

A Report  
On  
  
Active Learning  
With a Java Implementation

By  
Yash Gupta 2019A7PS1138P

Prepared in Partial Fulfillment of the Course  
BITS F464 Machine Learning

Birla Institute of Technology and Science (BITS), Pilani

December, 2022

# Table of Contents

<b>Introduction</b>	<b>3</b>
<b>1. An Overview of Active Learning</b>	<b>5</b>
1.1 What is Active Learning?	5
1.2 Motivation for Active Learning	6
1.3 Applications of Active Learning	7
1.4 Types of Active Learning	9
<b>2. The Problem: Handwritten Digit Recognition</b>	<b>11</b>
2.1 The Problem Statement	11
2.2 The Dataset	12
2.3 Data Preprocessing	12
2.4 The Underlying Model	14
<b>3. Demonstration of Active Learning</b>	<b>15</b>
3.1 The Broad Implementation Plan	15
3.2 Random Selection	16
3.3 Uncertainty Sampling Techniques	17
3.4 Query-by-Committee Techniques	19
3.5 Pool-Based Techniques: Results	20
3.6 Stream Based Techniques: Results	22
<b>4. Version Space</b>	<b>23</b>
4.1 The Concept of a Version Space	23
4.2 Greedy Shrinkage of Version Space	24
4.3 Discussion of Results	26
<b>5. Clustering</b>	<b>28</b>
5.1 Motivation	28
5.2 Methodology and Implementation	28
5.3 Discussion of Results	30
<b>Conclusion and Future Work</b>	<b>32</b>
<b>References</b>	<b>33</b>
<b>Dataset Used</b>	<b>34</b>
<b>Appendix: Source Code Files</b>	<b>35</b>

# Introduction

“Data is the oil of the 21st century, and analytics is the combustion engine [1].” Truer words couldn’t have been said considering the ever-astonishing rate of growth of the data science and analytics industry. However, getting the best out of the data at our disposal is not as easy as it appears at the first glance - much of it is unlabeled and of little use without thorough processing. Indeed, the race to build useful datasets has created a billion dollar industry. Grand View Research estimates the data labeling market revenues to grow to USD 38 bn by 2028. [2]

The traditional strategy in the corporate world - especially the computer vision industry - to get data useful for training ML models has been to have an in-house labeling team to “get the labeling job done” manually. The approach, however, is evidently expensive (human employees are almost always more expensive than an automated solution), subject to biases (thereby warranting the even costlier workforce), and certainly not something easy to scale. To address these problems and more, a significant portion of research in machine learning today has been directed towards strategies that require less data (at least less labeled data), use data more meaningfully, or that create synthetic data for training.

Such strategies, if developed sufficiently, can lead to innovative solutions in fields hitherto little explored - visual reasoning AI to crack IQ tests, for instance [3]. A feature commonly found in such fields is the need for highly specialized labeled data, which can be very difficult to procure. For example, for the verbal reasoning task, an approach could require examples of encoded examples along with answers. It almost always boils down to a human annotator sitting and manually producing such datasets, which can get laborious, expensive, and of course, limited in productivity.

The present report surveys one of the key strategies to circumvent this dearth of annotated examples: active learning. This growing area of research realizes cost-efficiency benefits by allowing the machine to select, or ask for, the examples it deems the best to learn from. In particular, we shall review the pool-based active

learning techniques and demonstrate their benefits in terms of gains in predictive performance. It will be followed by a brief demonstration of how such implementations can be rather simply transformed into stream based learning for low-storage capacity and high-velocity environments.

The report shall also briefly cover the concept of version space and demonstrate some key ideas based on it on a small fraction of the data. Finally, clustering is presented as another promising technique to overcome the lack of labeled examples.

First, however, let us briefly cover the basic concepts of active learning and how and why it matters to the machine learning practitioners today.

# 1. An Overview of Active Learning

The section briefly describes what is meant by the term “active learning” and how it is different from the run-of-the-mill machine learning techniques (technically the passive learning approaches). The motivation for using active learning shall be discussed and some of its prominent applications will be touched upon. Hence, we briefly state the three main types of active learning approaches in practice today. Let us begin with the basic idea of active learning.

## 1.1 What is Active Learning?

The key idea behind “active learning” is that a machine learning algorithm, if given the liberty to choose the examples from which it learns, can achieve a better performance (in terms of accuracy, for example). An active learner may pose *queries*, such as unlabeled instances to be labeled by an *oracle*, which might be a human annotator. [4]

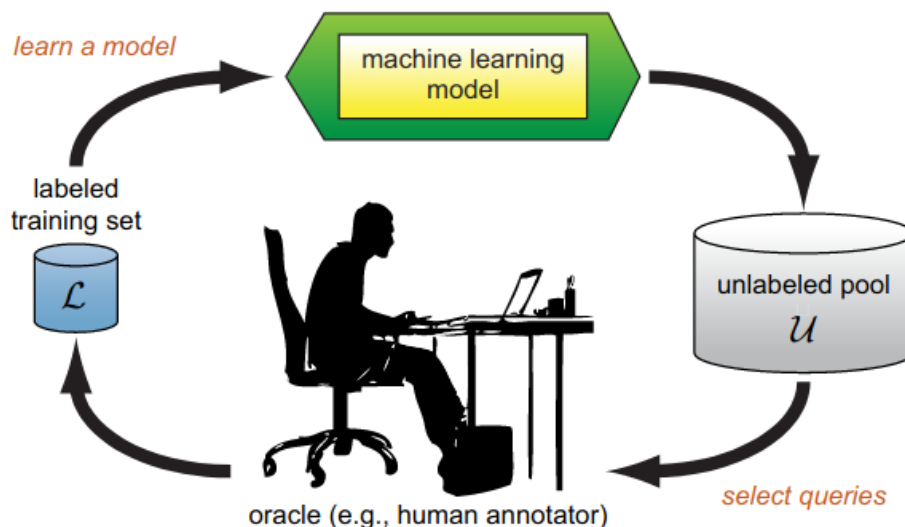


Figure 1.1: The pool-based active learning cycle.

Image Credit: Please refer [4]

A typical example of active learning would be a pool-based learning approach presented in Figure 1.1. As can be clearly seen, an active learning model is

different from a passive learning model in that it not only learns but it also “asks”, thereby learning from what it deems is required the most.

A general schema for the active learning approach can be presented as follows [5]. If we are considering a potential query,  $q$ , we need to evaluate the loss of the subsequent model,  $M'$  - obtained by updating the original model  $M$  with the query  $q$  and the response  $x$ . If we estimate the expected model loss as  $Loss(q) = E[Loss(M')]$ , then we arrive at the algorithm:

```
for i := 1 to totalQueries
  for each q in potentialQueries
    evaluate Loss(q)
  endfor
  ask query q for which Loss(q) is the lowest
  update model M with query q and response x
endfor
return model M
```

## 1.2 Motivation for Active Learning

In most ML and statistical tasks, gathering data is time consuming and costly; sometimes, it is prohibitively costly - for the visual reasoning example, for example, in the introduction. Therefore, it only makes sense to search for ways that can minimize the number of data instances required.

Active learning reduces the number of labeled instances required to achieve a given level of accuracy in the majority of reported results [4]. The essence of active learning is to maximize the utilization of the training samples [6]. Tech and finance giants, such as JP Morgan Chase & Co., fashion it as their go-to “learn more from less data” strategy [7].

To realize the potential of active learning, consider the following analogy of a teacher and a student. In the passive learning approach, a student learns by listening to the teacher’s lecture. In active learning, the teacher describes the concepts, the student asks questions, and the teacher spends more time explaining

the concepts that are difficult to understand for the student. The student and the teacher, therefore, interact and collaborate more effectively in the latter learning process.

JPMC actively uses ML development using active learning, where the annotator and modeler interact and collaborate. An annotator provides a small labeled dataset. The modeling team builds the model and generates input on what to label next. Within a few iterations, teams can build refined requirements, a labeled gold training set, active learner, and working machine learning model. They not only employ traditional uncertainty-, disagreement-, and information density-based sampling but also a business value based sampling, where data points with higher business value are focused on. Therefore, an active learning approach may even provide opportunities for incorporation of suggestions from domain experts for a particular financial application. [7]

### 1.3 Applications of Active Learning

Active learning, owing to its proven advantages over traditional ML techniques, has been gaining popularity among the ML practitioners. Some of the prominent applications of active learning include

- **Image Restoration:** Photographs can degrade because of several reasons - lossy image compression, noisy transfer of images over the internet servers, and the like - and thus require to be restored (see Figure 1.2).

