DEGREE PROJECT IN TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2021

Deep Active Learning for Image Classification using Different Sampling Strategies

Shahin Saleh

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Authors

Shahin Saleh ssaleh@kth.se
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology

Place for Project

Stockholm, Sweden

Examiner

Prof. Danica Kragic Jensfelt
Stockholm, Sweden
KTH Royal Institute of Technology

Supervisor

Dr. Hamid Reza Faragardi
Stockholm, Sweden
KTH Royal Institute of Technology

Abstract

Convolutional Neural Networks (CNNs) have been proved to deliver great results in the area of computer vision, however, one fundamental bottleneck with CNNs is the fact that it is heavily dependant on the ground truth, that is, labeled training data. A labeled dataset is a group of samples that have been tagged with one or more labels. In this degree project, we mitigate the data greedy behavior of CNNs by applying deep active learning with various kinds of sampling strategies. The main focus will be on the sampling strategies random sampling, least confidence sampling, margin sampling, entropy sampling, and K-means sampling. We choose to study the random sampling strategy since it will work as a baseline to the other sampling strategies. Moreover, the least confidence sampling, margin sampling, and entropy sampling strategies are uncertainty based sampling strategies, hence, it is interesting to study how they perform in comparison with the geometrical based K-means sampling strategy. These sampling strategies will help to find the most informative/representative samples amongst all unlabeled samples, thus, allowing us to label fewer samples. Furthermore, the benchmark datasets MNIST and CIFAR-10 will be used to verify the performance of the various sampling strategies. The performance will be measured in terms of accuracy and less data needed. Lastly, we concluded that by using least confidence sampling and margin sampling we reduced the number of labeled samples by 79.25% in comparison with the random sampling strategy for the MNIST dataset. Moreover, by using entropy sampling we reduced the number of labeled samples by 67.92% for the CIFAR-10 dataset.

Keywords

Convolutional Neural Network, Deep Active Learning, Deep Learning, Image Classification, Sampling Strategies, Semi-Supervised Learning.

Sammanfattning

Faltningsnätverk har visat sig leverera bra resultat inom området datorseende, men en fundamental flaskhals med Faltningsnätverk är det faktum att den är starkt beroende av klassificerade datapunkter. I det här examensarbetet hanterar vi Faltningsnätverkens giriga beteende av klassificerade datapunkter genom att använda deep active learning med olika typer av urvalsstrategier. Huvudfokus kommer ligga på urvalsstrategierna slumpmässigt urval, minst tillförlitlig urval, marginal baserad urval, entropi baserad urval och K-means urval. Vi väljer att studera den slumpmässiga urvalsstrategin eftersom att den kommer användas för att mäta prestandan hos de andra urvalsstrategierna. Dessutom valde vi urvalsstrategierna minst tillförlitlig urval, marginal baserad urval, entropi baserad urval eftersom att dessa är osäkerhetsbaserade strategier som är intressanta att jämföra med den geometri-baserade strategin K-means. Dessa urvalsstrategier hjälper till att hitta de mest informativa/representativa datapunkter bland alla oklassificerade datapunkter, vilket gör att vi behöver klassificera färre datapunkter. Vidare kommer standard dastaseten MNIST och CIFAR-10 att användas för att verifiera prestandan för de olika urvalsstrategierna. Slutligen drog vi slutsatsen att genom att använda minst tillförlitlig urval och marginal baserad urval minskade vi mängden klassificerade datapunkter med $79,25\%$, i jämförelse med den slumpmässiga urvalsstrategin, för MNIST-datasetet. Dessutom minskade vi mängden klassificerade datapunkter med $67,92\%$ med hjälp av entropi baserad urval för CIFAR-10-datasetet.

Nyckelord

Bildklassificering, Faltningsnätverk, Deep Active Learning, Djupinlärning, Semiövervakat lärande, Urvalsstrategier.

Acknowledgements

This work is carried out as a Graduate Master's thesis at the Royal Institute of Technology in Stockholm, Sweden. This work would not be possible without the enormous support I have received from people I will acknowledge here.

First of all, I want to express my sincerest gratitude to my supervisor Dr. Hamid Reza Faragardi who enriched this thesis with useful critiques. I also appreciate Dr. Faragardis way of working and the fast replies whenever I needed help. Furthermore, I want to thank Prof. Danica Kragic for being a great teacher that always answers emails with a smile. Not only in this course but other courses as well.

Finally, I want to give a special thanks to my beloved family for always supporting me. This thesis would not be possible without you!

Contents

Chapter 1

Introduction

1.1 Background

In recent years Deep Neural Networks (DNNs) have dominated as a machine learning tool to solve various kinds of tasks. DNN has been extensively explored and has achieved great performance in computer vision tasks such as object detection, face recognition, and image segmentation [42, 57]. One widely used DNN is the Convolutional Neural Network (CNN) that has been proved to deliver great results in the area of computer vision. However, CNNs need lots of labeled data to achieve state-of-the-art performance. This fundamental bottleneck causes the CNNs to be dependant on the ground truth, that is, labeled training data [57]. Compared to other machine learning models CNNs need relatively larger quantities of annotated training data to optimize its massive numbers of parameters [46].

With the growing number of data-generating devices such as smartphones, wearable electronics, and Internet of Things devices we have an abundance of data. Hence, the gathering of unlabelled data is easy and cheap to acquire, but the labeling of data is a time-consuming and expensive task since human efforts and expertise is often required. This is especially true within the area of medical image analysis where labeling is often time-consuming, expensive, and requires the intervention of medical experts that often charges hundreds of dollars an hour [52].

As mentioned before CNNs are greedy for labeled data, thus finding ways to reduce human efforts is of practical importance [20, 43]. One method that mitigates this problem is deep active learning, which is active learning applied to DNNs. Active

learning is a case of semi-supervised machine learning and has been studied for around three decades, but it recently started to gain attention [48]. Active learning is based on the idea that some samples are more important than other samples in terms of learning capabilities. The goal is to find the most informative unlabeled samples such that it is most representative for all the samples. Active learning scenarios can be divided into membership query synthesis [3], stream-based selective sampling [27], and pool-based sampling [29, 36]. These are different ways in which the learner queries the labels for instances. Membership query synthesis is the case where the learner synthesizes an instance similar to the dataset and sends it to an oracle (human annotator). Moreover, the main difference between stream-based selective sampling and pool-based sampling is that the former takes one sample at a time and makes an independent judgment whether we want to query the label, while the latter goes through each sample and chooses the best subset based on the evaluation of the entire dataset [9]. The approach that will be investigated in this research is the pool-based sampling scenario. Firstly, because the stream based selective sampling scenario takes one sample at a time, hence, resulting in insignificant impact on the CNNs accuracy due to local optimization methods [47]. Secondly, because the membership query synthesis mostly generates no recognizable semantic meaning for the human annotator when working in infinite problem domain [4].

The pool-based sampling approach works in cycles, in each cycle, it is trying to find a subset of informative samples amongst all the unlabelled samples. The subset is then sent to an oracle that labels the samples and adds them to the training dataset. The training dataset is then used to train the CNN. The cycle continues until we have reached the labeling budget [37]. Thus, by applying active learning using the pool-based sampling approach we can significantly reduce the amount of manually annotated data and hence reducing the cost of annotation [45].

1.2 Problem

Deep Neural Networks (DNNs) is a state-of-the-art machine learning tool that has shown great performance in solving various machine learning problems within the area of computer vision, speech recognition, and natural language processing. To train a DNN lots of data is needed since DNNs are very complex and often consist of millions of variables. The problem is that in many cases the data has to be labeled and labeling

data is both expensive and time-consuming since there is often human labor involved. The problem is not lack of data, instead, we have millions of terabytes of data that is produced each day. The problem is that produced data is mostly unlabelled and dependant on humans to label it. Imagine we have medical images that we need to set a diagnose on, usually, we would need a group of medical doctors to label these images with a diagnosis since we need to assure these images are correctly labeled and not biased by one medical doctor's opinion. This is a very expensive procedure since each medical doctor costs hundreds of dollars each hour. There are various kinds of methods that try to mitigate this problem, one of them is transfer learning. Transfer learning mitigates this problem by initializing a model with knowledge extracted from another model that has been trained on a similar problem [39]. In this project, we will focus on solving the data greedy bottleneck of Convolutional Neural Network (CNN) with the help of deep active learning. We will experiment with different active learning sampling strategies applied to two different datasets and measure their performance in terms of testing accuracy. The focus will be on the sampling strategies least confidence sampling, margin sampling, entropy sampling, and K-means sampling. The random sampling strategy was chosen since it is widely used as a baseline strategy [42, 45, 50]. Moreover, the least confidence sampling, margin sampling, and entropy sampling strategies are based on uncertainty, unlike the K-means sampling strategy, that is based on geometry. Thus, it will be interesting to further study and compare these strategies and investigate their performance. We will investigate which sampling strategy performs the best in terms of both accuracy and less data needed.

1.3 Research Question

This thesis aims to answer the following research question:

- *Which of the sampling strategies least confidence sampling, margin sampling, entropy sampling, and K-means sampling performs the best in terms of accuracy and less data needed?*

1.4 Scope and Delimitation

The scope of this study will be to explore the performance of the pool-based sampling scenario using various sampling strategies within the area of image classification.

Furthermore, two different CNN architectures with different complexities will be trained and applied to the MNIST and CIFAR-10 datasets respectively.

The fact that both the MNIST and CIFAR-10 dataset have 10 classes each is a delimitation in the sense that this study may not be generalizable to datasets with more or fewer than 10 classes. Furthermore, since we are using one specific kind of machine learning architecture the results could fluctuate between different architectures.

1.5 Purpose

A lot of research has been conducted towards creating various kinds of Deep Neural Network (DNN) architectures to enhance accuracy. Since the machine learning model does not get better than the data it is learning from we can ascertain that feeding our DNNs with high-quality data is important. The purpose of this project is to compare the performance of different sampling strategies and to see which one is superior in minimizing the amount of labeled data needed.

In this project, we will present the results of applying different sampling strategies on well-known and high-quality datasets.

1.6 Objective

The main objective of this project is to provide insights into the performance of different sampling strategies using the pool-based sampling scenario applied to Convolutional Neural Networks. Furthermore, we want to reduce the amount of labeled data by $>= 75\%$.

1.7 Research Methodology

In this research, we compare different sampling strategies applied to the pool-based scenario within the area of image classification. To answer the research question we begin with setting up a pool-based deep active learning environment using various sampling strategies. Thereafter, we implement two different Convolutional Neural Networks with different complexities using Pytorch. We will use the benchmark datasets MNIST and CIFAR-10 to evaluate the performance of the various sampling

strategies. Finally, we use accuracy and the less data needed metric to quantify our research question.

1.8 Ethics and Sustainability

It is vital to discuss the impact this research will have on ethics and sustainability. First and most importantly this research could spare the time of many different professions. Medical doctors are one of these professions, they will not need to spend time labeling more samples than they need to. This will enable medical doctors to focus on more important matters. Furthermore, we believe that this research could help various machine learning models to reach a higher accuracy at a faster rate, hence, enabling more accurate prediction with a lower cost within various areas including diagnosing patients based on X-ray images. As a result of improving the performance of various machine learning models, it is possible that humans get replaced by these models. If machine learning models perform better than humans then it is only a matter of time until we see more professions get replaced by machines. However, one ethical problem that arises when the deciding entity is a machine is who will be responsible when an error occurs. This is especially important when the machine is deciding about a diagnosis. To falsely claim that a patient is negative to cancer when the patient in fact is positive is very serious and normally someone is held responsible. However, when the deciding entity is a machine it gets more complicated. It is important that researchers and engineers keep the job loss consequence in mind when developing technologies that may lead to higher unemployment rates.

As previously mentioned we produce millions of terabytes of data each day. Parts of these data are collected, stored, and processed by data centers. As of today, there are hundreds of thousands of data centers all around the world and they all consume huge amounts of energy. The huge energy consumption is a result of having the data centers constantly running and cooled down. The processing of data and the cooling down process is sometimes powered by natural gas, coal, and other fossil fuels which is not renewable energy and hence has a negative impact on the environment.

The objective of this thesis is to reduce the number of labeled data and not to reduce the amount of total data, hence, we do believe that the proposed work in this thesis may have a short term negative impact on the environment, however, we do believe that on the long term all data centers will be fueled by renewable energy.

1.9 Outline

The remainder of this thesis is organized as follows. Chapter 2 introduces state-of-the-art methods that have been used within the deep active learning domain. Chapter 3 describes the deep active learning pipeline and the experimental setup that has been used in this thesis. Chapter 4 presents the results that are used to answer our research question. Chapter 5 concludes this thesis.

Chapter 2

Theory and Related Works

In this section, we begin by reviewing the different scenarios of active learning and its various strategies. We will also review the semi-supervised machine learning algorithm, active learning, and its application within the area of classification tasks. Furthermore, We will review deep active learning with a focus on the pool-based scenario. The focus will be on the pool-based scenario mainly for two reasons. Firstly, because using a CNN in combination with the stream-based sampling scenario is not recommended since retraining a CNN for one sample at a time has insignificant impact on the CNNs accuracy [47]. Secondly, because membership query synthesis works well when labels are obtained from experiments but not as good when labels are obtained from human annotators.[4].

