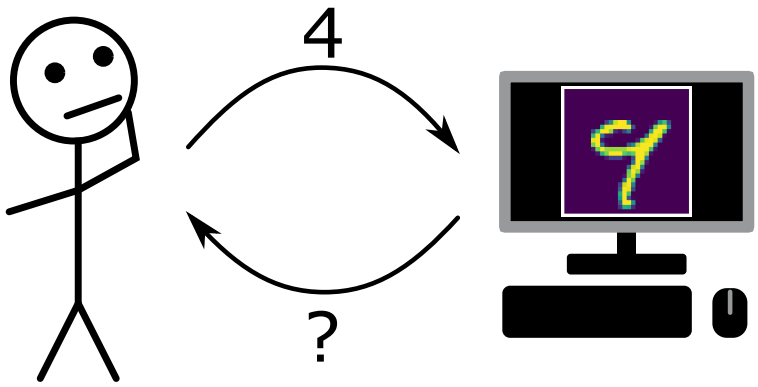
CHALMERS
UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG



Active Learning for Artificial Neural Network Models

A deeper look into practical implementation suggestions for active learning and how it is affected by the network capacities and training methods.

Master's thesis in Computer science and engineering

John Daniel Bossér
Erik Sörstadius

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

Active Learning for Artificial Neural Network Models

John Daniel Bossér
Erik Sörstadius

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

Active Learning for Artificial Neural
Network Models A deeper look into practical implementation suggestions for
active learning and how it is affected by the network
capacities and training methods.
John Daniel Bossér
Erik Sörstadius

Supervisor: Morteza Haghir Chehreghani, Data Science and AI
Examiner: Peter Damaschke, Data Science and AI

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A picture of an human (oracle) labeling a difficult image for an artificial neural network model running on a computer.

Typeset in LaTeX
Gothenburg, Sweden 2020

John Daniel Bossér, Erik Sörstadius
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Active learning is the field of choosing informative data to train machine learning models. This thesis covers eight separate substudies investigating how to maximize the test accuracy for deep feed-forward artificial neural network models using active learning. When performing active learning, a query strategy must be specified which is why four different query strategies were examined, namely the margin, entropy, least confident, and one suggested by the authors, the least squares. The data sets MNIST, Fashion-MNIST, and CIFAR-10 were used to see how the results generalize between data sets. With the eight substudies examined, we have concluded some suggestions that should be considered to improve the test accuracy:

(1) Among the query strategies examined, the *margin query strategy* consistently selected data which gave rise to the highest test accuracy. (2) The *cumulative training* method is most suitable to train feed-forward neural networks when using a query strategy. This means that the networks should be reset and retrained using all labeled data. (3) For improved performance, a query strategy should be used *after* the network has trained on some initially randomly selected data. (4) If the mean margin *informativeness measure*, used internally by the margin query strategies, starts to decrease during training, then one should consider gathering more unlabeled data or stop labeling to reduce cost. (5) The semi-supervised pseudo-label algorithm may be used to further increase test accuracy by utilizing the unlabeled data set. (6) To estimate the performance of a network without the presence of a dedicated labeled test set, one can use the randomly sampled data from (3) to create an upper and lower estimate of the test accuracy. We have shown, through empirical studies, that steps (1)-(6) are all associated with some benefit when performing active learning.

Keywords: Active Learning, Machine Learning, Artificial Neural Networks, Sampling technique

Acknowledgements

We want to thank, in first hand, our supervisor Morteza Haghir Chehreghani who with his encouragement and knowledge always had time to spare to help us. We also want to thank our examiner, Peter Damaschke, who we felt is genuinely interested in our thesis and who always lent a helping hand. Hampus Svensson and Karl Blomgren have also offered valuable help throughout the thesis. They read and gave feedback on the entirety of our report multiple times which was greatly appreciated. Finally, other people who we are grateful for in helping with the process of writing our thesis include Birgit Grohe, Anthony Norman, Adina Berg, Magnus Fries, and Lennart Bossér.

John Daniel Bossér and Erik Sörstadius, Gothenburg, June 2020

Contents

List of Figures

List of Tables

1

Introduction

Many repetitive and difficult classification problems have been outsourced to machine learning [1] [2, p. 1], which has the potential to be as good at classification as the data allows. For example, a common problem of interest is machine learning for image recognition and computer vision, i.e. automatically classifying what an image is depicts [3, p. 98]. Over the last couple of decades, a machine learning model inspired by the human brain, has been growing in popularity: *Artificial Neural Networks*. These models gained popularity through their high flexibility and accuracy. However, in order to train these models, one usually requires a large labeled data set. For example, Gargeya and Leng [4] used artificial neural networks to automate diabetic retinopathy, one of the leading causes behind preventable blindness. They trained the model using 75 137 publicly available fondus images and were thus able to differentiate diabetic retinopathy from healthy fundi with 94% sensitivity and 98% specificity. As an additional example, Mohanty et al. [5] used an artificial neural network to detect potential diseases in crops. Using a data set consisting of 54 000 images, they were able to achieve an accuracy of 99.35% in their prediction.

Labeling data sets, like the ones mentioned above, can be a slow and tedious process since it is not uncommon that a human is needed to review and classify the individual samples. It would, therefore, be useful to only label the most informative data, i.e. the data that the machine learning algorithm learns the most from, and use that for training. The study of choosing the most informative data, while not having access to all labels, is called *active learning* [6]. The difficulty lies in that there is no objectively superior way of determining how informative data is. Instead, *Query Strategies*[1], are used as heuristics. What is sought after is a query strategy that maximises the test accuracy, or minimises the loss, using as little data as possible [9, p. 4]. Much of the early work within active learning has been summarised by Settles in his survey from 2010 [9], as a part of his Ph.D. thesis. Settles reviewed query strategies and other approaches for active learning. We partly view and suggest our work in this thesis as an extension of his summery on pool-based training, and cover all of his suggested uncertainty based query strategies. These are the *least confident*, *margin*, and the *entropy* query strategies, which are presented in Section 2.2.1. This thesis also introduces another suggested query strategy, the *least squares* query strategy.

[1]Similar measures are known as acquisition functions in some statistical areas [7] [8].

The developed query strategy looks at the Euclidean distance between the network output and the true label. All query strategies are compared to random sampling, which in this thesis also is referred to as the *random* query strategy.

However, the work Settles covered did not include active learning for artificial neural networks. Instead, he focused more on other types of classifiers, such as support vector machines and decision trees. This thesis covers the knowledge gap of combining active learning and artificial neural network. Moreover, this thesis also looks into different choices of hyperparameters, for example:

- The number of neurons in the neural network.
- The number of times the training data trains the network, i.e. the number of epochs.

It also takes into consideration practical implementation difficulties like:

- An absence of test set.
- How to know that the active learning algorithm is effective compared to traditional methods.
- When to stop training.

1.1 Related Work

This section aims to cover some of the most recent advances in the field regarding active learning related to deep learning. It also aims to bring up work on similar problems but in slightly different circumstances.

1.1.1 Query Strategies

To perform active learning an informativeness measure has to be calculated for all data. The query strategy is the most essential part of active learning since it determines what data to label. Several papers have investigated which types of informativeness measures and query strategies that can be used to perform active learning. Houlsby et al. [10] showed that one can use Bayesian information-theoretic approaches as a query strategy if Bayesian neural networks are used. The Bayesian neural networks function like ordinary neural networks, with the difference that the trainable parameters are stochastic and described by probability distributions. Thus, the output of the Bayesian neural networks is stochastic, making it possible to analyze the output and select data for labeling using probability theory, such as Bayesian information-theoretic approaches. Another interesting approach was made by Gissin and Shalev-Shwartz [11] who interpreted the query strategy as a classification problem. They successfully trained a neural network to classify whether a sample is informative enough to be selected for labeling.

1.1.2 Sampling Techniques

There have been improvements in the field of sampling techniques for deep learning. A sampling technique is a way to gather a subset of data, i.e. a sample, from a larger data set. Unlike the algorithms used in the active learning setting, these algorithms have the luxury of having access to the labels of the samples while training. This opens up more possibilities for observing which samples to use for training. *Importance sampling* by Katharopoulos and Fleuret [12] is an example of this. They were able to sample the most informative samples by computing an upper bound of the gradient norm per sample. By sampling from a data set, using this information, they could minimize the loss per gradient descent step at a faster rate compared to random sampling. This is related to active learning in the way that they try to minimize the number of samples used for training while still achieving the best possible test accuracy.

1.1.3 Online Learning

Online data, or stream based data, is a type of data where a sample only can be considered once. There is thus no pool of data to select samples from, and one must decide if one wants to use the sample or not for training the model. This scenario could arise when the amount of data transmitted is larger than the storage capacity. Online learning considers this type of data intending to train a classifier on the received data. This type of problem introduces many new problems apart from those already present in the active learning scenario. For example, it introduces *concept drift* and *concept evolution*, which arise when the characteristic of the stream changes over time. Settles explains the basics behind online learning which is also called stream-based learning [9, p. 10]. The papers by Zhu et al. [13] and Saad et al. [14] also provide more information about online active learning.

1.1.4 Set Selection

Sener and Savarse [8] found that many of the active learning heuristics in Settles [9, p. 12] did not work satisfactorily when they were used in combination with convolutional neural networks (CNN's). CNN's are a type of network that trains filters to be convoluted over images. The active learning heuristics did not work for two reasons. First, the heuristics can only select either a single point or a few points at a time for training CNN. This gives no significant impact on the network, and thus not noticeably improving the result. Secondly, for each new point, one would need to retrain a new model with all labeled data, potentially with many epochs, to get satisfactory accuracy. Their solution to this is to view Active Learning as an *unlabeled core-set selection problem*. This means that they wish to find a good unlabeled subset of the training data that is comparable to the whole training set in terms of accuracy. Using this approach, they found a new active learning approach that worked better, by a large margin, for image classification with a CNN. The work in this thesis explores a similar idea to find good, although labeled, subset by using a combination of a query strategy and a neural network.

1.2 Aim

This thesis aims to give insights on when and how to use active learning for artificial neural networks and how this can be done effectively. We aim to document these insights and communicate them in a way that is applicable for projects in industry and academia, where data labeling can be expensive. With this in mind, the thesis also aims to cover potential solutions to some challenges that one might face when implementing an active learning method for deep learning.

1.3 Limitations

This thesis is limited to study active learning concerning feed-forward artificial neural networks, i.e. networks where the connections between the neurons do not create a cycle. It is also limited to query strategies that are based on uncertainty sampling, which are the most commonly used among the active learning frameworks [9, p. 12]. These include the least confident, entropy, least squares and margin query strategies which are explained in Section 2.2.1. Only pool-based active learning will be considered, meaning that all data can be considered for labeling at all times. Furthermore, the metric used for evaluating the performance is the classification accuracy of the neural network. Finally, three different data sets will be considered, namely the MNIST data set [15], the Fashion-MNIST data set [16] and the CIFAR-10 data set [17]. The MNIST data set consists of images of handwritten digits, the Fashion MNIST consists of low-resolution images of clothing, and the CIFAR-10 data set consists of colored images of various animals and objects. All three sets are commonly used to benchmark active learning algorithms.

1.4 Structure of the Thesis

The thesis is divided into four chapters, the last two of which include eight substudies. An overview of the structure of the last two chapters is presented in Figure 1.1. Each substudy is divided into six sections: Background, Method, Results, Discussion, Conclusion and Future Work. The Background section aims to explain the reason why the substudy in question has been investigated and why the results could be relevant. The Method section will state the experimental setups, and the Results section presents the plots with the key results highlighted. The Conclusion section summarises the experiment findings and how to use the conclusions. The Future Work section finally states questions that arose during the research which have not been examined in this thesis.

This chapter aims to introduce the reader to active learning, related work, and the importance of the work to be presented in this thesis.

The second chapter covers basic theory and background for neural networks, active learning, and query strategies. This chapter aims to prepare the reader with all algorithms and techniques that are used in the subsequent chapters.

The third chapter, named *Model-Centric Investigation*, mainly explores how the performance of different networks depends on several parameters and training methods. In particular, the substudies in the chapter compare these choices with simple random sampling. Random sampling is in this thesis interchangeably referred to as the random query strategy. While considering the effect of the above factors, the chapter covers two different substudies:

1. **Query Strategies and Training Methods:** The labeled data can be used to train a network in different ways. This study investigates and compares two of these methods. The first method, called *Incremental training*, keeps the weights from the previous iteration and trains on the new, small batch of data. The second method, called *Cumulative training*, is to gather all labeled data and use these to train a fresh network. The substudy also looks into how the performance of query strategies, in terms of test accuracy, is affected by the choice of method.

2. **Informativeness Measures** In this substudy, we take a closer look at the informativeness measures given by a query strategy, that is assigned to each point in the unlabeled data set. A high informativeness measure means that the query strategy deems the point to be informative. We investigate how the informativeness measures change as the neural networks learns more about the data and as the unlabeled data set starts to deplete. We investigate what the informativeness measures tell us about the training progress and the unlabeled data set.

The fourth chapter focuses more on novel ideas. Like the third chapter, it is structured in different substudies. As the name *Data-Centric Investigation* implies, these substudies try to investigate the distribution of the data selected by the query strategy and how it affects the training of a network. This chapter also assumes that the reader is aware of some conclusions from Chapter 3. To summarise, these cover:

1. **Evaluation of Sampled Data Sets:** We explore a similar idea to the coreset selection problem [8], see Section 1.1.4, where active learning is viewed as a problem of choosing the most informative data set. To compare two different data sets, a distinction between a selection network and an evaluation network is made. The distinction, for example, allows questions to be answered such as: "Can a small capacity network beneficially select data for a large capacity network?".

2. **Sample Distributions:** This substudy looks at the class distribution of the data selected by the query strategy with a network. Moreover, it is investigated why some distributions yield higher test accuracy than others.

3. **Random Start**: The goal of the different query strategies is to select informative data for the network to use for training, based on the current state and performance of the network. At initialization, the trainable parameters are

randomly sampled, as described in Section 2.3.5. We investigate if randomly initialized neural networks, aided by a query strategy, are biased towards some type of data. If so, can this bias be mitigated by initially training on randomly selected data before sampling with a query strategy?

4. **Estimates of the Test Accuracy**: As a followup to the *Random Start substudy* this substudy investigates if randomly sampled data is used to train the network initially, can this randomly sampled data be used to get an approximation of the test accuracy? Having a test set is a luxury that might not be available when using active learning since a test set requires labeled data. Thus, estimates of the test accuracy are of great interest. We construct two approximations corresponding to an upper estimate and lower estimate of the test accuracy and evaluate these to see how they relate to the accuracy on a larger test set.

5. **Course Plan Training**: This substudy looks into how the test accuracy can be further improved by changing the training order in the gradient descent method. The stochastic gradient descent method trains the network on randomly selected data. Could there be any benefits to being selective about the training order? Can the informativeness measures from the query strategies be used to create such an order? For this substudy, we created such an order based on the informativeness measures to see if there are any performance benefits.

6. **Semi-supervised Learning:** Since the active learning setting includes a lot of unlabeled data, we look into using this unlabeled data to improve the test accuracy of the network further. We do this by using *Semi-supervised learning*, which allows the network to assign pseudo-labels and thereby expanding the training set with parts of the unlabeled data. We investigate the benefits of performing semi-supervised learning after having sampled data with active learning.

