable classes, the SVs are the samples of the training set L that are closest to the hyperplane that specifies the decision boundary of the SVM classifier. Given the unlabeled learning set U, it is reasonable to assume that the samples closest to the decision boundary are the most interesting, as they have a higher probability of becoming SVs in the new learning set. Consequently, according to MS, the samples to be chosen

are those with the lowest absolute values of the discriminant function. In the case of nonlinearly separable classes, the same logic is used. The assumptions used in the binary instance can also be applied to a multiclass classification problem. The largest value among the discriminant functions provided by the binary T classifiers is used as the indicator for each sample. Samples with the lowest values of the indicator are then chosen, labeled by hand, and added to the training set:

Algorithm 9 MS method

Input: Consider the initial training set L, composed of n labeled samples of T different classes, the learning set U, composed of $m(m \gg n)$ unlabeled samples, N_s the number of samples to add at every iteration of the active learning process

while the predefined convergence condition is not satisfied **do**

Train a SVM classifier with the training set L, while estimating its free parameters by crossvalidation (CV)

for $u_j \in U$ with $j = 1, 2, ..., m$ **do**

Calculate the discriminant function values f_j for each binary SVM classifier

Count the number of votes of each class v_j

Identify the class $\omega_{MAX,j}$ with the maximum number of votes $v_{MAX,j}$

Let $f_{MIN,j}$ be the minimum absolute value of the discriminative function associated with $\omega_{MAX,j}$

end for

Select and label the N_s samples exhibiting the minimum values of $v_{MAX,j}$

Add the N_s selected samples to the training set L and remove them from U

end while

- **Posterior Probability**: another active learning technique is to estimate the posterior probability distribution of the classes $p_k = P(y = c_k|u)(k = 1, 2, ..., T)$. The posterior probability of each class is deter-

mined for each sample of the learning set U after the classifier has been trained using the training samples. In the case of binary classification, the optimal samples to choose are those with posterior probabilities close to 0.5, because they have the least uncertainty of choice. In multiclass problems, a more complex selection rule must be used. [77] provides a solution in which the samples with the greatest Kullback-Leibler divergence are chosen and added to the training set. This type of selection method can be applied to any classifier that produces a posterior probability estimate. SVM is not a probabilistic classification approach, so it does not directly produce probabilistic quantities as output. However, some strategies for extracting posterior probability estimates from the discriminant function values provided by SVM have been presented in the literature; the algorithm is discussed in detail below:

Algorithm 10 Posterior Probability method

Input: Consider the initial training set L, composed of n labeled samples of T different classes, the learning set U, composed of $m(m \gg n)$ unlabeled samples, N_s the number of samples to add at every iteration of the active learning process

while the predefined convergence condition is not satisfied **do**

Train a SVM classifier with the training set L, while estimating its free parameters by CV

Classify the learning set U and calculate for each sample $u_j(j = 1, 2, ..., m)$ the posterior probability of each class $p_k, j(k = 1, 2, ..., T)$

For each sample u_j, calculate the entropy $H(u_j)$ associated with the estimated posterior probabilities

Select and label the N_s samples characterized by the maximum values of entropy $H(u_j)$

Add the N_s selected samples to the training set L and remove them from U

end while

Here below there is the value of entropy $H(u_j)$:

$$H(U_j) = \sum_{k=1}^{T} -p_{k,j} log(p_{k,j}) \tag{2.32}$$

where $p_{k,j}$ is the posterior probability of ω_k given sample u_j;

- **Query by Committee**: this method, which uses the Query by Committee approach, selects learning samples to be added to the training set. Specifically, the samples with the greatest disagreement among different classifiers are chosen. This approach is proposed to solve the difficulties of multiclass active learning. Let s be a positive integer greater than one, which determines the feature sampling factor. We created s training subsets $L_1, L_2, ..., L_s$ from the initial training set L, where $L_g(g = 1, 2, ..., s)$ contains only the features $f(f = 1, 2, ..., d)$ that satisfy the constraint $(f1)module(s) = g - 1$. The number of samples in each subset is the same as the original number of samples, but with a factor s fewer features. Similarly, from the original learning set U, s learning subsets $U_1, U_2, ..., U_s$ are formed. At this point, each learning subset is considered individually and used to train an ensemble of c parallel SVM classifiers, with each classifier using a different kernel function to inject some diversity into the ensemble. As a result, a total of c parallel classifiers are applied. The learning samples are categorized after the training phase to estimate their labels. Specifically, estimates of cs are obtained for each sample. The algorithm is explained in detail below:

Algorithm 11 Query by Committee method

Input: Consider the initial training set L, composed of n labeled samples of T different classes, the learning set U, composed of m unlabeled samples, s the feature sampling factor

Construct the training subsets $L_g(g = 1, 2, ..., s)$

Construct the learning subsets $U_g(g = 1, 2, ..., s)$

Set the number of classifiers c to use in the ensemble for each training subset

Set N_s the number of samples to add at every iteration of the active learning process

while the predefined convergence condition is not satisfied **do**

Train $c \cdot s$ SVM classifier with the training subsets $L_g(g = 1, 2, ..., s)$, while estimating their free parameters by CV

Classify the learning subsets $U_g(g = 1, 2, ..., s)$ and calculate for each sample $u_j(j = 1, 2, ..., m)$ the number of occurrences of each class

For each sample u_j, calculate the entropy $H(u_j)$ associated with the occurrences of the estimated class labels

Select and label the N_s samples characterized by the maximum values of entropy $H(u_j)$

Add the N_s selected samples to the training set L and remove them from U

end while

Here below there is the value of entropy $H(u_j)$:

$$H(U_j) = \sum_{k=1}^{T} -rf_{k,j} log(rf_{k,j}) \tag{2.33}$$

where $rf_{k,j}$ is the relative frequency of class ω_k for sample u_j.

To demonstrate their performance, they conducted experimental research using both simulated data and real ECG signals from the MIT-BIH arrhythmia database. Overall, the results suggest that the presented strategies have

promising potential for selecting meaningful samples for the classification process to improve classification accuracy by reducing the number of labeled samples involved. Starting from a short, suboptimal training set, the techniques aim to select the most meaningful samples for the classification process from a large set of unlabeled data. Experimental results acquired on simulated and real ECG data suggest that the proposed approaches are capable of selecting relevant samples. In general, all proposed strategies outperform a purely random selection strategy in terms of accuracy and stability. In the comparison, the technique based on the MS principle seems to be the best because it quickly selects the most informative samples. Another interesting result is that active learning approaches can achieve slightly higher accuracies than the "full" classifier, demonstrating their usefulness in reducing the risks of mislabeling.

To classify ECG signals, Yufa Xia et al. [96] proposed an autonomous method based on convolutional neural network and active learning. In active learning, breaking-ties (BT) and modified BT algorithms are used to increase the performance of the model. Using the Association for the Advancement of Medical Instrumentation standard, they classified ECG signals into five types of heartbeats, as recommended by AAMI standards:

- **normal** (N): a normal beat or a bundle branch block beat;
- **ventricular** (V): a ventricular ectopic beat;
- **supraventricular** (S): a supraventricular ectopic beat;
- **fusion of normal and ventricular** (F): a fusion of a ventricular ectopic beat and a normal beat;
- **unknown heartbeats** (Q): a paced beat, a fusion of a paced and a 140 normal beat or a beat cannot be classified.

They acquire ECG signals by the wearable ECG device, they preprocessed them before the use; this preprocessing includes:

- de-noising;

- filtering;
- R-peak detection.

After that, ECG morphology and RR intervals are used as the ECG vectors, then these are divided into training set and test set. Then ECG training set and test set are sent to the classifier model. The training set consists of two parts:

- the initial training set;
- the active learning training set.

The classifier model contains two phases:

- learns appropriate feature representation of ECG signals using $1-D$ CNN and classifies the ECG signals by a softmax regression in which the active learning training set is empty;
- fine-tunes the classifier model by active learning.

Firstly, they train $1-D$ CNN using initial training set, this is utilized to extract the ECG feature values in an end-to-end way, and softmax regression is added in the end layer of CNN to identify ECG heartbeat types. The entire CNN is fine-tuned by minimizing the following cost function:

$$J(\theta_{DNN}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} 1(Y_i = k) log\left(\frac{exp(h_{\theta_{CNN}}(x_i))}{\sum_{k=1}^{K} ext(h_{\theta_{CNN}}(x_i))}\right) + \frac{\gamma}{2}\left(||W_{softmax}||_f^2 + \sum_{l=1}^{H}||W_l||_f^2\right) \quad (2.34)$$

where:

- the first term represents the cross entropy loss of the softmax layer;
- the second term denotes the weight decay penalty;
- $h_{\theta_{CNN}}(x_i)$ is the output of last fully connected layer in the CNN for an input x_i.

After training the classifier model, it is tested by the initial test set. The active learning phase is used to improve the performance of the classifier. During the active learning phase, the test set samples are divided into informative set and new test set. An expert labels the information set samples. The labeled information set is also included in the active learning training set. The new training set consists of the active learning training set and the initial training set. The classifier model is then retrained using the new training data. The new test set is used to test the classifier model. The active learning phase is a continuous process. To select the most informative ECG beats from the test set, the following algorithms are used:

- **BT algorithm** [59]: the target of this algorithm is to improve the value of $P(a) - P(b)$ amounts to breaking the tie between $P(a)$ and $P(b)$, here below in detail the algorithm:

Algorithm 12 BT algorithm

Input: LS initial training set, US an unclassified set of images

while stop rule **do**

build SVM using LS

$PA \leftarrow []$

$PB \leftarrow []$

for $u_j \in US$ **do**

Compute the PO (probabilistic outputs) of the classification results

Compute the a the class with highest probability

Compute the b the second class with highest probability

$PA \leftarrow P(a) + PA$

$PB \leftarrow P(b) + PB$

end for

for $image_j \in US$ **do**

if $image_j$ has the smallest difference in probabilities $(P(a) - P(b))$ (for the two highest probability classes) **then**

Remove the $image(s)$ from the US

Obtain the correct label from human experts and add the labeled $image(s)$ to the current training set

end if

end for

end while

- **MBT algorithm** [55]: this is a variation the the previous algorithm, in detail:

Algorithm 13 MBT algorithm

while stop rule **do**

$s =$ next class

select S_{U_s}

$\hat{x}_i^{MBT} = \underset{x_i, i \in S_{U_s}, k \in \zeta \setminus s}{argmax} \; p(y_i) = k | x_i, \hat{\omega})$

end while

where:

ζ: is a set of K labels;

S: is the set of index for the n pixels of a hyperspectral images;

ζ^n: is a set of images;

$y = (y_1, ..., y_n) \in \zeta^n$: be an image of labels;

K: is a set of labels;

k: is a label;

$x = (x_1, ..., x_n) \in \mathbb{R}^{d \times n}$: is an image of d-dimensional feature vectors;

ω: is the logistic regressors.

Then the most informative samples are selected and form the information set. The performance of this method performed well in the MIT-BIH arrhythmia database and also in the WDDB database. This implies that the presented ECG classification method can effectively classify ECG signals.

The main obstacles to automatic detection of arrhythmias by electrocardiogram (ECG) are the large variations among individuals and the high cost of labeling clinical ECG recordings. GuijinWang et al. [95] proposed Global Recurrent Neural Network (GRNN) as a comprehensive and updatable classification scheme to build a system with an autonomous feature learning scheme and an effective optimization mechanism. This study illustrates the feasibility of a global and updatable ECG beat classification system in practical applications. Based on morphological and temporal information, a recurrent neural network (RNN) is used to study the underlying properties of ECG beats. Active learning is used to identify the most informative beats and include them in the training set to improve system performance when new samples are collected. As the training set develops, the system is updated. This technique has three innovations:

- GRNN has an high capacity and fitting ability, it can categorize samples from several patients with a single model;

- when training and test samples are from different databases, the GRNN increases generalization performance, this can be characterized as the optimization mechanism finding the most informative samples to use as training data;
- RNN learns the fundamental differences between samples from different classes automatically.

This technique is known as GRNN since a single RNN is created for all patients. RNN is supplied a morphological vector and a Premature-or-Escape-Flag (PEF) to automatically learn underlying features. The PEF is offered as an innovative representation of temporal information to provide a likelihood of whether the sample is a premature-or-escape beat. A sample is defined as the morphological and PEF vectors from a single beat. The training dataset is divided into two parts:

- **initial training set**: this is applied a density-based clustering algorithm to group the samples in an unsupervised way, to generate it, samples are drawn at random from each group;
- **active training set**: this is initially empty.

The initial training set and the empty active training set are used to train a model. When a new record is introduced into the system, all samples are added to the test set in the first iteration. Active learning is used in each iteration to select the most informative samples in the current test set. In the current iteration, the selected samples are added to the active training set and the model is optimized on the expanded training set. The other samples in the current test set will be used to build the test set in the next iteration. The iteration for each record ends when the stop condition is met. All beats were classified into five categories, as recommended by the AAMI standards as in previous work. However, class Q was discarded because it is only minimally represented in the available databases. GRNN achieves excellent performance by using a single model for all patients, which increases the usability of the system. The

proposed optimization mechanism increases the performance and stability of the system as a whole. The visualization of features trained by GRNN shows that the model has learned the main underlying features from various classes. Further testing on LTSTDB-I [39] (This is a database containing 86 extended ECG recordings of 80 human participants chosen to show a variety of occurrences of ST-segment alterations, such as ST episodes of ischemia, non-ischemic axis-related ST episodes, episodes of delayed ST-level drift, and episodes comprising a combination of these phenomena. The database was designed to contribute to the creation and testing of algorithms that can distinguish between ischemic and nonischemic ST events, as well as to basic research on the mechanics and dynamics of myocardial ischemia, confirming the high performance of the proposed system. This research shows how a globally upgradable ECG beat classification system can be used in practical applications. Labeling efforts on clinical data could be widely used to iteratively improve the performance of the system.

2.5 New frontiers

This section explains a very interesting field of active learning, how it can be combined with other fields of AI (artificial intelligence) or with computer science more generally. Remarkably, it can give a huge improvement to the field in which it is applied. The following are some areas, with related work, in which it is applied.

2.5.1 NLP

One of the areas where active learning is most widely used is natural language processing (NLP), because many applications in this area require a lot of labeled data and the cost of this data is very high. In fact, there are few NLP datasets available for free, so the use of active learning can greatly reduce both the amount of labeled data needed and the experts required to label it accurately.

Ein-Dor et al. [23] proposed a large-scale empirical study on active learning techniques for BERT-based classification [18], covering a wide range of AL strategies and datasets. They focused on practical circumstances of binary text classification where annotation budget is limited and data are often partial. Their results show that AL can improve the performance of BERT, particularly in the more realistic scenario where the initial collection of labeled examples is generated through keyword-based queries, resulting in a biased sample of the minority class. First, BERT (Bidirectional Encoder Representations from Transformers) is a linguistic representation model. Unlike current linguistic representation models, BERT intends to pre-train deep bidirectional representations from unlabeled text, conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be refined with only one more output layer to produce models for a wide range of tasks, such as question answering and linguistic inference, without requiring significant changes to the task-specific architecture. BERT is conceptually and empirically simple. It has achieved tremendous results in eleven natural language processing tasks, including:

- GLUE score to 80.5% (7.7% point absolute improvement);
- MultiNLI accuracy to 86.7% (4.6% absolute improvement);
- SQuAD v1.1 question answering Test F1 to 93.2% (1.5 point absolute improvement);
- SQuAD v2.0 Test F1 to 83.1% (5.1 point absolute improvement).

This effort yielded 2,520 fine-tuning experiments (14 dataset-scenario combinations 5 initial seeds (1 basic model + (7 selection techniques 5 repetitions)). $BERT_{BASE}$ (110M parameters) was trained for 5 epochs in each fine-tuning run, with a learning rate of 5×10^{-5}, and the best model kept based on its performance on the dev set. In reality, dev sets may not be available, especially if the annotation budget is low. They found that ignoring dev data and establishing a constant number of epochs yields qualitatively identical, though

noisier, results. The experiments revealed that increasing the batch size has a significant effect on the stability of BERT results. However, due to the memory limitations of the GPU, increasing the batch size reduces the maximum sequence length. They found that limiting the batch size to 50 and the maximum sequence length to 100 tokens (after WordPiece tokenization) gives the best results. Their results show the potential of AL in addition to BERT, particularly in the latter case. In particular, a seed of training data generated by a simple query should capture only a few, perhaps obvious, features of the class to be evaluated. Research shows that the baseline BERT model has poor prediction performance, mainly due to low recall values. Although the random AL baseline is limited in its ability to help BERT emerge from this inadequate initial model, AL techniques prove to be extremely beneficial. Using the AL pipeline, BERT significantly improves its recall, generalizing beyond the limited data set to which it was initially exposed.