2.1 Active Learning Scenarios

In this subsection, we review the different scenarios for the semi-supervised machine learning algorithm, active learning. There are different scenarios in which the learner queries the oracle for labels. The three main scenarios are membership query synthesis, stream-based selective sampling, and pool-based sampling [49]. The differences amongst the scenarios are presented in Figure 2.1.

2.1.1 Membership Query Synthesis

Membership Query Synthesis is the first scenario investigated and developed by Dana Angluin [3] in 1988. This scenario queries samples by choosing any unlabeled sample

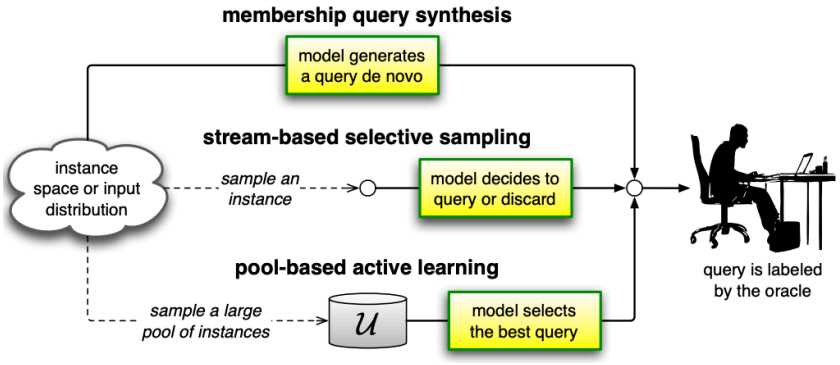

Figure 2.1: Active Learning Scenarios [49]

in the input space regardless of any underlying distribution. It typically queries for samples that are generated synthetically by the learner. Membership query synthesis works best with finite problem domains. When working with infinite problem domains the learner mostly generates samples that have no recognizable semantic meaning for the human oracle [4].

King et al. [25] implemented a useful real-world application using membership query synthesis. They developed a robot that could execute a series of autonomous biological experiments to discover metabolic pathways in a specific yeast. The result was a three-fold decrease in material costs since the robot could form synthetically produced pathways and test these with minimum human intervention.

2.1.2 Stream-Based Selective Sampling

Stream-Based Selective Sampling is a sequential process that draws one sample at a time from a data source. The learner then chooses if the label of the sample should be queried or discarded based on a query strategy. See Section 2.2 to find out more about different query strategies. One important assumption is that the cost of drawing an unlabeled sample from the data source is free [10].

The stream-based selective sampling scenario is well studied and used in several different areas. Three different applications where stream-based selective sampling has been used are i) Loy et al. [33] jointly optimized exploration-exploitation with minimal human supervision, ii) Liu et al. [32] addresses the problem of human activity recognition based on wearable sensors, and iii) and Smailovic et al. [53] analyzes whether the sentiment expressed in Twitter feeds can indicate stock price changes.

2.1.3 Pool-Based Sampling

The pool-based sampling scenario assumes that we already have a pool of data with a small subset that is labeled and the rest unlabeled. Similar to the stream-based selective sampling scenario the pool-based sampling scenario queries samples based on some query strategy, however, the main difference is that the query strategy is applied to all the unlabeled samples in the pool-based sampling scenario instead of one at a time as in the stream-based selective sampling scenario. The unlabeled samples with the most informativeness will be sent to the oracle and the rest will be discarded [49]. Informativeness could be quantified in several different ways, for example, by uncertainty or the geometrical position of a sample in the data space.

The pool-based sampling scenario has been widely used in several different areas within the machine learning domain including regression [62], text classification [36, 55, 63], image classification [1, 22, 41] and sound classification [17, 60].

2.2 Active Learning Strategies

To evaluate which samples should be sent to the oracle we need a sampling strategy. In this subsection, we will review the active learning strategies random sampling, least confidence sampling, margin sampling, entropy sampling [50], and K-means sampling [37]. We choose to focus on the pool-based sampling scenario since this is the scenario that will be investigated in this paper.

2.2.1 Random Sampling

The random sampling strategy is the case when the learner chooses to query the label for a sample randomly. This strategy is often used as a baseline to compare with other strategies to measure performance [34].

2.2.2 Least Confidence Sampling

The least confidence sampling strategy is the case when the learner chooses to query the label of samples which we are uncertain about. For example, assume there is a matrix $M_{p,s}$, where each row is a class and each column is a sample. Then for each sample, there is a vector S_c that contains the probability to belong to a certain class c. We then

choose to query the label of samples that has a small $max(M_{p,s})$ for $s \in \{1, 2, 3.., n\}$ where n is the number of samples. Since we want to avoid exceeding the labeling budget in the pool-based sampling scenario we typically want to control the number of samples that are queried. Hence, we typically choose to query k samples with the least confidence [58].

2.2.3 Margin Sampling

The margin sampling strategy is the case when the learner chooses to query the label of samples where the probability for the most probable class and the second most probable class are close to each other. Those samples that have a small margin between the most probable class and the second most probable class are queried. These samples are chosen to be queried since it is assumed that the learner needs more information from the oracle to discriminate between these two classes [44].

2.2.4 Entropy Sampling

The entropy sampling strategy is an uncertainty measurement that is previously used in several previous studies on the AL domain [8, 54, 66]. Entropy sampling is the case when the learner chooses to query the label of samples with the largest class prediction information entropy. The entropy is calculated as follows:

$$H(x) = -\sum_{y \in Y} P(y|x) log(P(y|x))$$

where $P(y|x)$ is the posteriori probability, H is the uncertainty measurement function and $Y = \{y_1, y_2.., y_k\}$ [67].

2.2.5 K-Means Sampling

James MacQueen [35] developed the K-means sampling strategy in 1967. The purpose of the paper was to describe a process for partitioning a N-dimensional dataset into k sets on the basis of a sample. Essentially, MacQueen is trying to find homogeneous groups of data points in a given dataset. Each homogeneous group is a cluster which is defined as a region where the density of data points is locally higher than in other regions in the N-dimensional space [30]. Each cluster is represented with a centroid in

the N-dimensional space. An example is illustrated in Figure 2.2 where each centroid is represented as a black dot. With respect to the pool-based sampling scenario, $K = m_{query}$ where m_{query} is the number of samples that will be labeled in each round. The data points that will be chosen to be labeled are the data points that is closest to the centroids. The Euclidean distance will be used to calculate the distance. The Euclidean distance d is the shortest route between two points p and q in the N-dimensional space and is calculated as follows [12]:

$$d\left(p,q\right) = \sqrt{\sum_{i=1}^{n}\left(q_i - p_i\right)^2}$$

Since the objective is to calculate if a certain data point is closer to the centroid than any other data point, hence, for optimization purposes the Euclidean distance d is calculated as follows:

$$d\left(p,q\right) = \sum_{i=1}^{n}\left(q_i - p_i\right)^2$$

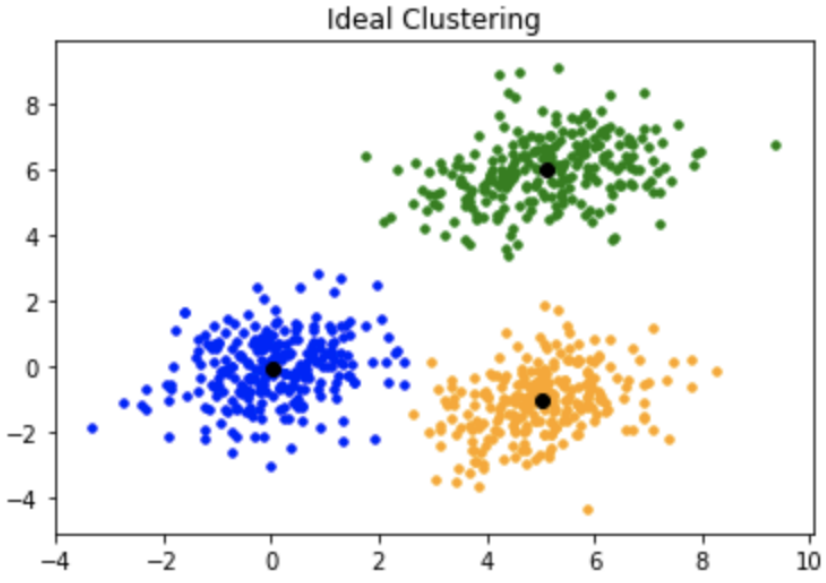

Figure 2.2: K-Means Sampling Strategy, where $K = 3$ [24].

2.3 Active Learning

Plenty of research has been conducted within the area of active learning for classification tasks. Tong and Chang [55] used active learning for Support Vector

Machines (SVMs) using a Radial Basis Function (RBF) kernel to effectively conduct relevance feedback for image retrieval. The paper focused on binary classification limited to linearly separable training data in the feature space. Furthermore, a pool-based approach was chosen with a query strategy that chooses data points close to the hyperplane. Joshi et al. [22] continued on the work of Tong and Chang to create a multi-class AL algorithm to recognize letters and digit images. Similar to Tong and Chang, Joshi et al. used active learning for SVM using an RBF kernel. However, it is complicated to use distance to the hyperplane as a query strategy since there are several hyperplanes in multiclass problems. Hence, Joshi et al. queried the samples that the model was most uncertain about (uncertainty sampling). Since SVM is not a probabilistic model the probability of each sample is not directly available. Hence, Joshi et al. quantified the uncertainty by creating a probability estimation that followed the approach proposed by Lin et al. [31].

Moreover, Bodo et al. [5] conducted two experiments using the clustering algorithm K-means to extract representative samples from the unlabeled dataset. What distinguished the two different experiments is the use of the two different kernel functions polynomial and RBF kernels. Furthermore, a group of SVMs was used to solve the binary classification problem. Using a group of SVMs to vote for which class each sample should have is called bootstrap aggregation. Bootstrap aggregation is used to reduce the average error of the model [6].

2.4 Deep Active Learning

The current state-of-the-art methods within the area of active learning are about deep active learning. Deep active learning has been widely used in several different data greedy areas. This is not surprising since Deep Neural Networks (DNNs) are greedy for data, however, by applying active learning we can reduce the amount of labeled data needed to train the model. Currently, the research focus is on the combination of different kinds of neural networks, active learning scenarios, and sampling strategies.

Ranganathan et al. [42] developed a Deep Belief Network in combination with the pool-based sampling scenario and the active learning strategies MV-Naive, MV-Aggressive, and MV-Conservative to solve an image recognition task. An et al. [2] developed a Recurrent Neural Network in combination with the pool-based sampling

scenario and the active learning strategies least confidence sampling and query-by-committee to solve a text classification task. Roy et al. [45] developed a Shot Multibox Detector in combination with the pool-based scenario and the active learning strategy query-by-committee.

Sener and Savarese [47] developed the core-set sampling strategy in 2018. The goal of the core-set strategy is to find a subset of samples that is most representative of the whole dataset. Sener and Savarese find this subset of samples geometrically by finding a set of points s that covers all other points as shown in Figure 2.3. The model tries to minimize the radius δ. Minimization of this boundary is essentially equal to the K-center problem proposed by Wolf [61]. However, Sener and Savarese use an efficient approximate to solve this combinatorial optimization problem.

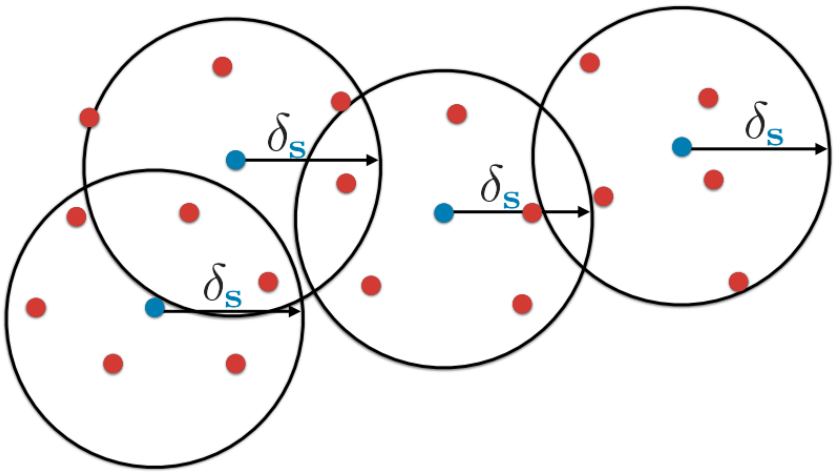

Figure 2.3: Core-Set Sampling Strategy [47].

2.5 Contribution

In this work, we will analyze the performance of several different sampling strategies applied to the pool-based selective sampling scenario. Compared to current research we will apply these strategies on a CNN for image classification tasks. Furthermore, we will quantify the accuracy and how much less labeled data is needed using various sampling strategies.

Chapter 3

Method

This chapter will elaborate on the method used to answer our research question that was given in Section 1.3. First of all, we will elaborate on the deep active learning pipeline that is used to query the labels of samples. Next, we explain the experimental setup used to run our experiments such as datasets, architectures, and the choice of hyperparameters. Furthermore, the evaluation metrics that will be used to quantify our research question will be elaborated. Lastly, the resources used in this research will be presented.

3.1 Deep Active Learning Pipeline

We will now elaborate further on the pool-based active learning scenario. We will follow the general procedure of the pool-based scenario described in the following papers [15, 16, 42, 45]. Moreover, the hyper parameters chosen are inspired by these papers. Due to limited computational resources, the only hyper parameter that will be tuned is the learning rate. We will now explain each step of the pool-based scenario pipeline that is presented in Figure 3.1.