1.5 Ethical Considerations

What is aimed to be developed is a method of choosing informative data to label. This method could be used in any setting, not limited to one type of ethical framework. The field of active learning has the potential to improve the effectiveness of weapons as well as making autonomous vehicles safer. It will thus be up to the user, who is performing active learning, to make the ethical analysis for their application. To research active learning three different data sets were used, namely MNIST, Fashion-MNIST, and CIFAR-10. They are all publicly available data and can be viewed or used by anyone. Public data has the benefit of allowing anyone to reproduce the results, which is a good thing for the scientific method. The data sets are filled with images depicting fashion objects, numbers, animals, and vehicles. The information they contain can not be linked to sensitive information or human individuals, which makes the analysis of data more ethically sound. Moreover, the

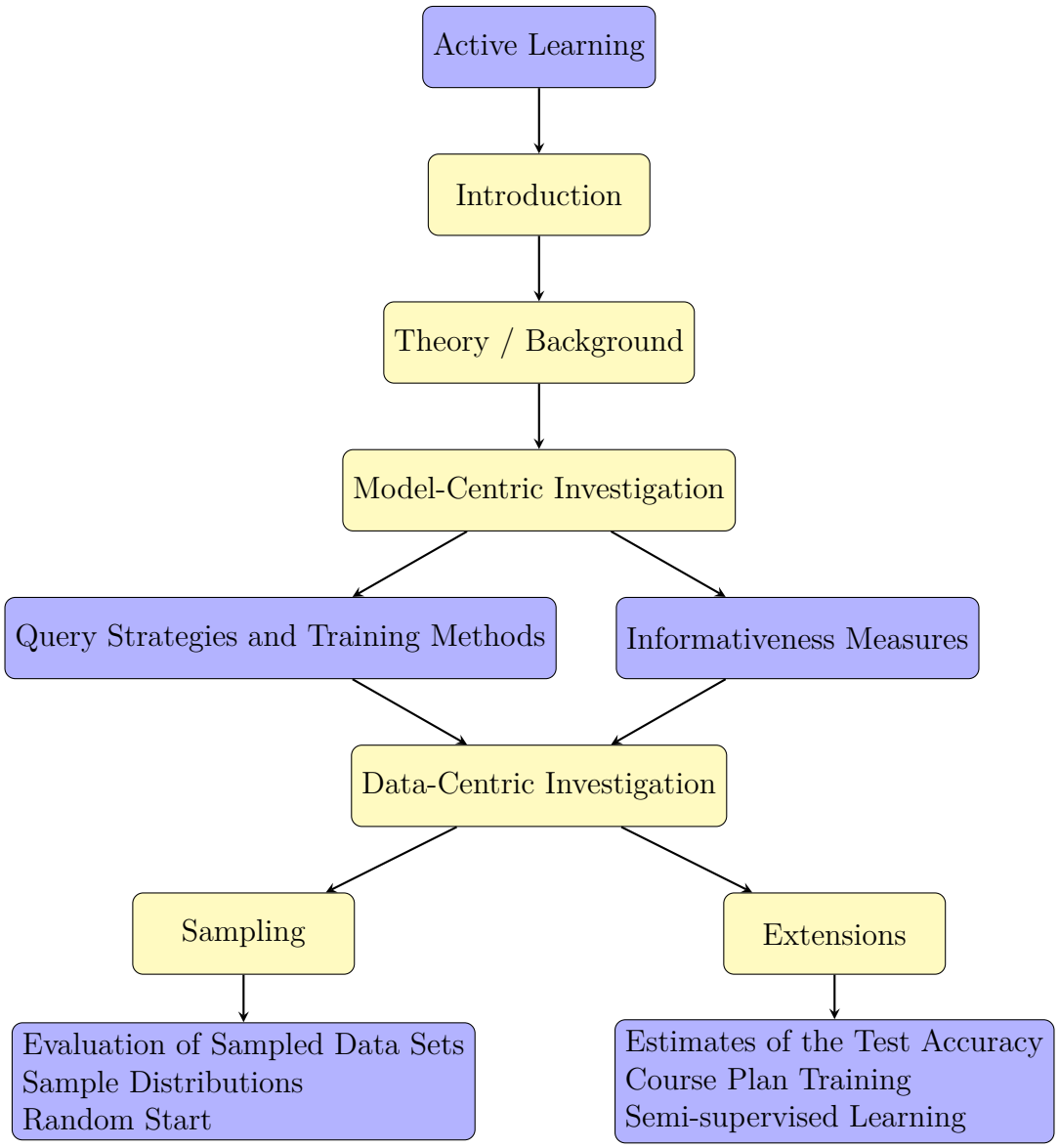


Figure 1.1: A map of the structure of the thesis. Blue boxes indicate what we have investigated, and yellow boxes describe the structure of the thesis.

methods investigated are used to make training of machine learning methods more efficient and to reduce human labour. We have taken all of this into consideration and concluded that the content of this thesis is worth researching.

2

Theory / Background

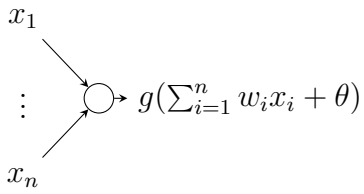
This chapter aims to cover some introductory theory and background needed to understand the results and how to implement and use active learning methods. This chapter consists of different parts. The first of which covers how artificial neural networks work, are trained and are used, see Section 2.1. The second part, in Section 2.2, introduces active learning and what types of problem usually arise within the field together with common components used to solve these problems. This is followed by a description of the query strategies examined in this thesis, see Section 2.2.1. The last part of the chapter, see Section 2.3, focuses on methodology for how the query strategies are used to train the networks, the data sets used, and the artificial neural network architectures used in the substudies.

2.1 Artificial Neural Networks

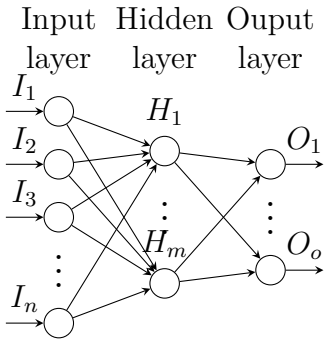
Artificial neural networks can be used for solving Machine Learning problems. A common application is to solve classification problems. For these problems, the usual goal is to identify what category, i.e. *the class*, each individual sample of the data belongs to. The data can, for example, be measurements or photos of objects. Artificial neural networks are inspired by the human brain, and just like the brain, an artificial neural network is built by several smaller units, artificial neurons, that are connected together, see Figure 2.1. These artificial neurons receive inputs, represented by numbers, which can be either data or input from other neurons. The output from a neuron is a weighted sum of all the inputs $\vec{x}$, shifted with a bias θ and passed through an *activation function* g,

$$\vec{x} \mapsto g\left(\sum_{i=1}^{n} w_i x_i + \theta\right),$$

where $\vec{w}$ is the weights of the neuron and n is the number of features, or in other words the dimensionality, of $\vec{x}$. When several of these artificial neurons are connected, they create an artificial neural network. In deep learning, the neurons are structured in layers. The first and last layers are usually referred to as the *input layer* and *output layer*, respectively. Layers between the input and output layers are called *hidden layers*. Such an arrangement of neurons is illustrated in Figure 2.1b.



(a) A single neuron. This neuron receives an input $\vec{x} = [x_1, \ldots, x_n]^T$ with n different features, creates a weighted sum of them with the weights $\vec{w} = [w_1, \ldots, w_n]^T$, and applies a bias θ to the weighted sum. This number is then fed through a non-linear function g which yields the output of the neuron.



(b) A neural network, consisting of several interconnected single neurons. This network consists of one input layer, one hidden layer, and one output layer. Each neuron has its own set of trainable weights and biases.

Figure 2.1: Shows the mathematics of a single neuron and how an artificial neural network is created by connecting many neurons.

The last layer of a neural network can vary depending on the task the network is to solve. In classification problems, it is desirable to have an output from the network that orders which class is most probable. In this case, the output of the last layer can be fed through a *softmax layer* that transforms the outputs according to

$$\sigma(\vec{O})_j^{(i)} = \frac{e^{O_j^{(i)}}}{\sum_j e^{O_k^{(i)}}},$$

where $O_j^{(i)}$ is the output of the neural network from output neuron j for data point $\vec{x}^{(i)}$, see Figure 2.1b. This creates an output $\vec{\sigma}^{(i)}$ such that $\sum_k \sigma_k^{(i)} = 1$ and $\sigma_c^{(i)} \in [0, 1]$ where each $\sigma_c^{(i)}$ can be interpreted as the probability estimates for the input $\vec{x}^{(i)}$ belonging to class c. The final output of the network is denoted $\hat{y}_{ic}$, where

$$\hat{y}_{ic} = \sigma_c^{(i)}, \tag{2.1}$$

which is the probability estimate for input $\vec{x}^{(i)}$ belonging to class c.

Artificial neural networks are trained by adjusting the weights and biases of their neurons by minimising a loss function L. The loss function determines the objective of the network and how well it currently solves its task. A common loss function for classification tasks is the *Sparse Categorical Cross Entropy* loss:

$$L(y, \hat{y}) = -\left(\frac{1}{N}\right) \sum_c \sum_{i=1}^{N} y_{ic} \log \hat{y}_{ic}, \tag{2.2}$$

where y_{ic} equals 1 if data point $\vec{x}^{(i)}$ belongs to class c and 0 otherwise, $\hat{y}_{ic}$ is the prediction of the neural network and N is the total number of points for which the

loss is calculated over. Usually, one defines a training set $T = \{(\vec{x}^{(i)}, y_i)\}$ consisting of data and labels for which the loss is calculated. A fit to the data is then achieved by minimising the loss w.r.t of the weights w_j and biases θ_k by numerically searching for where the gradient is 0, i.e. for which[1]

$$\frac{\partial L(y, \hat{y})}{\partial w_j} = 0,$$
$$\frac{\partial L(y, \hat{y})}{\partial \theta_k} = 0.$$

Ideally, one wants to find the weights w_j and biases θ_k for which the loss is minimal. In practice, however, finding this global optimum is difficult. Instead, numerical methods are usually employed to find values for w_j and θ_k that gives a close to optimal value of the target function. Usually, this is done with the stochastic gradient descent method [18].

In order to train artificial neural networks, a lot of data is needed. However, obtaining labeled data can be expensive. It is thus desirable to train with as little labeled data as possible, while still ensuring good performance of the network. The scope of this thesis is to combine the training of artificial neural networks with methods that aim to label as little data as possible. This method of finding and labeling the most informative data, i.e. data that yields a high test accuracy, is called *active learning* [9, p. 4].

2.2 Active Learning

Usually, a machine learning algorithm is trained using some labeled training set L, validated on some labeled test set T, and finally used to predict labels of some unlabeled data set U if the performance of the model is adequate. Each set consists of samples of e.g. images or measurements. These are fed to the machine learning algorithm to be used either for training or classification. Active learning is designed to deal with the scenario when the labeled training set L is small and needs to be expanded by labeling data from U. As an extension, one can also assume that the labeled test set T is small or non-existent, making an estimate of the performance of the algorithm difficult. Somehow data from U needs to be labeled but choosing which data to label is not trivial.

To solve this problem, active learning algorithms sort data according to an informativeness measure. A sample with a high informativeness measure indicates that the machine learning algorithm can increase its performance more by training on the sample, compared to a sample with lower informativeness measure. To use the data for training, it must first be assigned a label. This is done by querying an *oracle*, which supplies the true label of the data. In simulations, the oracle is often the

[1] Note that the prediction $\hat{y}_{ic}$ of the network is a function of the weights w_j and biases θ_k.

given labels for the data set but hidden from the active learning algorithm, but in real life the oracle can be a human or another algorithm.

The objective of active learning is often to maximise the test accuracy with the least amount of queries to the oracle as possible. However, it is possible that the test set does not exist, in the active learning scenario, since the data is assumed to be expensive to label. It is therefore not trivial how to measure the performance of the network.

Another question is how to query for labels efficiently. The main hypothesis is that this can be solved if the machine learning algorithm is allowed to take part in choosing what data to label [9, p. 4]. The most informative data can be chosen by observing the network output and applying a query strategy. It can, for example, ask for labels of data in which the output probability estimate $\hat{y}_{ic}$ is approximately equal for all classes c, indicating that the network is uncertain.

A lot of real-world applications for active learning exist. Imagine a scenario where Street View cars are gathering photographs of their surrounding continuously while driving. These cars collect a lot of data. If this set of images is used for training then labeling every image manually could be infeasible. Different objects in the photographs must be identified individually to make this set useful for training machine learning algorithm. An active learning algorithm could be useful in this hypothetical scenario. A human could manually label a small subset of all the collected images to train the machine learning algorithm. Active learning could then be used to determine what data should be labeled next to train the machine learning algorithm. If the active learning query strategy performs well, the amount of data needed to train the machine learning algorithm could theoretically be reduced. All non-informative images could, for example, be excluded from the training set by using active learning.

To generalise the above example, active learning is most useful in scenarios when the machine learning task has one or both of the following properties: (1) All data can not be labeled. (2) Obtaining labels is expensive. If all available data can be labeled, then active learning is redundant. To solve these tasks, active learning is set up consisting of four separate parts:

- **A data set** D consisting of unlabeled data U and labeled data L.
- **A classifier** C to be trained. This classifier usually (but not always) gives an output estimate $P(y = c|\vec{x})$ of point $\vec{x}$ belonging to class c. In this thesis, the only type of classifier examined is artificial neural networks.
- **A query strategy** A that given a classifier C aims to select data points $\vec{x} \in U$ to be labeled and then used for training.
- **An oracle** that can supply true labels to data in U. This can, for example,

be a human. The labeled data is then added to the pool of labeled data L.

As one can imagine, the query strategy is essential for active learning.

Labeling data may be expensive, implying that a dedicated test set does not necessarily exist. The test accuracy is therefore only used as a theoretical measure. Evaluation of the test accuracy after performing active learning is thus complicated. When not having access to a test set, a commonly used solution is to split up the training set into two part: one for training the machine learning algorithm, and one for evaluating the performance. This can be done under the condition that the training set is representative of the test set, which usually is true if the data sets have been sampled in the same way, and thus have a similar distribution. However, performing active learning breaks this condition, as the test set distribution and labeled set distribution will not necessarily be the same. This occurs since active learning is biased towards labeling informative data during construction of the training set. This further complicates the evaluation of the network performance in the absence of a dedicated test set. This thesis will look into ways to address this difficulty.

2.2.1 Query strategies

As mentioned, query strategies select data to be labeled by an oracle. Informative data can then be selected by considering the informativeness measure and be used for training a classifier C. Since the oracle may be a human, we want a strategy that selects as little data as possible but still ensures a good performance of the classifier C. In this section, we present five different query strategies, where one of them has been suggested by us: the *least squares* strategy. The remaining four are called random, least confident, entropy and margin query strategies [9, p. 12]. The query strategies in Settles [9, p. 12] have been slightly modified for consistency in this thesis. Some of query strategies informativeness measures are slightly altered, to ensure that the most informative data has the largest informativeness measure. We can then consider only maximisation without any loss of generality. The query strategies are furthermore generalised such that they can select an ordered batch of the most informative data points, instead of selecting a single sample.