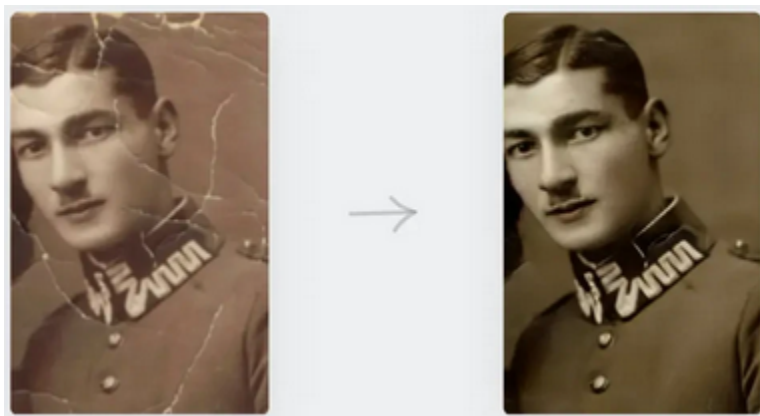


Figure 1.2: Image Restoration  
Image Credit: *Please refer [8]*

Unlike the conventional models which perform well for a particular fixed level of difficulty of restoration, it is possible to perform optimally on all difficulty levels with active learning. The model relies on a feedback mechanism, where at each epoch, the model guides its own learning toward the right proportion of subtasks per difficulty level. [8]

- **Medical Domain:** A versatile active learning workflow for optimization of genetic and metabolic networks has been proposed, maximizing the biological objective function, such as improvement of the protein production yield. [9]
- **Autonomous Driving:** To achieve high accuracy and performance expectations of the autonomous driving applications, vision-focussed deep learning models require large amounts of training data. But selecting the “right” instances capturing all possible scenarios and edge cases that can be confusing (see Figure 1.3) with active learning with active learning can increase the scalability, cost savings, and performance, as demonstrated by researchers at NVIDIA [9].



Figure 1.3: A potentially confusing instance for the autonomous driving model.  
Image Credit: *Please refer [9].*



- **Image Classification:** The Cost-Effective Active Learning for Deep Image Classification, or the CEAL, model is a novel attempt to build a competitive classifier with optimal feature representation via a limited amount of training instances in an incremental learning manner. Deep CNNs have been incorporated into active learning. Low confidence samples are picked up for manual annotation, while very high confidence samples are auto-annotated. [10]
- **Neural machine translation:** Active learning has been used for translation of one language to another. A curriculum learning framework has been developed, wherein the model is trained with easy samples first and then with progressively harder examples. Active learning based techniques have been employed to select samples for each competence level of the model. [8]

## 1.4 Types of Active Learning

Keeping the various niche flavors of active learning aside, there are three major variants of active learning depicted in Figure 1.4.

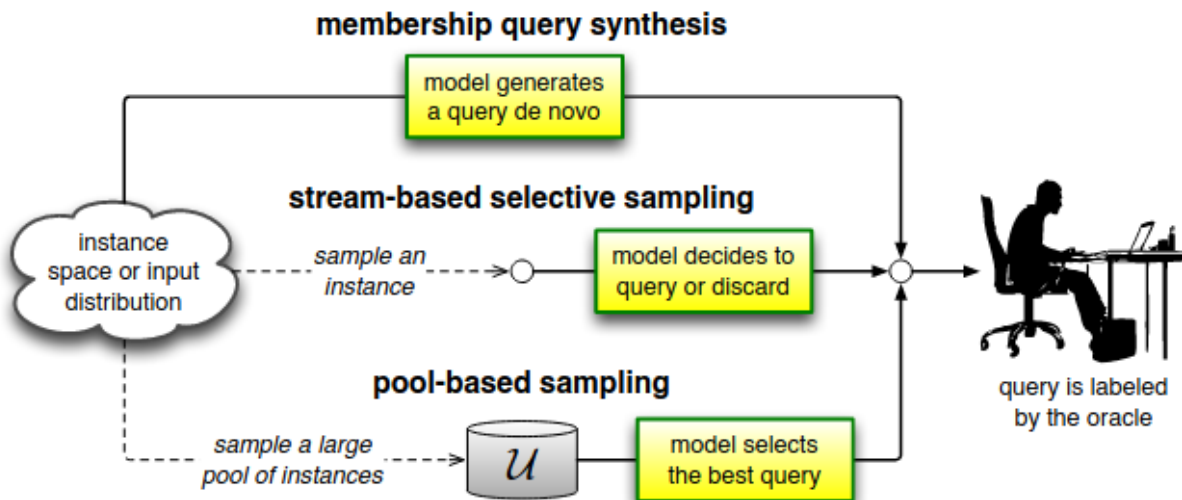


Figure 1.4: Major types of active learning.

Image Credit: Please refer [4].

### ***Membership Query Synthesis***

It involves the generation of synthetic data, or queries, for the oracle to annotate; that is, the active learner is allowed to create its own examples for labeling. It is generally only applicable to problems where it is easy to generate a data instance.

### ***Pool-based Sampling***

This most well known method attempts to evaluate the entire dataset for selecting the best query or set of queries. The active learner algorithm is often initially trained on a fully labeled part of the data, which is then used to determine which instances would be most beneficial to insert into the training set for the next active learning iteration. The downside of the method lies in its heavy memory requirements.

### ***Stream-based Sampling***

The algorithm chooses for each incoming instance from the stream whether it is beneficial enough to query or not. A natural disadvantage is that despite careful estimates of threshold, there is no guarantee that the model will remain in the budget or effectively use the budget by the end of training.

## 2. The Problem: Handwritten Digit Recognition

Owing to the successful performance of the homebrewed CNN model, the problem from Assignment-1 has been carried forth here for demonstration of active learning techniques, concepts of version space, and cost benefits of clustering. This section briefly describes our problem statement, the dataset used, the data preprocessing scheme adopted, and the underlying model which is augmented by the active learning techniques.

### 2.1 The Problem Statement

To briefly recapitulate the problem discussed in Assignment-1, it is a classic example of the image classification set of problems. Given an image of a handwritten digit, our goal is to assign a label to the image, the label being the digit represented by the image. The problem will be posed as a machine learning problem, that is, we shall train a classification model on a set of examples - handwritten digit images - with known true labels and test it on another set of instances, or images, where we do not reveal the true labels to the model.

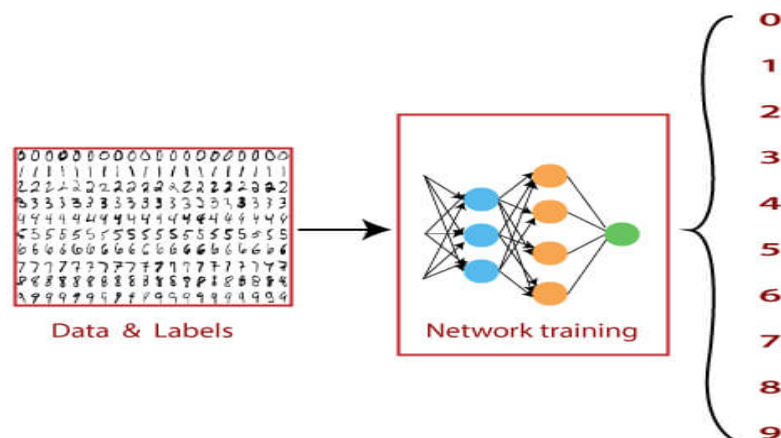


Figure 2.1 The Handwritten Digit Recognition Problem.  
Image Credit: Javatpoint, The Tensorflow MNIST Dataset.

Specifically, we would be using a simple convolutional neural network as our classification model. It may be noted that the problem is a multiclass classification problem where we have more than two classes. Therefore, we would have for the

output of our model the probabilities of the image being any particular digit, thereafter, choosing the digit with the highest probability as the assigned label.

To pose the problem more rigorously, we define a loss function which is to be minimized during the course of training:

$L = -\ln(p_c)$ , where  $p_c$  is the predicted probability for the correct class  $c$ . This loss function is sometimes termed the cross-entropy loss. We wish to minimize our classification error - especially the testing error - by minimizing the loss in the course of training.

## 2.2 The Dataset

The classic MNIST data set has been used for its simplicity and moderate memory requirements, making it suitable for a non-parallelized moderate-load personal-use CPU environment. There are 70,000 images of handwritten digits, each  $28 \times 28$  pixels (784 pixels total) and one channel (black-only grayscale). There are ten class labels associated with the images: 0, 1, 2, ..., 9 (Table 2.1). The dataset has been constructed from scanned documents available under the National Institute of Standards and Technology (NIST).

Table 2.1 The MNIST Dataset

Example Image										
Label	0	1	2	3	4	5	6	7	8	9

By default, the images have been split into two partitions: 60,000 for training and 10,000 for testing. However, due to the very high computational requirements involved in active learning techniques and version space minimization, only 2,500 images for training and 2,500 images for testing have been taken up for the present implementation.