In this work, Liu et al. [57] proposed an Active Learning with DivErse Interpretations (ALDEN) approach, motivated by the piece-wise linear interpretability of DNNs. ALDEN detects regions of linearly separable samples using local interpretations of DNNs. It then selects samples based on the variety of local interpretations and queries their labels. To solve the challenge of text categorization, they chose the term with the most interpretations to symbolize the whole sentence. As for active learning for text categorization, for a word in a specific sample used in a deep text classifier, its local interpretation can be calculated using the word $e_{i,j}$ embedding of the word $x_{i,j}$ as follows:

$$I_{i,j} = \frac{\partial \hat{y}_i}{\partial e_{i,j}} e_{i,j}^T + b \tag{2.35}$$

where $\hat{y}_i$ is the prediction of sample x_i. Consider that because to the complicated nonlinear feature interactions predicted by deep models, local interpretations (i.e. the contribution to model predictions) of the same word may differ across samples. It is now important to select samples with distinct local interpretations to maximize linear classifiers in different linearly separable regions. Meanwhile, since the different interpretations correspond to distinct

choice regions in the deep model, the samples with the most diverse interpretations can provide the most in-depth information for learning different decision logics in the deep model. Since the different samples contain varying numbers of words, it is necessary to begin the work of text classification by analyzing the local interpretations of the words in each sample. To this end, one looks for the most similar embedding of the word present in the samples labeled as close, which is expressed as:

$$N(\zeta, x_{i,j}) = \underset{x_m \in \zeta 1 \leq w \leq |x_m|}{argmin} ||e_{i,j} - e_{m,w}|| \tag{2.36}$$

The interpretation diversity of a word $x_{i,j}$ compared to the same word appeared in labeled samples can be calculated as follows:

$$D(\zeta, x_{i,j}) = \underset{x_m \in \zeta 1 \leq w \leq |x_m|}{min} ||I_{i,j} - I_{m,w}|| \tag{2.37}$$

Given that sentences contain varying quantities of words, it is impossible to use the local interpretations of all words in a sample directly. As a result, they implemented a pooling technique for active learning. Considering that in EGL-Word [100], the word with the highest EGL represents the entire phrase. In accordance with EGL-Word, the term is employed with the most diversified interpretation to reflect the entire sample for active learning. Formally, for a sample $x_i \in U$:

$$D(\zeta, x_{i,j}) = \underset{1 \leq j \leq |x_i|}{max} D(\zeta, x_{i,j}) \tag{2.38}$$

Now can be chosen the unlabeled sample with the most diversified interpretation for labeling based on the metric derived using Equation 2.38:

$$x = \underset{x_i \in U}{argmax} D(\zeta, x_i) \tag{2.39}$$

While there is a budget of K in each iteration, the above process is repeated for K times to select and label K samples. The below algorithm 14 summarizes the training procedure of the ALDEN approach:

Algorithm 14 The ALDEN approach

Input: Labeled set ζ, Unlabeled set U, budget K in each iteration and the number od iterations N

for $n = 1, 2, ..., N$ **do**

for $x_i \in \zeta \cup U$ **do**

Calculate prediction $\hat{y} = f(x_i|\theta_{n-1})$

for $1 \leq j \leq |x_i|$ **do**

Calculate the local interpretation $I_{i,j}$ using 2.35

end for

end for

for $k = 1, 2, ..., K$ **do**

for $x_i \in U$ **do**

for $1 \leq j \leq |x_i|$ **do**

Find the neighbor $N(\zeta, x_{i,j})$ of word $x_{i,j}$ using 2.36

Compute the diversity $D(\zeta, x_{i,j})$ of local interpretations of $x_{i,j}$ using 2.37

end for

Compute the diversity $D(\zeta, x_i)$ of local interpretations of x_i using 2.37

end for

Select and label the sample x having the most diverse local interpretations

$\zeta = \zeta \cup \{x\}$

$U = U \setminus x$

end for

Train a new model $f(x|\theta_n)$ in ζ

end for

return The final model $f(x|\theta_n)$

They applied linearly separable sample sections to the challenge of deep active learning, inspired by the local interpretability of DNNs. They proposed the

ALDEN approach for text classification, which selects and labels samples based on multiple interpretations of an unlabeled sample. They measured sample variety by using multiple interpretations of words in a sample. The ALDEN technique, according to experimental results, achieves excellent results on two datasets for text classification, using CNN and BiLSTM as classifiers.

Another interesting work in this field was proposed by Qian et al. [71] who proposed a bilingual active learning paradigm for relation classification, in which unlabeled instances are first jointly selected based on their prediction uncertainty scores in two languages and then manually labeled by an oracle. Instead of using a parallel corpus, instances in one language are translated into instances in the other language, and all instances in both languages are then entered into a bilingual active learning engine as mock parallel corpora. The results of the experiments conducted using the Chinese and English *ACE DRC 2005* corpora reveal that bilingual active learning beats monolingual active learning in relation classification. The idea behind the BAL paradigm is that while unlabeled uncertain instances in one language are informative for the learner of that language, unlabeled uncertain instances in both languages are informative for learners of both languages, potentially improving classification performance for both languages more than their individual active learners. This concept is expressed in the BAL algorithm 15, where n represents the batch size, i.e., the number of instances selected, labeled, and incremented at each iteration.

Algorithm 15 Bilingual Active Learning

Input:

- L_c and U_c labeled and unlabeled instances in Chinese, L_{et} and U_{et} their respective translation counterparts in English

- L_e and U_e labeled and unlabeled instances in English, L_{ct} and U_{ct} their respective translation counterparts in Chinese

- n, batch size

Initialize:

Add instances L_{ct} to L_c

Add instances L_{et} to L_e

while certain number of instances are labeled or performance is reached **do**

Learn the Chinese classifier SVM_c form L_c

Use SVM_c to classify instances in U_c and U_{ct}

Learn the Chinese classifier SVM_e form L_e

Use SVM_e to classify instances in U_e and U_{et}

Choose the n least confidently jointly predicted instances pairs $\{E_c|E_{et}\}$ from $\{U_c|U_{et}\}$, and have them labeled by an oracle

Choose the n least confidently jointly predicted instances pairs $\{E_e|E_{ct}\}$ from $\{U_e|U_{ct}\}$, and have them labeled by an oracle

Remove E_c from U_c and E_e from U_e

Add instances $E_c \cup E_{ct}$ to L_c with their manual labels

Add instances $E_e \cup E_{et}$ to L_e with their manual labels

end while

return SVM_c and SVM_e

The important element of this approach is the selection and labeling of unlabeled instances from U_c and U_e. When estimating the prediction uncertainty for an unlabeled instance in U_c, is examined not only its H_c uncertainty measure predicted by SVM_c, but also the H_{et} uncertainty measure predicted by SVM_e for its translation counterpart in U_{et}. In general, there are three ways to compute the averages of these two measures when they are considered together:

- arithmetic mean;
- geometric mean;
- harmonic mean.

Preliminary experiments suggest that there is no clear winner among these three averages, so only the geometric mean defined as follows is used:

$$H_g = \sqrt{H_c \times H_{et}} \tag{2.40}$$

Since they used the LC measure as the uncertainty score, when an instance in U_c cannot identify its translation equivalent in U_{et} due to a translation error or entity alignment failure, H_{et} is set to 1, the maximum. Since the larger H, the safer the prediction, the less likely the instance choice, unlabeled examples without translation counterparts are avoided. Accordingly, this study proposes a bilingual active learning paradigm for classifying Chinese-English relations. Given a small number of labeled connection instances and a large number of unlabeled relation instances in both languages, parallel pseudo-corpora were created that translated both labeled and unlabeled examples in one language to the other. After entity alignment, these labeled and unlabeled instances were loaded into a bilingual active learning engine. Experiments conducted on the *ACE DRC 2005* corpora in Chinese and English with the goal of categorizing relations reveal that bilingual active learning outperforms monolingual active learning in both Chinese and English. Furthermore, it is shown that BAL spanning two languages can compete with monolingual AL when the annotation scale is limited, even if the total number of labeled instances remains constant.

2.5.2 GAN

Work is also being done on implementing generative adversarial networks (GANs) [29] in the context of active learning. A GAN is a class of machine learning methods in which two neural networks are trained competitively in the

context of a zero-sum game. This type of structure allows the neural network to learn to generate new data with the same distribution as those used in the training phase. For example, it is possible to obtain a neural network that can generate hyperrealistic human faces, as demonstrated in 2018 by NVIDIA [46]. GANs consist of two components:

- G: the generator, that is a generative model, its purpose of the generative model is to produce new data;
- D: the discriminator, that is a discriminative model, its purpose is to learn how to distinguish real data from artificially generated ones.

Specifically, given a latent space z, having an a priori distribution $P_z(z)$, the generator represents a differentiable function $G(z;\theta_g)$ that produces the new data according to some distribution p_g, where θ_g are the parameters of the generative model. The discriminator represents a differentiable function $D(x;\theta_d)$, where θ_d are the parameters of the discriminative model, which gives the probability that x comes from the training data distribution p_{data}. The goal is to obtain a generator that is a good estimator of p_{data}. When this happens, the discriminator is "fooled" and can no longer distinguish p_{data} samples from p_g samples. The way to achieve this is through competitive training. The discriminative network is trained to maximize the probability of correctly classifying the training data samples and the generated samples. At the same time, the generative network is trained by minimizing:

$$log(1 - D(G(z)) \tag{2.41}$$

and maximizing the probability of the discriminator to consider the samples produced by the generative network, that is $x \sim p_g$, as coming from p_{data}. Learning therefore consists in optimizing a two-player minimax game (G and D):

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}(x)}[log\ D(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(z)))]] \tag{2.42}$$

which has a global optimum for $p_g = p_{data}$. The two networks are alternately trained by error back-propagation, keeping the parameters of the generative model unchanged during discriminator training and, vice versa, keeping the parameters of the discriminative network unchanged during generator training.

Jia-Jie Zhu et al. [102] proposed a new active learning approach by query synthesis using GAN; unlike normal active learning, the resulting algorithm adaptively synthesizes training instances for querying in order to increase learning speed. They generate queries according to the uncertainty principle, but their idea can work with other active learning principles; in some settings, this work outperforms traditional pool-based approaches. Their active learning approach, called Generative Adversarial Active Learning (GAAL), combines query synthesis with the uncertainty sampling principle. Using the uncertainty sampling principle, the intuition of the approach is to generate instances about which the current learner is uncertain. One option for the loss function is based on the uncertainty sampling principle. The distance (proxy) to the decision boundary in the case of a classifier with the decision function $f(x) = W\phi(x) + b$ is $|W\phi(x) + b|$. Formulate the synthesis of active learning as the following optimization problem, given a trained generating function G:

$$\min_{z} ||W^T \phi(G(z)) + b|| \tag{2.43}$$

where:

- z is the latent variable;
- G is obtained by the GAN algorithm.

By minimizing the loss, the generated samples will move closer to the decision boundary. With respect to pool-based active learning, the intuition is that it can generate more informative instances than those in the existing pool. After being labeled, the (s) solution of this optimization problem, $G(z)$, is used as the new training data for the next iteration. Its algorithm 16 is presented below:

Algorithm 16 GAAL

Train generator G on all unlabeled data by solving 2.42

Initialize labeled training dataset S by randomly picking a small fraction of the data to label

while labeling budget is reached **do**

Solve optimization problem 2.43 according to the current learner by descending the gradient 2.44

Use the solution $z_1, z_2, ...$ and G to generate instances for querying

Label $G(z_1), G(z_2), ...$ by human oracles

Add labeled data to the training dataset S and re-train the learner, update W, b

end while

with the following descending the gradient:

$$\bigtriangledown_z ||W^T \phi(G(z)) + b|| \tag{2.44}$$

A state-of-the-art classifier, such as convolutional neural networks, can be used. For this purpose, the feature map in the equation 2.43 can be replaced with a feed-forward function of a convolutional neural network. In this case, the linear SVM will be the output layer of the network. In step 4 of the algorithm 16, a different active learning criterion can be used. The authors emphasize that their contribution is a general framework rather than a specific criterion. The optimization problem is nonconvex, with many potential local minima. The goal is usually to find good local minima rather than the global minimum. To solve this problem, they used a gradient descent algorithm with momentum. They also restart the gradient descent to find other solutions regularly. Back-propagation is used to compute the gradient of D and G. This technique has shown promising results; in fact, it always succeeds in synthesizing points close to the boundary. This can lead to the generation of similar samples, reducing efficiency.

Nielsen et al. [65] proposed a GAN data augmentation method for image

classification that uses the prediction uncertainty of the classification network to determine the best GAN samples to add to the training set. To evaluate the uncertainty of the samples, the acquisition function framework, originally developed for active learning, was used. When training deep neural networks for supervised learning tasks, data augmentation is often used to increase the effective size of the training set. This technique is particularly useful when the size of the training set is small. For the experimental work, three different GAN models with increasing capacity were used:

- **Small-DCGAN** and **Large-DCGAN**: [72] that stands for Deep Convolutional GANs is set of constraints on the architectural topology of Convolutional GANs that make them stable to train in most settings;
- **PGGAN**: [45] this is a brand-new training method for generative adversarial networks. The key idea is to gradually increase the resolution of both the generator and the discriminator. Starting with a low resolution, they add new layers that model increasingly fine details as training progresses. This both accelerates and stabilizes the training, allowing for high-quality images to be produced.

The convolutional neural network (CNN) architecture of the classifier model was a simple five-layer network. They started the N iteration step with the currently trained classifier network and the training set. To complete the iteration step of a training cycle, samples from the data source are used to compute posterior estimates of the classifier network using the dropout MC [26] this is a theoretical framework that projects dropout training in deep neural networks (NNs) as an approximate Bayesian inference in deep Gaussian processes; as a direct result, a tool for modeling uncertainty with dropout NNs is provided by extracting information from previously discarded models; the problem of uncertainty representation in deep learning is mitigated without sacrificing computational complexity or test accuracy. Next, an acquisition function is used to process the posterior estimates and assign a score to each image sample. The samples with the highest scores are added to the training

set for the $N+1$ iteration step and used to train the resulting classifier. This process is repeated until the desired convergence or number of iterations is reached. The network of classifiers is initialized with random parameter values in the base case where $N = 0$. Two acquisition functions were used:

- Random sampling simply involves selecting random images from the data source to become part of the training set for the next iteration;
- BALD acquisition involves computing the score described in the following equation:

$$U(x) = H\left[\frac{1}{N}\sum_{n=1}^{N} P[(y|x,\omega_n)\right] - \frac{1}{N}\sum_{n=1}^{N} H[(y|x,\omega_n)] \tag{2.45}$$

where:

N is the number of MC samples;

ω_n are the parameters of the network sampled for the nth MC dropout sample;

x is an input data;

y is an output data;

P is the discriminative distribution.

Random sampling consists of simply selecting random images from the data source to be included in the training set for the next iteration. After the scores are calculated, the images with the highest scores are sampled and added to the training set for the next iteration. The algorithm is shown below:

Algorithm 17 GAN data augmentation using Bayesian acquisition

Input: Labelled dataset D_{raw}, Number of samples to append each iteration N_{iter}

Train GAN using D_{raw}

Generate sample set $D_{generated}$ from GAN

From augmented dataset $D_{augmented} \leftarrow \{D_{raw}, D_{generated}\}$

Set $D_{training} \leftarrow \{\}$

Initialize CNN patameters

for number of CNN training iterations **do**

  for $x_n \in D_{augmented}$ **do**

    Compute predicted CNN probabilities ϕ using MC-Dropout

    Evaluate acquisition function $U(\phi)$

  end for

  Append the samples x_n with the N_{iter} largest acquisition scores to $D_{training}$

  Retrain CNN using $D_{training}$

  Evaluate CNN balanced accuracy using test set

end for

Training a classification network with an integrated dataset from synthetic samples of GANs can improve the overall performance of the classifier. In addition, the acquisition function sampling mechanism has been shown to improve classifier performance, particularly for GANs with lower capacity. The results presented in this paper demonstrate that enhancing GANs with Bayesian uncertainty analysis is beneficial for image classification.

Mayer et al. [61] have proposed ASAL, a GAN-based AL method applicable to multiclass problems, as a new GAN-based active learning method that generates high-entropy samples. Instead of annotating synthetic samples directly, ASAL searches the pool for similar samples and includes them in training. As a result, the quality of new samples is high and annotations are reliable. The advantage of ASAL over traditional uncertainty sampling is its low execution complexity, which is sub-linear compared to linear. Using GANs

for fast pool-based active learning, this technique generates samples and retrieves similar real-world samples from the pool, which are labeled and added to the training set. Such a GAN can approximate the distribution of the pool data. The discriminator D ensures that the samples generated by the generator G are identical to the real-world samples. When the generator reaches convergence, it generates the function $G : \mathbb{R}^n \rightarrow X$, which maps the latent space variable $z \sim N(0_n, I_n)$ to the domain of the image X. The optimization problem describing the sample generation is as follows:

$$\begin{aligned} &maximize \;\; (H \circ h_{\theta^k})(x) \\ &subject \;\; to \;\; x = G(z) \end{aligned} \tag{2.46}$$

where:

- $H(q) := -\sum_{i=1}^{m} P(c = i|q) log[P(c = i|q)]$ with m the number of the categories;
- h is the classifier;
- θ^k are the weights at active learning cycle k;
- x is a sample.