The query strategies presented above have several aspects in common. The input to a query strategy A is the output probability estimates $P_C(y = c|\vec{x})$ from the classifier C for every class c and unlabeled data $\vec{x} \in U$. When using a neural network classifier, the outputs $\hat{y}_{ic}$, see Equation (2.1), are used as the probability estimates $P_C(y = c|\vec{x})$. Based on the input $\vec{x}$ of the neural network, the query strategy assigns an informativeness measure $I^A_{\vec{x}}$. A high informativeness measure means that the data $\vec{x}$ is considered informative and should be labeled by an oracle and used for training. The query strategies then sorts the data according to the informativeness measures, and returns an ordered batch B consisting of a predetermined number n data points

$$B = \{\vec{x}^{(1)}, \vec{x}^{(2)}, \ldots, \vec{x}^{(n)}\},$$

such that

$$I^A_{\vec{x}^{(1)}} > I^A_{\vec{x}^{(2)}} > \cdots > I^A_{\vec{x}^{(n)}},$$

and

$$I^A_{\vec{x}^{(n)}} > I^A_{\vec{x}} \;\; \forall \vec{x} \in U \setminus B.$$

In some of the coming substudies, the mean informativeness measure $\bar{I}^A_B$ of query strategy A of the selected batch B is sometimes of interest, where

$$\bar{I}^A_B = \frac{1}{n} \sum_{\vec{x} \in B} I^A_{\vec{x}}.$$

In the following sections, we present how the different query strategies assigns their informativeness measures $I^A_{\vec{x}}$.

2.2.1.1 Random

The random query strategy, R, assigns a uniformly distributed informativeness measure $I^R_{\vec{x}}$ to all data $\vec{x} \in U$. If active learning is not performed, more classical statistical methods can be used [19], for example cluster sampling. Using the random query strategy is equivalent to what is commonly done when not using active learning, which is why the other query strategies are commonly compared to random sampling. Sampling with the random query strategy is interchangeably referred to as *Random sampling* throughout the thesis.

2.2.1.2 Margin

The margin query strategy, M, assigns informativeness measures to each point $\vec{x} \in U$ as

$$I^M_{\vec{x}} = -[P_C(y = \tilde{c}_1|\vec{x}) - P_C(y = \tilde{c}_2|\vec{x})], \tag{2.3}$$

where $\tilde{c}_1$ and $\tilde{c}_2$ are the most probable and second most probable class, respectively, for the data $\vec{x}$ according to the classifier C.

This query strategy does not only consider data for which it is uncertain, it also considers how certain it is between the classes. The negative sign in $I^M_{\vec{x}}$ is needed to ensure that the samples with the smallest margin, i.e $P_C(y = c_1|\vec{x}) - P_C(y = c_2|\vec{x})$, has the largest informativeness measure. This query strategy tends to select data for which the classifier had a hard time deciding between the two most certain classes. It thus samples from the borders between clusters. Prior research suggests that the margin query strategy achieves a higher test accuracy than the other query strategies [9, p. 14].

2.2.1.3 Least Confident

The least confident strategy, LC, assigns informativeness measures $I^{LC}_{\vec{x}}$ to points $\vec{x} \in U$ based on the negative probability estimate of the most probable class c of $\vec{x}$

$$I^{LC}_{\vec{x}} = -\max_c P_C(y = c|\vec{x}). \tag{2.4}$$

After sorting with respect to the largest informativeness measures, this query strategy selects the batch for which the classifier is least certain in its predictions.

2.2.1.4 Entropy

The entropy query strategy, E, assigns an informativeness measure based on the entropy of a data $\vec{x} \in U$, as

$$I_{\vec{x}}^{E} = -\sum_{c_i} P_C(y = c_i|\vec{x}) \log P_C(y = c_i|\vec{x}). \tag{2.5}$$

After sorting and selecting a batch with the largest informativeness measures $I_{\vec{x}}^{E}$, the batch with the points corresponding to the largest entropy will be selected.

To help with the interpretation of this informativeness measure, we try to explain entropy intuitively. The entropy $H(Q)$ of a stochastic variable Q with a probability distribution P is defined as

$$H(Q) = -\sum_{i} P(q_i) \log P(q_i), \tag{2.6}$$

where q_i is a possible outcome of Q. Note that the $-\log P(q_i)$ term is large when the probability $P(q_i)$ of outcome q_i is small. The term can thus be interpreted as some kind of measurement of "surprise". Since $H(Q)$ is a weighted sum over the terms of surprise, weighted with probability estimates of the outcomes, it can be interpreted as the mean element of surprise. In our case the random variable is the assigned class y of the classifier C given that C is examining $\vec{x}$, i.e. our random variable is $y|\vec{x}$. Plugging this fact into the entropy formula in equation (2.6) gives equation (2.5). Thus, an interpretation of the query strategy is therefore that it selects the data that is most "surprising" to the classifier.

2.2.1.5 Least Squares

The least squares query strategy, LS, assigns informativeness measures to data $\vec{x} \in U$ as

$$I_{\vec{x}}^{LS} = -\sum_{c_i} \left[P_C(y = c_i|\vec{x}) - \frac{1}{N_c} \right]^2,$$

where N_c is the total number of classes in the data set. The idea behind this query strategy is that after sorting based on the informativeness measures and selecting a batch, it should have selected the data $\vec{x}$ for which the assigned class probabilities are almost equal to each other. That is, it assigns a high informativeness measure for points that satisfies $P_C(y = c_1|\vec{x}) \approx P_C(y = c_2|\vec{x}) \approx \cdots \approx P_C(y = c_{N_c}|\vec{x})$. For points where one class has a high probability, i.e. $P_C(y = \tilde{c}|\vec{x}) \approx 1$, the query strategy will assign a low informativeness measure.

2.3 Methodology

This section aims to give more practical implementation details on how to extend the training procedure of neural networks to include active learning. The methodology that the two following chapters, Chapter 3 and 4, have in common is also covered.

Label	MNIST	Fashion-MNIST	CIFAR-10
0	0	T-shirt/top	Airplane
1	1	Trouser	Automobile
2	2	Pullover	Bird
3	3	Dress	Cat
4	4	Coat	Deer
5	5	Sandal	Dog
6	6	Shirt	Frog
7	7	Sneaker	Horse
8	8	Bag	Ship
9	9	Ankle boot	Truck

Table 2.1: The content corresponding to each class for the data sets MNIST, Fashion-MNIST and CIFAR-10.

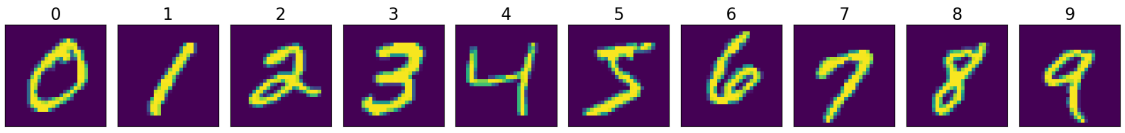
2.3.1 Data Sets

The MNIST [15] data set, the Fashion-MNIST [16] data set and the CIFAR-10 [17] data set was used for training and evaluation of the networks performance. The labels of the images corresponding to each class can be seen in Table 2.1 and examples of the images can be seen in Figure 2.2. All data sets were preprocessed by normalising the data.

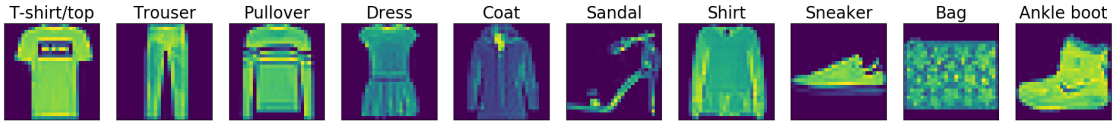
The MNIST data set consists of images of handwritten digits between 0 and 9. All images consist of gray scale values and have a resolution of 28×28 pixels. Some of the images can be seen in Figure 2.2a. The data set is split up in two different data sets: one training set, and one test set. The training set consists of 60000 images and is used to evaluate and optimise the loss in Equation (2.2). The test set consists of 10000 images, used to measure the network performance.

The Fashion-MNIST data set is in many ways similar to the MNIST data set. It also consists of gray scale values and have a resolution of 28×28 pixels. These images represent 10 different pieces of clothing, some of which can be seen in Figure 2.2b. This data set is known for being slightly more difficult to classify compared to MNIST [16]. This data set is, like the MNIST data set, split up in a training and test set, each consisting of 60000 and 10000 images, respectively.

The CIFAR-10 data set consists of RGB-coloured images with a resolution of 32×32 pixels, depicting 10 different classes that can be seen in Figure 2.2c. Since each image has three channels, one for each of the colors red, green, and blue, there are many more features in this data set compared to the MNIST and Fashion-MNIST data sets. This data set is split up into a training and test set, each consisting of 50000 and 10000 images, respectively.



(a) Example images from the MNIST data set.



(b) Example images from the Fashion MNIST data set.

(c) Example images from the CIFAR-10 data set.

Figure 2.2: Samples from the three different examined data sets where each image shows a photo of the object, with the true label above it.

2.3.2 Neural Network Architecture

For both the Fashion-MNIST and the MNIST data set the same type of networks were used. The networks consist of an input layer of 28×28 input neurons, a hidden layer with m neurons, a 0.2 dropout layer, and an output layer with 10 softmax output neurons. The dropout randomly sets 20% of the data to 0 during training, which reduces overfitting of the data. The rest of the inputs are propagated forward. The weights are initialised according to the Glorot uniform distribution and the biases are initialised to zero (0) [20]. The Adam optimiser [21], with default learning rate of 10^{-3}, was used to train the network. This network is referred to as $ANN1_m$, and its python implementation can be seen in Appendix A.1.1.

For the CIFAR-10 data set a more advanced convolution neural network was used, denoted called $CNN1$ throughout the thesis. The weights are initialised according to He uniform distribution and the biases are initialised to zero (0). To train and optimise the network, the stochastic gradient descent optimiser was used with a learning rate of 0.001 and a momentum of 0.9. The network was implemented with the python code shown in Appendix A.1.2.

Tensorflow's standard settings for the hyperparameters were used when training and evaluating the networks for a training set. These can be seen in Appendix A.1.3.

2.3.3 Training the Model

During selection and labeling, there are two ways of training the network after new data has been labeled. One method, called *Incremental Training*, is to use the trained network from the previous iteration and update it by training with only the recently labeled data. The other method, called *Cumulative training*, is to discard

the trained network from the previous iteration and use all labeled data to train a new network. In short: given a previously labeled data set L and a newly labeled batch B, the training methods can be summarised as

- *Incremental Training*: Update the weights and biases by training on batch B.
- *Cumulative Training*: Re-initialise the weights and biases and train on $L \cup B$.

A potential benefit of performing incremental training is that it should be faster since it is training on a smaller set of data each iteration. However, incremental training trains with data in a certain order, chosen by active learning. It is not known if this is beneficial or not. An investigation will be done to examine which of these training methods are preferable in an active learning setting.

2.3.4 Illustration of the Results

Each experiment is repeated several times and the mean of all simulations is computed. The results may vary a lot between runs since all algorithms are stochastic in nature. The stochasticity of the network originates both from the initialisation of weights and the stochastic gradient descent. Therefore, many reruns were needed to get the averaged test accuracy for all experiments. How many times each experiment was done depends on the time required to do each simulation. Shorter simulations could be redone many times and long simulations could only be done few times. Each plot shows the metric in question averaged over all reruns as a non-transparent line. To illustrate the variance of the reruns, an auxiliary area is filled with the corresponding transparent color to show one standard deviation above and below the mean. The test accuracy a is usually the most interesting metric, which is calculated as the fraction of data predicted correctly:

$$a = \frac{\sum_{i=1}^{N} y_i - \gamma_i}{N},$$

where N is the number of data, y_i is the correct label for the neural network input $\vec{x}_i$, $\gamma_i = \arg\max_c \hat{y}_{ic}$ is the predicted label of the network, and $\hat{y}$ is the output of neural network from Equation (2.1).

The majority of all plots in this thesis shows the accuracy for different methods on the y-axis, and the amount of data that has been labeled from the unlabeled set on the x-axis. The interesting part of each plot is often what accuracy is obtained for a subset of the entire data. Each plot should thus be understood as the x-axis showing the budget, i.e. how much data can be labeled. The plots will then show what accuracy can be obtained with this budget constraint.

2.3.5 Implementation

For all results in this thesis, TensorFlow 2.1 [22] was used to train the artificial neural networks, given a data set. To train a network with a query strategy A, in

short, we evaluate the neural network on all unlabeled data, and given the output of the network, we select a batch of samples to be labeled and trained on according to a query strategy. This can be summarised in the following steps:

1. Initialise a neural network.

2. (*Optional*) If any previously labeled data exists, use it to train the model.

3. (*Optional*) Label a number of randomly sampled data. This is motivated in the *Random Start* substudy, see 4.1.3, and the *Estimates of the Test Accuracy* substudy, see Section 4.1.4.

4. Evaluate the network on all the unlabeled data, i.e. $\forall \vec{x} \in U$ if it is possible. If the amount of CPU-time, GPU-time or memory is a limiting factor, evaluate the network on a random subset of all the unlabeled data. This gives an output probability estimate $P(c|\vec{x})$ for every class c for every point $\vec{x} \in U$ or for every point in the random subset.

5. Feed the output probability estimates $P(c|\vec{x})$ to the given query strategy A. The query strategy returns a batch B consisting of the $n_{\text{batch size}}$ most informative points to be used for training. See Section 2.2.1 for more information.

6. Ask the oracle for labels $y_{\vec{x}}$ for data $\vec{x} \in B$.

7. Change the set of all labeled points L to reflect the newly labeled data, i.e. $L \leftarrow L \cup B$.

8. Train the model with either incremental training or cumulative training:

   - *Incremental Training*: Update the weights and biases by training on batch B.

   - *Cumulative Training*: Re-initialise the weights and biases and train on $L \cup B$.

9. Evaluate and store the losses and accuracy obtained by the model on the training set $L \cup B$, and any potential test set.

10. Repeat by going to step 4 until a budget constraint is met, or if it is deemed that labeling more data is not valuable anymore.

11. (Optional) Perform the semi-supervised learning method *Gradual psuedo-label assignment* using both the labeled and unlabeled data. This method is investigated in the *Semi-supervised Learning* substudy, see Section 4.1.6.

3

Model-Centric Investigation

In this chapter different ways of performing active learning are compared. It is divided into two substudies, which are presented in parallel:

- An investigation of how to combine query strategies and training methods.
- How to interpret the informativeness measures.

The substudies can either be read as presented or a particular substudy can be read on its own.

The margin query strategy is appropriate for reducing the classification error. This means that it often achieves a high test accuracy [9, p. 14], which is the main metric used in this thesis to evaluate if active learning is beneficial. The research throughout this thesis also supports that the margin query strategy performs the best. Therefore, margin is the main query strategy that is analyzed throughout this chapter and Chapter 4. Chapter 4 assumes that the reader is aware of the results presented in this chapter.