## 2.3 Data Preprocessing

Originally, the images have been stored in a `.png` format under their respective digit directories, which themselves have been stored under the parent training and

testing directories. Java modules have been developed to read images directly from these directory structures and convert them into the familiar matrix representation. Images have been normalized to the interval  $[-0.5, 0.5]$  for faster and more reliable convergence of the network.

```
J Image.java > Image > matrix
1 public class Image {
2     /* a data structure to hold the image
3      * its matrix and label
4      * along with various active learning scores
5      */
6
7     public String uid; // a unique identifier for the image
8     public double[][] matrix; // the image matrix
9     public int label; // the true image class label
10
11     public double randomScore; // randomly chosen points
12     public double lcScore; // uncertainty sampling with least confidence
13     public double smScore; // uncertainty sampling with smallest margins
14     public double ratioConfScore; // uncertainty sampling with ratio of confidence
15     public double entropyScore; // uncertainty sampling with entropy
16     public double veScore; // qbc sampling with vote entropy
17     public double klScore; // qbc sampling with kl divergence
18
19     public Cluster cluster; // the cluster of the image
20     public int cost; // cost for labeling
21     public int clusterLabel; // the label from clustering
22
23     public int vsScore; // the version space score
24     public int worstVsScore; // the worst case version space score
25 }
```

Figure 2.2: The Image Object.

Image Source: Please refer to Image.java attached as a part of the assignment.

Unlike Assignment-1, this time image matrices are not used directly, rather we use custom homebrewed `Image` objects (see Figure 2.2), with the following information: a `String uid` for easier identification and hashing later, a `double[][] matrix` holding the image matrix, an `int label` representing the true class label of the image (which we may or may not choose to read depending on the selection of the instance with active learning or sampling), `double` scores for various active learning techniques (such as, `smScore` for smallest-margin sampling), clustering information (including the `Cluster` assigned along with the

subsequent cost of labeling, **double cost**), and version space information, such as the size of version space after using the image for training, **int vsScore**.

## 2.4 The Underlying Model

As previously mentioned, the CNN model developed in assignment-1 has been used for demonstration of active learning in the present assignment. To briefly review its architecture, it uses a simple network architecture, incorporating all the essential components - convolution, pooling, flattening, and probability assignment - yet nothing too complex, the reason being the ease of experimenting such a simple network on a personal-use machine and the clarity of backpropagation mechanics for such an architecture. The broad form of the network may be seen in Figure 2.3.

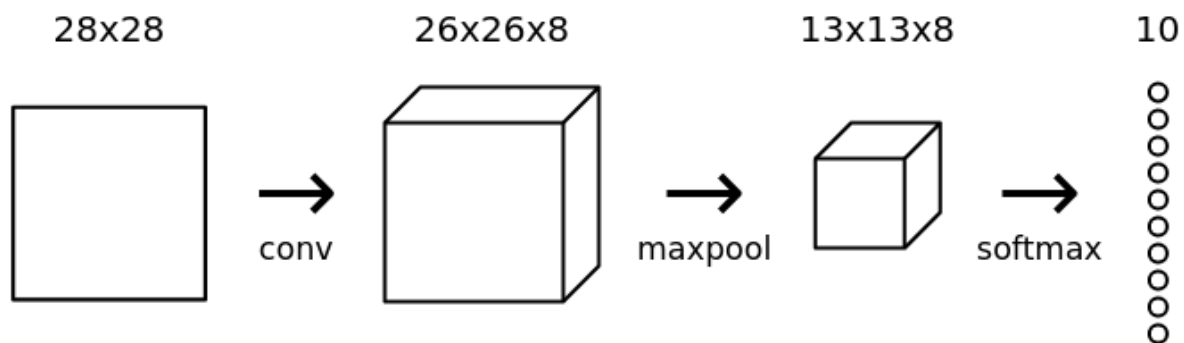


Figure 2.3 The Network Architecture Chosen  
Image Credit: Victor Zhou, Intro to CNNs in Keras

A softmax layer follows the maxpooling layer since we want to assign a probability to each of the 10 classes. Also note that the functionality of the traditional `tensorflow.keras.layers.Flatten()` has been incorporated into the fully-connected flatten layer itself with `double[][] MatrixOps.flatten(double[][] M)`.

### 3. Demonstration of Active Learning

This section delves into the crux of the assignment by providing a brief overview of the various pool-based active learning techniques, accompanied by their implementation scheme and results obtained. Specifically, we cover the uncertainty sampling techniques - least confidence, smallest margin, entropy based, and largest margin - and query-by-committee techniques - vote-entropy and KL-divergence. Also is covered the implementation of stream-based learning with a seamless transition from pool-based learning, retaining and reusing many of the modules in the process. Let us begin the section by taking a look at the broad plan to implement the different pool-based strategies.

#### 3.1 The Broad Implementation Plan

Our implementation shall leverage the power of inheritance and polymorphism in the object oriented language Java to reuse the features common to various active learning techniques, therefore, requiring changes only to the selection and ordering criteria.

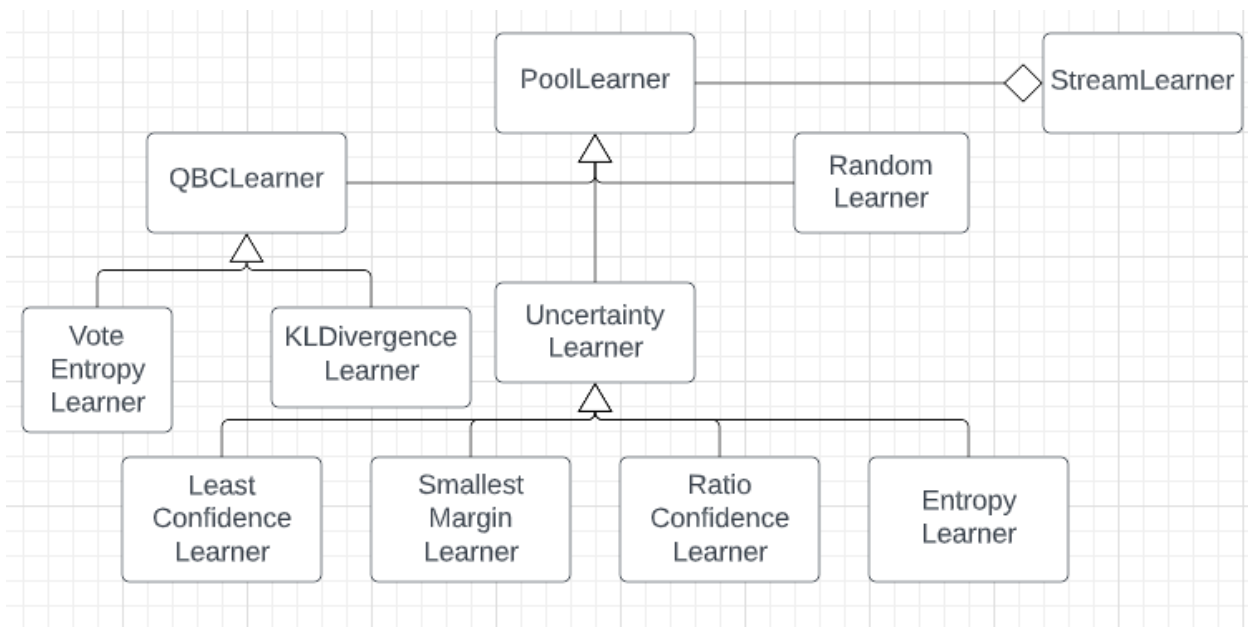


Figure 3.1: Class relations scheme for the active learning implementation  
Image Source: Created with LucidChart ©

As depicted in Figure 3.1, the abstract class `PoolLearner` forms the basic building block for our implementation of the active learning techniques. Three classes, namely `RandomLearner` for learning with random sample selection, `UncertaintyLearner` for uncertainty sampling, and `QBCLearner` for query-by-committee learning, inherit from this abstract class. The abstract classes `QBCLearner` and `UncertaintyLearner` in turn have non-abstract child classes, one for each selection criterion. Stream-based learning would be implemented using a pool-based learning as its backbone for estimation of a reasonable threshold via the class `StreamLearner` with a `PoolLearner` member.

The class `PoolLearner` provides implementation of the overloaded method `ArrayList<Image> getImages()` which fetches a specified number or fraction of images from the training pool according to the selection criterion. Further it has abstract methods `assignScore()` and `assignScores()` for assigning a value to image(s) based on the criterion under use and `orderImages()` to sort the images as per their assigned scores for subsequent fetching. These methods are implemented within the specific children classes, which will be discussed in the subsequent subsections.