Removing the constraint $x \in P$ by including the generator simplifies the problem, but the solution is altered. New samples are no longer selected from the pool but are created artificially. They solved the optimization problem in two steps:

- minimizing the objective with respect to z using the chain rule and gradient descent;
- recovering a synthetic sample x from z using G.

Since it is independent of the pool size, the solution of the problem 2.46 has a constant execution complexity of $O(1)$. Since it requires scanning each sample in the pool P, traditional uncertainty sampling has a linear complexity of $O(n)$ (where $n = |P$— is the pool size). The algorithm is explained in detail below:

Algorithm 18 ASAL

Input: Initialize the set X, Y by adding random pool samples to X^0 and their labels to Y^0. Train the generator G and the feature extractor F. Precompute the PCA, π and the set $S = F_{PCA}(x)|x \in X$.

while Labelling budget is exhausted **do**

Train classifier h_{θ^k} to minimize empirical risk $R(h_{\theta^k}) = \frac{1}{|x^k|} \sum_{(x,y)\in(X^k,Y^k)} l(h_{\theta^k}(x), y)$

Generate synthetic samples $\hat{x}$ with high entropy by solving 2.46

Compute the feature representations $\hat{x}$ of the generated samples: $\hat{f} = F_{PCA}(\hat{x})$

Retrieve real samples x that match $\hat{x}$ $x = p_i \in P|i = \underset{f\in S}{argmin}\, d(f, \hat{f})$

Annotate the samples x with labels y

Update the sets $X^{k+1} = X^k \cup x, Y^{k+1} = Y^k \cup y$

end while

Return: Trained Classifier h_{θ_k}

Failure case analysis revealed that the success of ASAL depends on the structure and size of the dataset, as well as the quality of the images and matches generated. ASAL is most effective when applied to large datasets. In this case, the sub-linear execution time quickly compensates for any preprocessing. ASAL is appropriate for interactive AL, where preprocessing is acceptable but sampling times are short.

2.5.3 Reinforcement Learning

Reinforcement learning (RL) is a machine learning technique that aims to create autonomous agents capable of choosing actions to achieve certain goals through interaction with their environment. It is one of the three main paradigms of machine learning, along with supervised and unsupervised learning. Unlike the other two, this paradigm deals with sequential decision problems, in which the action to be taken depends on the current state of the system and determines its future. The quality of an action is given by a numerical "re-

ward" value, inspired by the concept of reinforcement, which aims to encourage the agent's correct behavior. This type of learning is usually modeled through Markov decision processes (this is a random process in which the transition probability that determines the transition to a state of the system depends only on the immediately preceding state of the system "Markov property" and not on how this state was reached) and can be achieved with different types of algorithms, which can be classified according to the use of a model describing the environment, the methods of collecting experience, the type of representation. of the states of the system and the actions to be performed (discrete or continuous). Inputs to the system can come from a wide variety of sensors. For example, in the case of a robot that needs to learn how to move within a path, inputs can come from proximity sensors that must then be remapped into appropriate states. The state value function is one that associates a value relative to the degree of goodness of the situation with each state identified by the system and determined from the inputs. It is generally expressed in the following form:

$$V : S \to \mathbb{R} \tag{2.47}$$

The action value function is one that associates a value relative to the degree of goodness of combination with each pair consisting of state and action. It is generally expressed in the form:

$$Q : S \times A \to \mathbb{R} \tag{2.48}$$

Different reinforcement functions can be used to change the state value function and different policies to determine rewards and penalties, and all reinforcement functions can be reduced to the following basic formula:

$$v_{t+1} = (1 - \alpha)v_t(s) + \alpha\triangle_{t+1} \quad where \ 0 < \alpha \leq 1 \tag{2.49}$$

$\triangle_{t+1}$ is the reward or penalty assigned by the function to the action. This function changes the state value function from the next time it is invoked and

based on the evaluation of the current action made by the reward or penalty policy. The output represents one of the actions that the system can take. The decision is made to maximize the value of the action value function and depends entirely on the reinforcement distributed in the previous instants.

With the growing interest in deep reinforcement learning [42], researchers are trying to reformulate active learning as a reinforcement learning problem Fang et al. [24] have realized that the performance of heuristics (used to select a small subset of data for annotation) varies across datasets; to address these shortcomings, they introduce a new formulation, reformulating active learning as a reinforcement learning problem and explicitly learning a data selection policy, where the policy takes the role of the active learning heuristic; this method allows the selection policy learned through simulation on one language to be transferred to other languages. Below is the algorithm:

Algorithm 19 Learn an active learning policy

Input: data D, budget B

for $episode = 1, 2, ..., N$ **do**

$D_l \leftarrow \{\}$ and shuffle D

$\phi \leftarrow$ Random

for $i \in \{0, 1, 2, ..., |D|\}$ **do**

Construct s_i using x_i

The agent makes a decision according to $a_i = argmaxQ^{\pi}(s_i, a)$

if $a_i = 1$ **then**

Obtain the annotation y_i

$D_l \leftarrow D_l + (x_i, y_i)$

Update model ϕ based on D_l

end if

Receive r_i using held-out set

if $|D_l| = B$ **then**

Store $(s_i, a_i, r_i, Terminate)$ in M

Break

end if

Construct the new state s_{i+1}

Store transition (s_i, a_i, r_i, s_{i+1}) in M

Sample random minibatch of transitions $(s_j, a_j, r_j, s_{j+1)}$ from M

Perform gradient descent step on $L(\theta)$ with 2.50

Update policy π with θ

end for

end for

Return: the latest policy π

where:

- N: is the quantity of episodes;
- ϕ: is the learned model;

- s: is a state;
- x: is a sentence;
- y: is an output;
- a: is an action;
- π: is a policy;
- r is a reward;
- M: is the replay memory;
- L: is the loss;
- θ: are some parameters.

To update the parameters, they used a deep neural network to calculate the expected Q value (quality values). The Q-function (a function to compute quality) is implemented with a single hidden layer neural network, which takes the state representation as input and produces two scalar values that correspond to the $Q(s, a)$ values for $a \in \{0, 1\}$. The parameters of the DQN (Deep Q-Network) are learned using stochastic gradient descent, based on a regression objective to match the Q values predicted by the DQN and the Q values expected from the Bellman equation, $r_i + \gamma \, max_a Q(s_{i+1}, a; \theta)$. They used a replay memory of experience M to store each transition (s, a, r, s_0) used in an episode, after which they sampled a mini-batch of transitions from memory and then minimized the following loss function:

$$L(\theta) = \mathbb{E}_{s,a,r,s'}[(y_i(r, s') - Q(s, a; \theta))^2] \tag{2.50}$$

They formalized active learning using a Markov decision framework, in which active learning is defined as a series of binary annotation decisions applied to a data stream. On this basis, they created an active learning algorithm as a deep reinforcement learning policy. They demonstrated how these learned

active learning policies can be transferred between languages, providing consistent and significant improvements over basic methods, including traditional uncertainty sampling.

Human behavior and experience are inherently multi-modal, with significant individual and contextual heterogeneity. To achieve meaningful human-computer and human-robot interactions, multi-modal models of user states are needed. In this context, Rudovic et al. [79] proposed a new multi-modal AL approach that employs the concept of deep RL to find an optimal policy for active selection of user data, which is necessary to train target (modality-specific) models. They analyzed several multi-modal data fusion strategies and showed that the proposed model-level fusion combined with RL outperforms feature-level and modality-specific models as well as naive AL strategies such as random sampling and standard heuristics such as uncertainty sampling. Their multi-modal AL approach provides an optimal policy for selecting active data. Consequently, these data are used to retrain classification models to estimate the target output from video segments of fixed size. The proposed method consists of two sequential processes:

- training the classifiers for the target output;
- learning the RL model's Q-function for active data selection;

These two steps are executed in a loop in which the target classifiers are trained using data from each mode ($m = 1, ..., M$), resulting in M classifiers being trained in parallel. Model-level fusion is then performed based on their outputs to obtain the target label for the input. Following is the algorithm:

Algorithm 20 Multi-modal Q-learning (MMQL)

Input: Dataset $D = \{D^1, D^2, ..., D^M\}$, models $\{\varphi_0 = \{\varphi_0^1, \varphi_0^2, ..., \varphi_0^M\}, Q_0^\pi\} \leftarrow rand, B$

for $e := 1$ to $|D|$ **do**

$D_0^l \leftarrow \{\}$ and shuffle D

$\varphi_e \leftarrow \varphi_{e-1}$, $Q_e^\pi \leftarrow Q_{e-1}^\pi$

for $i := 1$ to $|D|$ **do**

$\hat{y}_i \leftarrow majorityVote(\varphi_e(x_i))$

construct a new state $s_i \leftarrow x_i$

the agent makes a decision according to $a_i = \underset{a}{argmax}\, Q_e^\pi(s_i, a)$

if $a_i = 1$ **then**

ask for label $D_e^l \leftarrow D_e^l \cup (x_i, y_i)$

end if

compute the reward $r_i \leftarrow R(a_i, \hat{y}_i, y_i)$

if $|D_e^l| = B$ **then**

Store (s_i, a_i, r_i, end) in M

Break

end if

construct the new state $s_{i+1} \leftarrow x_{i+1}$

store transition (s_i, a_i, r_i, s_{i+1}) in M

update Q_e^π using a batch form M

end for

update models φ_e using D_e^l

end for

Return: $\varphi \leftarrow \varphi_e$, $Q_*^\pi \leftarrow Q_e^\pi$

where:

- B: the number of possible samples;
- π: is an optimal policy;
- x: is a sentence;

- y: is an output;
- φ: is the learned model;
- s: is a state;
- a: is an action;
- r is a reward;
- M: is the replay memory.

Given the input multi-modal feature vectors x_i, the active learner decides whether or not to request the true label y_i. If the label is requested, the model receives a negative reward to reflect the high cost of obtaining the labels. If the label is not requested, the model receives a positive reward if the estimate is correct, otherwise it receives a negative reward. The following RL reward function encodes this aspect:

$$r_i(a_i, \hat{y}_i, y_i) = \begin{cases} r_{req} = -0.05, \; if \; a_i = 1 \\ r_{cor} = 1, \; if \; a_i = 0 \; \wedge \; \hat{y}_i = y_i \\ r_{inc} = -1, \; if \; a_i = 0 \; \wedge \; \hat{y}_i \neq y_i \end{cases} \tag{2.51}$$

where y_i is the target label obtained from the majority vote of mode-specific engagement classifiers. The parameters of the Q function are optimized, given the action-space pairs and rewards, minimizing Bellman's loss on the training data, which encourages the model to improve the expected reward estimate at each training iteration:

$$L_B^{(i)}(\Theta) = [Q_\Theta(s_i, a_i) - (r_i + \gamma \underset{a_{i+1}}{max} Q_\Theta(s_{i+1}, a_{i+1}))]^2 \tag{2.52}$$

where γ is a parameter set to 0.9. Instead of using heuristic strategies for active data selection, they used deep RL to actively select the most informative data samples for model fit to the target user. They showed that:

- the effectiveness of this method on a difficult multi-modal dataset of child-robot interactions during autism therapy;

- that the learned data-selection policy can generalize well to new children by allowing the pre-trained engagement classifiers to adapt quickly to the target child;

- how the proposed multi-modal AL approach can be used to efficiently personalize engagement classifiers to the target user using a small amount of data from actively selected users.

2.5.4 Quantum Computing

Quantum computing is a type of computation that uses quantum physics phenomena such as superposition, interference and entanglement [33]. Quantum computers are devices that perform quantum computations. Although current quantum computers are too small to outperform conventional (classical) computers in practical applications, larger realizations are expected to solve some computational problems, such as integer factorization (which is the basis of RSA cryptography), significantly faster than classical computers. One area of quantum information science is the study of quantum computing. There are various models for performing quantum computations, the most common of which are quantum circuits. The quantum Turing machine, quantum annealing and adiabatic quantum computation are other models. A qubit can be in a 1 or 0 quantum state, or in a superposition of the two. When measured, however, it is always 0 or 1; the probability of either outcome is determined by the quantum state of the qubit just before measurement. Efforts to build a physical quantum computer focus on technologies such as transmons, ion traps, and topological quantum computers, which aim to produce high-quality qubits. These qubits can be constructed differently depending on whether quantum logic gates, quantum annealing, or adiabatic quantum computation are used in the quantum computer computing paradigm. Currently there are several substantial obstacles to the construction of usable quantum computers. It is particularly difficult to maintain the quantum states of qubits because of quantum decoherence and state integrity. As a result, quantum computers

require error correction. A quantum computer can solve any computational problem that a classical computer can solve [64]. Conversely, any problem that a quantum computer can solve can also be solved by a classical computer, at least in theory, with sufficient time. Quantum computers are believed to be able to solve problems quickly that no classical computer could solve in a reasonable amount of time-a feat known as "quantum supremacy." Quantum complexity theory is the study of the computational difficulty of problems in relation to quantum computers.

Yongcheng Ding et al. [20] have proposed the use of active learning for efficient quantum information retrieval, a critical task in the design of quantum experiments. When dealing with large amounts of data, active learning is used for classification, with minimal loss of fidelity. As we know, active learning considers the cost of labeling, examines the most informative models (quantum states) to propose the least number of labels (measures) that provide the maximum knowledge gain. In this context, they provided a methodology to decide on the best experimental design for binary classification using AL algorithms. To achieve this goal, the estimation models are updated after identifying the qubit with the largest uncertainty using weak measurements in each cycle. This allows partial information to be extracted while slightly perturbing the qubits, resulting in reduced cost in terms of loss of fidelity. They presented AL techniques for extracting quantum information with an experimental design, also demonstrated a complete binary classification problem by extracting information from the qubits [31] using weak measurements, then compared Uncertain Sampling, Query-by-Committee and random sampling procedures, as well as labeling techniques using weak and strong measurements, and obtained the following result: with only 5 percent of labeled samples, they achieved about 90 percent rate estimation. They found that a weak measurement strategy outperformed a strong measurement method. Their approach incorporates the concept of trade-off and dynamic prediction, with the efficiency of a generative model.

Chapter 3

Active Learn by Gradient Variation

As discussed in the previous chapter a major challenge in active learning is to select the most informative instances to be labeled by an annotation oracle at each step. In this respect, one effective paradigm is to learn the active learning strategy that best suits the performance of a meta-learning model. Several strategies based on this approach, such as [5, 37, 22] have been discussed in Section 2.3.11. This chapter contains a strategy that first measures the quality of the instances selected in the previous steps and then trains a machine learning model that is used to predict the quality of instances to be labeled in the current step. The proposed approach selects the instances to be labeled as the ones producing the "maximum change" to the current classifier by using a proxy measure to estimate this change. That is, the key idea is to select instances to be submitted to the oracle to be labeled according to their "importance" in the training phase, which in turn is measured taking into account the "differences" between the learning gradient of the classification model when trained with or without the instance at hand. This approach can be instantiated with any classifier trainable via gradient descent optimization, and here we provide a formulation based on a deep neural network model, which has not been investigated in existing learning-to-active-learn approaches. The proposed instance selection approach is modeled as a regression problem,

that exploits the training gradient of a deep neural network model, and in general of any machine learning model whose training phase is based on the gradient descent algorithm. The experimental validation of this approach has shown promising results in scenarios characterized by relatively few initially labeled instances.