3.1 Background

In this chapter, the substudies have been designed to isolate the implications of the different choices which can be made while applying active learning with artificial neural networks. Among these, we have the choice of which query strategy to use, whether to train incrementally or cumulatively, and how to interpret the informativeness measures to evaluate the performance of active learning. We aim to generalise the results by examining if the same conclusion can be reached for a variety of data sets and hyperparameters and number of parameters of the model.

3.1.1 Query Strategies and Training Methods

In the theory chapter several query strategies were introduced, see Section 2.2.1, as well as different ways to train a network, see Section 2.3.3. Different combinations of training methods and query strategies may be useful under different circumstances, for example, for different network architectures, data sets and the amount of data

used for training. The performance of the network, given the different circumstances, must thus be investigated. This substudy examines the query strategies, the two proposed training methods and how these effects the performance of the network. Mainly two questions are investigated:

- Are different combinations of query strategies and training methods relevant in different scenarios and for different data sets?
- What query strategy and training method should be used to achieve the best possible test accuracy, for a given network?

The incremental and cumulative training methods can hypothetically benefit the training of the network in different ways. The benefit of the incremental method is that it should be computationally light since it trains only on a small batch for every iteration. However, the computational time might not be an issue in an active learning setting if it is cheap compared to the cost of asking an oracle for labels. A key difference between the methods is that the incremental training method acquires and therefore trains on the data in a certain order. The cumulative method, on the other hand, utilises the entire labeled data set and randomises the training order. Hypothetically, training on the data in a certain order could both be beneficial and disadvantageous. For example, a presumably non-beneficial way would be to train on informative data in the beginning and on non-informative data (when the interesting points are depleted) in the end.

The hyperparameters, *batch size* and *random subset size*, may also change the benefits of active learning, which is why these two parameters will be tested. The batch size determines how much data is labeled in one instance while the subset size determines how many data points, from the unlabeled set, can be considered for labeling. A smaller batch size means a higher resolution in picking the most informative data. It is however computationally heavy to have a small batch size since this means more iterations of active learning has to be performed. The subset size may be decreased if it is infeasible to consider all data due to memory limitations or simply to speed up the training. It is thus interesting to know how the batch size and the size of the random subset affects the training of the network when using a specific query strategy.

3.1.2 Informativeness Measures

Common for all query strategies is that they assign an informativeness measure to unlabeled data. We want to investigate how the informativeness measures relate to the performance of the network. For example, at some point, labeling new data may be too expensive compared to the improvement in the test accuracy. In an active learning setting, it would be beneficial if a stopping criterion could be constructed using only the training accuracy, the training loss and the informativeness measures of the selected data. We examine the test loss, training loss, test accuracy, and training accuracy, as well as the informativeness measure given by the query strategy.

Questions investigated include:

- When should new data be gathered, instead of labeling more of the existing data?
- When is it no longer beneficial to train the network?
- How can the informativeness measure be interpreted?

3.2 Method

3.2.1 Query Strategies and Training Methods

This substudy compares the test accuracy for networks with different combinations of training methods, query strategies. Two smaller experiments, investigating the batch size and random subset size, are also performed in this substudy.

The implementation used has been explained in Section 2.3.5, and is performed without any of the optional steps. For the main investigation of this substudy, several networks were tested, namely $ANN1_{10}$, $ANN1_{100}$, and $ANN1_{1000}$, to examine how the number of trainable parameters in the network influences the choice of query strategy and training method. Both the MNIST and Fashion-MNIST data sets were used in this substudy. The different networks were trained either incrementally or cumulatively using different query strategies with a batch size of 120. All query strategies, i.e. least confident, entropy, margin, and least squares, see Section 2.2.1, are compared to the random query strategy. The accuracy and loss of different training methods are also compared.

Two additional choices relevant to training were also analyzed. Experiments were designed to determine the importance of the batch size and the random subset size. The batch size was investigated by training an $ANN1_{100}$ network both incrementally and cumulatively with the margin query strategy using different batch sizes. The batch sizes analyzed were 30, 120, 480, and 960. To analyze the random subset size an $ANN1_{100}$ network was trained using the cumulative and incremental training method with the margin query strategy. The random subset sizes considered uniformly drawn samples of sizes 60000, 20000, 5000, and 2000. Note that using a subset size of 60000 is the same as considering all data, since in this case the query strategy strategy is allowed to select data from the whole unlabeled data set U. Both the MNIST and Fashion-MNIST data sets consist of 60000 samples. Samples were selected from the random subset using the query strategy with a batch size of 1000. Using this batch size allows for comparison when the random subset size and batch size are similar in size. The MNIST and Fashion-MNIST data sets allow for a random subset size of 60000. CIFAR-10, however, only allowed for a random subset size of 5000 with the available hardware.

3.2.2 Informativeness Measures

This substudy examined the relationship between the performance metrics, training metrics, and informativeness measure while training cumulatively with a query strategy. The metrics examined were test accuracy, train accuracy, test loss, and training loss. The mean informativeness measure $\bar{I}_B^A$ was calculated for all data in a batch B selected by query strategy A. We decided to limit this substudy to only a comparison between the margin query strategy and the random query strategy, based on the results from the *Query Strategies and Training Methods* substudy.

To examine how these metrics alter under the influence of different query strategies, both the margin query strategy M and the random query strategy R was used to select points from the unlabeled data set. The algorithm from Section 2.3.5 was used with cumulative training and without performing any optional steps to train the networks. We want a horizontal comparison of the informativeness measure of batches selected by different query strategies. It is difficult to compare the informativeness measures since the different query strategies output informativeness measures with different interpretations and ranges. Thus, we need a way to calculate how informative a single batch is, independently of how it was sampled. To do this, we need to evaluate the informativeness measures of a query strategy M on points selected by query strategy R.

To obtain the margin informativeness measure from the margin query strategy M on randomly sampled data, some extra steps were needed. The randomly sampled batch B was re-fed through the network trained with the random query strategy, which gives the probability estimates $P(c|\vec{x})$ for every point $\vec{x} \in B$. The probability estimates were then used to calculate the margin informativeness measure $I_{\vec{x}}^M$ for every point $\vec{x} \in B$, as in Equation 2.3. This is called "*margin on random informativeness measure*" in the results. The mean of the informativeness measures considered in the results when comparing the random query strategy R to the margin query strategy M. Note that the query strategy does not select data to be labeled in this case, it merely evaluates the informativeness measures for all points in B.

Each batch consisted of 120 images, and in total 500 batches were used for training. Two different types of neural networks were considered, namely $ANN1_{10}$ and $ANN1_{100}$, all using 1 epoch for training. $ANN1_{1000}$ gave a similar performance compared to the $ANN1_{100}$ network in the *Query Strategies and Training Methods* substudy, in Section 3.3.1. It was therefore not considered in this substudy.

The CIFAR-10 dataset was also used in this substudy, to see how the results generalize to a different type of data set and network. For this data set, the $CNN1$ network was used. The network was allowed to either train on the data with 50 epochs, or a maximum of 50 epochs with early stopping. The early stopping occurred when the validation accuracy, on a validation set consisting of 2000 points decreased over three successive epochs. The training was done with a batch size of 120.

3.2.2.1 Split Data

It is valuable to know when labeling more data is not beneficial anymore. To investigate this, an experiment was conducted to investigate the benefit of expanding the unlabeled data set, in comparison to labeling from the original unlabeled data set.

During the later stage of the training, when almost all data has been labeled, it is observed that the mean informativeness measure decreases, see Figure 3.4. There are two hypotheses to why $\bar{I}_B^M$ decreases. The first hypothesis is that the network learns the characteristics of the data such that no new data, with the same distribution, is informative anymore. This means that the decrease of $\bar{I}_B^M$ is due to the network becoming more certain. The second hypothesis is that when performing training with a query strategy, the most informative data is selected early, leaving uninformative data left in the end. This means that the decrease of $\bar{I}_B^M$ is due to exhaustion of informative data.

An experiment was performed to find which of the two hypothesis was the major contributor to the decrease in $\bar{I}_B^M$. The unlabeled data set was split into d subsets. The margin query strategy selected a fraction p points from the first subset. The next data set was then introduced to be considered for labeling, i.e. it was added to the unlabeled data set. This was done iteratively for all d subsets. The algorithm from Section 2.3.5 was used with cumulative training of an $ANN1_{100}$ network and the optional step of training with all previously labeled data. The margin query strategy was used to select data to be labeled from each subset. Two types of experiments were done with the following parameters:

d	p	batch size	number of samples in each subset
2	80%	120	30000
5	90%	200	12000

performed on both the MNIST and Fashion-MNIST data sets.

If the $\bar{I}_B^M$ increases drastically upon the introduction of a new data set, then it is a sign that the depletion of informative data is a major contributor to why the $\bar{I}_B^M$ decreases, i.e. supports hypothesis two above. If there is a considerable difference between the decrease of the $\bar{I}_B^M$ upon the introduction of the first compared to the last data set, then the interpretation is that the main reason, for the decrease in $\bar{I}_B^M$, is that the network is certain about the classification, i.e. supports hypothesis one above. Figure 3.10 shows the results of this experiment.

3.3 Results

In this section, the results are present from the two substudies. The most important results will be presented and later discussed in Section 3.4.

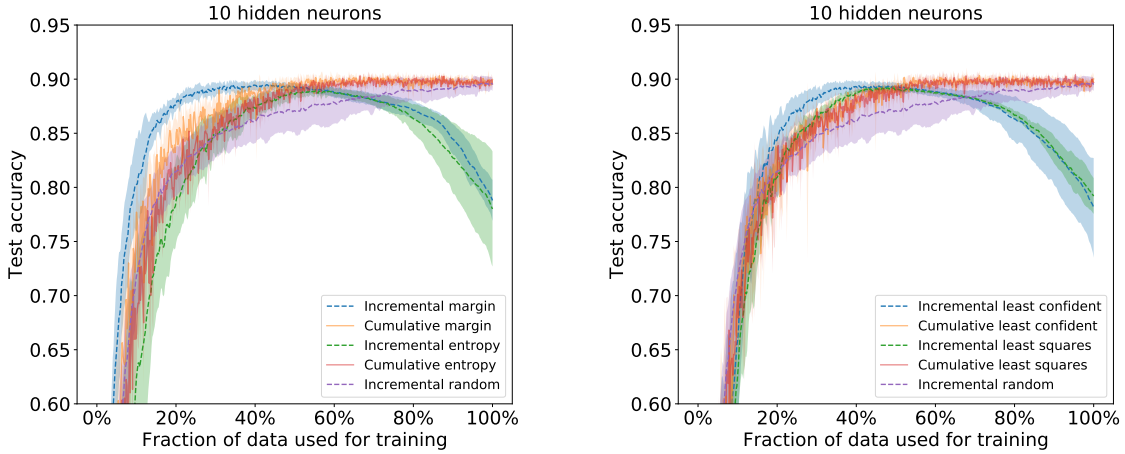
3.3.1 Query Strategies and Training Methods

This section presents the plots that show the test performance of networks trained with different query strategies and training methods. Figures 3.1, 3.2, and 3.3 show the test accuracies on the MNIST and Fashion-MNIST data sets for the $ANN1_{10}$, $ANN1_{100}$ and $ANN1_{1000}$ networks, respectively. Most important in this substudy is how the test accuracy is influenced by the choice of query strategy and training method. Note that the results for the cumulatively trained network with the random query strategy are missing from all figures. This is because it had a similar performance to the network trained incrementally with the random query strategy, and was thus left out to better highlight the comparison between the other query strategies. See Appendix A.3 for a comparison between networks trained incrementally and networks trained cumulatively with the random query strategy.

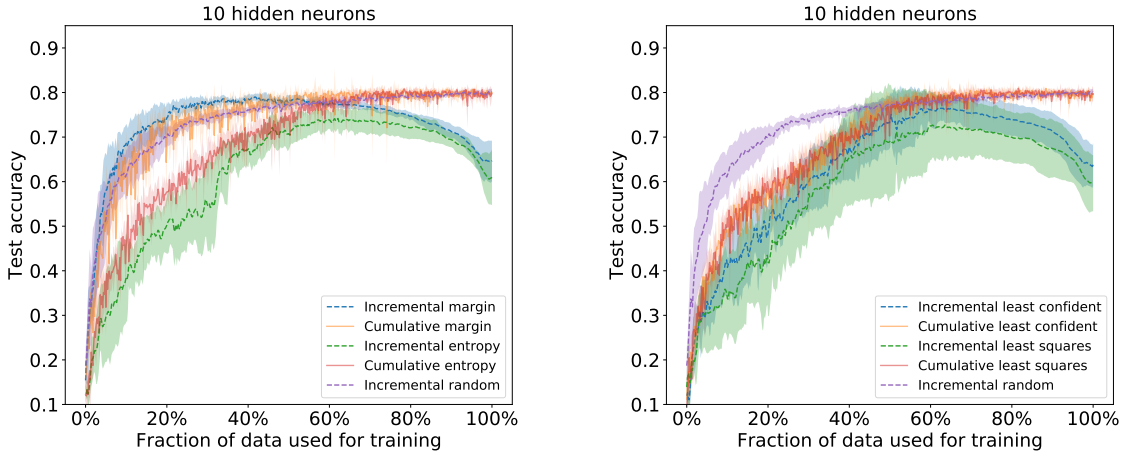
3.3.1.1 $ANN1_{10}$ Networks.

The performance of the different query strategies and training methods for $ANN1_{10}$ networks is shown in Figure 3.1. The figure shows that the test accuracies for the entropy, least confident and least squares have all similar performance for cumulatively trained networks on both the MNIST and Fashion-MNIST data sets, On the MNIST data set, these query strategies outperform the random query strategy after around 30% of the unlabeled data set has been labeled and used for training. The network trained cumulatively with the margin query strategy, outperforms the random query strategy after 10% of the data has been labeled. For the Fashion-MNIST data set, the entropy, least confident, and least squares query strategies are initially outperformed by the random query strategy for cumulatively trained networks. They reach a similar accuracy to the random query strategy after around 60% of the data has been labeled. The margin query strategy with cumulative training has a similar performance to the random query strategy on the Fashion-MNIST data set.

Training incrementally with the query strategies yield different results. For the MNIST data set, all query strategies perform better than the random query strategy when between 20% and 60% of the unlabeled data set has been labeled and used for training. Common between the query strategies is that they reach a maximal test accuracy when around 30% of the data has been used for training. Labeling and training on more than 30% of the unlabeled data set decreases the test accuracy. Some of these statements also hold true for the Fashion-MNIST data set, with the difference that the margin query strategy reaches a maximum at around 30% of training. The other query strategies reach a maximum after around 60%. Most query strategies are, however, not performing better than random for the incrementally trained network. Only the margin query strategy outperforms the random query



(a) MNIST.



(b) Fashion-MNIST.