## 3.2 Random Selection

The class `RandomLearner` gives the points arbitrary scores for a random selection of points for subsequent pool-based active learning. Note that it would be no different from feeding a passive learner a selection of arbitrarily chosen points, thereby foregoing any benefits we get from the inherent inquisitive nature of an active learner.

The random selection would serve as a baseline for benefits from inclusion of more points. We expect that the active learning techniques would outperform it with a more informed selection of points to benefit their training.



### 3.3 Uncertainty Sampling Techniques

One of the most common general frameworks for measuring informativeness is the uncertainty sampling, where the learner queries the instance that is most uncertain how to label [17]. The class `UncertaintyLearner` lays the framework for uncertainty sampling for the present assignment. It contains a `CNN` member variable, which will serve as the model (trained on the labeled instances) through which we will measure the uncertainty for each sample. Hence, one of the following criteria will be used for assigning scores to the images and their subsequent ordering and selection.

#### ***Least Confidence (LC)*** [`LeastConfidenceLearner.java`]

The strategy allows the active learner to select the unlabeled data samples for which the model is least confident in prediction or class assignment [18].

Mathematically, the best instance to be queried,  $x_{LC}^*$ , would be

$x_{LC}^* = \operatorname{argmax}_x [1 - \max_{j \in \text{classes}} P(j | x; \theta)]$ , where  $\theta$  parameterizes the model.

#### ***Smallest Margin*** [`SmallestMarginLearner.java`]

Also known as the *Best-versus-second-best (BvSB)* sampling, this strategy considers the instance with the least difference between the most probable and second-most probable class probabilities, that is,

$x_{SM}^* = \operatorname{argmin}_x P(\hat{y}_1 | x; \theta) - P(\hat{y}_2 | x; \theta)$ , where  
 $\hat{y}_1 = \operatorname{argmax}_{j \in \text{classes}} P(j | x; \theta)$  and  $\hat{y}_2 = \operatorname{argmax}_{j \in \text{classes} - \{\hat{y}_1\}} P(j | x; \theta)$ .

#### ***Entropy*** [`EntropyLearner.java`]

For classification with a large number of classes, the above two strategies do not fit optimally because they consider one or two most probable classes for sampling. The information lying in the remaining classes' probability distribution is unused [18]. Entropy sampling addresses this issue by measuring the disorder or impurity in the system, as follows,

$x_H^* = \operatorname{argmax} \left[ - \sum_{j=1}^m P(y_j | x; \theta) \log P(y_j | x; \theta) \right]$ , where  $m$  is the number of classes and  $\theta$  represents the current model parameters.

### ***Largest Margin*** [RatioConfidence.java]

This approach is meant specifically for margin-based classifiers, such as SVMs. In the SVM the support vectors are the most informative (in fact, the only informative points), hence the selected data points should be the ones falling on the margin. Therefore, we use the distance to the separating hyperplane as the selection criterion [2]. In the ***margin sampling*** variant, data points are chosen such that they fall within the region of margin a unit distance from the separating hyperplane, that is,

$x_{MS}^* = \operatorname{argmin}_x F(x; w)$ , where  $F(x; w)$  represents the distance between any data sample and the hyperplane of the class  $w$ . The strategy selects a single data sample for querying per iteration [18].

To select multiple samples per iteration for querying the human oracle, data samples with the lowest distance from support vectors can be chosen - one for each support vector. This strategy is sometimes called ***margin sampling - closest support vectors (MS-cSV)*** [2].

The preceding approaches present a problem, however: we do not have the geometric interpretation or parametric simplicity of the SVMs since we are using CNNs. Fortunately, for the deep learning based techniques, a close proxy is the ***ratio confidence sampling*** [19], wherein we take the ratio of the top two most confident predictions, that is,

$$x_{RC}^* = \operatorname{argmax}_x \frac{P(\hat{y}_2 | x; \theta)}{P(\hat{y}_1 | x; \theta)}, \quad \text{where} \quad \hat{y}_1 = \operatorname{argmax}_{j \in \text{classes}} P(j | x; \theta) \quad \text{and} \\ \hat{y}_2 = \operatorname{argmax}_{j \in \text{classes} - \{\hat{y}_1\}} P(j | x; \theta).$$

### 3.4 Query-by-Committee Techniques

The QBC approach involves maintaining a committee  $C = \{\theta^{(1)}, \dots, \theta^{(C)}\}$  of classifiers which are all trained with the set of currently labeled data  $L$ , but represent competing hypotheses. Each committee member then votes on the labelings of the query candidates. The most informative query is the one wherein most committee members disagree [4].

To successfully implement the QBC selection algorithm, one must:

- be able to construct a committee of models that represent different regions of the version space. [We achieve this with bagging for the CNN models]
- have some measure of disagreement among the committee members [The version space is non-null, as shown in Section 4].

The class `QBCLearner` lays the framework for QBC techniques. It keeps a committee of 5 CNNs (number kept flexible) trained on the set of labeled instances with the familiar 0.632 bagging, that is, we sample with replacement, so that on an average any sample of sufficient size contains only  $1 - \lim_{N \rightarrow \infty} \left(1 - \frac{1}{N}\right) = 1 - \frac{1}{e} = 0.632$  unique instances of the total.

This committee is used for assignment of scores using the following criteria one at a time.

#### ***Vote Entropy*** [`VoteEntropyLearner.java`]

Measuring the level of disagreement, the vote entropy learner chooses

$$x_{VE}^* = \underset{x}{\operatorname{argmax}} \left[ - \sum_{j=1}^m \frac{V(y_j)}{C} \log \left( \frac{V(y_j)}{C} \right) \right],$$
 where  $y_j$  ranges over each of the  $m$  possible classes (10 digits in our case),  $V(y_j)$  is the number of *votes* the present label receives, and  $C$  is the committee size (5 in our case).

#### ***Kullback-Leibler (KL) Divergence*** [`KLDivergenceLearner.java`]

The KL divergence is an information-theoretic measure of the difference between two probability distributions. This disagreement measure considers the most informative query to be the one with the largest average difference between the

label distributions of any one committee member and the consensus [4]. Mathematically,

$$x_{KL}^* = \underset{x}{argmax} \frac{1}{C} \sum_{c=1}^C D(P_{\theta^{(c)}} || P_C), \text{ where}$$

$$D(P_{\theta^{(c)}} || P_C) = \sum_{j=1}^m P(y_j | x; \theta^{(c)}) \log \left[ \frac{P(y_j | x; \theta^{(c)})}{P(y_j | x; C)} \right], \quad \theta^{(c)} \text{ represents any}$$

particular committee member, and  $C$  represents the committee as a whole, that is,

$$P(y_j | x; C) = \frac{1}{C} \sum_{c=1}^C P(y_j | x; \theta^{(c)}).$$

### 3.5 Pool-Based Techniques: Results

To gain insights into the efficacy of each active learning technique implemented above, a testbench has been developed (`TechniqueExperiment.java`). 3 trials each with 5 epochs of training have been conducted for each technique using 10%, 20%, 30%, 40%, and 50% of the training pool instances. We use 10% of the initial training pool of 2500 instances as the original labeled dataset of 250 images. For testing, 500 images from the testing partition have been used.

The performance of the CNN model has been recorded on the complete training pool and the testing set separately for each of the techniques and each query fraction. The results have been recorded in the form of learning curves as depicted in Table 3.1.

It can be noted that all the active learning techniques implemented outperform random learning if we consider the overall performance across the learning curve. The perturbations in the curve (hence its deviation from the ideal shape) can be attributed to the very small data size considered - only a pool of 2500 of the otherwise available 70000. It also explains the low accuracies across the curves; hence, we focus on the performance of the techniques relative to each other and the random selection, instead of the absolute numbers.