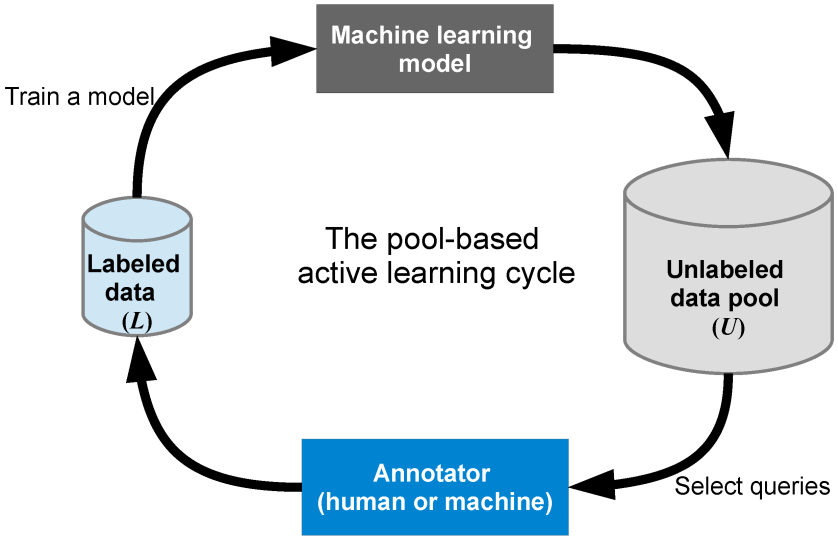

Figure 3.1: Pool-based scenario pipeline [23]

Step 1 - Initialization

Since the datasets that will be used in this experiment are already labeled we will begin by choosing a uniformly selected labeled subset L of our dataset D (see Section 3.2.2 for more details about datasets used). We will initially set $|L| = n$, where $n = 500$. The dataset L will sequentially increase with an amount of $|m_{query}| = 1000$ with each round. We will run each experiment for different numbers of rounds. We will also have an "unlabeled" dataset $U = D - L$ that will sequentially decrease with an amount of $|m_{query}| = 1000$ with each round.

Step 2 - Train a Model

The second step consists of training our CNN using the dataset L.

Step 3 - Accuracy

The third step consists of evaluating the CNN on the test set T and returning the accuracy of the CNN (more about evaluation in Section 3.2.4).

Step 4 - Sampling Strategy

Based on the sampling strategy (more about sampling strategies in Section 2.2) we will choose the top 1000 most informative/representative samples amongst the samples in the unlabelled dataset U and send them to the oracle.

Step 5 - Oracle

The annotator/oracle will label these 1000 samples and add these to the labeled dataset L and remove these from the unlabelled dataset U.

Step 6 - Cycle

We will now go back to step 2 and continue with these steps until we have reached our labeling budget.

3.2 Experimental Setup

3.2.1 Experimental Environment

Pytorch is the machine learning framework used to build the CNN architecture. Pytorch is an open-source Artificial intelligence (AI) framework initially developed by the Facebook AI research group to improve accuracy and performance in the AI community. The Pytorch framework is well known and easy to use as it is designed and assembled to work with Python. The reason for choosing Pytorch over other well-known machine learning frameworks such as Tensorflow is that the Pytorch code is simple to understand and it contains various kinds of optimizers and loss functions. Another reason is that the Pytorch framework operates under eager execution instead of the static execution graph used in Tensorflow [21]. Furthermore, the scientific computing framework Numpy was used to build the sampling strategies least confidence sampling, margin sampling and entropy sampling [18]. Moreover, the machine learning framework scikit learn was used to build the K-means sampling strategy [40].

3.2.2 Dataset Description

For the purpose of this academic research, we have conducted our experiments on the datasets MNIST and CIFAR-10. Both datasets are well known and frequently used in the academic world. These datasets are mostly used as benchmark datasets to measure the performance of various image classification models. Both of these high-quality datasets fulfill the requirements to be used in this thesis since they are well studied and used in the academic world.

The MNIST dataset consists of $70,000$ images of handwritten digits [65]. Each image consists of 28×28 black and white pixels belonging to one of 10 classes *0, 1, 2, 3, 4, 5, 6, 7, 8, and 9*. There are $7,000$ images per class. The dataset is divided into a training set and test set with respectively $60,000$ and $10,000$ images inside. Some examples of images taken from the MNIST dataset are represented in Figure 3.2.

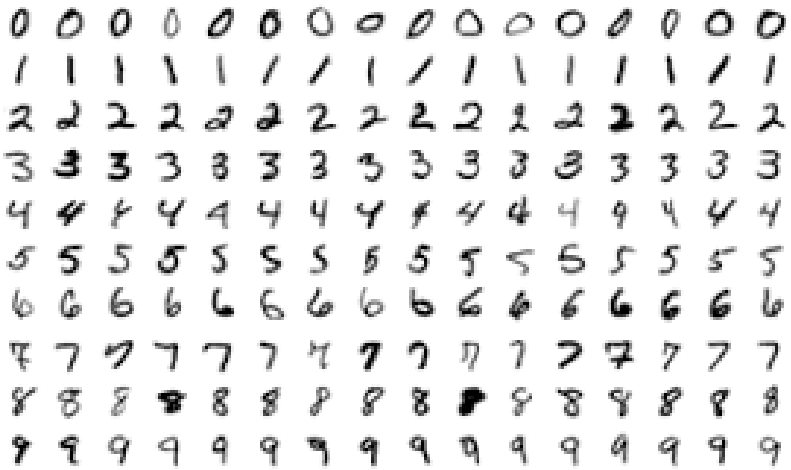

Figure 3.2: MNIST dataset where each row represent a class. [38]

The CIFAR-10 dataset consists of $60,000$ images representing different objects. Each image consists of 32×32 color pixels belongings to one of 10 different classes *airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck*. There are $6,000$ images for each class. The dataset is divided into a training set and test set with respectively $50,000$ and $10,000$ images inside. Each image is mutually exclusive which means that there are no overlapping classes. Some examples of images taken from the CIFAR-10 dataset are represented in Figure 3.3.

Both the MNIST and CIFAR-10 dataset is fully annotated and ready to use for our deep active learning pipeline. The test set will be used to measure the performance of our CNN for each round. We will have an initial uniformly selected labeled dataset $|L| = n$, where $n = 500$. Our CNN will initially train on this initial training set.

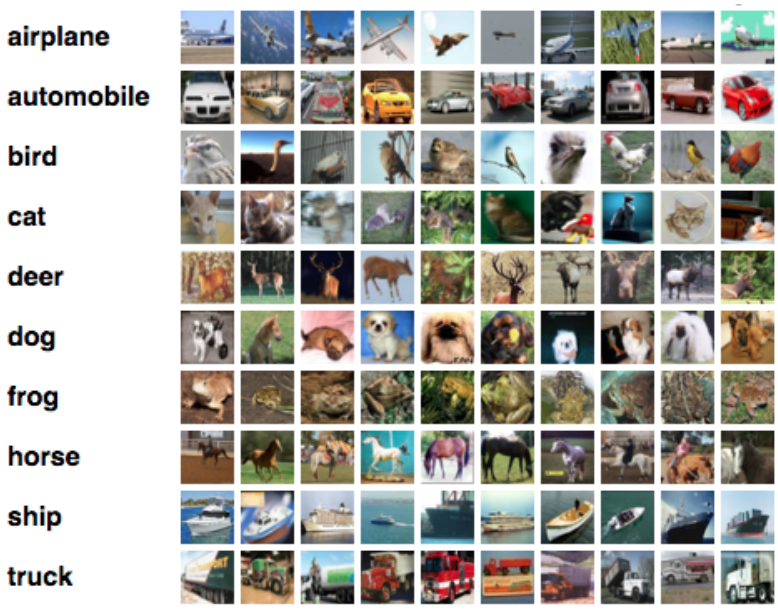

Figure 3.3: CIFAR-10 dataset where each row represent a class. [28]

3.2.3 Convolutional Neural Networks Architectures

Since we have two different datasets with different complexities we will use two different CNN architectures. For the MNIST dataset, we will use the architecture presented in Table 3.1. The architecture of Net1 is inspired by [13, 14, 26]. Moreover, for the CIFAR-10 dataset, we will use the architecture presented in Table 3.2. The architecture of Net2 is inspired by [7, 56].

Net1 - MNIST

Layer Type	in_channels	out_channels	kernel_size	Parameters
Convolution	1	10	5x5	-
Max Pooling	-	-	2x2	-
Relu	-	-	-	-
Convolution	10	20	5x5	-
Dropout	-	-	-	p = 0.5
Max Pooling	-	-	2x2	-
Relu	-	-	-	-
Linear	320	50	-	-
Relu	-	-	-	-
Dropout	-	-	-	p = 0.5
Linear	50	10	-	-
Softmax	-	-	-	-

Table 3.1: Net1 CNN Architecture used with the MNIST dataset

NET2 - CIFAR-10

Layer Type	in_channels	out_channels	kernel_size	Parameters
Convolution	3	64	3x3	-
Max Pooling	-	-	2x2	-
Relu	-	-	-	-
Convolution	64	128	3x3	-
Max Pooling	-	-	2x2	-
Relu	-	-	-	-
Convolution	128	256	3x3	-
Max Pooling	-	-	2x2	-
Relu	-	-	-	-
Linear	1024	128	-	-
Relu	-	-	-	-
Linear	128	256	-	-
Relu	-	-	-	-
Linear	256	10	-	-
Softmax	-	-	-	-

Table 3.2: Net2 CNN Architecture used with the CIFAR-10 dataset

Hyperparameters

The choice of hyperparameter can be found in Table 3.3. These hyperparameters was inspired by [7, 13, 14, 26, 56]. You may assume that those hyperparameters not mentioned are using their default values. It will be mentioned clearly in case we modify the hyperparameters during the experiment process.

Hyperparameter	Value
Optimizer	Stochastic Gradient Decent
Loss Function	Cross Entropy
Learning Rate	0.01
Momentum	0.5
Mini-Batch Size	64

Table 3.3: Hyperparameter settings for Net1 & Net2

3.2.4 Evaluation

In the literature, we found out that most papers researching Deep Active Learning used the evaluation metrics F1-score and accuracy to measure the performance of their machine learning models [16, 42, 50, 59, 64]. F1-score is preferred as a measuring

tool when the dataset is uneven. Accuracy is preferred as a measuring tool when false positives and false negatives have similar costs [11, 19, 51]. Since we have a balanced dataset and the cost of false positives and false negatives are similar we will choose accuracy as a measuring tool to measure the performance of our machine learning model. If we would use F1-score we would get a similar score as the accuracy score of the reasons mentioned previously. We calculate accuracy as described in Equation 3.1.

$$Accuracy = \frac{|\text{correctly classified samples}|}{|\text{all samples}|} \tag{3.1}$$

Furthermore, we will need to quantify how many fewer labeled samples we need when using various sampling strategies. We will use the metric Less Data Needed (LDN) which is calculated as in Equation 3.2. This metric takes the number of labeled samples needed for reaching the same maximum accuracy as the random sampling strategy divided by the number of labeled samples the random sampling strategy needed to reach its maximum accuracy. We compare with the random sampling strategy since this strategy only queries the labels for samples at random. Hence, it is possible to measure the performance of the other sampling strategies relative to the random sampling strategy to investigate the performance of using samples informativeness into account.

$$LDN = \frac{\text{samples needed for strategy } x \text{ to reach random sampling max accuracy}}{\text{samples needed for random sampling strategy to reach its max accuracy}} \tag{3.2}$$

3.2.5 Resources

We will run all experiments in Section 4 on a Macbook Pro from 2018 with the following specifications:

- **Processor:** 2,2 GHz 6-Core Intel Core i7
- **RAM:** 16 GB 2400 MHz DDR4
- **Graphics 1:** Radeon Pro 555X 4 GB
- **Graphics 2:** Intel UHD Graphics 630 1536 MB

Chapter 4

Results

In this chapter, we will present the experimental results. With the help of our results in this chapter, we will answer the research question described in Section 1.3. In Section 4.1 and Section 4.2 we will present the results of our deep active learning pipeline trained on the MNIST and CIFAR-10 dataset respectively. In each of these sections, we will present the result of each sampling strategy with various learning rates. At the end of these aforementioned sections we will extend the results by training the Net1 and Net2 models with a learning rate set to 0.05 with a labeling budget of $30,500$ (see Sections 4.1.6 and 4.2.6). Furthermore, we will demonstrate the extended results of each sampling strategy together to get a more comparing view of the performance of the different sampling strategies. In Section 4.3 we will sum up the important findings.

4.1 MNIST Results

The results presented in this chapter are generated with the use of Net1 (See Section 3.2.3).

4.1.1 Random Sampling

The random sampling strategy will be used as a baseline to compare with other sampling strategies. The results presented in Figure 4.1 shows the impact of applying random sampling on the MNIST dataset. In Table 4.1 a more detailed view of the ten first rounds and their accuracy are demonstrated. Each row in Table 4.1 represents the

results for some specific learning rate and each column represents the accuracy of the model trained on some specific number of labeled samples.