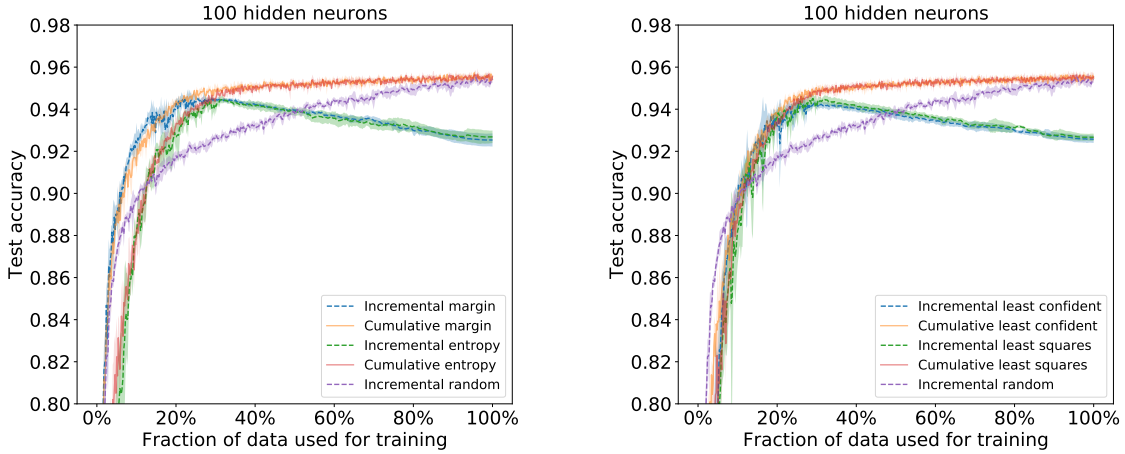
Figure 3.1: Test accuracy achieved for training an $ANN1_{10}$ network on the MNIST and Fashion-MNIST data sets, averaged over 5 reruns. The figures include the combination of both training methods and all query strategies.

strategy until 50% of the unlabeled Fashion-MNIST data set has been labeled and trained on.

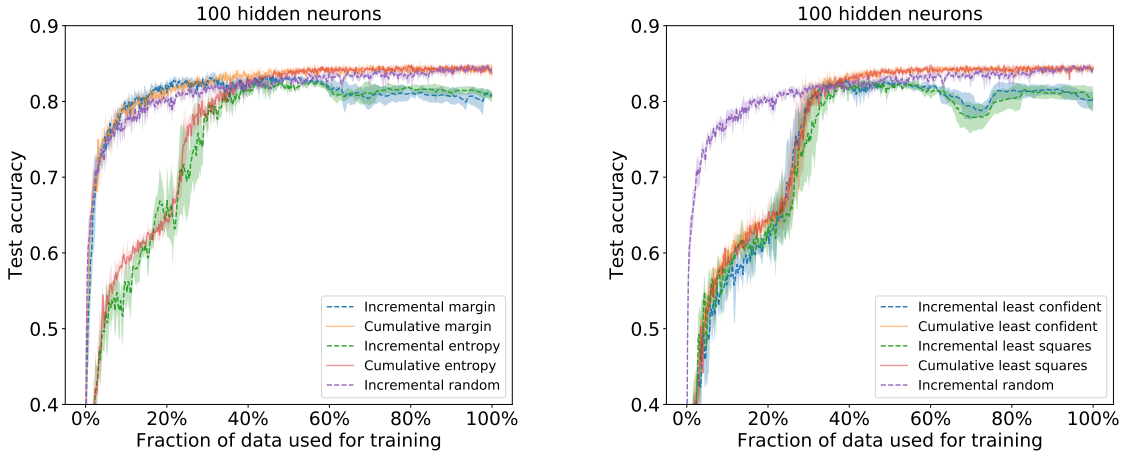
The $ANN1_{10}$ network trained incrementally reaches a higher test accuracy than the cumulative trained network up until 50% of the data had been used for training. This holds true for both the MNIST and Fashion-MNIST data sets with the margin query strategy as well as the least confident query strategy for the MNIST data set.

3.3.1.2 $ANN1_{100}$ and $ANN1_{1000}$ Networks.

The test accuracy of the different query strategies for $ANN1_{100}$ and $ANN1_{1000}$ networks can be seen in Figure 3.2 and Figure 3.3, respectively. The content of the figures are similar, which allows highlighting of the most important results efficiently. The figures show that the networks trained cumulatively with the entropy, least



(a) MNIST

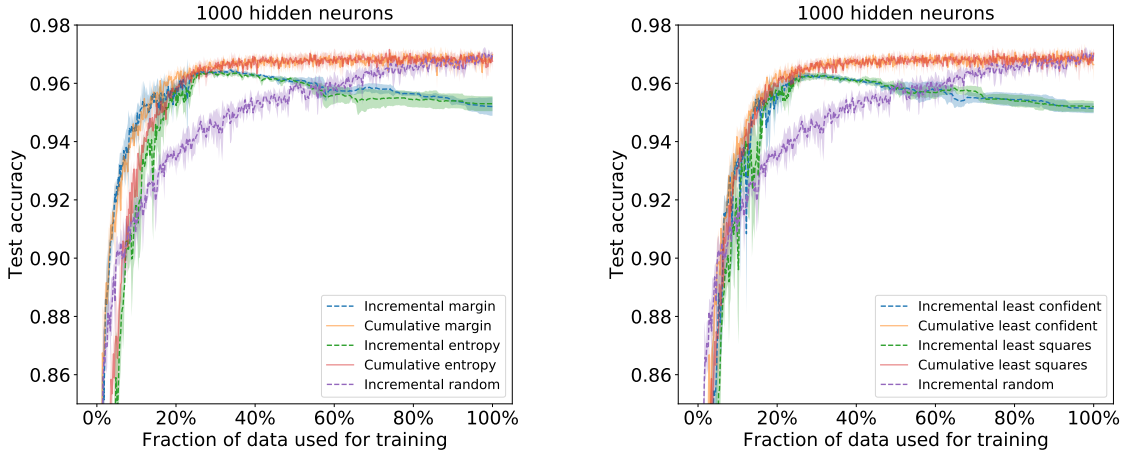


(b) Fashion-MNIST

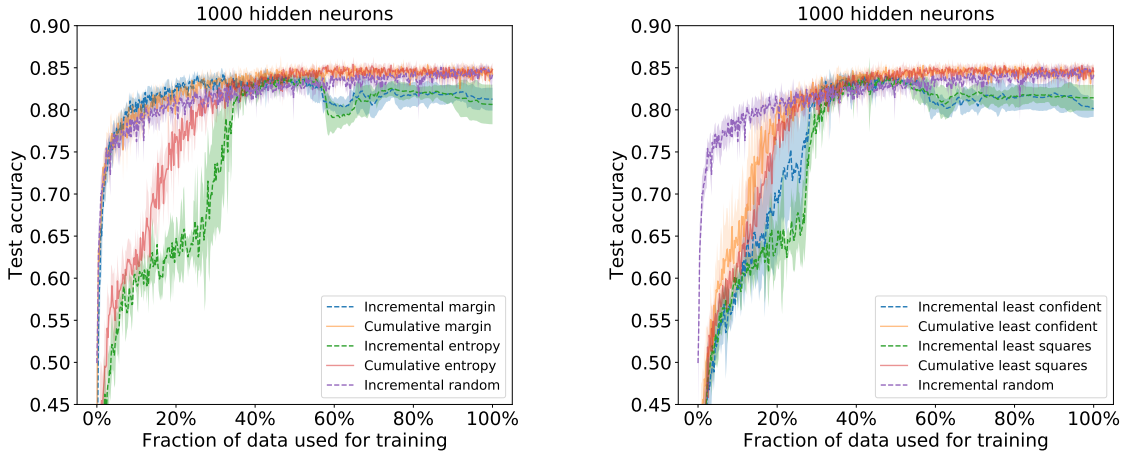
Figure 3.2: Test accuracy achieved for training an $ANN1_{100}$ network on the MNIST and Fashion-MNIST data sets, averaged over 5 reruns. The figures include the combination of both training methods and all query strategies.

confident, and least squares query strategies have a similar performance for both data sets. On the MNIST data set, these query strategies outperform the random query strategy after 10% of the unlabeled data set has been labeled and used for training. The network trained cumulatively with the margin query strategy outperforms the network trained with the random query strategy. For the Fashion-MNIST data set, the entropy, least confident, and least squares query strategies are initially outperformed by the random query strategy for cumulatively trained networks. They reach a similar accuracy to the random query strategy after around 40% of the data has been labeled. The margin query strategy with cumulative training outperforms the random query strategy for this data set as well.

When training incrementally on the MNIST data set, all query strategies are performing better than the random query strategy between 10% and 50% of the unlabeled data set has been labeled and used for training. Common for all of them,



(a) MNIST



(b) Fashion-MNIST

Figure 3.3: Test accuracy achieved for training an $ANN1_{1000}$ network on the MNIST and Fashion-MNIST data sets, averaged over 5 reruns. The figures include the combination of both training methods and all query strategies.

except for margin, is that they reach a maximal test accuracy when around 30% of the data has been used for training. Labeling and training on more than 30% of the unlabeled data set decreases the test accuracy. The margin query strategy reaches a maximal test accuracy earlier, at around 25%. On the Fashion-MNIST data set, the other query strategies reach a maximum test accuracy after around 45%, while for the margin query strategy the same happens after approximately 30%. All query strategies with an incrementally trained network, except for margin, achieve a lower test accuracy than the random query strategy. Only the margin query strategy outperforms the random query strategy before 35% of the unlabeled Fashion-MNIST data set has been labeled and used for training.

3.3.1.3 Training Metrics

Figure 3.4 shows the training accuracy, training loss, and mean informativeness measure $\bar{I}_B^M$ for every batch B selected by the margin query strategy M, for both the incremental and cumulative training methods training the $ANN1_{10}$ network. Plots for the $ANN1_{100}$ and $ANN1_{1000}$ can be seen in the Appendix, in Figure A.1 and A.2, respectively. The $ANN1_{100}$ and $ANN1_{1000}$ behave similarly. The figures show that $\bar{I}_B^M$ starts to decrease earlier, the training loss is higher and the training accuracy is lower for incremental training compared to cumulative training. The difference in training accuracy and the training loss between the training methods increases when almost all data in the unlabeled data set has been used for training, disadvantageously for the incrementally trained networks.

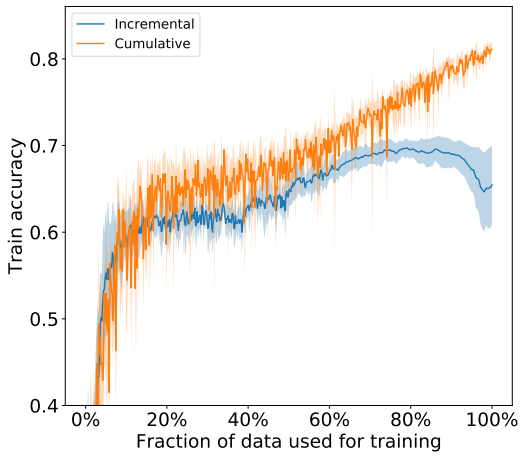
3.3.1.4 Optimal Batch Size Choice

The impact of the batch size on the test accuracy on an $ANN1_{100}$ network can be seen in Figure 3.5. The figure shows that the performance of the incrementally trained network is sensitive to the choice of batch size, where a larger batch size impacts the performance negatively. The batch size size of 120, which was previously used, seems to let the network perform well with the cumulative method. The cumulative training method seems to be resilient to varying batch size while the incremental training seems to not be resilient, see Figure 3.5.

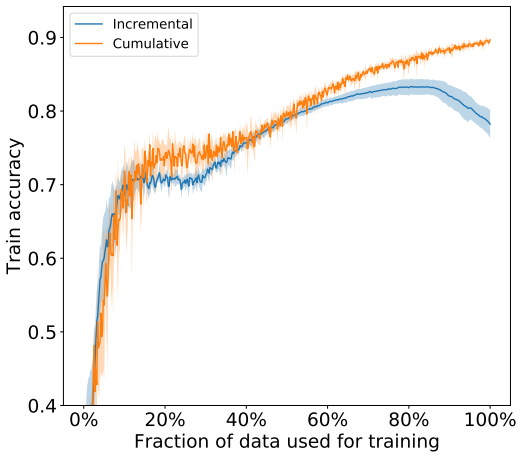
3.3.1.5 Subset Training

The results in Figure 3.6 shows the impact of the random subset size, on the test accuracy of an $ANN1_{100}$ network, trained cumulatively with the margin query strategy. Note that the random query strategy is not used to select data, it only limits what data the margin query strategy can choose from. The result for the MNIST data set, Figure 3.6a, shows that even if only 3% of the original data set is considered, the margin query strategy still has a better performance compared to the random query strategy. Moreover, the figure shows that the performance decreases for smaller subsets. Also, the random subset can consist of 8%, or 5000 points out of 60000, without losing any significant loss in test accuracy.

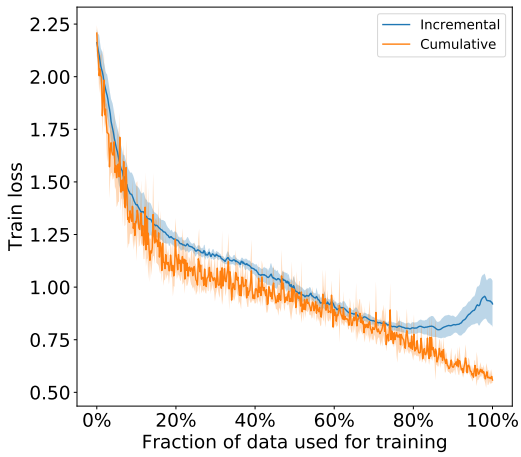
For the Fashion-MNIST data set, see Figure 3.6b, the size of the random subset does not impact the performance as much as for MNIST. The test accuracy was comparable for all subset sizes considered.



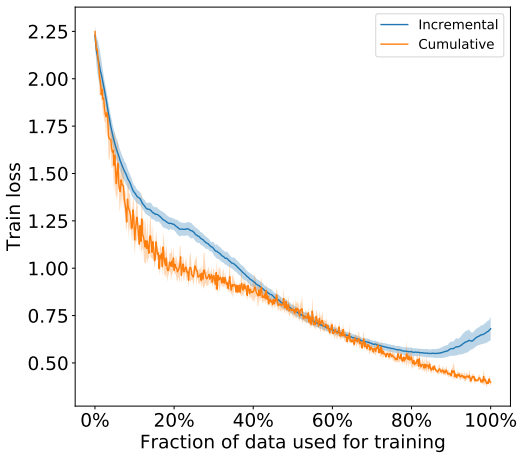
(a) Training accuracy Fashion-MNIST



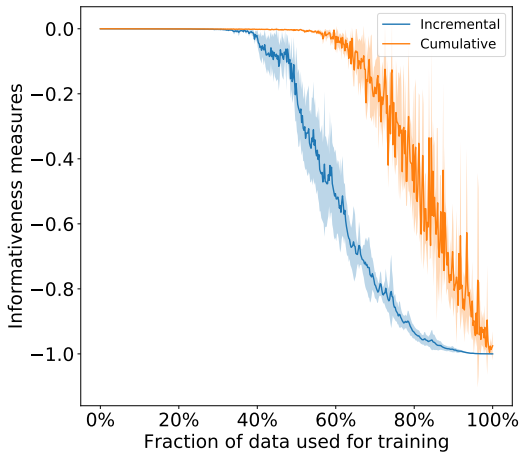
(b) Training accuracy MNIST



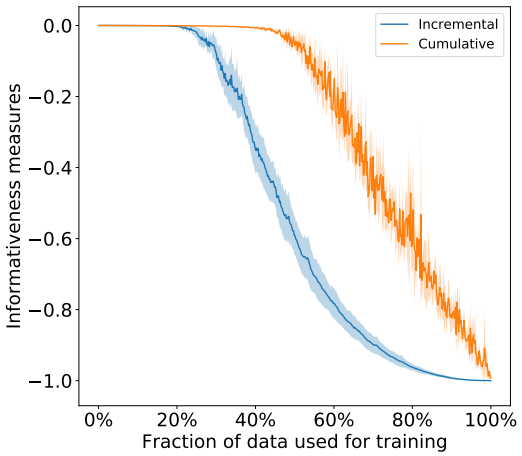
(c) Training loss Fashion-MNIST



(d) Training loss MNIST

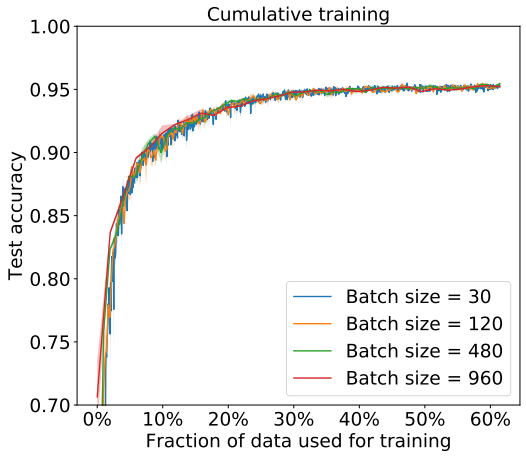


(e) Mean informativeness measures Fashion-MNIST.