3.1 Main contributions

The learning-to-active-learn approach originally incorporates a regression-based meta-learning approach within a maximum model change framework. It can be summarized as follows:

- Starting from a classification model trained on a small set of instances, is defined an iterative active learning scheme that, in order to decide the bunch of instances to be labeled by an annotation oracle, it predicts which instances will yield the maximum change to the current classifier;
- Is defined a meta-learning process upon two key ingredients: a notion of the importance of unlabeled instances (from the pool of active learning choices) that expresses the contribution that each instance provides to the learning of the classifier; and a regression model to be trained on pairs of labeled instances with associated importance scores;
- Is designed our approach to profitably exploit the learning capabilities of (Deep) Neural Network models trained on a classification task. Nonetheless, the proposed learning-to-active-learn approach is actually versatile w.r.t. the supervised learning model, as long as the gradient descent is used as the training optimization method;
- While taking advantage of a deep neural network model, is also faced a challenge related to how to score the importance of instances to drive the active learning process. Indeed, one cannot rely on the differences between the parameters of a classifier trained on a set of instances and the

parameters of a classifier trained conditionally to the presence/absence of a given instance, since such differences are expected to be negligible in the case of neural network classifiers. Therefore, was investigated different strategies of instance importance scoring by considering variations in the learning gradient of the neural network model. In this regard, the key idea was to account for the similarity of direction of two gradients, the one unbiased and the other one biased w.r.t. a candidate instance for labeling at each step of the active learning process. Measuring the importance score of an instance is not a simple task in the case that neural network classifiers are considered, as the differences between the classifier trained using the whole training set and the classifier trained without considering the instance are typically very small in the case of neural network classifiers;

- The experimental evaluation conducted on CIFAR-10 image data, and including a comparison with random and LCS baselines, has shown promising results by the proposed approach in terms of percentage increase in accuracy, due to the active learning process driven by the proposed instance importance scoring strategies, which tends to improve as the number of initially available labeled instances gets smaller.

3.2 Proposed Approach

In this section first is introduced the general characteristics of the proposed active learning framework, which is inspired by the one proposed in [52]. This approach is designed to be versatile w.r.t. the supervised learning model, as long as the gradient descent is used as the training optimization method. Therefore, this approach is particularly well suited to be equipped with a (Deep) Neural Network model. A classification problem consists in associating every instance taken from a predefined domain $\mathcal{D}$ with a label taken from a fixed universe of labels $\mathcal{L}$. Assuming the presence of a set of instance-label pairs $LI \subseteq \mathcal{D} \times \mathcal{L}$ and a set of unlabeled instances $UI \subseteq \mathcal{D}$, where for

each pair $\langle x, y\rangle \in LI$, x is an instance in $\mathcal{D}$ and y is the label associated with x. The proposed approach is comprised of two phases:

- In the initialization phase, a neural network is trained with the labeled instances in LI. An initial set of unlabeled instances is randomly selected from UI and these instances are submitted to the oracle to be labeled, thus obtaining a new set of labeled instances, denoted as NLI;
- In the iterative phase, several pool-based active learning steps are performed. In each step, the set NLI of newly labeled instances is used to train the classifier together with the set LI.

When retraining the classifier, the *importance* of every instance x in NLI during the training is measured so as to assign an *importance score* to x. Next, a regressor is trained using the instances in NLI that aim to predict the importance scores. Finally, the top-k instances having the greatest importance score are selected for oracle labeling and, once labeled, they replace the set NLI so to start the next active learning step. The concept of importance score is at the core of this approach. Following the model change framework [85], the importance score of an instance x measures the impact of having x in the training set for the obtained classifier. That is, the importance score of a (labeled) instance x w.r.t. a set of labeled instances is a measure of the difference between the parameters of the classifier θ trained over LI and the parameters of the classifier $\hat{\theta}$ trained over $LI \cup \{\langle x, y\rangle\}$, where y is the label of x. Unfortunately, in the case of neural network classifiers, for the most commonly used training algorithm, such as the stochastic gradient, there is (almost) no difference between the parameters of the classifier trained using LI and the parameters of the classifier trained conditionally to the presence/absence of a given instance, i.e., trained using $LI \cup \{\langle x, y\rangle\}$. Indeed, cannot relied on the differences between the parameters of a classifier trained on a set of instances and the parameters of a classifier trained conditionally to the presence/absence of a given instance, since such differences are expected to be negligible in the case of neural network classifiers. To overcome this issue, was defined different

notions of importance score (cf. Section 3.2.2). In the next section is provided a detailed description of the proposed approach.

3.2.1 The *LAL-IGradV* algorithm

Algorithm 21 shows the general schema of the proposed approach, named ***L**earning to **A**ctive **L**earn by Instance **I**mportance based **Grad**ient **V**ariation* (*LAL-IGradV*). *LAL-IGradV* receives in input a (small) set of labeled instances LI, a set of unlabeled instances UI, a deep neural network model DNN, a regressor model R, the number *epochs* of active learning epochs, and the number k of unlabeled instances to select for oracle labeling at each active learning epoch. The algorithm first trains DNN using LI (line 2), randomly selects k unlabeled instances from UI and asks the oracle to label them, thus obtaining the initial set NLI of oracle-labeled instances (lines 3-4). Then, at each epoch (lines 5-14), *LAL-IGradV* performs the following steps:

- The neural network model is trained using the labeled instances in LI and NLI (line 7). During the training process, every instance $x \in NLI$ is associated with its importance score r_x. The computation of the importance scores of the instance in NLI is performed using one of the techniques described in Section 3.2.2;

- A regressor R is trained on the set $\{(x, r_x)|x \in NLI\}$. Note that the choice of the regressor model actually used in this and the subsequent steps is orthogonal w.r.t. the proposed approach; however, it is essential that the chosen regression model must be trainable using a small set of labeled instances;

- NLI instances are added to LI (line 10);

- The regressor R is applied to the instances in UI so that, given an instance x, it predicts its importance score $\widehat{r_x}$ (line 11);

- The algorithm selects the top-k instances from UI ($topK$) having the highest predicted importance scores, and these instances in $topK$ are

submitted to the oracle for labeling (lines 12-13). Finally, NLI is replaced with *topK* (line 14).

Algorithm 21 *LAL-IGradV*

1: **Input:** LI: set of labeled instances, UI: set of unlabeled instances, DNN: deep neural network model, R: importance score regressor, *epch*: maximum number of epochs, k: number of relevant instances to select
2: Train DNN on LI
3: $NLI \leftarrow$ Select k instances from UI uniformly at random
4: The oracle annotates the instances in NLI
5: **for** $i = 1 \dots epch$ **do**
6: **for** $x \in NLI$ **do**
7: Train DNN on $LI \cup NLI$ and compute importance score r_x
8: **end for**
9: Train R on the set of pairs $\{\langle x, r_x \rangle \,|\, x \in NLI\}$
10: $LI \leftarrow LI \cup NLI$
11: Apply R to UI instances to predict importance scores ($\widehat{r_x}$)
12: $topK \leftarrow$ Select top-k instances from UI by importance score $\widehat{r_x}$
13: The oracle annotates the instances in $topK$
14: $NLI \leftarrow topK$
15: **end for**

3.2.2 Importance scoring strategies

Let $f(x_i, \theta)$ be the output of a DNN model f characterized by a vector of parameters θ for an input x_i and let $X = \{x_1, \dots, x_n\}$ be a set of instances used for training f, where each sample $x_i \in X$ is associated to a label y_i. The training of the DNN f over X requires solving the following:

$$\arg\min_{\theta} \left(\sum_{x_i \in X} (L(y_i, f(x_i, \theta)) + reg(\theta) \right) \tag{3.1}$$

where:

- $L(y_i, f(x_i, \theta))$ is the loss of the model for instance x_i;
- $reg(\theta)$ is the regularization of the parameters.

The training of f is accomplished by iteratively updating the parameters θ, through two steps:

- computing the change in all weights w.r.t. the change in error, i.e., the *gradient*, defined as:

$$\delta(X) = \frac{\partial}{\partial \theta} \sum_{x_i \in X} (L(y_i, f(x_i, \theta)) + reg(\theta)) \tag{3.2}$$

- updating θ using $\delta(X)$, i.e., $\theta_{k+1} = \theta_k - \eta \times \delta(X)$, where η is the update step size.

Four strategies are proposed to associate each instance in NLI with its importance score during the training of the DNN classifier. The goal shared by the various techniques is to modify the training of the neural network model by accounting for the importance of the instances in NLI involved in each training step. Each of the proposed techniques makes use of the gradient corresponding to the instances currently in LI and NLI, i.e., $\delta(LI \cup NLI)$, hereinafter simply denoted as δ. The four proposed techniques differ in the way the importance of an instance x in NLI is calculated with respect to the single epoch. We will use symbol δ_x to denote the value of the gradient $\delta(\{x\})$, and $\delta_{\neg x}$ to denote the value of the gradient $\delta(LI \cup NLI \setminus \{x\})$. In the following, we describe our proposed techniques for computing the importance scores:

- **Direct similarity** (DS): given an instance x in NLI, this strategy compares the learning gradient of the neural network at the current epoch, δ, with the gradient calculated with respect to x only, i.e., δ_x. The importance score of x at the current epoch is defined as the cosine similarity between δ and δ_x, i.e., $r_x = cos(\delta, \delta_x)$. The rationale of this strategy is that an instance $x \in NLI$ is likely to be more important for the training of DNN at the current epoch if there is a small difference between

the directions of the gradients δ and δ_x, as reflected by a high value of the cosine similarity between the two gradients. That is, the more the learning behavior of the neural network considering the whole training set is similar to the one of the same neural network trained on x only, the higher the importance of x is;

- **Ranked direct similarity** (*RDS*): this strategy first applies the *DS* technique, then the importance scores of the instances in NLI computed by *DS* are ordered and divided into three bins, which correspond to the top quartile of the importance scores, the bottom quartile, and the union of the second and third quartiles. The instances falling into the top quartile will be associated with score 1, the ones falling into the bottom quartile with score 0, and the other instances with score 0.5;

- **Leave-one-out distance** (*LD*): given an instance x in NLI, this strategy compares δ with the gradient calculated when leaving out x, i.e., $\delta_{\neg x}$. The importance score of x at the current epoch is defined as the complement of the cosine similarity (i.e., cosine distance) between δ and $\delta_{\neg x}$, i.e., $r_x = 1 - cos(\delta, \delta_{\neg x})$. The rationale of this strategy is that an instance $x \in NLI$ is likely to be more important for the training of DNN at the current epoch if leaving it out will lead to large differences between the learning behavior of the neural network considering the whole training set and the learning behavior of the same neural network trained without x, i.e., a large change in the direction of the gradient $\delta_{\neg x}$ w.r.t. the gradient δ, as reflected by a high value of the cosine distance between the two gradients. The learning gradient for the neural network at the current epoch is calculated with respect to all instances except one instance $x \in NLI$, denoted by $grad_x$. Therefore, the importance of x at the current epoch is measured as one minus the cosine similarity between δ and δ_x. The idea behind this calculation is to consider example x the more relevant the more the update that the network weights receive in the real training phase (associated with the direction of the gradient *grad*) and

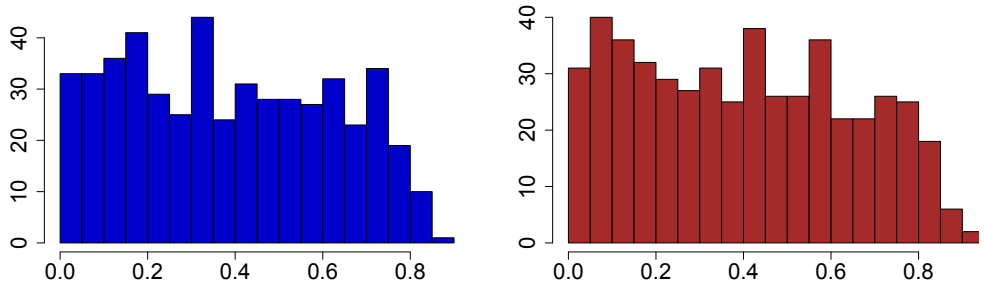


Figure 3.1: Histograms of the $k = 500$ measurements of *DS* (left) and *LD* (right) strategies at the first epoch of active learning

different from what it would have been not considering the example x during this training period (associable with the direction of the $grad_x$ gradient);

- **Ranked leave-one-out distance** (*RLD*): analogously to *RDS* w.r.t. *DS*, the *RLD* strategy adds the same discretization step over the importance scores computed by *LD*.

All the techniques just described are performed on each data as many times as there are active learning epochs, the score value that is actually used is the average of all values calculated during these epochs.

Figure 3.1 shows the distributions of the importance scores yielded by *DS* and *LD* at the first epoch of active learning. As it can be observed, both distributions span over the full regime of admissible values, despite the high dimensionality of the gradient vectors being compared.

3.3 Experimental Evaluation

3.3.1 Data

For the evaluation of this proposal was used the well-known CIFAR-10 dataset [1] [53], which consists of 60000 instances representing 32x32 colour images, labeled using 10 mutually exclusive classes, with 6000 images per class.

[1] https://www.cs.toronto.edu/~kriz/cifar.html

The dataset is organized into 50000 instances as the training set and 10000 instances as the test set. The latter contains exactly 1000 randomly-selected images from each class, while the training set is comprised of five training batches, which contain 5000 images from each class. The training set was divided into two parts, the one corresponding to the set of labeled instances (LS), and the other corresponding to the set of unlabeled instances (US), and at the time of labeling by the specialist of the relevant examples identified by the proposed technique, the dataset label is used to simulate the expert.

3.3.2 Baseline methods

The performance of this methods was compared with a Random baseline and the LCS method [83]. The Random baseline, hereinafter denoted as Rnd, simply selects k instances to be annotated at each epoch uniformly at random from the set of unlabeled instances. The LCS method follows an uncertainty sampling approach, therefore it estimates the uncertainty of a specific instance and exploits it as criterion for the unlabeled instance selection. More precisely, given an instance x and a classification model θ, the uncertainty of x w.r.t. θ ($\phi(x)$) is measured as:

$$\phi(x) = (1 - P_\theta(y^*|x)) \times \frac{m}{m-1} \tag{3.3}$$

where:

- $P_\theta(y^*|x)$ denotes the probability that the model θ assigns to the label y^* for the instance x;
- y^* is the label for which θ yields the maximum probability on x (i.e., $y^* = \arg\max_y P_\theta(y|x)$);
- m is the cardinality of the set of labels.

Note that the uncertainty function ranges between $[0, 1]$, where 1 is the most uncertain score.

3.3.3 Settings and assessment criteria

In the experimental evaluation, it was used 6 Convolutional Neural Network (CNN) 2D layers, with 3 input channels, kernel size 3, stride size 3, padding size 1, ReLU activation function and max pool layers. The CNN module has on top a fully-connected network with an input layer of size 4096, one hidden layer with input size 4096 and output size 1024, another hidden layer with input size 1024 and output size 512, an output layer of size 10 (i.e., number of classes), and a dropout layers with probability 0.1. In our *LALIGradV* algorithm, the DNN model was trained using cross entropy as loss function and Adam optimizer (with learning rate 1e-4 and weight decay 5e-4), a number of epochs equal to 10 for both the initialization step of training (Line 2) and the training steps in the main loop (Line 8). Also, the maximum number of iterations of the algorithm, i.e., number of epochs in the active learning process (*epch*) was set to 10. Unless otherwise specified, the number k of instances to select from UI was set to 500; the size of LI, respect UI, was experimentally varied. As the regressor (R), it was used two models: the Gradient Boosting Regressor, with least absolute deviations (LAD) loss function and 200 estimators, for the *DS* and *LD* strategies, and the Random Forest Classifier, with maximum depth 5, for the *RDS* and *RLD* strategies. To simulate the oracle for annotating the instances, it was resorted to the availability of class label information for the CIFAR-10 data: whenever an instance was used in the UI set, it was masked its actual label during the learning process, and it was unveiled the label only if the instance was selected within the *topK* set of instances to annotate. A batch of tests was carried out for each technique as follows:

- 10 epochs of neural network training were employed for each selection policy (Line 2 and 7/8);
- 10 training epochs (*epch*) of the iterative active learning process in question were used for each technique, at each step 500 ($|topK|$) units of elements are taken and chosen for each epoch of active training;

- During the tests, the size of the "pre-annotated" training set was varied in order to observe the gain that would be obtained with this technique as the labeled data decreased;

- In order to simulate the human annotator, the annotated dataset was taken and they were "annotated" after the selection of the examples using the label already provided in the dataset.

To assess the performance of the methods, it was considered the accuracy of the classifier during the various training batches, in absolute terms as well as in terms of percentage increase w.r.t. the early accuracy of the classifier itself or the accuracy of a reference method. More precisely, was computed:

- the *accuracy at the initial step of training* of *LAL-IGradV* (line 2), denoted as $A^{(0)}$, and the accuracy at the end of the active learning process, denoted as A;

- the *percentage increase in the accuracy* of *LAL-IGradV*, which is defined as $100 \times (A - A^{(0)})/A^{(0)}$;

- the *percentage increase in the accuracy* of *LAL-IGradV* w.r.t. *Rnd*, resp. *LCS*, which is defined as $\%_{Rnd} = 100(A - A_{Rnd})/A_{Rnd}$, respect $\%_{LCS} = 100(A - A_{LCS})/A_{LCS}$, where A_{Rnd} and A_{LCS} denote the accuracy at the end of the active learning process for *Rnd* and *LCS*.

The starting neural network is the same for all the tests, only the second part varies, the one in which a greater amount of unlabeled data comes into play, moreover all the random seeds are set in such a way as to obtain the reproducibility of the tests.