The highest accuracy achieved is 97.14% with a learning rate of 0.05 and with 9500 labeled samples. In Section 4.1.6 we will extend the results by evaluating the model for more labeled samples with a learning rate set to 0.05.

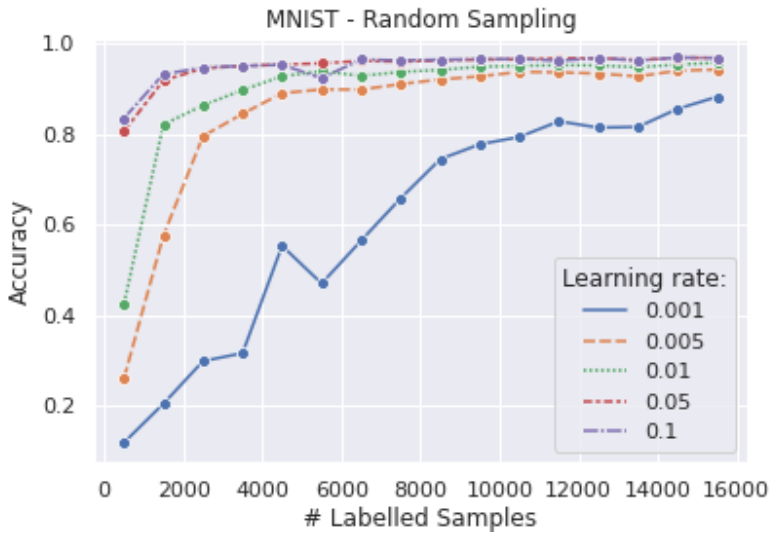

Figure 4.1: Comparison of different learning rates using Random Sampling strategy applied to the MNIST dataset.

eta \ #	500	1500	2500	3500	4500	5500	6500	7500	8500	9500
0.001	0.1176	0.2088	0.3029	0.3207	0.5582	0.4748	0.5699	0.6636	0.7500	0.7829
0.005	0.2595	0.5811	0.8009	0.8502	0.8961	0.9042	0.9041	0.9161	0.9264	0.9332
0.01	0.4240	0.8266	0.8691	0.9028	0.9335	0.9445	0.9343	0.9430	0.9475	0.9546
0.05	0.8066	0.9242	0.9509	0.9562	0.9594	0.962	0.9674	0.9658	0.9684	**0.9714**
0.1	0.8354	0.938	0.9528	0.9563	0.9588	0.9293	0.9706	0.9685	0.9701	0.9710

Table 4.1: Detailed comparison of different learning rates (eta) using Random Sampling strategy applied to the MNIST dataset.

4.1.2 Least Confidence Sampling

The results presented in Figure 4.1 shows the impact of applying least confidence sampling on the MNIST dataset. In Table 4.2 a more detailed view of the ten first rounds and its accuracy are demonstrated. Each row in Table 4.2 represents the results for some specific learning rate and each column represents the accuracy of the model trained on some specific number of labeled samples.

The highest accuracy achieved is 98.82% with a learning rate of 0.05 and with 7500 labeled samples. In Section 4.1.6 we will extend the results by evaluating the model for more labeled samples with a learning rate set to 0.05.

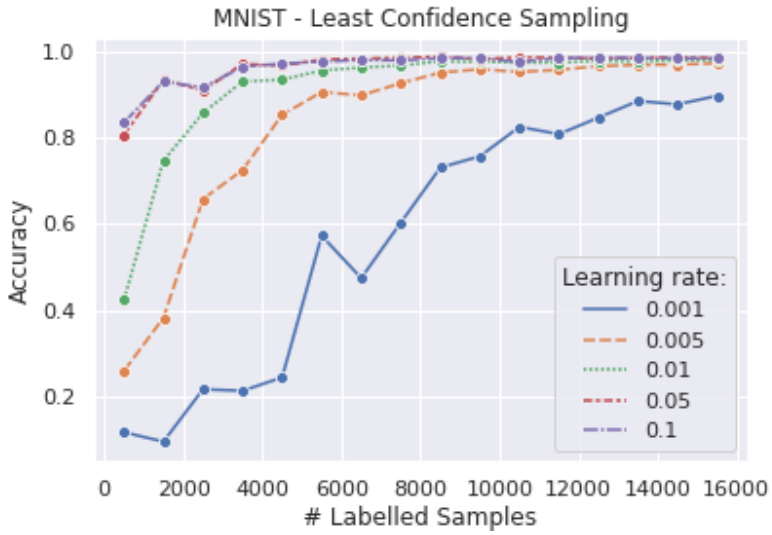

Figure 4.2: Comparison of different learning rates using Least Confidence Sampling strategy applied to the MNIST dataset.

eta \ #	500	1500	2500	3500	4500	5500	6500	7500	8500	9500
0.001	0.1176	0.0959	0.2172	0.2137	0.2446	0.573	0.4749	0.6032	0.7308	0.7571
0.005	0.2595	0.3827	0.6582	0.7251	0.8532	0.906	0.8983	0.9264	0.9511	0.9588
0.01	0.424	0.7471	0.8574	0.9306	0.9346	0.956	0.9627	0.9677	0.9771	0.9764
0.05	0.8066	0.9345	0.9096	0.9711	0.9673	0.9812	0.9834	0.9851	**0.9882**	0.9821
0.1	0.8354	0.9305	0.9162	0.9647	0.9719	0.9766	0.9801	0.9793	0.9846	0.9844

Table 4.2: Detailed comparison of different learning rates (eta) using Least Confidence Sampling strategy applied to the MNIST dataset.

4.1.3 Margin Sampling

The results presented in Figure 4.3 shows the impact of applying margin sampling on the MNIST dataset. In Table 4.3 a more detailed view of the ten first rounds and their accuracy are demonstrated. Each row in Table 4.3 represents the results for some specific learning rate and each column represents the accuracy of the model trained on some specific number of labeled samples.

The highest accuracy achieved is 98.58% with a learning rate of 0.05 and with 8500 labeled samples. In Section 4.1.6 we will extend the results by evaluating the model for more labeled samples with a learning rate set to 0.05.

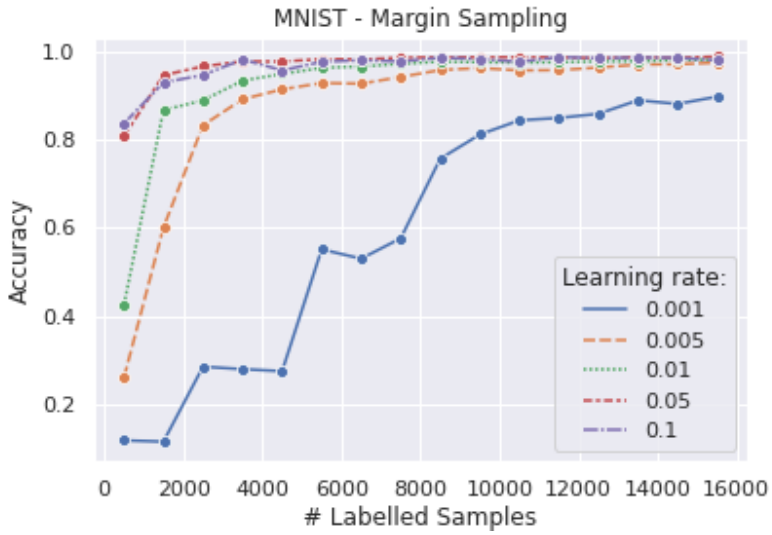

Figure 4.3: Comparison of different learning rates using Margin Sampling strategy applied to the MNIST dataset.

eta \ #	500	1500	2500	3500	4500	5500	6500	7500	8500	9500
0.001	0.1176	0.1148	0.2846	0.2791	0.2745	0.5505	0.5294	0.5756	0.7561	0.8107
0.005	0.2595	0.5999	0.8317	0.8907	0.9131	0.9274	0.9260	0.9412	0.9569	0.9614
0.01	0.4240	0.8663	0.8880	0.9329	0.9480	0.9619	0.9649	0.9709	0.9758	0.9748
0.05	0.8066	0.9451	0.9652	0.9763	0.9762	0.9817	0.9809	0.9848	**0.9858**	0.9858
0.1	0.8354	0.9286	0.9455	0.9808	0.9561	0.9756	0.9790	0.9765	0.9848	0.9817

Table 4.3: Detailed comparison of different learning rates (eta) using Margin Sampling strategy applied to the MNIST dataset.

4.1.4 Entropy Sampling

The results presented in Figure 4.4 shows the impact of applying entropy sampling on the MNIST dataset. In Table 4.4 a more detailed view of the ten first rounds and their accuracy are demonstrated. Each row in Table 4.4 represents the results for some specific learning rate and each column represents the accuracy of the model trained on some specific number of labeled samples.

The highest accuracy achieved is 98.54% with a learning rate of 0.05 and with 8500 labeled samples. In Section 4.1.6 we will extend the results by evaluating the model for more labeled samples with a learning rate set to 0.05.

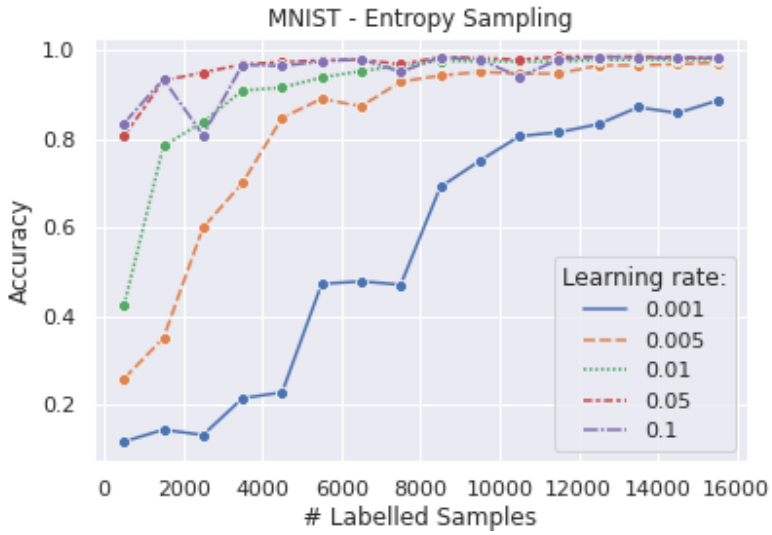

Figure 4.4: Comparison of different learning rates using Entropy Sampling strategy applied to the MNIST dataset.

eta \ #	500	1500	2500	3500	4500	5500	6500	7500	8500	9500
0.001	0.1176	0.1445	0.1326	0.2155	0.2287	0.4735	0.4792	0.4717	0.6919	0.7513
0.005	0.2595	0.3512	0.6004	0.7011	0.8484	0.8910	0.8742	0.9296	0.9434	0.9519
0.01	0.4240	0.7846	0.8391	0.9097	0.9164	0.9390	0.9538	0.9654	0.9765	0.9758
0.05	0.8066	0.9321	0.9503	0.9687	0.9740	0.9777	0.9799	0.9686	**0.9854**	0.9831
0.1	0.8354	0.9351	0.8092	0.9677	0.9651	0.9753	0.9801	0.9518	0.9848	0.9798

Table 4.4: Detailed comparison of different learning rates (eta) using Entropy Sampling strategy applied to the MNIST dataset.

4.1.5 K-Means Sampling

The results presented in Figure 4.5 shows the impact of applying margin sampling on the MNIST dataset. In Table 4.5 a more detailed view of the ten first rounds and their accuracy are demonstrated. Each row in Table 4.5 represents the results for some specific learning rate and each column represents the accuracy of the model trained on some specific number of labeled samples.

The highest accuracy achieved is 97.83% with a learning rate of 0.05 and with 8500 labeled samples. In Section 4.1.6 we will extend the results by evaluating the model for more labeled samples with a learning rate set to 0.05.

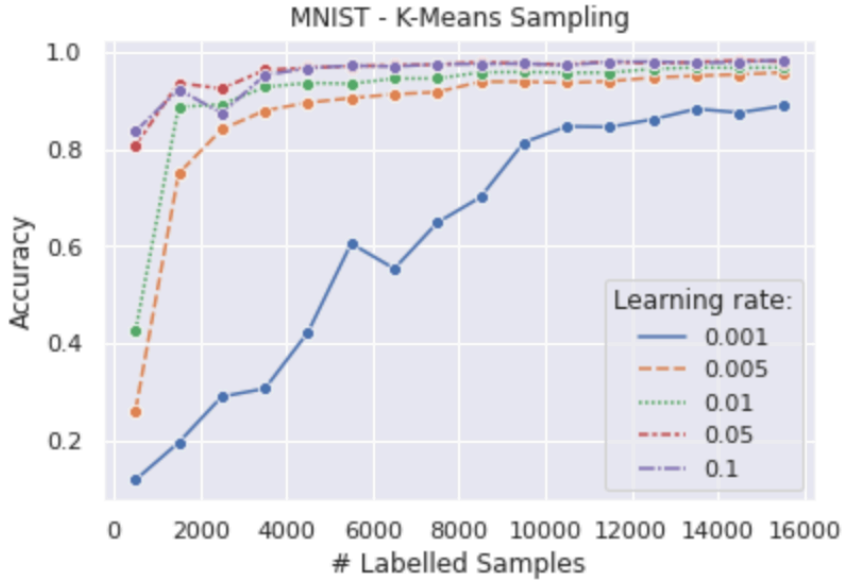

Figure 4.5: Comparison of different learning rates using K-Means Sampling strategy applied to the MNIST dataset.

eta \ #	500	1500	2500	3500	4500	5500	6500	7500	8500	9500
0.001	0.1176	0.1945	0.2891	0.3057	0.4220	0.6047	0.5528	0.6479	0.7002	0.8128
0.005	0.2595	0.7486	0.8398	0.8785	0.8950	0.9043	0.9121	0.9167	0.9380	0.9381
0.01	0.4240	0.8862	0.8910	0.9270	0.9355	0.9338	0.9445	0.9447	0.9571	0.9581
0.05	0.8066	0.9353	0.9236	0.9620	0.9665	0.9702	0.9717	0.9733	**0.9783**	0.9751
0.1	0.8354	0.9221	0.8731	0.9513	0.9661	0.9710	0.9693	0.9740	0.9746	0.9754

Table 4.5: Detailed comparison of different learning rates (eta) using K-Means Sampling strategy applied to the MNIST dataset.