Table 3.1: Learning Curves for the Pool-based Active Learning Techniques

Dataset Segment	Learning Curves																																																								
Training pool of 2500 images	<p>Training Pool Accuracy against Queried Instances</p> <table><thead><tr><th>Queried Instances - Fraction of Total Pool</th><th>Random</th><th>Least Confidence</th><th>Entropy</th><th>Smallest Margin</th><th>Ratio Confidence</th><th>Vote Entropy</th><th>KL Divergence</th></tr></thead><tbody><tr><td>0.0</td><td>0.735</td><td>0.735</td><td>0.735</td><td>0.735</td><td>0.735</td><td>0.735</td><td>0.735</td></tr><tr><td>0.1</td><td>0.800</td><td>0.770</td><td>0.770</td><td>0.790</td><td>0.780</td><td>0.760</td><td>0.790</td></tr><tr><td>0.2</td><td>0.780</td><td>0.800</td><td>0.770</td><td>0.800</td><td>0.790</td><td>0.820</td><td>0.780</td></tr><tr><td>0.3</td><td>0.780</td><td>0.780</td><td>0.780</td><td>0.820</td><td>0.790</td><td>0.830</td><td>0.790</td></tr><tr><td>0.4</td><td>0.770</td><td>0.800</td><td>0.800</td><td>0.830</td><td>0.800</td><td>0.850</td><td>0.810</td></tr><tr><td>0.5</td><td>0.790</td><td>0.820</td><td>0.830</td><td>0.850</td><td>0.800</td><td>0.860</td><td>0.830</td></tr></tbody></table>	Queried Instances - Fraction of Total Pool	Random	Least Confidence	Entropy	Smallest Margin	Ratio Confidence	Vote Entropy	KL Divergence	0.0	0.735	0.735	0.735	0.735	0.735	0.735	0.735	0.1	0.800	0.770	0.770	0.790	0.780	0.760	0.790	0.2	0.780	0.800	0.770	0.800	0.790	0.820	0.780	0.3	0.780	0.780	0.780	0.820	0.790	0.830	0.790	0.4	0.770	0.800	0.800	0.830	0.800	0.850	0.810	0.5	0.790	0.820	0.830	0.850	0.800	0.860	0.830
Queried Instances - Fraction of Total Pool	Random	Least Confidence	Entropy	Smallest Margin	Ratio Confidence	Vote Entropy	KL Divergence																																																		
0.0	0.735	0.735	0.735	0.735	0.735	0.735	0.735																																																		
0.1	0.800	0.770	0.770	0.790	0.780	0.760	0.790																																																		
0.2	0.780	0.800	0.770	0.800	0.790	0.820	0.780																																																		
0.3	0.780	0.780	0.780	0.820	0.790	0.830	0.790																																																		
0.4	0.770	0.800	0.800	0.830	0.800	0.850	0.810																																																		
0.5	0.790	0.820	0.830	0.850	0.800	0.860	0.830																																																		
Testing partition of 500 images	<p>Test Accuracy against Queried Instances</p> <table><thead><tr><th>Queried Instances - Fraction of Total Pool</th><th>Random</th><th>Least Confidence</th><th>Entropy</th><th>Smallest Margin</th><th>Ratio Confidence</th><th>Vote Entropy</th><th>KL Divergence</th></tr></thead><tbody><tr><td>0.0</td><td>0.660</td><td>0.660</td><td>0.660</td><td>0.660</td><td>0.660</td><td>0.660</td><td>0.660</td></tr><tr><td>0.1</td><td>0.720</td><td>0.710</td><td>0.690</td><td>0.720</td><td>0.750</td><td>0.720</td><td>0.730</td></tr><tr><td>0.2</td><td>0.730</td><td>0.730</td><td>0.730</td><td>0.750</td><td>0.720</td><td>0.770</td><td>0.720</td></tr><tr><td>0.3</td><td>0.750</td><td>0.730</td><td>0.740</td><td>0.770</td><td>0.740</td><td>0.770</td><td>0.740</td></tr><tr><td>0.4</td><td>0.750</td><td>0.740</td><td>0.770</td><td>0.790</td><td>0.750</td><td>0.800</td><td>0.780</td></tr><tr><td>0.5</td><td>0.760</td><td>0.770</td><td>0.810</td><td>0.810</td><td>0.770</td><td>0.825</td><td>0.800</td></tr></tbody></table>	Queried Instances - Fraction of Total Pool	Random	Least Confidence	Entropy	Smallest Margin	Ratio Confidence	Vote Entropy	KL Divergence	0.0	0.660	0.660	0.660	0.660	0.660	0.660	0.660	0.1	0.720	0.710	0.690	0.720	0.750	0.720	0.730	0.2	0.730	0.730	0.730	0.750	0.720	0.770	0.720	0.3	0.750	0.730	0.740	0.770	0.740	0.770	0.740	0.4	0.750	0.740	0.770	0.790	0.750	0.800	0.780	0.5	0.760	0.770	0.810	0.810	0.770	0.825	0.800
Queried Instances - Fraction of Total Pool	Random	Least Confidence	Entropy	Smallest Margin	Ratio Confidence	Vote Entropy	KL Divergence																																																		
0.0	0.660	0.660	0.660	0.660	0.660	0.660	0.660																																																		
0.1	0.720	0.710	0.690	0.720	0.750	0.720	0.730																																																		
0.2	0.730	0.730	0.730	0.750	0.720	0.770	0.720																																																		
0.3	0.750	0.730	0.740	0.770	0.740	0.770	0.740																																																		
0.4	0.750	0.740	0.770	0.790	0.750	0.800	0.780																																																		
0.5	0.760	0.770	0.810	0.810	0.770	0.825	0.800																																																		

Overall, the *vote entropy* technique emerges as the best method in terms of both training and testing learning curves. Among the uncertainty sampling methods, the *smallest margin* method performs the best. The benefits of using these techniques have been summarized in Table 3.2.

Table 3.2 Accuracy Gains over Random Selection

Type	Best Technique	Average Training Accuracy Gain	Max Training Accuracy Gain	Average Testing Accuracy Gain	Max Testing Accuracy Gain
QBC	Vote Entropy	0.033 ←	0.080 ←	0.030 ↔	0.070 ↔
Uncertainty Sampling	Smallest Margin	0.028 →	0.060 →	0.030 ↔	0.070 ↔

← indicates a win, ↔ a tie, and → a loss.

The overall ranking of techniques can be stated approximately as: (1) vote entropy (2) smallest margin, (3) KL divergence, (4) entropy, (5) least confidence, (6) ratio of confidence, and (7) random selection.

### 3.6 Stream Based Techniques: Results

We run the stream based learning with the entropy criterion for the purpose of demonstration. The threshold for selection of instances from the stream is set so that ~20% of the instances are queried for annotation by the oracle and hence training. This threshold is estimated as the entropy of the last chosen instance from a pool of 2250. Passing the same pool as the stream, it is verified that 450 out of 2250 instances - exactly 20% - of the instances are accepted.

Now passing a previously unseen stream, we note that 433 out of 2250 instances - about 19.2% - from the new stream are accepted, which affirms the reliability of our threshold estimation. Coming to the gain in performance, training accuracy jumps from 0.74 to 0.77 and testing accuracy jumps from 0.67 to 0.71. The fact that the gains are slightly lower than pool-based learning (testing accuracy increasing from 0.67 to 0.71) can be attributed to the fact that each image is used for only iteration as opposed to 5 due to the real-time characteristic of a stream.

## 4. Version Space

The present section builds up the concept of version space and discusses greedy strategies for version space minimization. It concludes with a brief discussion of results obtained with a Java implementation of ordering of data points on the basis of their effect on the version space.

### 4.1 The Concept of a Version Space

Version space of the currently labeled data can be defined as the hyper-area of the hypotheses consistent with the data [11]. More formally,

$V = \{f \in H \mid \forall i \in [1 \dots n], y_i f(x_i) > 0\}$  for a binary classification problem, where  $n$  is the number of training examples,  $H$  is the hypothesis space, and the allowed labels are  $-1$  and  $+1$  [12].

Therefore, the version space gives us the measure of the number of classifiers consistent with the training data. This seemingly abstract concept can be very useful for selection of the best training instances when we consider its Vapnik dual. Consider the training instances as lines (defined by their vectors) and the classifiers as points (defined by their weight vectors), as shown in Figure 4.1.

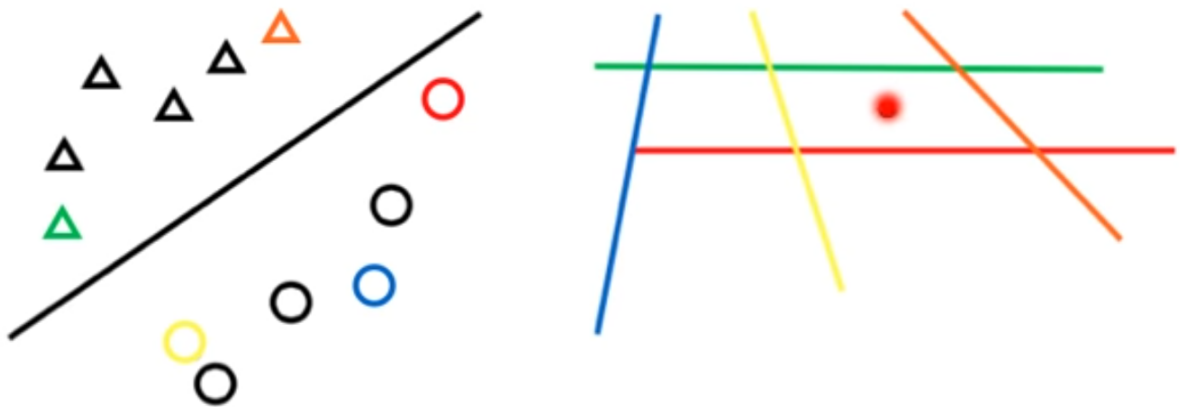


Figure 4.1: The Vapnik dual transformation (left to right).  
Image Credit: Please refer [12].