3.3.4 Results

Table 3.1 reports on the performance of our *LAL-IGradV* variants corresponding to the four importance scoring techniques, for varying percentages of the set of unlabeled instances (UI); for example, row '10%' indicates that

Table 3.1: Performance of the proposed methods: initial and final accuracy, percentage increase w.r.t. *Rnd* and w.r.t. *LCS*, and active learning time (sec) averaged over the epochs, for various percentage values of unlabeled instances.

	$A^{(0)}$	*DS*				*RDS*				*LD*				*RLD*			
		A	$\%_{\mathsf{Rnd}}$	$\%_{\mathsf{LCS}}$	time	A	$\%_{\mathsf{Rnd}}$	$\%_{\mathsf{LCS}}$	time	A	$\%_{\mathsf{Rnd}}$	$\%_{\mathsf{LCS}}$	time	A	$\%_{\mathsf{Rnd}}$	$\%_{\mathsf{LCS}}$	time
10%	0.793	0.831	2.32	0.43	186	0.832	2.44	0.54	191	0.831	2.28	0.39	625	0.828	1.90	0.01	769
20%	0.783	0.826	1.90	0.75	178	0.825	1.79	0.65	217	0.824	1.72	0.57	623	0.822	1.46	0.32	796
30%	0.784	0.827	1.95	0.50	170	0.828	2.06	0.61	250	0.826	1.75	0.30	620	0.822	1.46	0.32	827
40%	0.763	0.819	4.01	1.08	170	0.811	3.04	0.13	295	0.811	3.02	0.11	620	0.811	2.96	0.05	872
50%	0.733	0.801	5.97	2.84	162	0.800	5.82	2.70	352	0.799	5.80	2.67	619	0.779	3.07	0.03	1002
60%	0.728	0.801	6.32	3.21	162	0.798	5.96	2.86	423	0.795	5.57	2.48	614	0.777	3.20	0.18	1089
70%	0.708	0.778	6.49	2.50	154	0.778	6.38	2.40	513	0.773	5.82	1.86	607	0.760	4.01	0.12	1190
80%	0.640	0.705	5.39	1.82	139	0.704	5.27	1.71	613	0.700	4.62	1.08	604	0.694	3.78	0.27	1310
90%	0.570	0.644	5.89	2.22	129	0.636	4.60	0.98	732	0.632	3.95	0.35	602	0.636	4.59	0.97	1395

10% of the instances of the CIFAR-10 training set was used as UI and the remaining 90% of the training set as LI.

Looking at the table, several remarks stand out. First of all, it is not surprising to notice that the accuracy values (i.e., columns corresponding to A and $A^{(0)}$) tend to decrease as the percentage of unlabeled instances gets higher, since the *LAL-IGradV* method is forced to handle progressively reduced sets of labeled instances on its initial training. More interestingly, the percentage increase of each of the *LAL-IGradV* variants w.r.t. both *Rnd* and *LCS* is always positive (up to 6.5% against *Rnd* and up to 3.2% against *LCS*) and it tends to improve with higher percentages of unlabeled instances, with peaks around 70% against *Rnd* and around 50-60% against *LCS*. As concerns the impact of the importance scoring technique, is observed that all the *LAL-IGradV* variants are able to improve upon the accuracy at the initial training step. Moreover, the direct similarity based techniques, i.e., *DS* and *RDS*, reveal to be more efficient [2] as well as more accurate than the leave-one-out distance based techniques, for each percentage of unlabeled set. This would indicate that the direct similarity based techniques are more effective than the leave-one-out distance based techniques. This fact was intuited because to a

[2]Experiments were carried out on an Intel Core i7 CPU @2.90GHz, 32GB RAM, with NVIDIA GeForce RTX 2070 Super GPU

higher sensitivity of the approach in capturing the gradient direction change due to the individual contribution of an instance rather than to the masking of a single instance in the training gradient, which would result in a more diluted signal of variation of the training gradient. Figure 3.2 focuses on the percentage increase in accuracy that each active learning method achieves by varying the fraction of unlabeled instances. As expected due to the advantage of performing an active learning task, the percentage increase values tend to improve for higher fractions of unlabeled instances. The trends are steeper for that *LAL-IGradV* methods, particularly for *DS* and *RDS*, followed by *LCS*. Indeed, it is worth emphasizing that *LAL-IGradV* methods achieve the best performance gain against the two baselines as the fraction of labeled instances becomes smaller.

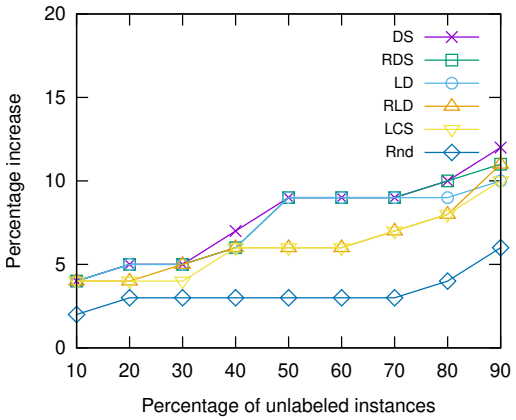
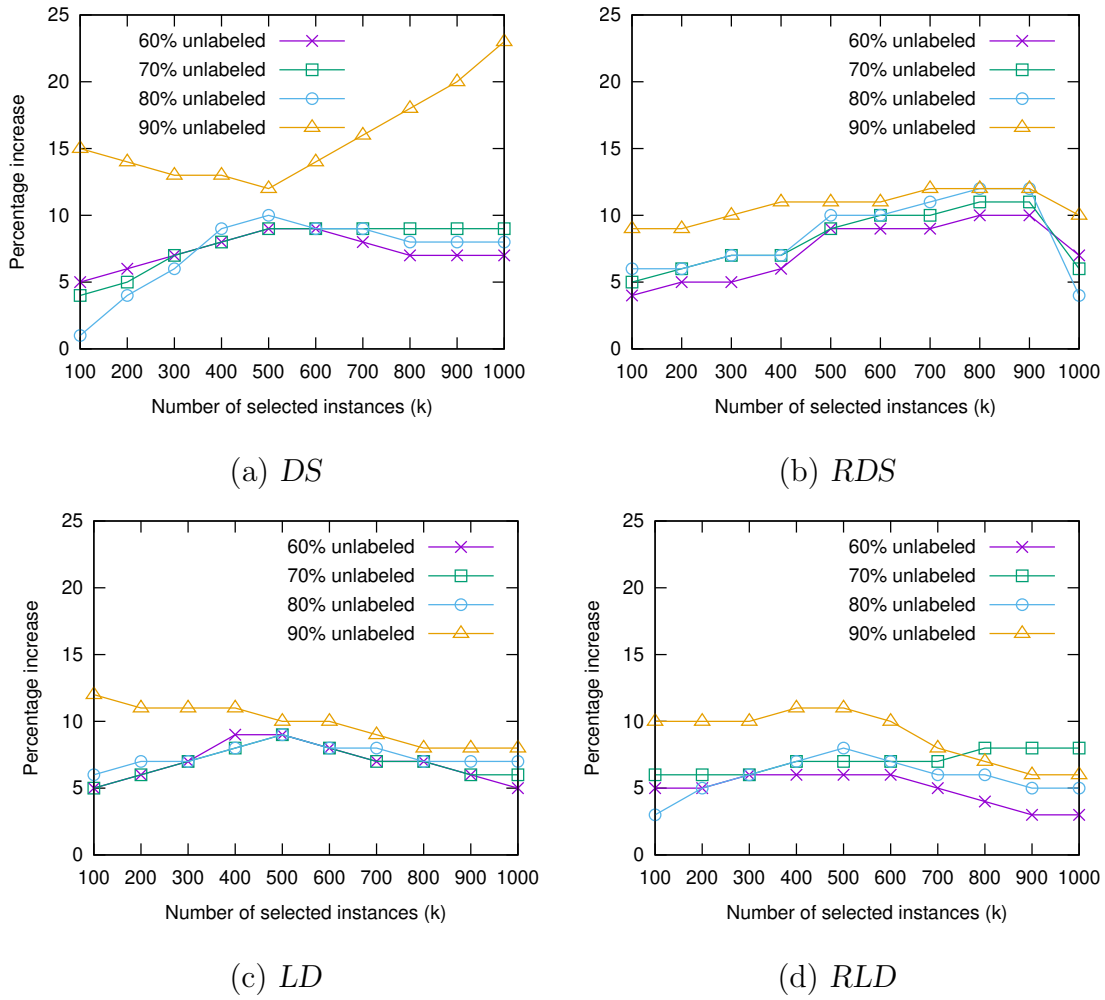


Figure 3.2: Percentage increase of accuracy for the various active learning methods, with varying percentage of unlabeled instances, and number of selected instances (k) equal to 500.

In Figs. 3.3 and 3.4, is delved into the trends of accuracy percentage-increase obtained by a particular active learning method, for varying k, i.e., number of unlabeled instances to be selected at each epoch of the active learning process. At a first glance, in each of the plots, was noticed that the curve of the percentage increase values over k is more likely to change for larger fractions of the set of unlabeled instances, with the most evident changes corresponding to 90%.



(a) *DS* (b) *RDS*

(c) *LD* (d) *RLD*

Figure 3.3: Percentage increase due to active learning based on our *LAL-IGradV* variants, by varying the number of selected instances (k) and the percentage of labeled instances.

A few interesting remarks can be drawn from Fig. 3.3. When portions of UI below 90% are selected, can be observed a relatively small range of variation of the percentage increase values (approximately from 5% to 10%), with peaks around $k = 500$ for the *DS* and *LD* variants, and around $k = 900$ for the *RDS* and *RLD* variants. This would hint at higher requirements (i.e., higher k) needed for the importance scoring strategies that compute discretized importance scores. Another remark is on the curves corresponding to the use of 90% of the set of unlabeled instances: compared to the curves corresponding to lower fractions of UI, the percentage increase values are higher on average,

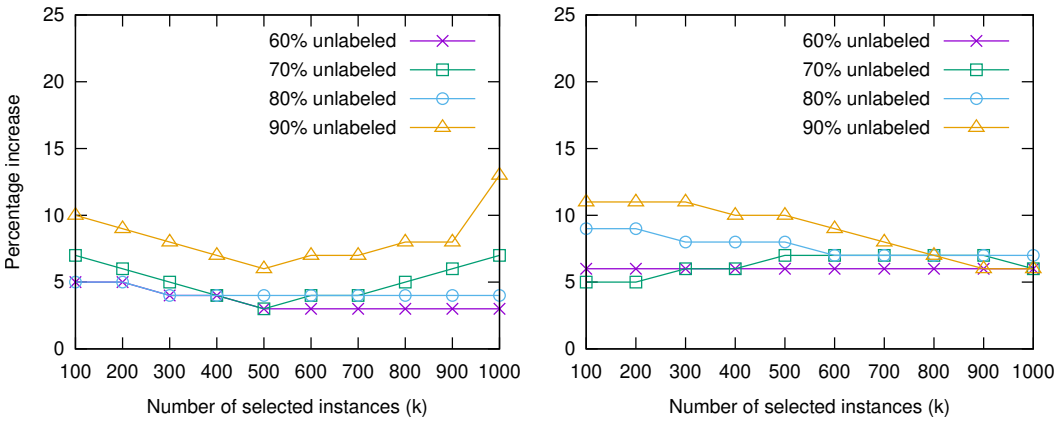


Figure 3.4: Percentage increase due to active learning based on *Rnd* (left) and on *LCS* (right), by varying the number of selected instances (k) and the percentage of labeled instances.

and the trends are quite different, especially for the *DS* variant where can be observed a minimum (rather than a maximum) for $k = 500$. Apart from this exception, it is worth noticing that better percentage increase of accuracy do not necessarily correspond to a higher number k of selected instances. This might be explained since the more unlabeled instances are selected for labeling, the more the method is less likely to make a correct choice for changing the most the current model, as the latter is being trained only on few instances, thus lacking full knowledge on the class distribution of all the instances for available training. Concerning the baseline methods (Fig. 3.4), two different situations occur between the *Rnd* plot (on the left) and the *LCS* plot (on the right). The former shows a decreasing trend until mid values of k (i.e., around 500 instances) followed by a rising trend, which sheds light on the divergent behavior of a random selection of the unlabeled instances w.r.t. all the other instance selection methods. Also, the *LCS* plot shows curves that tend to monotonically decrease, respect remain substantially unchanged, for larger, respect smaller, fractions of UI, which again puts in evidence how our *LAL-IGradV* variants behave differently from an uncertainty sampling approach like *LCS*.

The proposed *LAL-IGradV* has shown that a learning-to-active-learn by

instance importance based gradient variation improves significantly upon not only a random baseline but also an uncertainty sampling approach like *LCS*. *LAL-IGradV* methods are all able to increase the accuracy at the initial training step, and tend to improve with higher percentages of unlabeled instances. Yet, higher percentages of unlabeled instances lead to an increased gain against *LCS* and random baseline. *LAL-IGradV* methods are also not particularly demanding in terms of number (k) of selected instances to label at each active learning epoch.

Chapter 4

Active Learn by Gradient Variation with Variational AutoEncoders

The *LAL-IGradV* algorithm, discussed in the previous chapter, is able to outperform both a random baseline as well as an uncertainty sampling approach like *LCS*. This chapter will introduce the ***L****earning to* ***A****ctive* ***L****earn by Instance* ***I****mportance based* ***Grad****ient* ***V****ariation with* ***V****ariational* ***A****uto****E****ncoders* (*LAL-IGradV-VAE*) algorithm, an enhancement of *LAL-IGradV* which improves its performance by exploiting a variational autoencoder as a pre-processing step. The adopted variational autoencoder aims to learn low-dimensional latent representations for the input data samples which are ultimately provided as input to the model selector presented in the previous chapter. The rationale behind the new algorithm is that using the original data samples to train the selector model is too expensive for two reasons: (i) due the lack of samples, the selector is unable to optimally learn the rules for estimating the relevance score; and (ii) high-dimensional data highly impact the efficiency of the method in terms of both training and prediction time.

4.1 Background

This section provides preliminaries about the technical aspects of the proposed algorithm *LAL-IGradV-VAE*. In particular, an introduction to autoencoder and variational autoencoder is provided next.

4.1.1 AutoEncoders

An AutoEncoder (AE) [80] consists of a neural network that is trained to reconstruct its input. It is a type of artificial neural network used to learn efficient coding of unlabeled data, thus falling within the realm of unsupervised learning. The encoding is validated and refined by attempting to reconstruct the input from the encoding. The AE learns a representation (encoding) of a data set, typically for dimensionality reduction, training the network to ignore insignificant data also well known as noise. An AutoEncoder is characterized by the following main features:

- specificity of the data: it means that if, for example, an autoencoder is trained to reproduce images of people it cannot be used to reproduce images of animals;
- loss in the reconstruction process: an output is not exactly the same as the input, so if the target is to obtain a faithful reproduction, the AutoEncoder is not the optimal solution;
- unsupervised learning paradigm: only unlabelled data are required to train an autoencoder.

4.1.1.1 The architecture

Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{F} \subseteq \mathbb{R}^p$ be the space of input data instances and the space of compressed data instances, respectively. An autoencoder consists of two main functions, namely an encoder ϕ and a decoder ψ, with the following objectives:

- the **encoder** $\phi : \mathcal{X} \rightarrow \mathcal{F}$ takes an input data instance and it provides a compressed representation for it;
- the **decoder** $\psi : \mathcal{F} \rightarrow \mathcal{X}$ receives the compressed version of a data instance and outputs the original instance with the least possible error.

The goal, as formally defined in [4], is to learn the encoder and decoder functions such that:

$$\phi, \psi = \underset{\phi,\psi}{argmin} ||\mathcal{X} - (\psi \circ \phi)\mathcal{X}||^2 \tag{4.1}$$

In the simplest scenario where the autoencoder consists of a single hidden layer, the encoding stage takes a vector $x \in \mathcal{X}$ and maps it to a new vector $h \in \mathcal{F}$: $h = \sigma(Wx + b)$. The computed vector h is usually referred to as latent representation, while σ is an activation function (e.g. a sigmoid function). Furthermore, W is a weight matrix and b is a bias vector. Weights and biases are typically initialized randomly and then iteratively updated via back-propagation. Subsequently, the decoding phase takes as input an encoded vector h and it reconstructs the original data x by computing $\hat{x}$ as $\hat{x} = \sigma'(W'h + b')$ where σ', W', and b' are not necessarily related to their counterparts in the encoder stage. In general, an autoencoder can have an arbitrary number of hidden layers, such as the one shown in Figure 4.1. Also, the encoded vector $\phi(x)$ can be considered as a compressed representation of the input x since the feature space $\mathcal{F}$ has typically a lower dimensionality than the input space $\mathcal{X}$, i.e. $p < d$.