4.1.6 Summary - MNIST

In Figure 4.6 we have trained our network sequentially until we reach our labeling budget of $30,500$. We notice that the random sampling strategies perform worse, in terms of accuracy, compared with the other sampling strategies. The highest accuracy achieved with the use of random sampling is 98.00%. This accuracy was achieved after labeling $26,500$ samples.

The highest accuracy achieved with the use of least confidence sampling is 98.82%. This accuracy was achieved after labeling $8,500$ samples. However, it achieved an accuracy of 98.12% after only $5,500$ labeled samples, compared to random sampling that needed $26,500$ labeled samples to achieve the same accuracy. This implies that we need to label 79.25% less training data than the random sampling strategy needed to achieve the same performance in terms of accuracy.

The highest accuracy achieved with the use of margin sampling is 98.82%. This accuracy was achieved after labeling $28,500$ samples. However, it achieved an accuracy of 98.17% after only $5,500$ labeled samples, compared to random sampling that needed $26,500$ labeled samples to achieve a smaller accuracy of 98.12%. This implies that we need to label 79.25% less training data than the random sampling strategy needed to achieve the same performance in terms of accuracy.

The highest accuracy achieved with the use of entropy sampling is 98.80%. This accuracy was achieved after labeling $17,500$ samples. However, it achieved an accuracy of 98.54% after only $8,500$ labeled samples, compared to random sampling that needed $26,500$ labeled samples to achieve a smaller accuracy of 98.12%. This implies that we need to label 67.92% less training data than the random sampling strategy needed to achieve the same performance in terms of accuracy.

The highest accuracy achieved with the use of K-means sampling is 98.75%. This accuracy was achieved after labeling $29,500$ samples. However, it achieved an accuracy of 98.18% after only $14,500$ labeled samples, compared to random sampling that needed $26,500$ labeled samples to achieve a smaller accuracy of 98.12%. This implies that we need to label 45.28% less training data than the random sampling strategy needed to achieve the same performance in terms of accuracy.

Figure 4.6: Comparison of different sampling strategies applied to the MNIST dataset using a learning rate of 0.05. The sampling strategies compared are Random Sampling (RS), Least Confidence Sampling (LCS), Margin Sampling (MS), Entropy Sampling (ES), and K-Means Sampling (KMS).

An interesting measurement to look at is the overall accuracy gap between the random sampling strategy and the rest of the sampling strategies. The random sampling accuracy gap is shown in Figure 4.7. For the first 4.500 samples we observe an uptrend in the accuracy gap, hence, the sampling strategies successfully find representative samples early in the process. This do also imply that the sampling strategies help the network to converge at a faster rate compared to random sampling. In the bigger picture, an interesting finding is an overall downtrend in Figure 4.7. The observation is that as more samples are labeled the effect of using a sampling strategy decreases. In other words, while the random sampling strategy continues to reach higher accuracy, the other sampling strategies have already begun to converge which leads to the overall downtrend. To further illustrate the uptrend and downtrend a brown and black line has been fitted to the average accuracy gap respectively in Figure 4.8. The trend line has been shifted upwards to make it more apparent.

Lastly, an interesting observation is that the random sampling accuracy gap converges to 0 as we train with more labeled samples. This is intuitive since we should get close to the same accuracy for all various sampling strategies as we are training with the same training set using the same network. This is also seen in Figure 4.6 where we are approaching the same accuracy as we move towards labeling all samples.

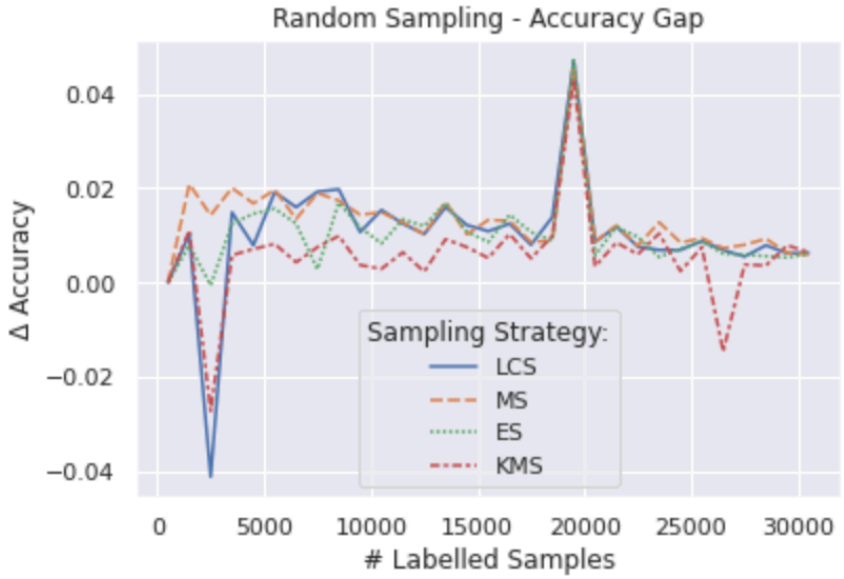

Figure 4.7: Comparison of the accuracy gap in Figure 4.6 between Random Sampling (RS) and Least Confidence Sampling (LCS), Margin Sampling (MS), Entropy Sampling (ES), and K-Means Sampling (KMS).

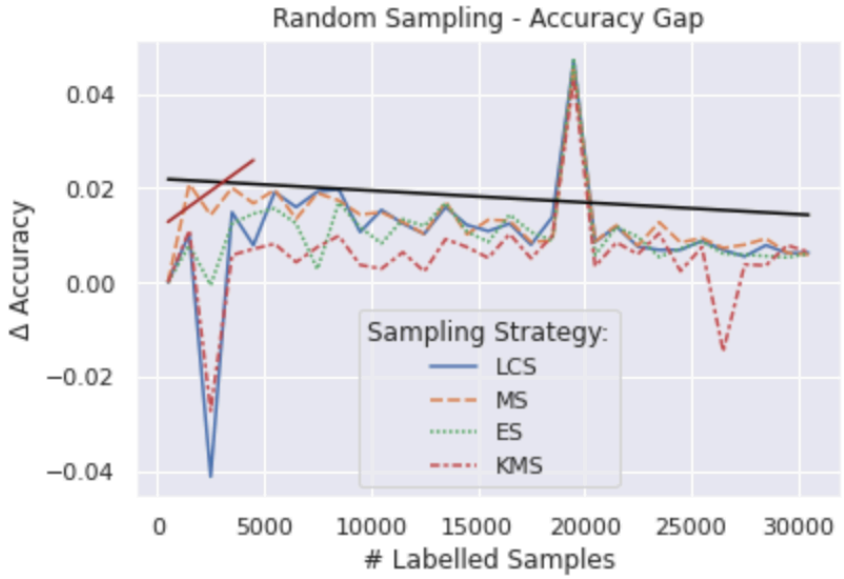

Figure 4.8: Comparison of the accuracy gap in Figure 4.6 between Random Sampling (RS) and Least Confidence Sampling (LCS), Margin Sampling (MS), Entropy Sampling (ES), and K-Means Sampling (KMS). The black line is the fitted line for the average accuracy gap shifted a little bit upward for clarity.

4.2 CIFAR-10 Results

The results presented in this chapter are generated with the use of Net2 (See Section 3.2.3).

4.2.1 Random Sampling

The Random Sampling strategy will be used as a baseline to compare with other sampling strategies. The results presented in Figure 4.9 shows the impact of applying random sampling on the CIFAR-10 dataset. In Table 4.6 a more detailed view of the ten first rounds and their accuracy are demonstrated. Each row in Table 4.6 represents the results for some specific learning rate and each column represents the accuracy of the model trained on some specific number of labeled samples.

The highest accuracy achieved is 56.82% with a learning rate of 0.1 and with 9500 labeled samples. In Section 4.2.6 we will extend the results by evaluating the model for more labeled samples with a learning rate set to 0.05.

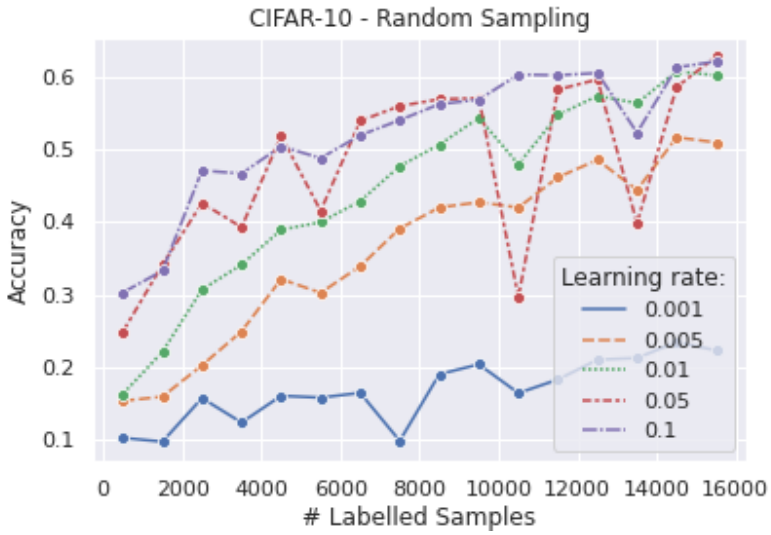

Figure 4.9: Comparison of different learning rates using Random Sampling strategy applied to the CIFAR-10 dataset.

# / eta	500	1500	2500	3500	4500	5500	6500	7500	8500	9500
0.001	0.1024	0.09730	0.1572	0.1236	0.1604	0.1579	0.1644	0.0983	0.1901	0.2037
0.005	0.1534	0.1592	0.2024	0.2491	0.3211	0.3022	0.3388	0.3907	0.4197	0.4269
0.01	0.1628	0.2210	0.3067	0.3409	0.3890	0.3995	0.4285	0.4766	0.5062	0.5429
0.05	0.2488	0.3306	0.4159	0.3825	0.5082	0.4051	0.5289	0.5494	0.5587	0.5603
0.1	0.3027	0.3325	0.4706	0.4664	0.5039	0.4877	0.5191	0.5400	0.5620	**0.5682**

Table 4.6: Detailed comparison of different learning rates (eta) using Random Sampling strategy applied to the CIFAR-10 dataset.

4.2.2 Least Confidence Sampling

The results presented in Figure 4.10 shows the impact of applying least confidence sampling on the CIFAR-10 dataset. In Table 4.7 a more detailed view of the ten first rounds and their accuracy are demonstrated. Each row in Table 4.7 represents the results for some specific learning rate and each column represents the accuracy of the model trained on some specific number of labeled samples.

The highest accuracy achieved is 59.54% with a learning rate of 0.1 and with 9500 labeled samples. In Section 4.2.6 we will extend the results by evaluating the model for more labeled samples with a learning rate set to 0.05.

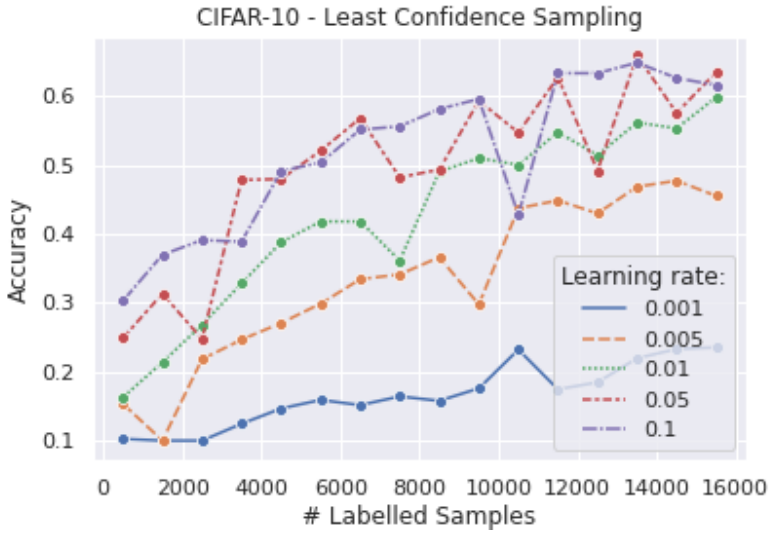

Figure 4.10: Comparison of different learning rates using Least Confidence Sampling strategy applied to the CIFAR-10 dataset.