Now it is evident that the blue circle is not as useful in minimizing the version space, that is, narrowing down our list of useful classifiers, as compared to the

yellow circle. Notice the similarities between such a formulation and the support vector machines. Indeed, the maximum margin of the SVM based model would be the size of the largest hypersphere fitting in the version space and centered at the weight vector of the SVM.

We can approximate the version space by using a committee of classifiers. Then, in the dual formulation, it would consist of all points where the classifiers in the committee disagree. The version space can thus be understood as the region still unknown to the overall model class - therefore, we can approach the training problem as a version space minimization problem.

## 4.2 Greedy Shrinkage of Version Space

One method of choosing the instances best for querying would be to prefer ones which reduce the version space by most - that is, greedy reduction of version space. However, to be conservative we want the version space to reduce the most in the worst case for each point, making it akin to binary search, where we aim to approximately halve the version space in each move.

Consider the following greedy strategies for version space minimization in case of SVMs. In Figure 4.2 (left), we choose the point **b**, which approximately halves the version space and passes closest to the center of the maximum-margin hypersphere,  $w_i$ . This strategy is called the *simple margin approach* [12]. However, it fails in cases like Figure 4.2 (right), here **e** would have been the right point to query greedily instead of **a**.

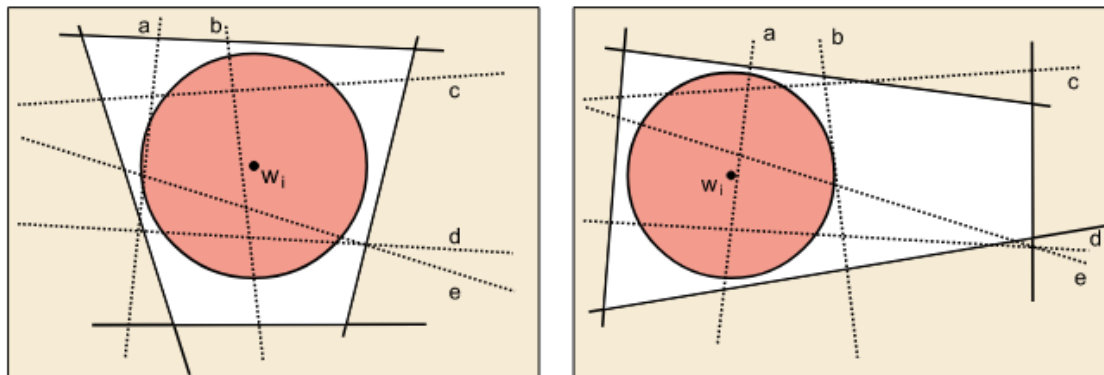


Figure 4.2: Simple Margin strategy: Success (left) and Failure (right)  
Image Credit: Please refer [13].



A more sophisticated approach will train the SVM again with each point, considering each of the possible labels. Then we choose the point where the maximum margins are about equal (ratio closest to one) since that would approximately halve our version space. This approach is sometimes called the **ratio margin approach**. It is shown in Figure 4.3, where we query the correct label in each case (**b** in left and **e** in right).

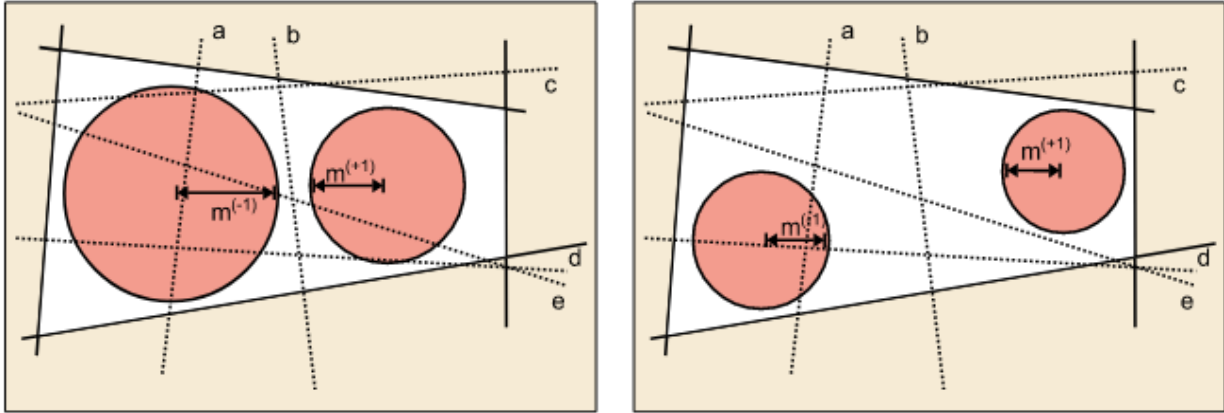


Figure 4.3: Ratio Margin strategy: Success (left) and Success (right)  
Image Credit: Please refer [13].

Since we do not use the SVMs in the present assignment, we can not directly make the use of geometric concepts like the maximal margins. Instead, we define the version space in terms of the number of points where our CNN committee (such as that used in QBC based learning) disagrees. Then the goal would be to choose the points that reduce this version space by the most in terms of the points still remaining.

Taking ideas from the preceding discussion, we consider the version space reduction as the minimum of the cases where we train the model with the various labels (0 to 9) for the image. The image which produces the highest worst-case reduction - that is, the least size of version space remaining - wins. Mathematically, we choose  $\operatorname{argmin}_x \max_{j \in [0..9]} VS^{after}(x, j)$ , where  $VS^{after}(x, j)$  is the size of the version space - in terms, of points of disagreement for the committee - after training with the instance  $x$  considering the class label  $j$  (adapted from [11]).

Such a technique is guaranteed to be approximately as good as any other strategy for minimizing the number of collected labels [14].

The modules for version space reduction based on the above idea have been implemented in `VersionSpaceReducer.java`. Modules to assign “true version space scores” to the points have also been developed. The true score is the size of the version space after training with the image *if we know the true label of the image*, that is, if the label of the image is revealed. Mathematically, we choose  $\text{argmin}_x VS^{after}(x, y)$ , where  $y$  is the true class label.

### 4.3 Discussion of Results

We consider a total training sample of 2500 images of which 250 are labeled. This labeled group of images is used to train a committee of 5 CNNs using a bagging based approach. We then find the size of the version space over the unlabelled 2250 images - the images of interest from which we wish to greedily query for subsequent training. Over 10 trials, the version space size varied from about 880 to 950, with the mean value of 917 (out of 2250).

Since the computations for ordering points on the basis of version space shrinkage are prohibitively expensive in terms of runtime, the ordering has been done with only 250 points from the pool. The version space size for the following trial was recorded to be 92 (out of 250). The points were ordered using their true and worst-case version scores, resulting in the following statistic (Table 4.1).