4.1.1.2 Reconstruction Loss

The autoencoder is trained by minimizing a loss function which measures the reconstruction error of the compressed data instances with respect to their original counterparts. More formally, given a dataset with m data samples, the following average loss is minimized:

$$L = \frac{1}{m} \sum_{j=1}^{m} L(x^{(j)}, \hat{x}^{(j)}) \tag{4.2}$$

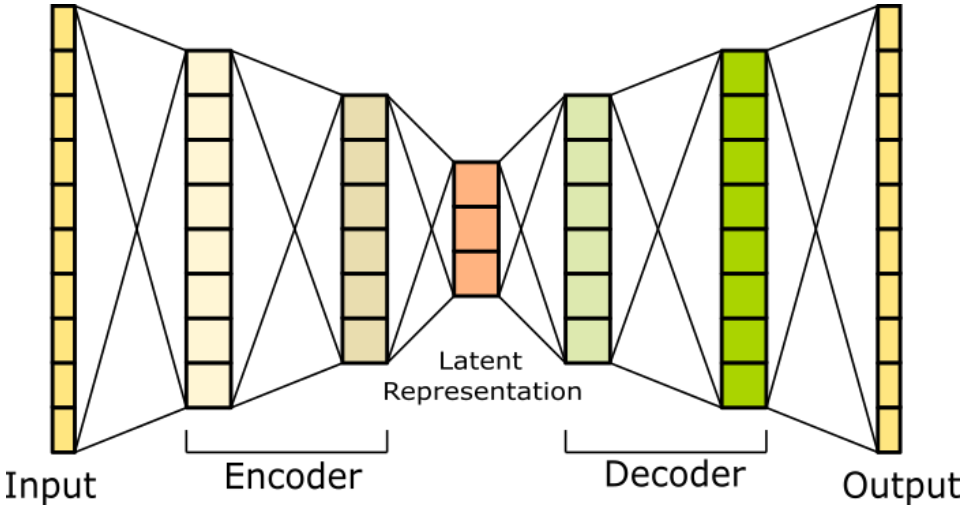


Figure 4.1: An example architecture of AutoEncoder.

where $L(x^{(j)}, \hat{x}^{(j)})$ quantifies the per-instance difference between the original data instance x and its compressed representation $\hat{x}^{(j)}$. The choice of $L(\cdot, \cdot)$ function depends on the nature of data. If the input is categorical, the following cross-entropy loss function can be adopted:

$$L(x, \hat{x}) = -\sum_{i=1}^{d}[x_i \, log(\hat{x}_i) + (1 - x_i) log(1 - \hat{x}_i)] \tag{4.3}$$

On the other hand, when the input is real-valued, one possibility is to consider the squared error which computes the difference between $\hat{x}$ and x as follows:

$$L(x, \hat{x}) = ||x - \hat{x}||^2 = ||x - \sigma'(W'(\sigma(Wx + b)) + b'||^2 \tag{4.4}$$

4.1.1.3 Applications

Autoencoders have a number of applications, including anomaly detection, denoising of images, and dimensionality reduction. The goal of an autoencoder is to reconstruct the data that lies within a subspace of the input domain. This means that the autoencoder should be able to recreate only the input data that exists within this subspace. However, the limitations of the model mean that it can only reconstruct what it was trained on, so any variations or anomalies in the input data will not be detected by the model.

Another use of autoencoders is in image compression. As the encoded inputs have a smaller dimension than the original inputs, the encoder can be used for compression, with the hidden (encoded) representations retaining all or most of the information from the original input, but taking up less space.

4.1.2 Variational AutoEncoders

Starting from the basic autoencoder model, several variants have been proposed in the literature. In particular, here we focus on the Variational AutoEncoders (*VAE*) variant that allows to boost the representation capabilities of autoencoders [48]. The main differences between *VAE* and the basic model are the following:

- in the encoding stage the input x is passed to the encoder module. Instead of directly generating a hidden representation h as done in AE models, the generation process in *VAE* involves two components: $\mathbb{E}(z)$ and $\mathbb{V}(z)$ where z is the latent random variable with a Gaussian distribution with mean $\mathbb{E}(z)$ and variance $\mathbb{V}(z)$. In practice, Gaussian distributions are used as the encoded distribution, but other distributions can be used as well. The encoder will be a function that maps from $\mathcal{X}$ to $\mathbb{R}^{2d} : x \mapsto h$ (here h is used to represent the concatenation of $\mathbb{E}(z)$ and $\mathbb{V}(z)$);

- z is sampled from the above encoded distribution; specifically, $\mathbb{E}(z)$ and $\mathbb{V}(z)$ are passed into a sampler to generate the latent variable z;

- z is passed into the decoder to generate $\hat{x}$. The decoder will be a function from $\mathcal{Z}$ to $\mathbb{R}^n : z \mapsto \hat{x}$, where $\mathcal{Z}$ is the latent space.

In the *VAE* formulation, h is simply the vector $\mathbb{E}(z)$ for classic autoencoders. In short, the main distinction between *VAE* and AE is that *VAE* have a larger latent space than AE since they are generative models that use a probabilistic distribution to describe data generation. Given an observed dataset $X = \{x_i\}_{i=1}^N$ of N independent and identically distributed random variables

samples, it is assumed that a generative model for each data x_i is conditioned on an unobserved random latent variable z_i that represent the generative distribution parameters. This generative model is also a probabilistic decoder. An approximate posterior distribution over the latent variable z_i is assumed symmetrically given a data x_i denoted by recognition, which is equivalent to a probabilistic encoder and governed by the parameters. Finally, a prior distribution for the latent variables z_i is assumed, denoted by $p_\theta(z_i)$. The parameters are unknown and must be learned from data. The observed latent variables z_i can be interpreted as a recognition model $q_\phi(z|x)$. The marginal log-likelihood is expressed as a sum of the individual data points $log\ p_\theta(x_1, x_2, ..., x_N) = \sum_{i=1}^{N} log\ p_\theta(x_i)$, and each point can be rewritten as follows:

$$log\ p_\theta(x_i) = D_{KL}(q_\phi(z|x_i)||p_\theta(z|x_i)) + \zeta(\theta, \phi; x_i) \tag{4.5}$$

The first term is the Kullback-Leibler divergence of the approximate recognition model from the true posterior, and the second term is the variational lower bound on the marginal likelihood, which is defined as:

$$\zeta(\theta, \phi; x_i) \triangleq E_{q_\phi(z|x_i)}\left[-log\ q_\phi(z|x) + log\ p_\theta(x - z)\right] \tag{4.6}$$

Because the Kullback-Leibler divergence is non-negative, $\zeta(\theta, \phi; x_i)$ is a lower bound on the marginal log-likelihood, and because the marginal log-likelihood is independent of the parameters and, maximizing the lower bound improves our posterior approximation with respect to the Kullback-Leibler divergence. The variational lower bound can be expanded further as follows:

$$\zeta(\theta, \phi; x_i) = -D_{KL}(q_\phi(z|x_i)||p_\theta(z)) + \mathbb{E}_{q_\phi(z|x_i)}[log\ p_\theta(x_i|z)] \tag{4.7}$$

Variational inference follows by maximizing $\zeta(\theta, \phi; x_i)$ all data points with respect to θ and ϕ. Given a dataset $X = \{x_i\}_{i=1}^{N}$ with N data points, we can estimate the marginal likelihood lower-bound of the full dataset $\zeta(\theta, \phi; X)$ using a mini-batch $X^M = \{x_i\}_{i=1}^{M}$ of size M as follows:

$$\zeta(\theta, \phi; X) \approx \tilde{\zeta}^M(\theta, \phi; X^M) = \frac{N}{M} \sum_{i=1}^{M} \zeta(\theta, \phi; x_i) \tag{4.8}$$

Mean-field theory Variational Bayes starts with a factorized approximate posterior and then performs closed form optimization updates (which usually require conjugate priors). *VAE*, on the other hand, takes a different route where the gradients of $\tilde{\zeta}^M(\theta, \phi; X^M)$ are approximated using the reparameterization trick and stochastic gradient optimization.

4.1.2.1 Loss function

Data from the input space are encoded to the latent space using an encoder and noise, and then the data from the latent space is "decoded" to the output space. To move from the latent to the input space (the generative process), we can either learn the distribution (of the latent code) or enforce some structure. In particular, the *VAE* imposes structure on the latent space. The training of *VAE* models is done by minimizing a loss function. As a result, the loss function includes a reconstruction term as well as a regularization term.

- The reconstruction term is on the final layer;
- The regularization term is applied to the latent layer in order to impose a specific Gaussian structure on the latent space. This is accomplished by employing a penalty term $l_{KL}(z, N(0, I_d))$. Without this term, *VAE* will behave like a traditional autoencoder, which may result in overfitting and losing all the desired generative properties of a *VAE*.

4.1.2.2 Sampling z (reparameterization trick)

The purpose of the reparameterization trick is to make the VAE's stochasticity amenable to gradient-based optimization. By introducing random noise through a differentiable function, the gradients of the VAE's parameters can be backpropagated through the stochastic layers of the model, allowing it to be trained with standard gradient-based optimization algorithms. The reparameterization trick so is a simple method for estimating $\zeta(\theta, \phi; x_i)$ from a small

sample of size L. Considering Equation 4.6, where can be reparameterized the random variable $\tilde{z} \sim q_\phi(z|x)$ with a differentiable transformation $g_\phi(\epsilon|x)$ and an auxiliary noise variable ϵ drawn from some distribution $p(\epsilon)$. $\zeta(\theta, \phi; x_i)$ is approximated using this technique as follows:

$$\zeta(\theta, \phi; x_i) \approx \tilde{\zeta}(\theta, \phi; x_i) = \frac{1}{L} \sum_{l=1}^{L} log\, p_\theta(x_i, z_{(i,l)}) - log\, q_\phi(z_{(i,l)|x_i}) \tag{4.9}$$

where $z_{(i,l)} = g_\phi(\epsilon_{i,l}, x_i)$ and $\epsilon_{i,l}$ is a random noise drawn from $\epsilon_l \sim p(\epsilon)$. Because we want to optimize the mini-batch estimates from Equation 4.8, by plugging Equation 4.9 we get the following differentiable expression:

$$\hat{\zeta}^M(\theta, \phi, X) = \frac{N}{M} \sum_{i=1}^{M} \tilde{\zeta}(\theta, \phi, x_i) \tag{4.10}$$

which can be derived according to θ and ϕ and plugged into an optimizer framework. Here below there is the algorithm for the full optimization procedure for *VAE*:

Algorithm 22 Optimization procedure for *VAE*

$(\theta, \phi) \leftarrow$ Initialize Parameter
while not Convergence of (θ, ϕ) **do**
  $X^M \leftarrow$ Random minibatch of M datapoints
  $\epsilon \leftarrow L$ random samples of $p(\epsilon)$ (Gradients of Equation 4.10)
  $(\theta, \phi) \leftarrow$ update parameters based on g
end while
return (θ, ϕ)

where g refers to a differentiable function that maps random noise to the parameters of a probability distribution. L is frequently set to 1 as long as M is large enough. $M = 100$ and $L = 1$ are typical values. The lower bound on the log-likelihood $log\, p_\theta(x_i)$ is given by Equation 4.9. The equation is changed in [12] to:

$$\zeta(\theta, \phi; x_i) = \frac{1}{L} \sum_{l=1}^{L} log \frac{1}{k} \sum_{j=1}^{k} \frac{p_\theta(x_i, z_{(j,l)})}{q_\phi(z_{(j,l)|x_i})} \tag{4.11}$$

Rather than taking the gradient of a single randomized latent representation, the generative network's gradients are learned by a weighted average of the sample over different samples from its (approximated) posterior distribution. Weights are simply the likelihood functions $q_\phi(z_{(j,l)|x_i})$.

4.2 Proposed Approach

This section introduces a variant of the active learning framework *LAL-IGradV*, which adopts the *VAE* as a preprocessing step in order to reduce the dimensionality of the input data instances. Let $LI \subseteq \mathcal{D} \times \mathcal{L}$ be a set of instance-label pairs and $UI \subseteq \mathcal{D}$ be a set of unlabeled instances, where for each pair $\langle x, y \rangle \in LI$, x is an instance in $\mathcal{D}$ and y is the label associated with x. The proposed approach consists of two main phases:

- An initialization phase where (i) a *VAE* is trained with all the samples in UI yielding a new set of (compressed) unlabeled samples; (ii) a neural network is trained with the labeled instances in LI; and (iii) an initial set of unlabeled instances is randomly selected from UI and these instances are submitted to the oracle to be labeled, thus obtaining a new set of labeled instances, denoted as NLI;

- An iterative phase, where several pool-based active learning steps are performed. In each step, the set NLI of newly labeled instances is used to train the classifier together with the set LI with the support of the *VAE* previously trained.

When retraining the classifier, the *importance* of every instance x in NLI during the training is measured so as to assign an *importance score* to the latent representation of x. Next, a regressor is trained using the instances in NLI that aim to predict the importance scores. Finally, the top-k instances having the greatest importance score are selected for oracle labeling and, once labeled, they replace the set NLI so to start the next active learning step. In the next section, a detailed description of the proposed approach is provided.

4.2.1 The *LAL-IGradV-VAE* algorithm

Algorithm 23 shows the general schema of the proposed approach, named ***L**earning to **A**ctive **L**earn by Instance **I**mportance based **Grad**ient **V**ariation with **VAE*** (*LAL-IGradV-VAE*). *LAL-IGradV-VAE* receives in input a (small) set of labeled instances LI, a set of unlabeled instances UI, a deep neural network model DNN, a regressor model R, a VAE, the number *epch* of active learning epochs, and the number k of unlabeled instances to select for oracle labeling at each active learning epoch. The algorithm first trains DNN using the samples in LI (line 2), then a VAE model is trained with data instances in UI such as to obtain a summarized representation of the instances in the dataset (line 3); it randomly selects k unlabeled instances from UI and asks the oracle to label them, thus obtaining the initial set NLI of oracle-labeled instances (lines 4-5). Then, at each epoch (lines 6-15), $LAL\text{-}IGradV$ performs the following steps:

- The neural network model is trained using the labeled instances in LI and NLI (line 8). During the training process, every instance $x \in NLI$ is associated with its importance score r_x. The computation of the importance scores of the instance in NLI is performed using one of the techniques described in Section 3.2.2;
- A regressor R is trained on the set $\{\langle x_l, r_x\rangle \mid x_l$ is the representation in the latent space of $x\}$ where x is an instance in UI. Note that the choice of the regressor model actually used in this and the subsequent steps is orthogonal w.r.t. the proposed approach; however, it is essential that the chosen regression model must be trainable using a small set of labeled instances;
- NLI instances are added to LI (line 11);
- The regressor R is applied to the latent representation of the instances in UI so that, given an instance x, it predicts its importance score $\widehat{r_x}$ (line 12);

- The algorithm selects the top-k instances from UI ($topK$) having the highest predicted importance scores, and these instances in $topK$ are submitted to the oracle for labeling (lines 13-14). Finally, NLI is replaced with $topK$ (line 15).

Algorithm 23 *LAL-IGradV-VAE*

1: **Input:** LI: set of labeled instances, UI: set of unlabeled instances, DNN: deep neural network model, VAE: variational autoencoder model, R: importance score regressor, $epch$: maximum number of epochs, k: number of relevant instances to select
2: Train DNN on LI
3: Train VAE on all instances in $UI \cup LI$
4: $NLI \leftarrow$ Select k instances from UI uniformly at random
5: The oracle annotates the instances in NLI
6: **for** $i = 1 \dots epch$ **do**
7: **for** $x \in NLI$ **do**
8: Train DNN on $LI \cup NLI$ and compute importance score r_x
9: **end for**
10: Train R on the set of pairs $\{\langle x_l, r_x \rangle \mid x_l$ is the representation in the latent space of $x\}$
11: $LI \leftarrow LI \cup NLI$
12: Apply R to UI instances to predict importance scores ($\hat{r}_{x_t}$)
13: $topK \leftarrow$ Select top-k instances from UI by importance score $\hat{r}_{x_t}$
14: The oracle annotates the instances in $topK$
15: $NLI \leftarrow topK$
16: **end for**

As it can be noticed, with the exception of an additional preprocessing step through VAE, the *LAL-IGradV-VAE* technique is almost identical to *LAL-IGradV*. However, this small difference improves the performances both in terms of quality of results as well as in computational terms because the selector in *LAL-IGradV-VAE* works on data with a smaller size.