# / eta	500	1500	2500	3500	4500	5500	6500	7500	8500	9500
0.001	0.1024	0.1000	0.1000	0.1244	0.1466	0.1590	0.1515	0.1641	0.1577	0.1758
0.005	0.1534	0.1000	0.2180	0.2465	0.2705	0.2986	0.3341	0.3410	0.3660	0.2975
0.01	0.1628	0.2137	0.2680	0.3279	0.3875	0.4179	0.4178	0.3613	0.4900	0.5096
0.05	0.2488	0.3131	0.2472	0.4781	0.4793	0.5204	0.5675	0.4818	0.4930	0.5928
0.1	0.3027	0.3695	0.3908	0.3885	0.4911	0.5032	0.5510	0.5557	0.5808	**0.5954**

Table 4.7: Detailed comparison of different learning rates (eta) using Least Confidence Sampling strategy applied to the CIFAR-10 dataset.

4.2.3 Margin Sampling

The results presented in Figure 4.11 shows the impact of applying margin sampling on the CIFAR-10 dataset. In Table 4.8 a more detailed view of the ten first rounds and their accuracy are demonstrated. Each row in Table 4.8 represents the results for some specific learning rate and each column represents the accuracy of the model trained on some specific number of labeled samples.

The highest accuracy achieved is 60.71% with a learning rate of 0.05 and with 8500 labeled samples. In Section 4.2.6 we will extend the results by evaluating the model for more labeled samples with a learning rate set to 0.05.

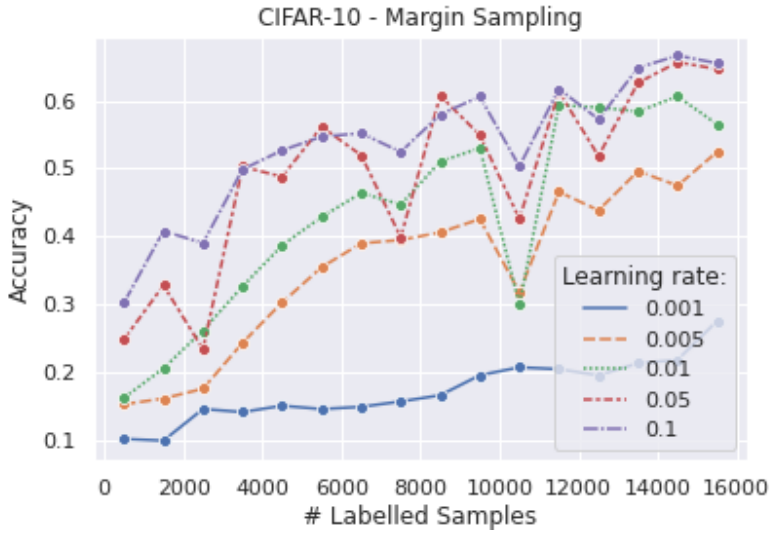

Figure 4.11: Comparison of different learning rates using Margin Sampling strategy applied to the CIFAR-10 dataset.

# / eta	500	1500	2500	3500	4500	5500	6500	7500	8500	9500
0.001	0.1024	0.1000	0.1465	0.142	0.1513	0.1464	0.1497	0.1576	0.1667	0.1962
0.005	0.1534	0.1619	0.1763	0.2438	0.3031	0.3550	0.3901	0.395	0.4062	0.4258
0.01	0.1628	0.2063	0.2621	0.3266	0.3873	0.4289	0.4641	0.4471	0.5104	0.5303
0.05	0.2488	0.3283	0.2343	0.5029	0.4880	0.5616	0.5196	0.3990	**0.6071**	0.5492
0.1	0.3027	0.4081	0.3905	0.4988	0.5269	0.5470	0.5520	0.5245	0.5796	0.6059

Table 4.8: Detailed comparison of different learning rates (eta) using Least Confidence Sampling strategy applied to the CIFAR-10 dataset.

4.2.4 Entropy Sampling

The results presented in Figure 4.11 shows the impact of applying entropy sampling on the CIFAR-10 dataset. In Table 4.9 a more detailed view of the ten first rounds and their accuracy are demonstrated. Each row in Table 4.9 represents the results for some specific learning rate and each column represents the accuracy of the model trained on some specific number of labeled samples.

The highest accuracy achieved is 59.13% with a learning rate of 0.05 and with 9500 labeled samples. In Section 4.2.6 we will extend the results by evaluating the model for more labeled samples with a learning rate set to 0.05.

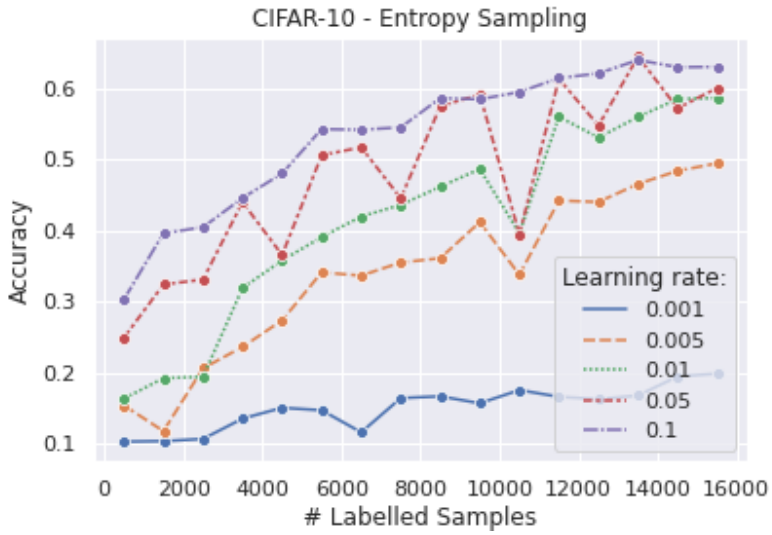

Figure 4.12: Comparison of different learning rates using Entropy Sampling strategy applied to the CIFAR-10 dataset.

# / eta	500	1500	2500	3500	4500	5500	6500	7500	8500	9500
0.001	0.1024	0.1031	0.1062	0.1348	0.1501	0.1466	0.1155	0.1637	0.1662	0.1564
0.005	0.1534	0.1172	0.2056	0.2358	0.2732	0.3407	0.3362	0.3547	0.3607	0.4116
0.01	0.1628	0.1918	0.1938	0.3187	0.3576	0.3904	0.4194	0.4358	0.4614	0.4863
0.05	0.2488	0.3237	0.3307	0.4395	0.3653	0.5057	0.5169	0.4451	0.5747	**0.5913**
0.1	0.3027	0.3958	0.4044	0.4455	0.4801	0.5425	0.5416	0.5450	0.5861	0.5846

Table 4.9: Detailed comparison of different learning rates (eta) using Least Entropy Sampling strategy applied to the CIFAR-10 dataset.

4.2.5 K-Means Sampling

The results presented in Figure 4.13 shows the impact of applying K-means sampling on the CIFAR-10 dataset. In Table 4.10 a more detailed view of the ten first rounds and their accuracy are demonstrated. Each row in Table 4.10 represents the results for some specific learning rate and each column represents the accuracy of the model trained on some specific number of labeled samples.

The highest accuracy achieved is 59.13% with a learning rate of 0.1 and with 8500 labeled samples. In Section 4.2.6 we will extend the results by evaluating the model for more labeled samples with a learning rate set to 0.05.

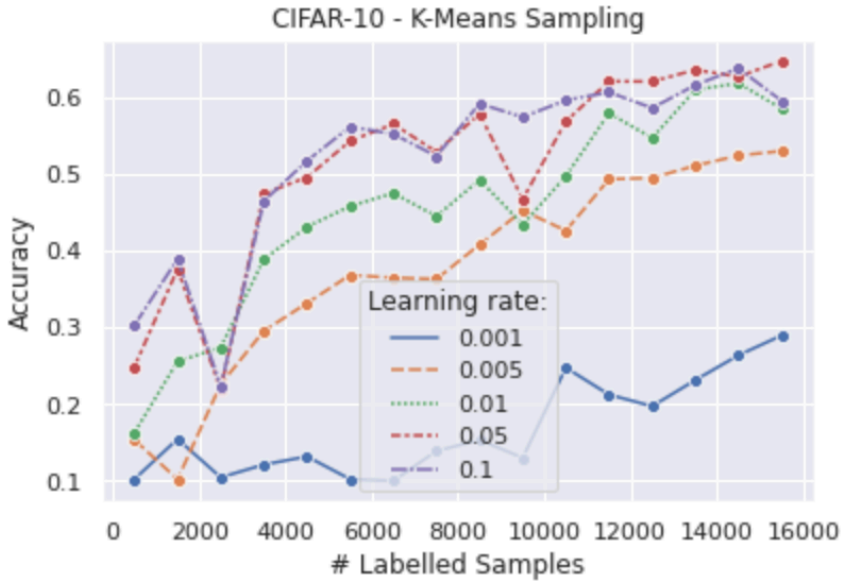

Figure 4.13: Comparison of different learning rates using Entropy Sampling strategy applied to the CIFAR-10 dataset.

# / eta	500	1500	2500	3500	4500	5500	6500	7500	8500	9500
0.001	0.1024	0.1545	0.1046	0.1212	0.1318	0.1022	0.1000	0.1389	0.1532	0.1301
0.005	0.1534	0.1021	0.2259	0.2954	0.3315	0.3681	0.3645	0.3628	0.4078	0.4521
0.01	0.2556	0.2738	0.3887	0.4308	0.4578	0.4747	0.4446	0.4922	0.4336	0.4970
0.05	0.2488	0.3759	0.2194	0.4741	0.4954	0.5427	0.5647	0.5286	0.5777	0.4669
0.1	0.3027	0.3891	0.2232	0.4646	0.5167	0.5613	0.5517	0.5230	**0.5913**	0.5736

Table 4.10: Detailed comparison of different learning rates (eta) using Least Entropy Sampling strategy applied to the CIFAR-10 dataset.

4.2.6 Summary - CIFAR

In Figure 4.14 we have trained our network sequentially until we reach our labeling budget of $30,500$. We notice that the random sampling strategies perform worse, in terms of accuracy, compared with the other sampling strategies. The highest accuracy achieved with the use of random sampling is 65.99%. This accuracy was achieved after labeling $29,500$ samples.

The highest accuracy achieved with the use of least confidence sampling is 71.57%. This accuracy was achieved after labeling $29,500$ samples. However, it achieved an accuracy of 66.31% after only $16,500$ labeled samples, compared to random sampling that needed $29,500$ labeled samples to achieve a smaller accuracy of 65.99%. This implies that we need to label 44.06% less training data than the random sampling strategy needed to achieve the same performance in terms of accuracy.

The highest accuracy achieved with the use of margin sampling is 71.75%. This accuracy was achieved after labeling $29,500$ samples. However, it achieved an accuracy of 66.10% after only $16,500$ labeled samples, compared to random sampling that needed $29,500$ labeled samples to achieve a smaller accuracy of 65.99%. This implies that we need to label 44.06% less training data than the random sampling strategy needed to achieve the same performance in terms of accuracy.

The highest accuracy achieved with the use of entropy sampling is 71.53%. This accuracy was achieved after labeling $29,500$ samples. However, it achieved an accuracy of 66.11% after only $16,500$ labeled samples, compared to random sampling that needed $29,500$ labeled samples to achieve a smaller accuracy of 65.99%. This implies that we need to label 44.06% less training data than the random sampling strategy needed to achieve the same performance in terms of accuracy.

The highest accuracy achieved with the use of K-means sampling is 69.72%. This accuracy was achieved after labeling $29,500$ samples. However, it achieved an accuracy of 66.02% after only $19,500$ labeled samples, compared to random sampling that needed $29,500$ labeled samples to achieve a smaller accuracy of 65.99%. This implies that we need to label 33.90% less training data than the random sampling strategy needed to achieve the same performance in terms of accuracy.

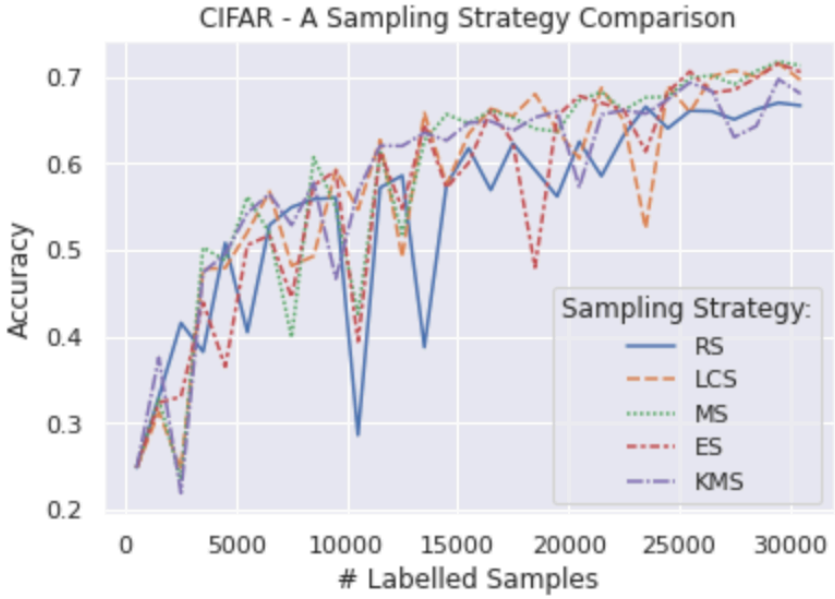

Figure 4.14: Comparison of different sampling strategies applied to the CIFAR-10 dataset using a learning rate of 0.05. The sampling strategies compared are Random Sampling (RS), Least Confidence Sampling (LCS), Margin Sampling (MS), Entropy Sampling (ES), and K-Means Sampling (KMS).