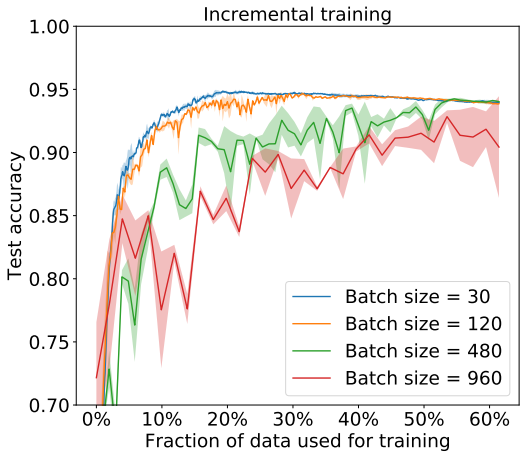


(f) Mean informativeness measures MNIST.

Figure 3.4: Shows different metrics for the two training methods using an $ANN1_{10}$ network with the margin query strategy for both the MNIST and Fashion-MNIST data sets for 5 reruns. The corresponding test accuracy can be seen in Figure 3.1.

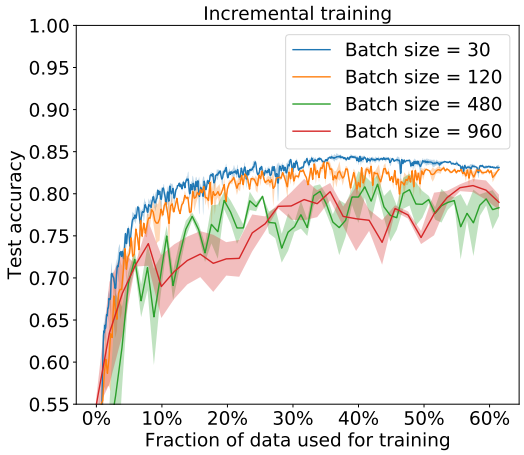


(a) Cumulative training and MNIST data set.



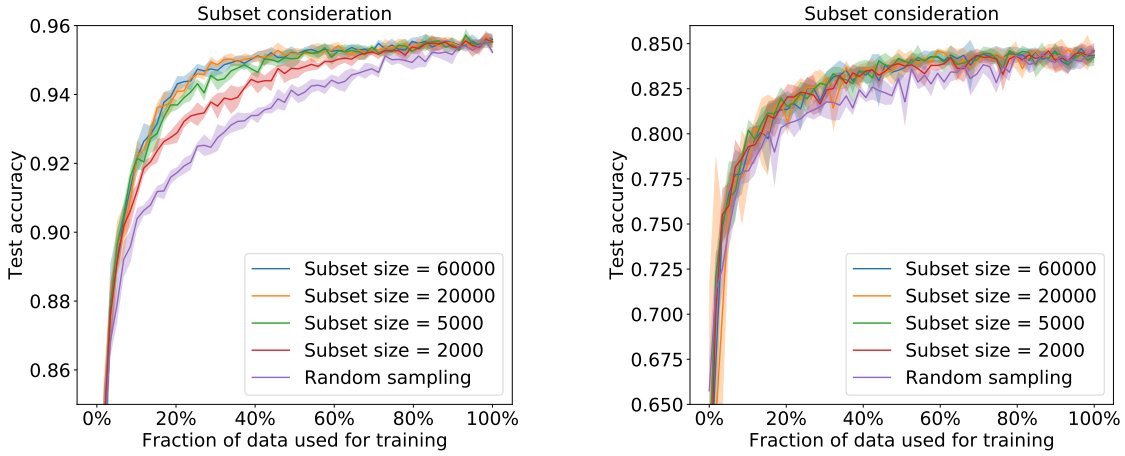
(b) Incremental training and MNIST data set.

(c) Cumulative training and Fashion-MNIST data set.



(d) Incremental training and Fashion-MNIST data set.

Figure 3.5: The test accuracy for a number of different batch sizes for two different data sets, Fashion-MNIST and MNIST for an $ANN1_{100}$ network. Points were selected for labeling using the margin query strategy, averaged over 2 reruns.

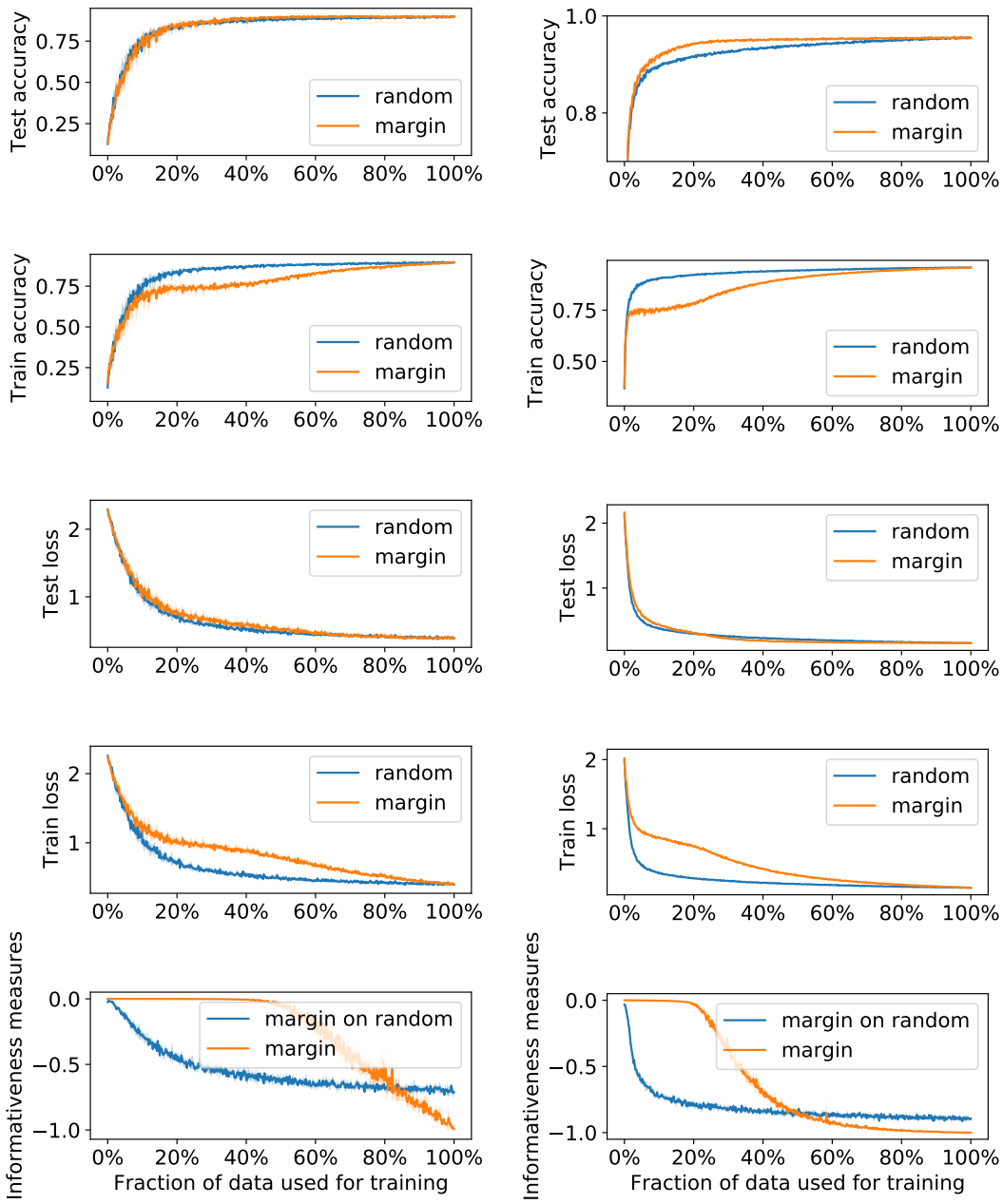


(a) MNIST data set

(b) Fashion-MNIST data set

Figure 3.6: The test accuracy for an $ANN1_{100}$ network trained cumulative training method with the margin query strategy, for varying sizes of subsets. A batch size of 1000 was used for training. The experiment was averaged over 5 reruns.

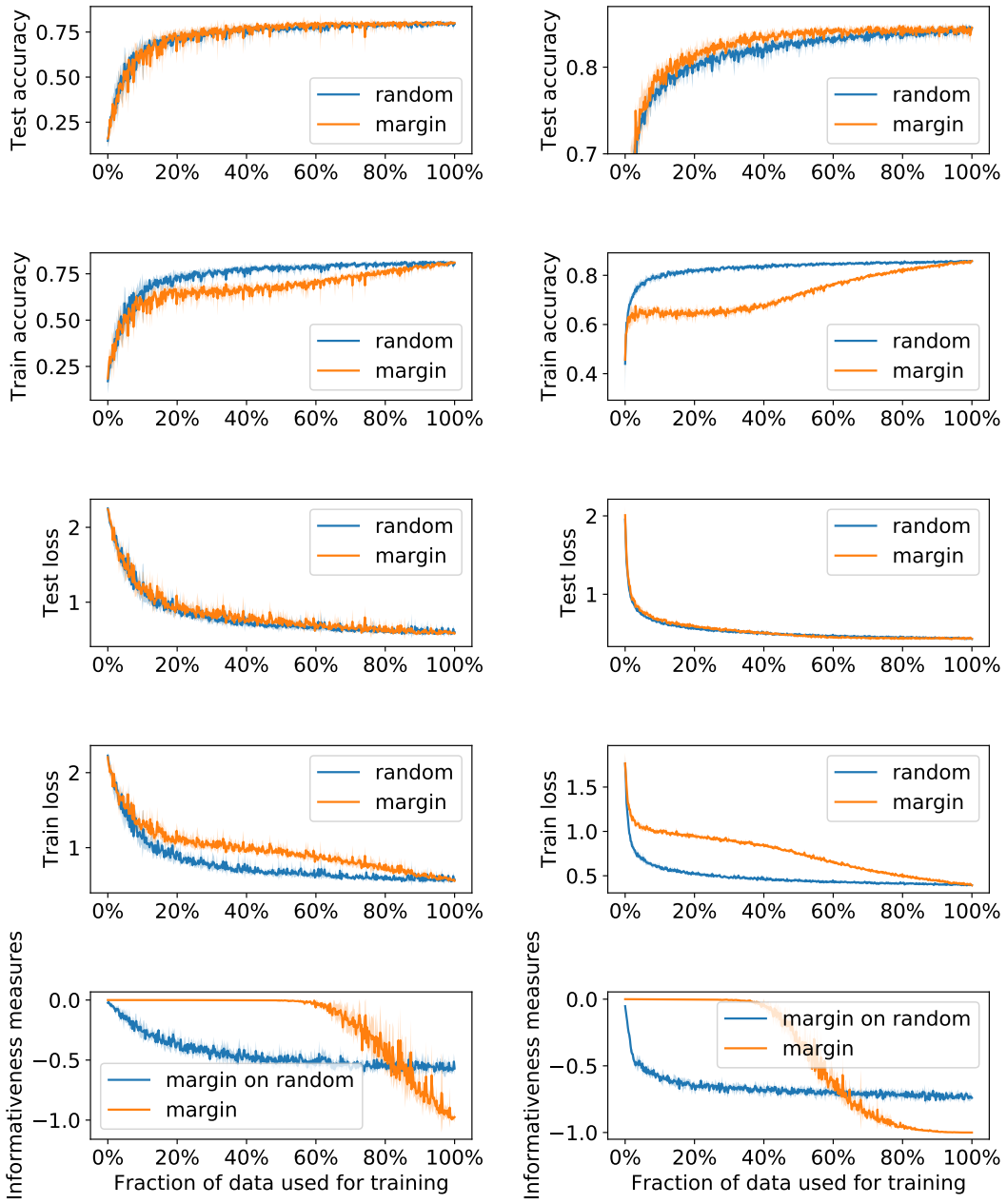
3.3.2 Informativeness Measures



(a) $ANN1_{10}$ **(b)** $ANN1_{100}$

Figure 3.7: Shows test accuracy, training accuracy, test loss, training loss, and the mean informativeness measure from the margin query strategy while training the MNIST data set. These results were obtained with the cumulative training method and repeated 5 times.