4.3 Experimental Evaluation

4.3.1 Data and baseline methods

For the evaluation of this proposal was used the well-known CIFAR-10 dataset which consists of 60000 instances, divided is two parts: 50000 instances corresponding to the training set, and the remaining 10000 instances corresponding to the test set, consequently, the training set has been divided into other two parts corresponding to the set of labeled instances (LI), and the set of unlabeled instances (UI). Also, at the time of labeling by the specialist of the relevant instances identified by the proposed technique, the dataset label is used to simulate the expert. Likewise, the CIFAR-100 [54] dataset was used and divided in the same manner as CIFAR-10, it differ in the number of target classes in the instances, with CIFAR-100 having 100 classes instead of 10.

The performance of *LAL-IGradV-VAE* was compared with *LAL-IGradV* and consequentially with *Random*, *LCS* and *LAL* methods, which were presented in Section 3.3.2.

4.3.2 Settings and assesment criteria

In the experimental evaluation, two neural network were used:

- *one for the testing of the proposed technique*: a Convolutional Neural Network (CNN) 2D layers, with 3 input channels, kernel size 3, stride size 3, padding size 1, ReLU activation function and max pool layers. The CNN module has on top a fully-connected network with an input layer of size 4096, one hidden layer with input size 4096 and output size 1024, another hidden layer with input size 1024 and output size 512, an output layer of size 10 (i.e., number of classes), and a dropout layers with probability 0.1. It was trained using cross entropy as loss function and Adam optimizer (with learning rate 1e-4 and weight decay 5e-4) on the experimental evaluation made on CIFAR-10, for the tests conducted with the CIFAR-100 dataset, a MobileNet [36] with the following archi-

tecture was used: it starts with a stem that consists of two convolutional layers. The first layer, BasicConv2d, takes input images with 3 channels and applies a 3x3 convolution with (32 * α) output channels (α was set to 1). The second layer, DepthSeperabelConv2d, uses depthwise separable convolution, which involves applying a 3x3 depthwise convolution followed by a 1x1 pointwise convolution. This layer transforms the (32 * α) input channels into (64 * α) output channels. Following the stem, there are four Conv layers. Each Conv layer performs downsampling and includes multiple depthwise separable convolutional layers. In Conv1, the first depthwise separable convolutional layer takes (64 * α) input channels and applies a 3x3 convolution with (128 * α) output channels, using a stride of 2 for downsampling. The second depthwise separable convolutional layer maintains the same number of input and output channels (128 * α) and applies a 3x3 convolution. Conv2 follows a similar pattern as Conv1. The first depthwise separable convolutional layer takes (128 * α) input channels and applies a 3x3 convolution with (256 * α) output channels, using a stride of 2 for downsampling. The second depthwise separable convolutional layer maintains the same number of input and output channels (256 * α) and applies a 3x3 convolution. Conv3 continues the downsampling process. The first depthwise separable convolutional layer takes (256 * α) input channels and applies a 3x3 convolution with (512 * α) output channels, using a stride of 2 for downsampling. Following this, there are several subsequent depthwise separable convolutional layers, each with (512 * α) input and output channels. Finally, Conv4 performs the last downsampling step. The first depthwise separable convolutional layer takes (512 * α) input channels and applies a 3x3 convolution with (1024 * α) output channels, using a stride of 2 for downsampling. The second depthwise separable convolutional layer maintains the same number of input and output channels (1024 * α) and applies a 3x3 convolution. After the Conv layers, there is a fully connected layer (FC) that takes the features from the previous

layers and maps them to the desired number of classes. The number of input features to this linear layer is (1024 * α), and the output size is determined by the $class_{num}$ parameter. Lastly, the adaptive average pooling layer (Avg) computes the average value for each channel across the spatial dimensions of the feature maps, resulting in a 1x1 output. Overall, the architecture follows a pattern of downsampling and increasing the number of channels in each layer, utilizing depthwise separable convolutions for efficient computation. For both the networks was used a number of epochs equal to 10 for both the initialization step of training and the training steps in the main loop;

- *one for the dimensionality reduction step*: this is a *VAE* where the encoder is equipped with a convolutional layer 2D with 3 input channels, kernel size 4, stride size 2, padding size 1, followed by a batch normalization layer 2D and other two layers one convolutional and one batch normalization, all with ReLU activation function. Instead the decoder has a Convolutional Transpose 2D layer on top with 3 input channels, kernel size 4, stride size 2, padding size 1, followed by a batch normalization layer 2D and other two layers one Convolutional Transpose 2D layer and one sigmoid layer. It was trained using MSE as loss function and Adam optimizer (with learning rate 3e-04 and weight decay 1e-5), a number of epochs equal to 10 for the training, that was used all the CIFAR-10/CIFAR-100 dataset.

Unless otherwise specified, the number k of instances to select from UI was set to 500 when was used the dataset CIFAR-10 and was set to 1000 for CIFAR-100; the size of LI was set to the 10% of the CIFAR-10 test set and the remaining samples become UI. As the regressor (R), it was used the Gradient Boosting Regressor model, with least absolute deviations (LAD) loss function and 200 estimators. To simulate the oracle for annotating the instances, it was resorted to the availability of class label information for the CIFAR-10/CIFAR-100 data: whenever an instance was used in the UI set, it was masked its

actual label during the learning process, and it was unveiled the label only if the instance was selected within the *topK* set of instances to annotate. A batch of tests was carried out for each technique as follows:

- 10 epochs of neural network training;
- 10 epochs of *VAE* training;
- 10 training epochs (*epch*) of the iterative active learning process in question were used for each technique, at each step 500/1000 ($|topK|$) units of elements are taken and chosen for each epoch of active training;
- During the tests, the size of the latent space representation obtained by the *VAE* was varied in order to observe and establish the opportune size of it;
- In order to simulate the human annotator, the annotated dataset was taken and they were "annotated" after the selection of the instances using the label already provided in the dataset.

To assess the performance of the methods, it was considered the accuracy of the classifier during the various training batches, in absolute terms as well as in terms of percentage increase w.r.t. the early accuracy of the classifier itself or the accuracy of a reference method. More precisely, was computed:

- the *accuracy at the initial step of training* of *LAL-IGradV-VAE*, denoted as $A^{(0)}$, and the accuracy at the end of the active learning process, denoted as A;
- the *percentage increase in the accuracy* of *LAL-IGradV-VAE*, which is defined as $100 \times (A - A^{(0)})/A^{(0)}$;
- the *percentage increase in the accuracy* of *LAL-IGradV-VAE* w.r.t.:

  Rnd which is defined as $\%_{Rnd} = 100(A - A_{Rnd})/A_{Rnd}$;

  LCS, which is defined as $\%_{LCS} = 100(A - A_{LCS})/A_{LCS}$;

LAL-IGradV which is defined as $\%_{lal} = 100(A - A_{lal})/A_{lal}$;

where A_{Rnd}, A_{LCS} and A_{lal} denote the accuracy at the end of the active learning process for *Rnd*, *LCS* and *LAL-IGradV*.

4.3.3 Results

In the following section are described the tests performed to assess the effectiveness of the techniques on the CIFAR-10 and CIFAR-100 datasets.

4.3.3.1 Experiments on CIFAR-10

Table 4.1 reports on the performance of our *LAL-IGradV-VAE DS* variant only, because is the best performed with the previous technique therefore was used the new variant of the algorithm with *DS* importance scoring strategy and 500 samples to select at each *AL* iteration, varying percentages of the set of unlabeled instances (*UI*); as was done for *LAL-IGradV*.

Table 4.1: Performance of the proposed methods: initial and final accuracy, percentage increase w.r.t. *Rnd*, *LAL-IGradV* (*DS*) and *LCS*, and active learning time (sec) averaged over the epochs, for various percentage values of unlabeled instances.

	$A^{(0)}$	*LAL-IGradV-VAE* (*DS*)				
		A	$\%_{\mathsf{Rnd}}$	$\%_{\mathsf{LCS}}$	$\%_{\mathsf{LAL\text{-}IGradV(DS)}}$	time
10%	0.793	0.822	1.17	-0.69	-1,24	177
20%	0.783	0.813	1.13	0.01	-0.63	167
30%	0.784	0.832	0.33	-1.11	-1.72	165
40%	0.763	0.811	2.69	-0.13	-0.26	167
50%	0.733	0.803	8.18	1.27	-1.40	159
60%	0.728	0.777	2.54	-0.39	-3.27	158
70%	0.708	0.789	3.31	-0.44	-2.84	150
80%	0.640	0.722	8.75	5.55	3.94	135
90%	0.570	0.665	9.30	6.05	5.13	125

Looking at the table, several remarks stand out. First of all, it is not surprising to notice that the accuracy values (i.e., columns corresponding to A and $A^{(0)}$)

tend to decrease as the percentage of unlabeled instances gets higher, since the *LAL-IGradV-VAE* method is forced to handle progressively reduced sets of labeled instances on its initial training. More interestingly, the percentage increase of *LAL-IGradV-VAE* w.r.t. *Rnd* is always positive, instead this is not true w.r.t. *LCS* and *LAL-IGradV* when there are more labeled samples (up to 9.30% against *Rnd*, up to 6.05% against *LCS* and up to 5.13% against *LAL-IGradV* (*DS*)) anyway it tends to improve with higher percentages of unlabeled instances. Remarkable is the fact that *LAL-IGradV-VAE* obtains better time performances[1] as well as an higher accuracy when the amount of the unlabeled instances increase. The proposed *LAL-IGradV-VAE* has shown that a learning-to-active-learn by instance importance based gradient variation improves significantly upon not only a random baseline but also an uncertainty sampling approach like *LCS* and more in particular w.r.t. the original version of the algorithm *LAL-IGradV*. *LAL-IGradV-VAE* increases the accuracy at the initial training step, and tend to improve with higher percentages of unlabeled instances when there are a less quantity of unlabeled instances.

Concerning the training of the *VAE* model, the impact of the latent space representation size was analyzed by running different experiments with values in $\{2048, 1024, 512, 256, 128, 64\}$. The results of these experiments are presented in Figure 4.3. It is worth noticing that reducing the latent space representation size allows to yield better results.

More precisely, the best results were obtained with the latent space representation size set to 64, where an about 5% improvement of the performances was reached. This means that the reduction of the size of the instances, presented to the instance selector, helps it to learn in a better way the association with its score despite the presence of a little amount of data.

[1]Experiments were carried out on an Intel Core i7 CPU @2.90GHz, 32GB RAM, with NVIDIA GeForce RTX 2070 Super GPU

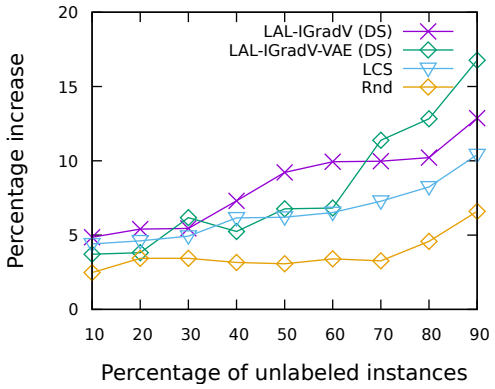


Figure 4.2: Percentage increase of accuracy for the various active learning methods, with varying percentage of unlabeled instances, and number of selected instances (k) equal to 500.

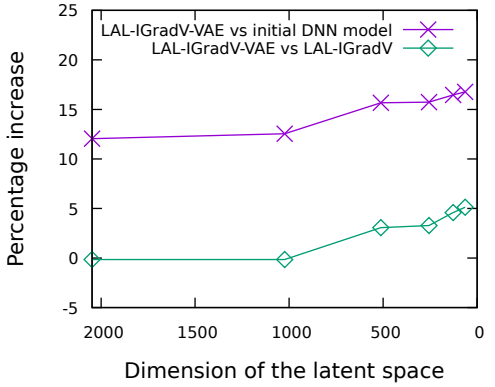


Figure 4.3: Percentage increase of accuracy varying the size of the latent space representation of the instances.

4.3.3.2 Experiments on CIFAR-100

As in the previous section here there is a table 4.2 that explains the performance obtained with the dataset CIFAR-100 of *LAL-IGradV DS* w.r.t.

LCS, *Random* and *LAL* (sec. 2.3.11.3) and 1000 samples to select at each *AL* iteration, varying percentages of the set of unlabeled instances (UI); as was done in the previous experiments.

Table 4.2: Performance of the proposed methods: initial and final accuracy, percentage increase w.r.t. *Random*, *LAL-IGradV* (*DS*), *LCS* and *LAL*, and active learning time (sec) averaged over the epochs, for various percentage values of unlabeled instances.

	$A^{(0)}$	*LAL-IGradV-VAE* (*DS*)					
		A	$\%_{\mathsf{Rnd}}$	$\%_{\mathsf{LCS}}$	$\%_{\mathsf{LAL}}$	$\%_{\mathit{LAL\text{-}IGradV(DS)}}$	time
10%	0.621	0.645	2.11	-0.11	2.01	0,27	455
20%	0.607	0.631	2.37	0.31	2.14	0.37	501
30%	0.577	0.607	1.23	-0.17	1.91	-0.72	487
40%	0.503	0.549	3.77	1.17	0.13	-0.78	499
50%	0.470	0.517	5.28	1.35	3.27	1.11	481
60%	0.411	0.488	5.97	2.28	4.19	2.27	470
70%	0.391	0.457	6.19	3.71	4.11	1.91	461
80%	0.327	0.398	7.73	6.57	6.77	2.04	407
90%	0.301	0.453	9.97	7.01	7.51	3.17	375

Looking at the table, it is evident that the accuracy trends, as observed in the experiments conducted on the CIFAR-10 dataset, have remained largely unchanged. However, notable attention should be given to the fact that, in this case, the technique *LAL-IGradV-VAE* (*DS*) incorporating the VAE demonstrates superior performance and higher gains compared to other techniques. This can be attributed to the fact that as the model size increases due to the complexity of the CIFAR-100 dataset, the technique *LAL-IGradV-VAE* (*DS*) is better able to generalize and effectively identify the most suitable instances for labeling. It is worth noting, however, that the *LAL* technique achieves unexceptional performance. This is due to the fact that when using the model parameters along with features of the instances (in the experiments, those provided by the VAE), the ML model, which should learn how to estimate the quality score of the instances, fails to do so adequately due to the explosion in the quantity of input data provided, therefore, the proposed techniques are

effective for AL tasks and perform exceptionally well with neural networks. They leverage the network's state to understand how a given data point can be influential to it. Moreover, unlike other techniques such as LAL, they do not suffer from parameter explosion, which hampers their performance with complex models feel like neural networks.

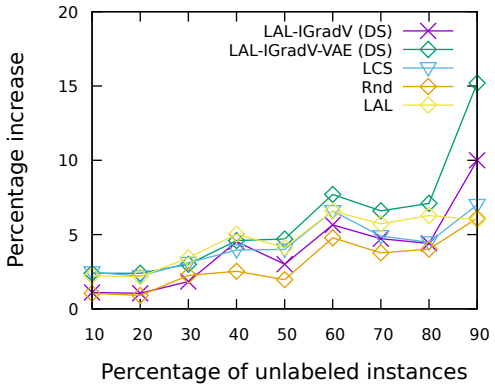


Figure 4.4: Percentage increase of accuracy for the various active learning methods, with varying percentage of unlabeled instances, and number of selected instances (k) equal to 1000.

In the figure 4.4, it is possible to observe how the improvement of the model in terms of accuracy varies with the amount of labeled data. As seen in the previous experiments, the proposed techniques perform better as the initially labeled data decreases, especially the variant with the VAE. The dimension of the latent space used is set to 64, which performed well in the experiments with CIFAR-10 dataset.

4.4 Limitations and possible enhancements

LAL-IGradV-VAE has shown satisfactory results in terms of a significantly positive change in the accuracy of the classifier (also better than *LAL-IGradV*), and this performance improvement is emphasized for increasingly

large sets of unlabeled instances, which makes *LAL-IGradV-VAE* useful in these practical and more challenging scenarios. Nonetheless, several aspects of this approach need to be further investigated and enhanced. This importance scoring strategies might be improved in different ways. The importance of an instance could be measured not only in terms of its own contribution to the model change but also w.r.t. other instances, including both labeled and unlabeled ones, according to some instance locality principle. In this regard, it would be worthy to consider the data instance features, so as to identify an instance's neighborhood to evaluate in each step of importance scoring. Features of the regressor (meta-features) could also be incorporated into the instance selection steps, although this would require to identify those features that are suited to a specific type of regressor. From an efficiency viewpoint, it would also be important to define theoretical properties on the gradient direction change in function of the number of top-k instances to be annotated and/or the size of the batch of unlabeled instances available for the active learning process, in order to prune the candidates thus speeding up the active learning of the model. Besides enhancements on the importance scoring and top-k selection policies, different choices might be investigated about the architecture and setting of the deep neural network classifier. Our experimental evaluation focused on image data, for which CNN models are known to be effective; clearly, the choice of the neural network architecture might be dependent on the type of the input data and on the target learning task. *LAL-IGradV-VAE* exhibited quite different behavior w.r.t. not only random instance selection, but also compared to an uncertainty sampling method like *LCS*, *LAL*, and *LAL-IGradV*.