An interesting measurement to look at is the overall accuracy gap between the random sampling strategy and the rest of the sampling strategies. The random sampling accuracy gap is shown in Figure 4.15. An interesting finding is that we have an overall uptrend, however, the uptrend is not as evident for the second half of Figure 4.15. To further illustrate this we plot three different trend lines in Figure 4.16. The black line is the overall trend and has been shifted upward for clarity, the brown line is the trend for the first half, and the pink line is the trend for the second half. The brown line is a consequence of having too few labeled samples to be representative of the dataset. Since the CIFAR-10 dataset is more complex than the MNIST dataset we see that the uptrend continues for more rounds compared to the MNIST dataset. However, the uptrend also indicates that the sampling strategies help our network to converge at a faster rate compared to the random sampling strategy. Hence, as we label more samples we begin to find a representative labeled pool of samples. The result is that

the rate at which we reach higher accuracy slows down for the sampling strategies compared to the random sampling strategy, hence the brown trend-line.

Another interesting observation is that the random sampling accuracy gap converges to 0 as we train with more labeled samples. This is intuitive since we should get close to the same accuracy for all various sampling strategies as we are training with the same training set using the same network. This is also seen in Figure 4.14 where we are approaching the same accuracy as we move towards labeling more samples.

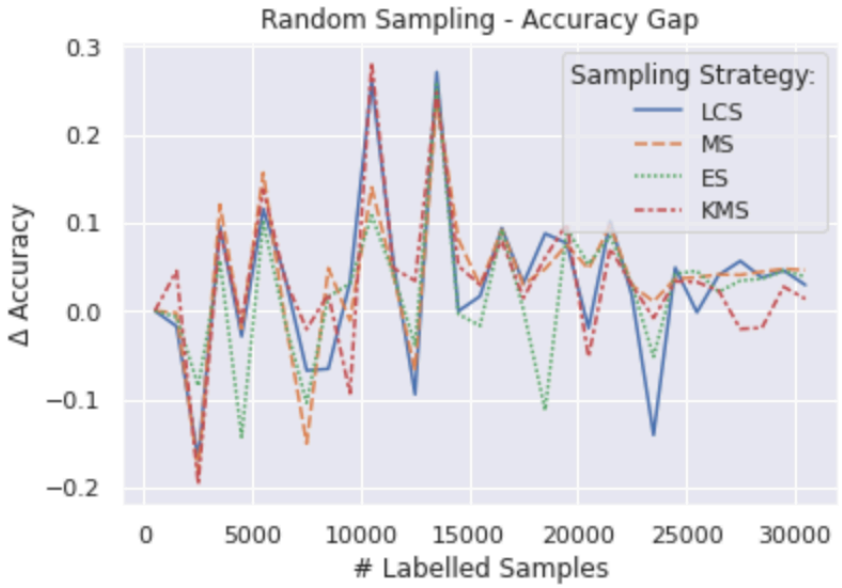

Figure 4.15: Comparison of the accuracy gap in Figure 4.14 between Random Sampling (RS) and Least Confidence Sampling (LCS), Margin Sampling (MS), Entropy Sampling (ES), and K-Means Sampling (KMS).

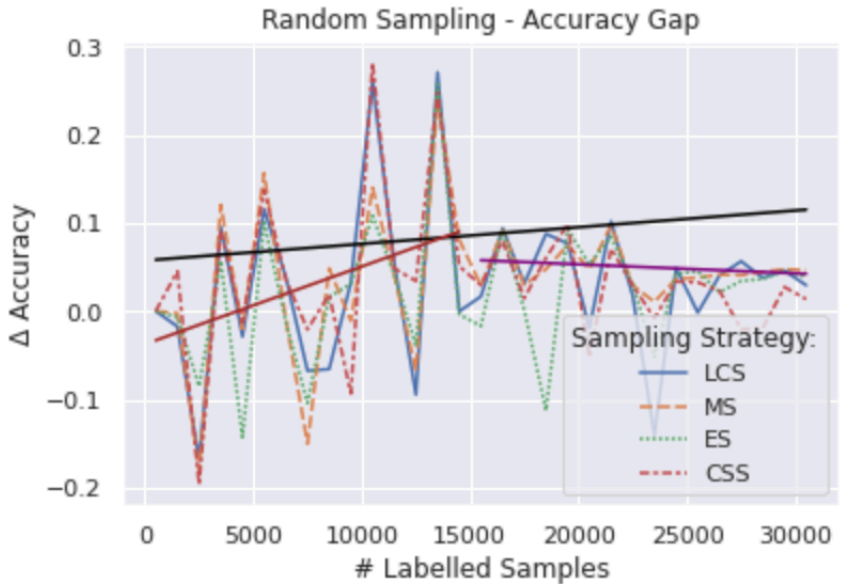

Figure 4.16: Comparison of the accuracy gap in Figure 4.14 between Random Sampling (RS) and Least Confidence Sampling (LCS), Margin Sampling (MS), Entropy Sampling (ES), and K-Means Sampling (KMS). The black line is the overall trend and has been shifted upward for clarity, the brown line is the trend for the first half, and the pink line is the trend for the second half.

4.3 Results Summary

In Table 4.11 and Table 4.12 we further quantify the results gathered in this research. In Table 4.11 we observe that the least confidence sampling and margin sampling strategy performs equally well. Both strategies need 79.25% less labeled data to get as good performance as the random sampling strategy. Both strategies achieve the same results, however, from Section 4.1.6 we know that the maximum accuracy for the least confidence sampling strategy was reached at 8.500 labeled samples compared to margin sampling that needed 28.500 labeled samples to reach its maximum accuracy. Another observation is that the K-means sampling strategy does not perform as well as the rest of the sampling strategies in terms of less data needed.

In Table 4.12 we observe that the margin sampling strategy performs best in terms of maximum accuracy achieved. However, a more important metric to look at is how much less data is needed to reach random sampling maximum accuracy. With respect to this metric, we see that the entropy sampling strategy performs significantly better than its counterparts. Just as in Table 4.11 the least confidence sampling and margin sampling strategy perform almost equally well. Furthermore, we observe that the K-

Sampling Strategy	Max Accuracy	Max Accuracy Diff	Less Data Needed (%)
Random Sampling	98.00%	-	-
Least Confidence Sampling	98.82%	+0.82%	79.25%
Margin Sampling	98.82%	+0.82%	79.25%
Entropy Sampling	98.80%	+0.80%	67.92%
K-Means Sampling	98.75%	+0.75%	45.28%

Table 4.11: A sampling comparison on the MNIST dataset with a labeling budget of 30.500. Random sampling is compared with the other strategies.

means sampling strategy performs worst.

Sampling Strategy	Max Accuracy	Max Accuracy Diff	Less Data Needed (%)
Random Sampling	65.99%	-	-
Least Confidence Sampling	71.57%	+5.58%	44.06%
Margin Sampling	71.75%	+5.76%	44.06%
Entropy Sampling	71.53%	+5.54%	67.92%
K-Means Sampling	69.72%	+3.73%	33.90%

Table 4.12: A sampling comparison on the CIFAR-10 dataset with a labeling budget of 30.500. Random sampling is compared with the other strategies.

Chapter 5

Conclusions

The primary goal of this thesis was to investigate the performance of different Deep Active Learning sampling strategies within the area of image classification. The research was done by setting up an experimental environment where the outcome of the experiments was recorded. Moreover, we used the recorded results to compare different sampling strategies with each other. In this chapter, we will discuss the positive effects and the drawbacks of the results. Furthermore, we will discuss validity, limitations, and future work.

5.1 Discussion

This thesis focused on the pool-based scenario and its various sampling strategies to provide further insights in exploring different sampling strategies. Furthermore, several important findings were observed and presented.

First and most importantly we found significant differences between the baseline random sampling strategy and the rest of the sampling strategies. The results we obtained indicate that the proposed sampling strategies improve the performance of Net1 and Net2 where both are Convolutional Neural Network (CNN) based architectures. The sampling strategies assisted the CNNs to maximize the extracted knowledge gathered by choosing the most informative samples and at the same time minimizing the number of samples humans need to label manually. Furthermore, we found out that the sampling strategies support the CNN models to converge at a faster rate. Moreover, Net1 and Net2 reached a higher accuracy score when integrating

deep active learning. However, the benefits amongst the sampling strategies least confidence sampling, margin sampling, and entropy sampling which all are based on uncertainty were limited. For example, we discovered that the least confidence sampling and margin sampling strategies performed equally well on the MNIST dataset and almost equally well on the CIFAR-10 dataset. Furthermore, the least confidence sampling and margin sampling strategy performed the best with regards to the MNIST dataset in terms of both maximum accuracy achieved and less data needed. They both achieved an accuracy of 98.82% and a less data needed score of 79.25%. With regards to the Cifar-10 dataset, The highest accuracy score of 71.75% was achieved by the margin sampling strategy and the highest less data needed score of 67.92% was achieved by the entropy sampling strategy. Hence, entropy sampling helped our CNN to converge the fastest with regards to CIFAR-10 dataset. The sampling strategy that performed the worst was the K-means sampling strategy. The K-means strategy successfully performed significantly better than random sampling however it was far from reaching the performance of the uncertainty-based strategies.

Investigating further into the learning rate capabilities of Net1 and Net2 we discovered two different stages in which we learn at different rates. The first stage is the stage where our sampling strategies improve our accuracy at a fast rate. However, as soon as the sampling strategies begin to find a representative dataset we enter the second stage. In the second stage, the learning rate starts to decrease. Furthermore, for the less complex dataset MNIST we found out that the sampling strategies were very fast at finding a representative dataset, hence, the first stage did not last long. However, for the more complex dataset CIFAR-10 we found out that the first stage lasted longer since the sampling strategy could not find a representative dataset in the early rounds. This is intuitive since CIFAR-10 is a much more complex dataset. Furthermore, we discovered that the accuracy gap was larger for the CIFAR-10 dataset compared to MNIST, hence, the sampling strategies had a greater impact when applied to the CIFAR-10 dataset in terms of accuracy gap. However, since we converge faster when using the MNIST dataset we get a significantly better score on the less data needed metric. Another finding we discovered was the accuracy gap peaked as soon as the sampling strategies found a representative dataset. In other words, the accuracy gap peaked when we transitioned from the first to the second stage.

5.1.1 Limitations

We acknowledge that the results presented in this thesis should be interpreted with caution. To enhance the performance of deep learning models it is a good practice to tune the hyperparameters, however, in this thesis, the only hyperparameter that has been tuned is the learning rate. The hyperparameter tuning could have been done manually or automatically using an optimization algorithm. However, since we have limited resources we were not able to tune more hyperparameters than the learning rate. Moreover, we did not run the experiments with a labeling budget equal to the whole dataset for the same reason mentioned above. Hence, the results recorded from using the CIFAR-10 dataset should especially be interpreted with caution since the model has not converged within the labeling budget we set. Furthermore, the initial plan was to implement more sampling strategies, however, it was not possible within the given time constraint. Hence, we chose to implement sampling strategies we were most experienced with.

5.1.2 Future Work

There are still lots of improvement opportunities for this research project.

First of all, we concluded that the deep active learning model performed better for the MNIST dataset compared to the CIFAR-10 dataset. This could be a consequence of our limited computational resources since we were not able to use a higher labeling budget. For the MNIST dataset, we noted that the model had already begun to converge early in the rounds. However, the model did not seem to fully converge for the CIFAR-10 dataset, thus this could be interesting to further investigate.

It would also be interesting to investigate which samples the sampling strategies has in common. By doing this we could extract the features of these images to investigate if we could exploit these newly found characteristics. It would be interesting to see any correlation between these images.

One sampling strategy we did not manage to implement within the given time constraint was the Query by Committee sampling strategy. This sampling strategy uses the disagreement between several different machine learning models using the same labeled dataset. This would be very interesting to further investigate.

Finally, it would be important to tune the machine learning models' hyperparameters

since we want to get as good results as possible. Hence, performing more experiments using various hyperparameters to record which one yields the best performance is very important.

Bibliography

[1] Ahmad, Muhammad, Khan, Asad, Khan, Adil Mehmood, Mazzara, Manuel, Distefano, Salvatore, Sohaib, Ahmed, and Nibouche, Omar. “Spatial prior fuzziness pool-based interactive classification of hyperspectral images”. In: *Remote Sensing* 11.9 (2019), p. 1136.

[2] An, Bang, Wu, Wenjun, and Han, Huimin. “Deep Active Learning for Text Classification”. In: *Proceedings of the 2nd International Conference on Vision, Image and Signal Processing*. 2018, pp. 1–6.

[3] Angluin, Dana. “Queries and concept learning”. In: *Machine learning* 2.4 (1988), pp. 319–342.

[4] Angluin, Dana. “Queries revisited”. In: *International Conference on Algorithmic Learning Theory*. Springer. 2001, pp. 12–31.

[5] Bodó, Zalán, Minier, Zsolt, and Csató, Lehel. “Active learning with clustering”. In: *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*. JMLR Workshop and Conference Proceedings. 2011, pp. 127–139.