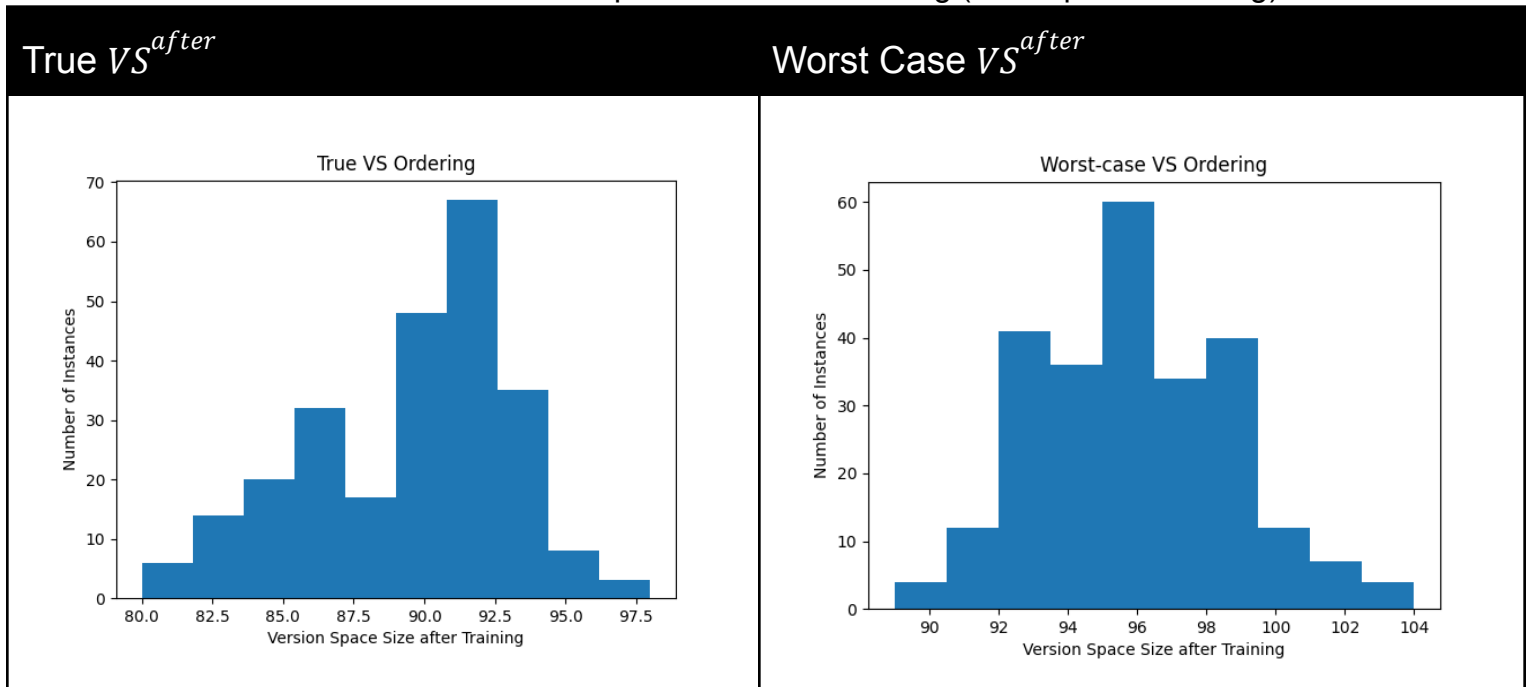
Table 4.2 Version Space Shrinkage Results

Statistic	True Scores (based on training with true labels)	Worst-case Scores (based on training with worst labels)
Average Version space size after training $VS^{after}$	89.44	95.68
Standard Deviation of $VS^{after}$	3.60	2.78

Max $VS^{after}$	98	104
Min $VS^{after}$	80	89

Table 4.2 shows the points plotted in histograms by the distribution of their true  $VS^{after}$  scores and their worst case  $VS^{after}$  scores, as defined in Section 4.2.

Table 4.2 Distribution of Version Space sizes after training (the required ordering)



All in all, it can be noted that the mean true version space size after training with any particular instance is lower than the corresponding worst-case score, as is intuitive. Further, the standard deviation of the true scores is higher, showing the greater variability in benefits of actually labeling and training with an instance, thereby accentuating the significance of proper active learning techniques. Also note that most points do not offer any shrinkage in version space size whatsoever, as signified by the peak in plot on the left at  $\sim 92$ . Some points can even adversely affect the version space size (the portion on the right of the peak in the left plot). Anyhow, it is evident that we can greatly benefit by choosing first the best points - those with  $VS^{after}$  scores of  $\sim 80$ .

## 5. Clustering

The section discusses an alternative to active learning by augmenting the labeled examples at our disposal with clustering. This approach forms the backbone of several semi-supervised learning studies. Specifically, the motivation for such an approach is discussed followed by our implementation, including the algorithm used. Finally, the results and benefits obtained by the clustering are briefly discussed.

### 5.1 Motivation

Cluster analysis is an unsupervised task that divides the data into groups, or clusters, that are meaningful, useful, or both [15]. Being an unsupervised learning task, it does not require labels, hence is readily applicable to problems where data is aplenty, but labeled data is scarce or expensive.

It is reasonable to say that a bird that quacks like a duck, swims like a duck, and walks like a duck is probably a duck. A generalization of such reasoning would be to argue that similar data points would be likely to have the same class labels. Thus, if we obtain clusters with enough purity, we can extend the labels of the instances known to us to the other instances present in the cluster. Semi-supervised learning techniques such as the preceding have been shown to produce satisfactory results in the literature. [16] This idea shall form the basis for the training technique implemented in this section.

Using clustering to generate more labeled instances (from unlabeled instances) can be very cost efficient, for a human annotator more often than not can be the bottleneck expense in the dataset generation. Further, it is not subject to human-induced biases and lack of authenticity as prevalent in crowdsourced annotation. [2]

### 5.2 Methodology and Implementation

The key steps in our implementation would be

- Train a CNN using a small number of training instances - all labeled (250 in our case). Note the performance of the CNN in terms of training, complete pool, and test accuracy (testing size of 2500).
- Run the K-Means algorithm on the unlabeled pool of 2250 (take a size larger than the labeled instances for realism), assigning each image a cluster
- Sample a fraction - say 20% - of instances from each cluster and ask for their labels. Note their labels and assign the majority label to each image in the cluster. The queried instances are assigned their true labels, not the cluster labels, though since there is no additional cost incurred due to such choice.
- Calculate the cost of such clustering -  $Cost = N_{queried} \times costPerLabel + (N_{tot} - N_{queried}) \times 0$ , where  $N_{queried} = 0.2 \times N_{tot}$  if we use a sampling fraction of 20%.
- Also calculate the accuracy of the classification as  $Accuracy = \frac{\#(true\ label == cluster\ label)}{\#(total)}$ .
- Train the CNN with the data augmented with the new examples and note the resulting training and testing performance accuracies.

### ***The Clustering Algorithm***

It will also be worthwhile taking a look into the ***K-Means algorithm*** used for clustering, as implemented in `KMeans.java`.

```

K-Means({x_1, x_2, ..., x_N}, K)
  (u_1, u_2, ..., u_K) := AssignSeeds(K)
  while !isStoppingCondition()
    for k := 1 to K
      cluster_1, ..., cluster_K := {}, ..., {}
      for n := 1 to N
        j := argmin (j) ||u_j - x_n||
        cluster_j = cluster_j union x_n
      for k := 1 to K
        u_k = sum x in cluster_k (x) / |cluster_k|
  return {u1, u2, ..., uk}

```

We use a fixed number of iterations of the clustering process as the stopping condition (the outermost loop above). We set it to 100 iterations. Euclidean distance has been used as the measure of dissimilarity or distance.


Choosing the initial centroids is often the most crucial choice affecting the resultant clusters [15]. Instead of using random initialization of seeds, the seeds have been computed as arithmetic means of the already labeled training instances, each seed representing a different digit. On a related note, the number of clusters  $K$  has been set to  $K = 10$ , one for each digit 0, 1, ..., 9.

### 5.3 Discussion of Results

Multiple trials of clustering each with 100 iterations and 10 clusters have been performed. The average accuracy of the subsequent labeling, as defined in Section 5.2, was recorded to be 0.74, that is, 74% of the unlabeled instances were assigned their true labels via the cluster based label assignment scheme.

Overall, it was found that each cluster had in most cases a unique label. Therefore, each of the digits was effectively represented. However, in some poor performance trials, the following digits, shown in Table 5.1, were confused (two clusters representing the same digit on a majority assignment basis):

Table 5.1 Digits confused during clustering

5 and 6		4 and 9	
			

This observation can be attributed to the very similar shapes of each pair of confused digits when the writing strokes are considered. The digits within each pair differ by just one tiny stroke, hence their euclidean distances can be expected to be small. A more sophisticated clustering scheme can use other features - such as those extracted from the lower CNN layers - for clustering instead of the Euclidean distances on the original normalized matrices.

Since the accuracy of label assignments was 0.74, training with the augmented pool of images did more harm than good to the CNN model. The training accuracy dropped from 0.86 to 0.83 and the testing accuracy dropped from 0.71 to 0.67.

Assuming that each label costs Rs. 100 and it takes an hour of time, we realize a total cost of Rs. 45500 with cluster based label assignments as opposed to a cost of Rs. 225000 had we opted for labeling each instance manually. Therefore, we save Rs. 179500, a little less than 80% savings. Similarly, it will take 455 hours instead of 2250 hours it would have taken in absence of clustering. The average cost per image is Rs. 20.22 and the average time is 0.2022 hours, or 12.13 minutes (compared with Rs. 100 and 60 minutes otherwise).

If we perform the same experiment with 40% of the 90% unlabeled pool (as stated in the assignment problem statement), we get a cost of Rs. 18500 (savings of Rs. 71500) or cost per image of Rs. 20.56. We require 185 hours total (saving 715 hours) or 12.34 minutes per image. The accuracy of label assignment is 0.72. The classifier shows the same effects on its performance as the preceding case.

Therefore, we conclude that although the cluster-based labels can not be expected to produce the same accuracy as with true labels, it can effectively reduce our labeling cost and time by a lot, while also providing insights into the structure and organization of the data.