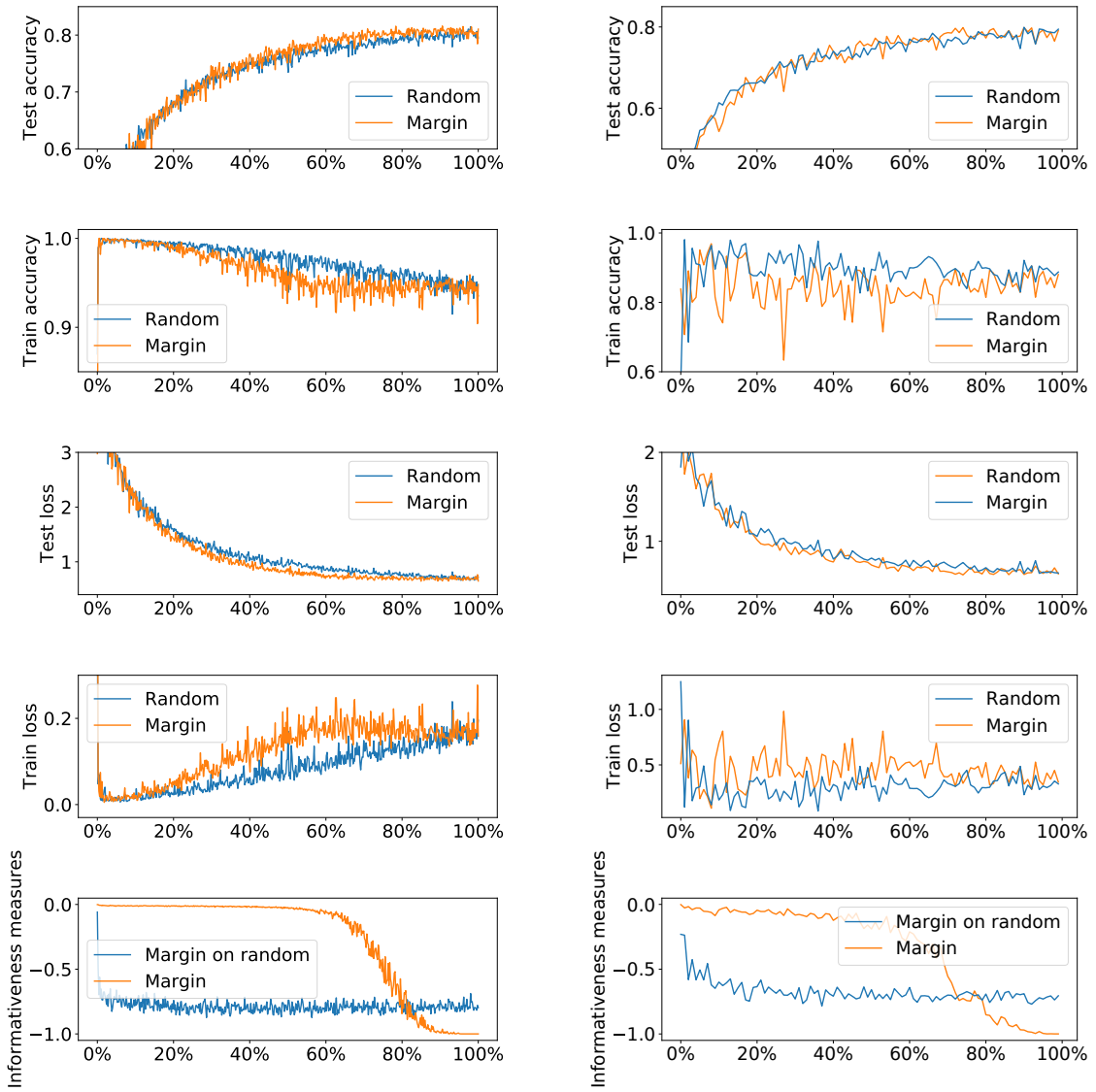
The results can be seen in Figures 3.7 and 3.8 for the MNIST and Fashion-MNIST data sets, respectively. Note, in particular, that the trends between the data sets are very similar. These figures show how the mean informativeness measures of the data in the selected batch are changing in relation to the test accuracy, training accuracy, test loss, and the training loss. Looking at the training accuracy, there is



(a) $ANN1_{10}$. **(b)** $ANN1_{100}$.

Figure 3.8: Shows test accuracy, training accuracy, test loss, training loss, and the mean informativeness measure from the margin query strategy while training on the Fashion-MNIST data set. These results were done with the cumulative training method and repeated 5 times.

a notable difference between the random and margin query strategies. For networks trained with the random query strategy, the training accuracy is similar to the test accuracy. For the margin counterpart, the training accuracy is lower compared to the test accuracy. Another notable remark is that there exists a phase where the training accuracy is approximately constant, for both networks, while training with



(a) Results for CIFAR-10 with no early stopping

(b) Results for CIFAR-10 with early stopping.

Figure 3.9: Shows test accuracy, training accuracy, test loss, training loss, and the mean informativeness measure from the margin query strategy while training on the CIFAR-10 data set. These results were done with the cumulative training method and repeated 1 time. A random subset size of 5000 of the unlabeled data set was used during the runs, due to memory limitations.

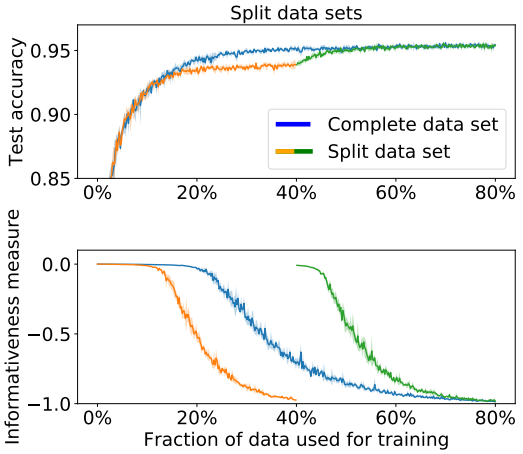
the margin query strategy. For an $ANN1_{10}$ network, this phase is when the network has trained on between 10% to 30% of the total training set. For an $ANN1_{100}$ network, this is between 3% to 20%. The figures show that when the training accuracy starts to increase after this plateau, the test accuracy has also reached a plateau. The figures also show that the test loss of the networks trained with the margin query strategy is higher, for some period, compared to the network trained with the random query strategy. This difference between the margin and random

query strategy is more significant when comparing the training loss. Figures 3.7 and 3.8 also show that the margin query strategy keeps the mean informativeness measure high in the early stage of training. For an $ANN1_{10}$ network, the mean informativeness measure is close to 0 until about 23% of the training data set has been labeled. This also observed for the $ANN1_{100}$ network until around 18% of the data set has been used for training. The mean margin informativeness measures evaluated on the batch selected by the random query strategy decreases from the start of the training. Lastly, a comparison between the two networks shows that the mean informativeness measures changes in a quicker rate for an $ANN1_{100}$ network compared to an $ANN1_{10}$ network.

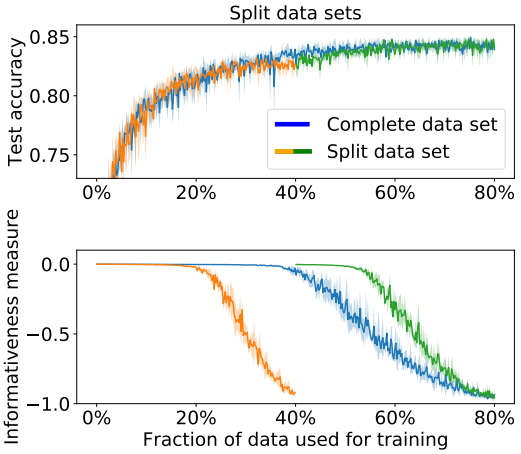
Figure 3.9 illustrates how the different metrics change while training a $CNN1$ network with either the random or the margin query strategy, on the CIFAR-10 data set. Figure 3.9a shows how these metrics evolve when letting the network train 50 epochs on the data. Figure 3.9b shows the corresponding results but when early stopping was used. Early stopping was based on the accuracy on a validation set containing 2000 samples. The networks trained with the margin query strategy give a similar test accuracy as the network trained with the random query strategy. The training accuracy is higher for the network trained with the random query strategy. For the test loss, there are no notable differences between the query strategies. The training loss is slightly larger for the network trained with the margin query strategy. Lastly, the figure also shows that the mean margin informativeness measure, for the network trained with the margin query strategy, is close to 0 until 60% of the data has been labeled and used for training. The margin informativeness measure evaluated on the points selected by the random query strategy decreases quickly early in the training, after which it remains approximately constant throughout the training until the whole unlabeled data set has been used for training.

3.3.2.1 Split Data

The difference between introducing new data sets gradually and considering all data from the starts is shown in Figures 3.10 and 3.11. The figures show a large decrease in the mean informativeness measure at the introduction of a new subset. The phase where the mean informativeness measure is close to 0 was the longest when the first subset was introduced. In particular, it becomes shorter for each introduced subset as can be seen in Figure 3.11. The line corresponding to "Complete data set" is given for a comparison with having access to all unlabeled data from the beginning.

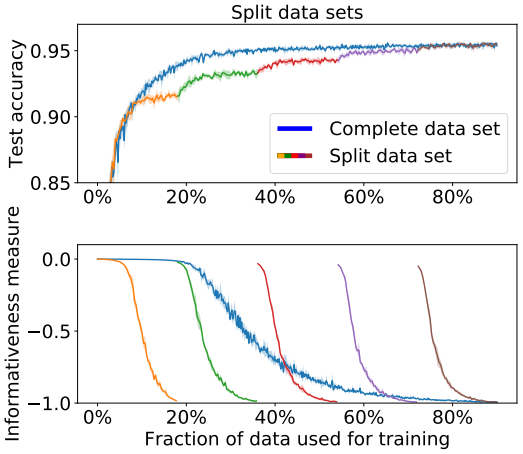


(a) MNIST data set

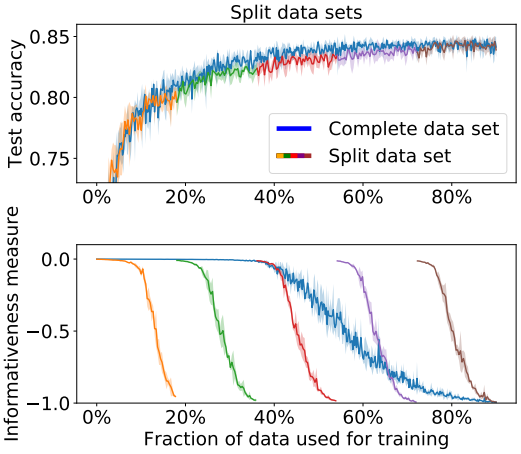


(b) Fashion-MNIST data set

Figure 3.10: Shows the effect on the informativeness measure and test accuracy when new unlabeled subsets are added, for a total of two, to the unlabeled training set, averaged over 5 reruns.



(a) MNIST data set



(b) Fashion-MNIST data set

Figure 3.11: Shows the effect on the informativeness measure and test accuracy when new unlabeled subsets are added, for a total of five, to the unlabeled training set, averaged over 5 reruns.

3.4 Discussion

3.4.1 Query Strategies and Training Methods

From the results in the Figures 3.1, 3.2 and 3.3 it can be seen that in general smaller capacity networks have smaller test accuracy compared to high capacity networks, independently of the query strategy used. This is due to the fact that larger networks have substantially more parameters that can be used to describe the classification problem. Larger networks with more parameters also seem to learn at a significantly faster rate compared to smaller networks. This remark is coherent with previous research [23].

Comparing the different query strategies, the margin query strategy consistently outperforms the other strategies for all examined network sizes and training methods. Prior studies have shown that the margin query strategy usually gives a high test accuracy, compared to other query strategies [24]. Another interesting remark is that the other query strategies all perform similarly. A common attribute among them is that they usually are outperformed by the random strategy initially. This is probably because a query strategy on a network initialised with random weights can not determine what is informative. The network might be biased towards certain types of images due to the initialisation. For comparison, the random query strategy, would not be subject to this bias. A potential solution to lessen the effect of the bias is investigated in the *Random Start* substudy, see Section 4.1.3. Moreover, the results in Figures 3.1, 3.2 and 3.3 indicate that all query strategies, using cumulative training, tend to perform better compared to the random strategy in the later stages of the training for larger networks.

The Figures 3.1, 3.2 and 3.3 also allows for comparison between the training methods. In summary, the cumulative training performs better than incremental when a lot of data is selected. Incremental training is reasonable if only a fraction of the total unlabeled data set is labeled. For the MNIST data set, the incremental and the cumulative method performs similarly up to 1/6'th of all images have been labeled and used for training. We can see that cumulative training appears to have some advantages since the model selection accuracy never decreases, in comparison to incremental training. In Figure 3.1a shows slightly higher test accuracy for the margin query strategy with incremental, compared to the cumulative method. This can be seen when around 10% to 40% of the whole unlabeled data set has been used for training. Harnessing this performance benefit seems very difficult since it occurs for a short time during the labeling process. A heuristic, creating an alternative order of training, was investigated in *Course Plan Training* substudy, see Section 4.1.5.

Networks trained with the incremental training methods seem to perform subpar, for all query strategies, when a large portion of the unlabeled data has been labeled. This phenomena can be seen in Figures 3.1, 3.2 and 3.3 for all network sizes. Incremental training initially leads to a high accuracy which later decreases as more and

more data is used for training. This is probably a consequence of the training order created by the incremental approach. For comparison cumulative training does not show the same drawbacks for the margin query strategy. This can be seen for all results including the cumulative method, in Figures 3.1, 3.2 and 3.3. The accuracy never decreases when selecting new data with any query strategy while using the cumulative method. The training order is always randomised since the method adds all the training data in a pool from which the stochastic gradient descent is allowed to choose freely. This removes the problems induced by ordering the training set.

A figure of the training loss and the training accuracy, corresponding to the results in Figure 3.1 can be seen in Figure 3.4. The figure shows that the mean margin informativeness measures start to decrease earlier for the incremental method compared to the cumulative. In the discussion from the *Informativeness Measures* substudy, see Section 3.4.2, we concluded that the main contributor to the decrease is mostly due to the data set being depleted of informative data. Thus, a reasonable explanation of why the mean margin informativeness measure decrease faster for the incremental strategy is that the networks find the remaining data set non-informative earlier. This, in turn, could happen because the networks only trains on one small informative batch at a time. Figure 3.1a shows that the test accuracy starts to decrease for the incremental training when having labeled a large fraction of the data set. When the mean margin informativeness measures decreases, we subsequently see that the test accuracy reaches a plateau, after which the accuracy starts to decrease. Moreover, note that while the incrementally trained network reached a test accuracy plateau, the accuracy of the cumulatively trained network is still increasing. During this phase of the training, the cumulatively trained network increases in accuracy more quickly than the corresponding network with incremental training. Also note that in this phase, for the MNIST data set, where 20%-40% of the data has been used for training, the mean informativeness measure is lower for the cumulatively trained network. This is coherent with the conclusions of *Informativeness Measures* substudy, see Section 3.5.2, which says that a high mean margin informativeness measure can be linked to a faster learning rate.

The influence the batch size has on the test accuracy on an $ANN1_{100}$ network can be seen in Figure 3.5. The cumulative training method seems to be resilient to the varying batch size for both the MNIST and Fashion-MNIST data sets. The incremental training method seems to be more affected by a large batch size compared to the cumulative training method, where an increased batch size is correlated with a worse performance. Why incremental training is more sensitive to a large batch can be intuitively understood. If the incremental training method selects a large batch size, then a lot of similar data will be used for training sequentially. This is presumably not beneficial for the network. The cumulative training does not suffer from this negative consequence. Even though the cumulative training method selects a large amount of similar data, the effect of the choice of the selected batch will be diluted by the data previously sampled.