[6] Breiman, Leo. “Bagging predictors”. In: *Machine learning* 24.2 (1996), pp. 123–140.

[7] Chansung, Park. “CIFAR-10 Image Classification in TensorFlow”. In: (2018). URL: `https://towardsdatascience.com/cifar-10-image-classification-in-tensorflow-5b501f7dc77c`.

[8] Chen, Jinying, Schein, Andrew, Ungar, Lyle, and Palmer, Martha. “An empirical study of the behavior of active learning for word sense disambiguation”. In: *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. 2006, pp. 120–127.

[9] Cheng, Yu, Chen, Zhengzhang, Liu, Lu, Wang, Jiang, Agrawal, Ankit, and Choudhary, Alok. "Feedback-driven multiclass active learning for data streams". In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2013, pp. 1311–1320.

[10] Cohn, David, Atlas, Les, and Ladner, Richard. "Improving generalization with active learning". In: *Machine learning* 15.2 (1994), pp. 201–221.

[11] Czakon, Jakub. "F1 Score vs ROC AUC vs Accuracy vs PR AUC: Which Evaluation Metric Should You Choose?" In: (2021). URL: https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc.

[12] Danielsson, Per-Erik. "Euclidean distance mapping". In: *Computer Graphics and image processing* 14.3 (1980), pp. 227–248.

[13] David Arpin, Nadia Yakimakha. "Amazon SageMaker now supports PyTorch and TensorFlow 1.8". In: (2020). URL: https://aws.amazon.com/blogs/machine-learning/amazon-sagemaker-now-supports-pytorch-and-tensorflow-1-8/.

[14] "Distributed deep learning training using PyTorch with HorovodRunner for MNIST". In: *Databricks* (). URL: https://aws.amazon.com/blogs/machine-learning/amazon-sagemaker-now-supports-pytorch-and-tensorflow-1-8/.

[15] Feng, Chen, Liu, Ming-Yu, Kao, Chieh-Chi, and Lee, Teng-Yok. "Deep active learning for civil infrastructure defect detection and classification". In: *Computing in civil engineering 2017*. 2017, pp. 298–306.

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

[17] Han, Wenjing, Coutinho, Eduardo, Ruan, Huabin, Li, Haifeng, Schuller, Björn, Yu, Xiaojie, and Zhu, Xuan. "Semi-supervised active learning for sound classification in hybrid learning environments". In: *PloS one* 11.9 (2016), e0162075.

[18] Harris, Charles R., Millman, K. Jarrod, Walt, Stéfan J. van der, Gommers, Ralf, Virtanen, Pauli, Cournapeau, David, Wieser, Eric, Taylor, Julian, Berg, Sebastian, Smith, Nathaniel J., Kern, Robert, Picus, Matti, Hoyer, Stephan, Kerkwijk, Marten H. van, Brett, Matthew, Haldane, Allan, Río, Jaime Fernández

del, Wiebe, Mark, Peterson, Pearu, Gérard-Marchant, Pierre, Sheppard, Kevin, Reddy, Tyler, Weckesser, Warren, Abbasi, Hameer, Gohlke, Christoph, and Oliphant, Travis E. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[19] Huilgol, Purva. “Accuracy vs. F1-Score”. In: *Medium* (2019). URL: https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2.

[20] Jeon, Jiwoon, Lavrenko, Victor, and Manmatha, Raghavan. “Automatic image annotation and retrieval using cross-media relevance models”. In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. 2003, pp. 119–126.

[21] Jia, Bill. “Announcing PyTorch 1.0 for both research and production”. In: *Facebook Engineering* (2018).

[22] Joshi, Ajay J, Porikli, Fatih, and Papanikolopoulos, Nikolaos. “Multi-class active learning for image classification”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 2372–2379.

[23] kale, ajit. “A Meta-Learning Approach To One-Step Active Learning”. In: (). URL: https://medium.com/@kaleajit27/apaperaday-week1-a-meta-learning-approach-to-one-step-active-learning-5ffea59099a2 (visited on 05/31/2021).

[24] Khosla, Savya. “ML | K-means++ Algorithm”. In: (2018). URL: https://www.geeksforgeeks.org/ml-k-means-algorithm/.

[25] King, Ross D, Whelan, Kenneth E, Jones, Ffion M, Reiser, Philip GK, Bryant, Christopher H, Muggleton, Stephen H, Kell, Douglas B, and Oliver, Stephen G. “Functional genomic hypothesis generation and experimentation by a robot scientist”. In: *Nature* 427.6971 (2004), pp. 247–252.

[26] Koehler, Gregor. “MNIST Handwritten Digit Recognition in PyTorch”. In: *Nextjournal* (2020). URL: https://nextjournal.com/gkoehler/pytorch-mnist.

[27] Krishnamurthy, Vikram. “Algorithms for optimal scheduling and management of hidden Markov model sensors”. In: *IEEE Transactions on Signal Processing* 50.6 (2002), pp. 1382–1397.

[28] Krizhevsky, Alex. “The CIFAR-10 dataset”. In: (). URL: https://www.cs.toronto.edu/~kriz/cifar.html (visited on 05/31/2021).

[29] Lewis, David D and Gale, William A. “A sequential algorithm for training text classifiers”. In: *SIGIR’94*. Springer. 1994, pp. 3–12.

[30] Likas, Aristidis, Vlassis, Nikos, and Verbeek, Jakob J. “The global k-means clustering algorithm”. In: *Pattern recognition* 36.2 (2003), pp. 451–461.

[31] Lin, Hsuan-Tien, Lin, Chih-Jen, and Weng, Ruby C. “A note on Platt’s probabilistic outputs for support vector machines”. In: *Machine learning* 68.3 (2007), pp. 267–276.

[32] Liu, Rong, Chen, Ting, and Huang, Lu. “Research on human activity recognition based on active learning”. In: *2010 International Conference on Machine Learning and Cybernetics*. Vol. 1. IEEE. 2010, pp. 285–290.

[33] Loy, Chen Change, Hospedales, Timothy M, Xiang, Tao, and Gong, Shaogang. “Stream-based joint exploration-exploitation active learning”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 1560–1567.

[34] Luo, Tong, Kramer, Kurt, Goldgof, Dmitry B, Hall, Lawrence O, Samson, Scott, Remsen, Andrew, Hopkins, Thomas, and Cohn, David. “Active learning to recognize multiple types of plankton.” In: *Journal of Machine Learning Research* 6.4 (2005).

[35] MacQueen, James et al. “Some methods for classification and analysis of multivariate observations”. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[36] McCallumzy, Andrew Kachites and Nigamy, Kamal. “Employing EM and pool-based active learning for text classification”. In: *Proc. International Conference on Machine Learning (ICML)*. Citeseer. 1998, pp. 359–367.

[37] Mittal, Sudhanshu, Tatarchenko, Maxim, Çiçek, Özgün, and Brox, Thomas. “Parting with illusions about deep active learning”. In: *arXiv preprint arXiv:1912.05361* (2019).

[38] *MNIST database*. 2017. URL: https://en.wikipedia.org/wiki/MNIST_database.

[39] Pan, Sinno Jialin and Yang, Qiang. “A survey on transfer learning”. In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[40] Pedregosa, Fabian, Varoquaux, Gaël, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, Blondel, Mathieu, Prettenhofer, Peter, Weiss, Ron, Dubourg, Vincent, et al. “Scikit-learn: Machine learning in Python”. In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[41] Qi, Guo-Jun, Hua, Xian-Sheng, Rui, Yong, Tang, Jinhui, and Zhang, Hong-Jiang. “Two-dimensional active learning for image classification”. In: *2008 IEEE conference on computer vision and pattern recognition*. IEEE. 2008, pp. 1–8.

[42] Ranganathan, Hiranmayi, Venkateswara, Hemanth, Chakraborty, Shayok, and Panchanathan, Sethuraman. “Deep active learning for image classification”. In: *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2017, pp. 3934–3938.

[43] Ren, Pengzhen, Xiao, Yun, Chang, Xiaojun, Huang, Po-Yao, Li, Zhihui, Chen, Xiaojiang, and Wang, Xin. “A Survey of Deep Active Learning”. In: *arXiv preprint arXiv:2009.00236* (2020).

[44] Roy, Nicholas and McCallum, Andrew. “Toward optimal active learning through monte carlo estimation of error reduction”. In: *ICML, Williamstown* (2001), pp. 441–448.

[45] Roy, Soumya, Unmesh, Asim, and Namboodiri, Vinay P. “Deep active learning for object detection.” In: *BMVC*. 2018, p. 91.

[46] Schröder, Christopher and Niekler, Andreas. “A Survey of Active Learning for Text Classification using Deep Neural Networks”. In: *arXiv preprint arXiv:2008.07267* (2020).

[47] Sener, Ozan and Savarese, Silvio. “Active learning for convolutional neural networks: A core-set approach”. In: *arXiv preprint arXiv:1708.00489* (2017).

[48] Settles, Burr. “Active learning”. In: *Synthesis lectures on artificial intelligence and machine learning* 6.1 (2012), pp. 1–114.

[49] Settles, Burr. “Active learning literature survey”. In: (2009).

[50] Shen, Yanyao, Yun, Hyokun, Lipton, Zachary C, Kronrod, Yakov, and Anandkumar, Animashree. “Deep active learning for named entity recognition”. In: *arXiv preprint arXiv:1707.05928* (2017).

[51] Shung, Koo Ping. “Accuracy, Precision, Recall or F1?” In: *Towards Data Science* (2018). URL: `https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9`.

[52] Smailagic, Asim, Costa, Pedro, Noh, Hae Young, Walawalkar, Devesh, Khandelwal, Kartik, Galdran, Adrian, Mirshekari, Mostafa, Fagert, Jonathon, Xu, Susu, Zhang, Pei, et al. “Medal: Accurate and robust deep active learning for medical image analysis”. In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2018, pp. 481–488.

[53] Smailović, Jasmina, Grčar, Miha, Lavrač, Nada, and Žnidaršič, Martin. “Stream-based active learning for sentiment analysis in the financial domain”. In: *Information sciences* 285 (2014), pp. 181–203.

[54] Tang, Min, Luo, Xiaoqiang, and Roukos, Salim. “Active learning for statistical natural language parsing”. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. 2002, pp. 120–127.

[55] Tong, Simon and Chang, Edward. “Support vector machine active learning for image retrieval”. In: *Proceedings of the ninth ACM international conference on Multimedia*. 2001, pp. 107–118.

[56] Trencseni, Marton. “Solving CIFAR-10 with Pytorch and SKL”. In: (2019). URL: `https://bytepawn.com/solving-cifar-10-with-pytorch-and-skl.html`.

[57] Voulodimos, Athanasios, Doulamis, Nikolaos, Doulamis, Anastasios, and Protopapadakis, Eftychios. “Deep learning for computer vision: A brief review”. In: *Computational intelligence and neuroscience* 2018 (2018).

[58] Wang, Dan and Shang, Yi. “A new active labeling method for deep learning”. In: *2014 International joint conference on neural networks (IJCNN)*. IEEE. 2014, pp. 112–119.

[59] Wang, Keze, Zhang, Dongyu, Li, Ya, Zhang, Ruimao, and Lin, Liang. “Cost-effective active learning for deep image classification”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.12 (2016), pp. 2591–2600.

[60] Wang, Yu, Mendez, Ana Elisa Mendez, Cartwright, Mark, and Bello, Juan Pablo. “Active learning for efficient audio annotation and classification with a large amount of unlabeled data”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 880–884.

[61] Wolf, Gert W. *Facility location: concepts, models, algorithms and case studies. Series: Contributions to Management Science: edited by Zanjirani Farahani, Reza and Hekmatfar, Masoud, Heidelberg, Germany, Physica-Verlag, 2009, 549 pp.,€171.15,* $219.00, £144.00, ISBN 978-3-7908-2150-5 (hardprint), 978-3-7908-2151-2 (electronic)$. 2011.

[62] Wu, Dongrui. “Pool-based sequential active learning for regression”. In: *IEEE transactions on neural networks and learning systems* 30.5 (2018), pp. 1348–1359.

[63] Xu, Zhao, Yu, Kai, Tresp, Volker, Xu, Xiaowei, and Wang, Jizhi. “Representative sampling for text classification using support vector machines”. In: *European conference on information retrieval*. Springer. 2003, pp. 393–407.

[64] Yang, Lin, Zhang, Yizhe, Chen, Jianxu, Zhang, Siyuan, and Chen, Danny Z. “Suggestive annotation: A deep active learning framework for biomedical image segmentation”. In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2017, pp. 399–407.

[65] Yann LeCun Corinna Cortes, Christopher J.C. Burges. “THE MNIST DATABASE of handwritten digits”. In: (). URL: `http://yann.lecun.com/exdb/mnist/` (visited on 05/31/2021).

[66] Zhu, Jingbo and Hovy, Eduard. “Active learning for word sense disambiguation with methods for addressing the class imbalance problem”. In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. 2007, pp. 783–790.

[67] Zhu, Jingbo, Wang, Huizhen, Yao, Tianshun, and Tsou, Benjamin K. “Active learning with sampling by uncertainty and density for word sense disambiguation and text classification”. In: *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. 2008, pp. 1137–1144.

TRITA -EECS-EX-2021:283