This learning-to-active-learn approach proposed has key novelty is twofold: the integration of a regression-based meta-learning approach within a maximum model-change framework, and the definition of policies for scoring the instance importance based on the amount of change in the learning gradient of a deep neural network model.

Chapter 5

Conclusions

The main contribution of this thesis is an active learning method, dubbed *LAL-IGradV-VAE*, that was developed to improve neural network training by exploiting a central point of neural network training, i.e. the gradients. In fact, a careful study was done to determine the right way to extrapolate a learning score for a sample using the learning gradients. In this technique the gradients, within the NN, were linearized and used as a multidimensional vector. These vectors were used to calculate the change in the gradient of a sample used during training through the cosine distance, after which all the distances of the samples, used in an AL training phase, were calculated and then the average of them corresponding to the relevance score. These scores are used to train a Gradient Boosting regressor or a Random Forest classifier, depending on the policy adopted, and then this model is used to estimate the score of unlabeled samples. The experimental evaluation presented in this thesis showed that the proposed technique is very fast and also, at the end of training phase, it is able to provide a model with 5% higher accuracy than other techniques such as *Random*, *LCS* and *LAL*. Also, it was identified that using the original sample to train the selector model is too costly for two reasons: (i) due to the lack of samples, the selector is not able to best learn the rules for estimating the relevance score, and (ii) the data is too large which results in a long training and prediction time. For these reasons, a VAE was introduced as a preprocessing step to learn a compressed representation of

the input data samples before applying the *LAL-IGradV-VAE* method. More precisely, the latent space representations provided by the VAE was used to reduce the sample size and obtain a region in which it is located, thus the computation time was reduced and the accuracy of the model was increased up to 9.97% compared with other techniques such as *Random*, *LCS* and *LAL*.

5.1 Future works

The proposed technique actually obtains very good results and can be extended and used in a real-world environment, in fact, one only needs to develop a new VAE for the specific case and start the process. For example, it can be applied in the medical field, where it can significantly reduce the cost of data labeling which, because only a few specialized people who know the application domain can do it, can be very expensive, not to mention the fact that it would also save a lot of training time. In addition, the proposed work can be further improved, for example, by using a set of spatial features to help the selector model choose the best samples or by identifying the slowest steps and trying to speed them up further.

Bibliography

[1] S. I. Ali and W. Shahzad. A feature subset selection method based on symmetric uncertainty and ant colony optimization. In *2012 International Conference on Emerging Technologies*, pages 1–6, 2012.

[2] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, Apr. 1988.

[3] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[4] P. Baldi. Autoencoders, unsupervised learning, and deep architectures. In I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver, editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 37–49, Bellevue, Washington, USA, 02 Jul 2012. PMLR.

[5] Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *J. Mach. Learn. Res.*, 5:255–291, 2004.

[6] S. Begum, R. Sarkar, D. Chakraborty, S. Sen, and U. Maulik. Application of active learning in dna microarray data for cancerous gene identification. *Expert Systems with Applications*, 177:114914, 2021.

[7] A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[8] J. Bernard, M. Zeppelzauer, M. Lehmann, M. Müller, and M. Sedlmair. Towards user-centered active learning algorithms. *Computer Graphics Forum*, 37(3):121–132, 2018.

[9] M. Bilgic, L. Mihalkova, and L. Getoor. Active learning for networked data. In *ICML*, pages 79–86, 2010.

[10] D. Bouneffouf. Exponentiated gradient exploration for active learning, 2014.

[11] D. Bouneffouf, R. Laroche, T. Urvoy, R. Feraud, and R. Allesiardo. Contextual bandit for active learning: Active thompson sampling. In C. K. Loo, K. S. Yap, K. W. Wong, A. Teoh, and K. Huang, editors, *Neural Information Processing*, pages 405–412, Cham, 2014. Springer International Publishing.

[12] Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders, 2015.

[13] E. Y. Chang, K. Goh, S. Tong, and C. wei Chang. Support vector machine concept-dependent active learning for image retrieval. *IEEE Transactions on Multimedia*, 2005.

[14] C. Chao, M. Cakmak, and A. L. Thomaz. Transparent active learning for robots. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 317–324, 2010.

[15] S. Chen and R. Rosenfeld. A survey of smoothing techniques for me models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50, 2000.

[16] W. Chu, M. Zinkevich, L. Li, A. Thomas, and B. L. Tseng. Unbiased online active learning in data streams. In C. Apté, J. Ghosh, and P. Smyth, editors, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 195–203. ACM, 2011.

[17] G. Contardo, L. Denoyer, and T. Artieres. A meta-learning approach to one-step active learning, 2017.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[19] C. Dima, M. Hebert, and A. Stentz. Enabling learning from large datasets: applying active learning to mobile robotics. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 1, pages 108–114 Vol.1, 2004.

[20] Y. Ding, J. D. Martín-Guerrero, M. Sanz, R. Magdalena-Benedicto, X. Chen, and E. Solano. Retrieving quantum information with active learning. *Phys. Rev. Lett.*, 124:140504, Apr 2020.

[21] R. Du, S. Wang, Q. Wu, and X. He. Learn concepts in multiple-instance learning with diverse density framework using supervised mean shift. In *2010 International Conference on Digital Image Computing: Techniques and Applications*, pages 643–648, 2010.

[22] S. Ebert, M. Fritz, and B. Schiele. RALF: A reinforced active learning formulation for object class recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3626–3633, 2012.

[23] L. Ein-Dor, A. Halfon, A. Gera, E. Shnarch, L. Dankin, L. Choshen, M. Danilevsky, R. Aharonov, Y. Katz, and N. Slonim. Active Learning for BERT: An Empirical Study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7949–7962, Online, Nov. 2020. Association for Computational Linguistics.

[24] M. Fang, Y. Li, and T. Cohn. Learning how to active learn: A deep reinforcement learning approach, 2017.

[25] S. Flesca, D. Mandaglio, F. Scala, and A. Tagarelli. Learning to active learn by gradient variation based on instance importance. In *2022 26th*

International Conference on Pattern Recognition (ICPR), pages 2224–2230, 2022.

[26] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning, 2015.

[27] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[28] C. L. Giles, K. D. Bollacker, and S. Lawrence. Citeseer: an automatic citation indexing system. In *INTERNATIONAL CONFERENCE ON DIGITAL LIBRARIES*, pages 89–98. ACM Press, 1998.

[29] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[30] Y. Guo and R. Greiner. Optimistic active-learning using mutual information. In M. M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 823–829, 2007.

[31] L. Gyongyosi and S. Imre. A survey on quantum computing technology. *Computer Science Review*, 31:51–71, 2019.

[32] F. Herrera, S. Ventura, R. Bello, C. Cornelis, A. Zafra, D. Sánchez-Tarragó, and S. Vluymans. *Multiple Instance Learning*, pages 17–33. Springer International Publishing, Cham, 2016.

[33] J. Hidary. *Quantum Computing: An Applied Approach*. Springer, 2019.

[34] S. C. H. Hoi, R. Jin, and M. R. Lyu. Large-scale text categorization by batch mode active learning. In L. Carr, D. D. Roure, A. Iyengar, C. A. Goble, and M. Dahlin, editors, *WWW*, pages 633–642. ACM, 2006.

[35] M. Hopwood, P. Pho, and A. V. Mantzaris. Exploring a link between network topology and active learning. In *2021 Twelfth International*

Conference on Ubiquitous and Future Networks (ICUFN), pages 81–86, 2021.

[36] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[37] W. Hsu and H. Lin. Active learning by learning. In *Proc. of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2659–2665, 2015.

[38] H. O. Ilhan and M. F. Amasyali. Active learning as a way of increasing accuracy. *International Journal of Computer Theory and Engineering*, 6:460–465, 2014.

[39] F. Jager, G. B. Moody, A. Taddei, G. Antoli, M. Emdin, A. Smrdel, B. Glavi , C. Marchesi, and R. G. Mark. A long-term ST database for development and evaluation of ischemia detectors. In *Computers in Cardiology 1998*, pages 301–304, Piscataway, 1998. IEEE Society Press.

[40] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379, 2009.

[41] G. Jun and J. Ghosh. An efficient active learning algorithm with knowledge transfer for hyperspectral data analysis. In *IGARSS 2008 - 2008 IEEE International Geoscience and Remote Sensing Symposium*, volume 1, pages I–52–I–55, 2008.

[42] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *J. Artif. Intell. Res.*, 4:237–285, 1996.

[43] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with gaussian processes for object categorization. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007.

[44] A. Karegowda, A. Manjunath, and J. M.A. Comparative study of attribute selection using gain ratio and correlation based feature selection. *Int J Inf Technol Knowl Manage*, 2, 01 2010.

[45] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017.

[46] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks, 2018.

[47] R. King, K. Whelan, F. Jones, P. Reiser, C. Bryant, S. Muggleton, D. Kell, and S. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature: international weekly journal of science*, 427(6971):247–252, Jan. 2004.

[48] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2013.

[49] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132, 1995.

[50] S. Koldijk, M. Sappelli, S. Verberne, M. A. Neerincx, and W. Kraaij. The swell knowledge work dataset for stress and user modeling research. *Proceedings of the 16th International Conference on Multimodal Interaction*, 2014.

[51] K. Konyushkova, R. Sznitman, and P. Fua. Learning active learning from data, 2017.

[52] K. Konyushkova, R. Sznitman, and P. Fua. Learning active learning from data. In *Proc. of the Annual Conference on Neural Information Processing Systems*, pages 4225–4235, 2017.

[53] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 (canadian institute for advanced research).

[54] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-100 (canadian institute for advanced research).

[55] J. Li, J. M. Bioucas-Dias, and A. Plaza. Hyperspectral image segmentation using a new bayesian approach with active learning. *IEEE Transactions on Geoscience and Remote Sensing*, 49(10):3947–3960, 2011.

[56] X. Li and Y. Guo. Adaptive active learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[57] Q. Liu, Y. Zhu, Z. Liu, Y. Zhang, and S. Wu. Deep active learning for text classification with diverse interpretations. In *Proceedings of the 30th ACM International Conference on Information amp; Knowledge Management*, CIKM '21, page 3263–3267, New York, NY, USA, 2021. Association for Computing Machinery.

[58] Y. Liu. Active learning with support vector machine applied to gene expression data for cancer classification. *Journal of Chemical Information and Modeling*, 44(6):1936–1941, 2004.

[59] T. Luo, K. Kramer, D. B. Goldgof, L. O. Hall, S. Samson, A. Remsen, and T. Hopkins. Active learning to recognize multiple types of plankton. *J. Mach. Learn. Res.*, 6:589–613, 2005.

[60] O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1997.

[61] C. Mayer and R. Timofte. Adversarial sampling for active learning, 2018.

[62] A. K. Mccallum. Employing em in pool-based active learning for text classification. In *In Proceedings of the 15th International Conference on Machine Learning*, pages 350–358. Morgan Kaufmann, 1998.

[63] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[64] M. Nagy and S. G. Akl. Quantum computation and quantum information. *INTERNATIONAL JOURNAL OF PARALLEL, EMERGENT AND DISTRIBUTED SYSTEMS*, 21:1–59, 2006.

[65] C. Nielsen and M. Okoniewski. Gan data augmentation through active learning inspired sample acquisition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[66] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web, 1999.

[67] H. Papadopoulos, V. Vovk, and A. Gammerman. Conformal prediction with neural networks. In *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007)*, volume 2, pages 388–395, 2007.

[68] E. Pasolli and F. Melgani. Active learning methods for electrocardiographic signal classification. *IEEE Transactions on Information Technology in Biomedicine*, 14(6):1405–1416, 2010.

[69] N. Pillai, E. Raff, F. Ferraro, and C. Matuszek. Sampling approach matters: Active learning for robotic language acquisition. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 5191–5200, 2020.

[70] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, and H.-J. Zhang. Two-dimensional active learning for image classification. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[71] L. Qian, H. Hui, Y. Hu, G. Zhou, and Q. Zhu. Bilingual active learning for relation classification via pseudo parallel corpora. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[72] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[73] A. Ragav and G. K. Gudur. Bayesian active learning for wearable stress and affect detection, 2020.

[74] S. Rajan, J. Ghosh, and M. Crawford. An active learning approach to hyperspectral data classification. *Geoscience and Remote Sensing, IEEE Transactions on*, 46:1231 – 1242, 05 2008.

[75] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang. A survey of deep active learning.

[76] Y. Ren, C. C. Aggarwal, and J. Zhang. Activeiter: Meta diagram based active learning in social networks alignment. *IEEE Transactions on Knowledge and Data Engineering*, 33(5):1848–1860, 2021.

[77] N. Roy and A. Mccallum. Toward optimal active learning through sampling estimation of error reduction. In *In Proc. 18th International Conf. on Machine Learning*, pages 441–448. Morgan Kaufmann, 2001.

[78] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *ICML*, 2001.

[79] O. Rudovic, M. Zhang, B. Schuller, and R. Picard. Multi-modal active learning from human data: A deep reinforcement learning approach. In *2019 International Conference on Multimodal Interaction*, ICMI '19, page 6–15, New York, NY, USA, 2019. Association for Computing Machinery.

[80] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. 1986.

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

[82] B. Settles and M. Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079, Honolulu, Hawaii, Oct. 2008. Association for Computational Linguistics.

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

[84] B. Settles, M. Craven, and S. Ray. Multiple-instance active learning. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1289–1296. Curran Associates, Inc., 2007.

[85] B. Settles, M. Craven, and S. Ray. Multiple-instance active learning. In *Proc. of the Twenty-First Annual Conference on Neural Information Processing Systems*, pages 1289–1296, 2007.

[86] G. Shafer and V. Vovk. A tutorial on conformal prediction. 2007. cite arxiv:0706.3188Comment: 58 pages, 9 figures.

[87] F. Shahmohammadi, A. Hosseini, C. E. King, and M. Sarrafzadeh. Smartwatch based activity recognition using active learning. In *2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, pages 321–329, 2017.

[88] Y. Shen, H. Yun, Z. C. Lipton, Y. Kronrod, and A. Anandkumar. Deep active learning for named entity recognition, 2018.

[89] M. P. Singh and R. Tiwari. Article:correlation-based attribute selection using genetic algorithm. *International Journal of Computer Applications*, 4(8):28–34, August 2010. Published By Foundation of Computer Science.

[90] R. Singh, N. Palmer, D. Gifford, B. Berger, and Z. Bar-Joseph. Active learning for sampling in time-series experiments with application to gene expression analysis. In *Proceedings of the 22nd International Conference*

on Machine Learning, ICML '05, page 832–839, New York, NY, USA, 2005. Association for Computing Machinery.

[91] A. T. Stentz, C. Dima, C. Wellington, H. Herman, and D. Stager. A system for semi-autonomous tractor operations. *Autonomous Robots*, 13(1):87 – 103, July 2002.

[92] F. C. Steward. Symposia of the society of experimental biology. number xi. the biological action of growth substances. *Journal of the American Chemical Society*, 80(1):253–254, 1958.

[93] L. M. Tiwari, S. Agrawal, S. Kapoor, and A. Chauhan. Entropy as a measure of uncertainty in queueing system. In *2011 National Postgraduate Conference*, pages 1–4. IEEE;, 2011.

[94] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, mar 2002.

[95] G. Wang, C. Zhang, Y. Liu, H. Yang, D. Fu, H. Wang, and P. Zhang. A global and updatable ecg beat classification system based on recurrent neural networks and active learning. *Inf. Sci.*, 501:523–542, 2019.

[96] Y. Xia and Y. Xie. A novel wearable electrocardiogram classification system using convolutional neural networks and active learning. *IEEE Access*, 7:7989–8001, 2019.

[97] J. Xu, L. Song, J. Y. Xu, G. J. Pottie, and M. van der Schaar. Personalized active learning for activity classification using wireless wearable sensors. *IEEE Journal of Selected Topics in Signal Processing*, 10(5):865–876, 2016.

[98] Z. Xu, R. Akella, and Y. Zhang. Incorporating diversity and density in active learning for relevance feedback. pages 246–257, 04 2007.

[99] J.-X. Zhang, D.-B. Chen, Q. Dong, and Z.-D. Zhao. Identifying a set of influential spreaders in complex networks. *CoRR*, abs/1602.00070, 2016.

[100] Y. Zhang, M. Lease, and B. C. Wallace. Active discriminative text representation learning, 2016.

[101] J. Zhu, H. Wang, E. Hovy, and M. Ma. Confidence-based stopping criteria for active learning for data annotation. *ACM Trans. Speech Lang. Process.*, 6(3), apr 2010.

[102] J.-J. Zhu and J. Bento. Generative adversarial active learning, 2017.

[103] X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 58–65, 2003.