## Conclusion and Future Work

While Assignment-1 demonstrated how a CNN can provide excellent results for an image classification problem when the data is plentiful, the present assignment creates a situation, where the data is severely limited by restricting our training to a very small pool of images. Though the images themselves are accessible, we do not allow our routines to use their labels without incurring a corresponding cost (Section 5.3). It is hoped that this report provides some promising solutions to training a model effectively under such a condition, which is quite common in both industry and research, especially in hitherto unexplored areas.

The suite of active learning techniques prove their usefulness by giving gains as high as 0.07 in terms of accuracy (Section 3.5) over a random selection (which is often the case when feeding data to a passive model). Further, as demonstrated in Section 3.6, the pool-based techniques can be easily transformed for a stream-based scenario, thus extending their usefulness to low memory environments, commonly found in cellular phones and embedded systems. Section 4 provided insights into the theoretical basis of active learning techniques and gave a quick visualization of the ordering of points with respect to greedy version space shrinkage. Further, clustering techniques were presented as a useful alternative to the active learning techniques. Although, the performance of the model did not stack up very well against the active learning techniques, the cost saving benefits were significant (Section 5.3).

The arsenal of techniques presented give us a very promising launchboard for further investigation and application to a more varied group of real-world scenarios. A good starting point would be to test their effectiveness with larger pools and streams. Textual, structured, and time-series data should also be tested with, so as to establish a clear scope of application for each technique. It is sincerely hoped that the report gives valuable insights for such experiments and more.



# References

- [1] “Joe Kaeser: Data is the 21st century's oil, says Siemens CEO Joe Kaeser,” *The Economic Times*. [Online]. Available: <https://economictimes.indiatimes.com/magazines/panache/data-is-the-21st-centurys-oil-says-siemens-ceo-joe-kaeser/articleshow/64298125.cms?from=mdr>. [Accessed: 30-Nov-2022].
- [2] Aravind Kandiah, “It’s 2022 and Data Labeling Still Sucks”, *Bifrost.ai*. [Online]. Available: <https://www.bifrost.ai/post/its-2022-and-data-labeling-still-sucks>. [Accessed: 30-Nov-2022]
- [3] “Trends in 2022, AI Isn't So Bad for Jobs, Price Optimization Vs. Price Hike, Visual Reasoning Advances”, *TheBatch @ Deeplearning.ai*. Issue: November 16, 2022.
- [4] Settles B., “Active Learning Literature Survey”, Computer Sciences Technical Report, University of Wisconsin-Madison. Available: <https://burrsettles.com/pub/settles.activelearning.pdf>. [Accessed: 30]
- [5] Tong, Simon. “Active learning: theory and applications.” Stanford University (2001).
- [6] T. Pu, “Application of active learning algorithm in handwriting recognition numbers,” *Journal of Physics: Conference Series*, vol. 1861, no. 1, p. 012060, 2021.
- [7] “Learning more from less data with active learning,” *Learning More from Less Data with Active Learning*. [Online]. Available: <https://www.jpmorgan.com/insights/technology/active-learning>. [Accessed: 30-Nov-2022].
- [8] Rohit Kundu, “Active Learning in Machine Learning,” *V7Labs*. [Online]. Available: <https://www.v7labs.com/blog/active-learning-guide>. [Accessed: 02-Dec-2022].
- [9] AC John, “Active Learning: Strategies, Tools, and Real-world Use Cases,” *Neptune.ai*. [Online] Available: <https://neptune.ai/blog/active-learning-strategies-tools-use-cases>. [Accessed: 02-Dec-2022]
- [10] K. Wang, D. Zhang, Y. Li, R. Zhang and L. Lin, "Cost-Effective Active Learning for Deep Image Classification," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 12, pp. 2591-2600, Dec. 2017, doi: 10.1109/TCSVT.2016.2589879.
- [11] Golovin, D., “Active Learning 2: The Myopic Version-Space Shrinking Strategy”, CS 253: Advanced Topics in Machine Learning lectures at Carnegie Mellon University, Fall Semester 2010.

- [12] Shayan Doroudi, “Active Learning for Support Vector Machines (SVMs)”. [Online]. Available: <https://www.youtube.com/watch?v=ztCXSAUAe38>. [Accessed: 02-Dec-2022].
- [13] S. Tong and D. Koller, “Support vector machines active learning with applications to text classification”, *Journal of Machine Learning Research* 2001.
- [14] Dasgupta, S., “Analysis of a Greedy Active Learning Strategy” in *Advances in Neural Information Processing Systems 17 [NIPS]*, Dec 2004.
- [15] Tan, P.-N., Steinbach, M., Kumar, V. (2005). Introduction to Data Mining. Addison Wesley. ISBN: 0321321367.
- [16] Peikari, M., Salama, S., Nofech-Mozes, S. *et al.* A Cluster-then-label Semi-supervised Learning Approach for Pathology Image Classification. *Sci Rep* 8, 7193 (2018). <https://doi.org/10.1038/s41598-018-24876-0>.
- [17] B. Settles and M. Craven, “An analysis of active learning strategies for sequence labeling tasks,” *Proceedings of the Conference on Empirical Methods in Natural Language Processing - EMNLP '08* 2008.
- [18] Hardik Dave, “Active Learning Sampling Strategies”, *Medium*. [Online]. Available: <https://medium.com/@hardik.dave/active-learning-sampling-strategies-f8d8ac7037c8>. [Accessed: 02-Dec-2022].
- [19] U. Patel, H. Dave, and V. Patel, “Hyperspectral image classification using uncertainty and diversity based Active Learning,” *Scalable Computing: Practice and Experience*, vol. 22, no. 3, pp. 283–293, 2021.

## Dataset Used

**The MNIST Database of Handwritten Digits:** The dataset is available in public domain as .gz files at the link: <http://yann.lecun.com/exdb/mnist/>. The .png format data used in the present implementation can be found at the link: <https://www.kaggle.com/datasets/jidhumohan/mnist-png>.

## Appendix: Source Code Files

Following is a list of source code files - containing the implementation of the network, various auxiliary functions, plotting utilities, and a consolidated runtime script - attached with the present report.

### *Java Files*

- **Driver.java**: a menu-driven demonstration of active learning techniques, stream learning techniques, version space reduction, and clustering based labeling.
- **CNN.java**: the CNN constructor and training and testing modules
- **Committee.java**: the CNN constructor and training and testing modules for committee of CNNs developed using bagging
- **Conv3x3.java**: the convolutional layer - both forward and backward props
- **MaxPooling.java**: the pooling layer - both forward and backward props
- **Dense.java**: the fully connected softmax layer - both forward and backward props
- **ImageOps.java**: modules to open and process images
- **MatrixOps.java**: modules to emulate NumPy operations like adding matrices, matrix dot products, flattening matrices, and the like
- **VectorOps.java**: modules to emulate NumPy operations for vectors
- **PrettyPrint.java**: printing results on to console or required output files in a human friendly format
- **Image.java**: custom data structure to hold images
- **PoolLearner.java**: abstract class laying the framework for pool-based active learning
- **UncertaintyLearner.java**: abstract class laying the framework for pool-based active learning with uncertainty sampling
- **QBCLearner.java**: abstract class laying the framework for pool-based active learning with query-by-committee
- **LeastConfidenceLearner.java**: least confidence based uncertainty sampling

- `SmallestMarginLearner.java`: smallest margin based uncertainty sampling
- `EntropyLearner.java`: entropy based uncertainty sampling
- `RatioConfidenceLearner.java`: ratio of confidence based uncertainty sampling
- `VoteEntropyLearner.java`: vote entropy based QBC sampling
- `KLDivergenceLearner.java`: KL divergence based QBC sampling
- `VersionSpaceReducer.java`: the ordering of points according to greedy version space shrinkage
- `StreamLearner.java`: stream based active learning
- `Cluster.java`: data structure with cluster information
- `KMeans.java`: K-Means algorithm for clustering and subsequent label assignment to the pool
- `TechniqueExperiment.java`: run an active learning technique for certain number of trials
- `AggregateResults.java`: obtain mean performance across the trials

### ***Python Files***

- `learningCurve.py`: plotting the learning curves using the .csv files obtained by running the `AggregateResults.java`
- `plotVersionSpaceDistr.p`: plotting the histogram of version space distribution using output of `VersionSpaceReducer.java`

### ***GNUPlot Files***

- `plotEpochAcc.p`: plotting the training accuracy against epochs
- `plotStepAcc.p`: plotting the training accuracy against hundreds of steps
- `plotEpochLoss.p`: plotting the training loss against epochs
- `plotStepLoss.p`: plotting the training loss against hundreds of steps

### ***Shell Scripts***

- `run.sh`: an all-in-one script to compile all relevant Java files, run the driver, and plot the results obtained