Lastly, the influence of the random subset size can be seen in Figure 3.6. Recall that

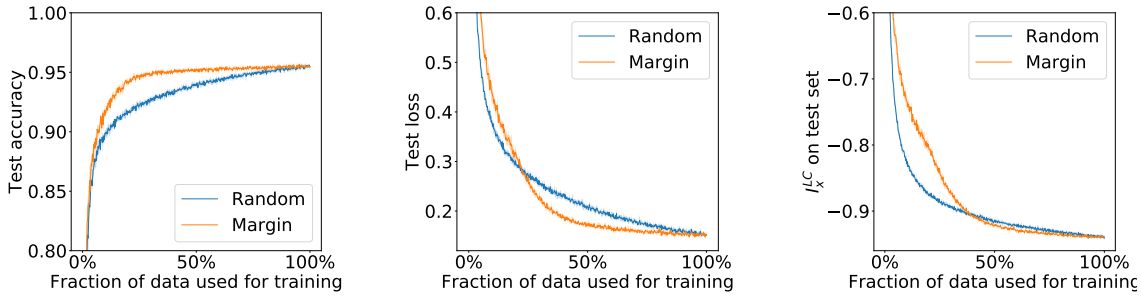
the random subset size determines how much of the unlabeled data set the query strategy is allowed to query. The figure shows that a small random subset size may impact the performance of the network negatively. This was most clearly seen for the MNIST data set, see Figure **??**. The random subset size did not influence the performance as much for networks trained on the Fashion-MNIST data set, see Figure **??**. It is natural to associate a worse performance with a smaller random subset size, at least when using a query strategy that performs better compared to random sampling. In the extreme case, where the random subset size is equal to the batch size, the selected data won't be affected by the choice of query strategy, and will thus be equivalent to random sampling. Figure 3.5 shows that it is possible to only allow the query strategy to consider a random subset of all unlabeled data if needed, but for optimal performance should the random subset be as large as possible.

3.4.2 Informativeness Measures

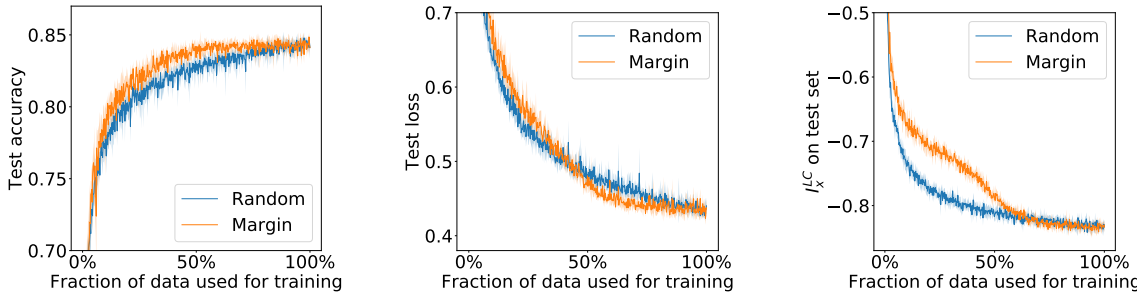
In Figures 3.7 and 3.8 we see that the test accuracy is higher than the training accuracy when using the margin query strategy. It is usually expected that the training accuracy is higher than the test accuracy when training a classifier [25, p. 31]. This is common for models that are optimised on a training data set and evaluated on a test set. The fact that the test accuracy surpasses the train accuracy could indicate that the margin query strategy selects difficult data for labeling and thus training. The distribution of the test data and the selected training data may thus be different when the margin query strategy is used. This is further investigate in Section 4.1.2. Even though the margin strategy selects difficult data, the labeled data is still useful to help the model understand the underlying distribution of the test data. This follows from the fact that the test accuracy being higher than the training accuracy for the margin query strategy. However, the test accuracy is lower than the training accuracy for the random query strategy. The distribution of the labeled data set is investigated more in the substudy *Sample Distributions*, in Section 4.1.2.

Another unexpected finding is the fact that the test loss is higher for the margin query strategy than for random sampling *while* the test accuracy is higher. Recall that the loss is a measure of how well the output fits the true data labels in the test data, see Equation (2.2). This means that the output probabilities of the network with the margin query strategy have a worse fit to the true labels. At the same time, the network also has a better test accuracy compared to the network trained with the random query strategy. This means that the prediction of the margin network often agrees with the correct label, while the output of the network shows greater uncertainty in its prediction. The phenomenon, where the prediction is as accurate but with a worse fit to the true labels is interpreted as the network having "low self-esteem".

To test this interpretation, the least confident informativeness measure from Equation (2.4) was evaluated on the whole test set during training. The relationship



(a) MNIST



(b) Fashion-MNIST

Figure 3.12: Shows the test accuracy and loss for an $ANN1_{100}$ network trained cumulatively. The rightmost figure shows the mean least confident informativeness measure evaluated on the points in the test set at different stages in the training averaged over 5 reruns.

between the test accuracy, test loss, and least confident mean informativeness measure can be seen in Figure 3.12. Recall that the least confident informativeness measure for a given data point is $-\max_c P_{ANN1_{100}}(y = c|\vec{x})$, which means a low informativeness measure is correlated with a high confidence in the prediction of the $ANN1_{100}$ network. The figure shows that the mean least confident informativeness measure, evaluated on the test set, is in general higher when using the margin query strategy compared to the random query strategy. This further indicates that usage of the margin query strategy makes the network more uncertain in its predictions, while also making the network better at predicting the data.

In Figures 3.7, 3.8 and 3.9 we see how the mean margin informativeness measures $\bar{I}_B^M$ and the margin on random informativeness are changing during training. The mean margin informativeness measure on the randomly selected points are decreasing immediately as the training start. Note that all batches come from the same distribution since they are all randomly sampled. The decrease in the mean margin on random informativeness measure can therefore only be caused by the network learning about the data. Thus, an increase in certainty of the network can be correlated to a decrease in the mean margin informativeness measure. Also note that we see the most change in test accuracy while the margin on random informativeness measures are at its highest. This is coherent with the noted correlation since this where see the most change in the informativeness measure during the training. On the other hand, when training the network with the margin query strategy, we see

that the margin informativeness measures are close to zero (0) for several batches through the training. Since we are actively selecting the most informative data with the margin query strategy, the decline in margin informativeness measure may be caused by a combination of the network becoming more certain in its prediction and that the unlabeled data set is running out of informative data. However, when using the margin query strategy, the training accuracy can be correlated with a slow increase in test accuracy and a declining mean margin informativeness measure. The network learned the most while the mean informativeness measures where close to zero (0). Combining the two observations makes it possible to link a high margin informativeness measure with a fast learning rate.

If the mean margin informativeness measure can be linked to a fast learning rate, this means that the network trained with margin query strategy learns faster compared to its random query strategy counterpart. This statement is supported by comparing the mean informativeness measure in the bottom of Figures 3.7 and 3.8. The mean informativeness measure from the margin query strategy is higher, when a small fraction of data has been used, compared to the mean margin on random informativeness measure. Indeed, the test accuracy for the margin network is higher compared to its random counterpart, which could be a result of an initially faster learning rate. This is easier to observe for the $ANN1_{100}$ network than the $ANN1_{10}$ network. There is also an intersection in the graphs showing the mean informativeness measures. After the intersection, the rate of learning, i.e. how fast the test accuracy increases for every batch, for the random query network is faster compared to the margin query network. This can be interpreted as the random query strategy network is starting to catch up. After this point, continued training would make the network perform equally well as its margin counterpart. This is observed in the figures: the test accuracy of the network trained with the random query strategy gets closer to the test accuracy of the margin counterpart when approaching 100% fraction of data used in training.

For CIFAR-10, see Figure 3.9, the training accuracy is larger than the test accuracy, unlike what was seen on the MNIST and Fashion-MNIST data sets where 1 epoch was used for training. This is due to the network training with 50 epochs allowing the training set to be almost perfectly modeled by the network. This makes the training accuracy close to 1, as can be seen in the figure when less than 10% of the data has been labeled. The distribution of the selected training set and the test set may still be different in this case. The train accuracy can however not be used to determine this, due to the fact that the network trains for several epochs on a small set. Comparing the test accuracy between the $CNN1$ network with 50 epochs trained with the margin query strategy to the network trained with random query strategy, we potentially see slightly higher test accuracy, after around 60% of the data has been used for training. The difference is small, and might not be significant.

There is a risk that the amount of epochs trained on the selected training data affects the $CNN1$ network and the query strategy's ability to find an informative

batch. Training several epochs on few data samples could potentially make the network confident in its prediction of the unlabeled data. This is the motivation as to why early stopping was used in an additional test. The results can be seen in Figure 3.9b. This may have caused the opposite of the expected effect: the network trained with the margin query strategy was more certain in its predictions, which can be seen by comparing the margin informativeness measures in Figures 3.9a to 3.9b. This might be due to the batch size being larger for the run with the early stopping. However, the training accuracy also got lower and the training loss got higher. This indicates that the selected batches were more difficult for the network to use for training when early stopping was used.

To summarise the results seen on the CIFAR-10 data set, we see the same evolution in the training metrics as seen in the MNIST and Fashion-MNIST data sets. However, the margin query strategy only seems to increase the test accuracy slightly for the CIFAR-10 data set, compared to the random query strategy. Moreover, the comparison between the test and train accuracy does not hold in this case, probably due to the fact that the network trains several epochs on very small tests set when only a small fraction of the data has been labeled.

3.4.2.1 Split Data

Looking at Figures 3.10 and 3.11, it can be seen that a lot of data is given a high informativeness measure, for several batches in the first subset. When the subsequent subsets are added to the unlabeled data set, it is observed that the phase, when the mean informativeness measure is close to 0, is shorter. This means that the network, looking at an identically distributed data set as in the previous subset, identifies less informative data. What is more remarkable is that the mean informativeness measure increases drastically when a new subset is introduced. Even at introduction of the last subset in Figure 3.11, the mean informativeness measure increases from -1 to 0 instantly. This tells us that the decrease of the mean informativeness measure observed in 3.3.2 is mostly caused by the data sets being depleted of interesting data.

Note that when new subsets are introduced, the test accuracy increases significantly for the MNIST data set, as seen in Figures 3.10 and 3.11. We observe this increase while the mean informativeness measure is high, meaning that a lot of informative data is trained on. This reinforces the idea that the rate of increase in test accuracy is correlated to the margin mean informativeness measure. The more informative the data is, the larger is the increase in accuracy. We also see in Figure 3.11 that it would be preferable to have all data from the start compared to gathering data during the active learning process. The test accuracy when all data is considered is always higher than the test accuracy obtained when considering a subset of all data.

From the experiment illustrated in Figures 3.10 and 3.11, we conclude that the mean margin informativeness measure is high while there are informative data in the unlabeled training set. This is coherent with what we observe in Figures 3.7 and 3.8, which shows that the training accuracy increases at the same time as

the mean informativeness measure starts decreasing. In the figures, a low training accuracy seems to be correlated with a high mean informativeness measure. As the training set is being depleted of difficult points, the training accuracy naturally starts to increase. It is thus natural to correlate a high mean margin informativeness measure with fast learning of the network. Decreasing mean margin informativeness measures could thus be interpreted as either:

(a) An indication that the user is not gaining much accuracy by querying the oracle for labels in the training set and should stop training the network.

(b) An indication that more unlabeled data should be acquired in order to improve accuracy.

3.5 Conclusion

3.5.1 Query Strategies and Training Methods

- The test accuracy converges faster for networks with more neurons in the hidden layer.
- Performing active learning with any query strategy risks a worse performance compared to random random sampling.
- Decreasing the random subset size impacts the performance of the network, but for a large random subset, with more than $\frac{1}{3}$ of the entire data set considered, no decrease in test accuracy was observed for the MNIST or Fashion-MNIST data sets. A fraction of $\frac{1}{12}$ showed a small decrease in accuracy for the MNIST data set.
- The margin query strategy seems to be the preferred query strategy to use for the tested neural network architectures.

Incremental Training

- The test accuracy of an incrementally trained network starts to decrease when the mean margin informativeness measure decreases.
- The incrementally trained networks perform worse later in the training, probably due to the order the data was trained on.
- The mean margin informativeness measure decreases earlier in the training when the incremental method is used, compared with the cumulative method.
- The batch size has a large impact when training with the incremental method.

Cumulative Training

- The cumulative training method suffers no negative consequences from labeling the data in a certain order.
- The query strategies entropy, least confident and least squares behave similarly and are observed to outperform the random query strategy when a fairly large fraction of the data has been labeled, but do never outperform the margin query strategy.
- The batch size has little impact when training with the cumulative method.

3.5.2 Informativeness Measures

All conclusions are made for a network with a low number of epochs.

- A high mean margin informativeness measure can be associated with a fast increase in the test accuracy. A low mean margin informativeness measure can be associated with a slow increase in the test accuracy.
- A significant decrease in the mean margin informativeness measures is indication that one should either stop labeling data, since at this point one can only expect small improvements in test accuracy. This can be done without the presence of a test set.
- If sampling is cheap and labeling is expensive, then, as soon as the mean margin informativeness measure decreases, new unlabeled data should be sampled. It is better to have access to as a large as possible unlabeled data set from the beginning.
- There is a connection between a high mean margin informativeness measure and a fast increase in test accuracy. The training accuracy at this stage seems to be increasing slowly.
- After performing active learning with the margin query strategy, the labeled data set does not appear to have have the same distribution as the test set.
- The mean informativeness measure from the margin query strategy decreases due to the unlabeled training set being depleted of informative data.
- A network which is well trained considers less data to be informative, compared to an identical network which has trained using less data.
- After performing the margin query strategy the network is less certain in the label of data, even though it is more accurate. A network with this characteristic is said to have “low self-esteem”.

- The incremental test accuracy starts to decrease when there is no informative data left. This is not necessarily true for the cumulative training method.

3.6 Future Work

It could for some applications be important to estimate the uncertainty of the model. Having performed active learning could make this difficult, as was seen in Section 3.3.2, where the network could perform better on the test set than on the train set, linked to the mentioned *low self-esteem*. Maybe this problem could be mitigated by including some random samples, with the trade-off of having a worse test accuracy while gaining higher certainty in the network.

Mixing two query strategies, for example, sampling half of a batch with random sampling and the other half with a query strategy could be beneficial. A similar idea to this is investigated in the *Random Start* substudy, which can be read about in Section 4.1.3. Investigating how to combine two or more query strategies would be interesting. Additionally, more than two query strategies could be used at the same time, for example by employing an ensemble of query strategies.

4

Data-centric investigation

This chapter consists of two types of studies. The first type of study is a deeper investigation of the different subsets that are generated by active learning. This type of study investigates what characterises subsets that are associated with a high test accuracy and a low test accuracy. The second type of study treats different extensions, building on the results found in chapter 3. This chapter is divided into six substudies which will be presented in parallel:

- Evaluation of the sampled data sets.
- Random start.
- Estimates of the test accuracy.
- Sample distribution.
- Course plan training.
- Semi-supervised learning.

Many of the same conclusions can be drawn for an $ANN1_m$ network, for a varying number m of neurons in the hidden layer. This chapter can either be read as presented or a particular substudy can be read on its own. It is assumed that the reader is aware of the results in the previous chapter, Chapter 3. Examples of conclusions carried into this chapter include:

- The cumulative training method should always be used.
- The batch size for querying data does not have a large impact, when using cumulative training.
- The size of the subset which is considered can be smaller than the entire data set with non-existing or small penalties.
- The margin query strategy usually performs better than random sampling.

An $ANN1_{100}$ network was used as the standard network for the MNIST and Fashion-MNIST data sets. For the CIFAR-10 data set the $CNN1$ from Section 2.3 was used.

4.1 Background

4.1.1 Evaluation of Sampled Data Sets

Active learning can be seen as a data selection problem. The benefit of this is that the data can be sampled with one type of network and used to train another prediction model. This substudy will look into how to determine if a selected data set is good, given a model, and how to obtain such a data set. We suggest an empirical approach that involves, at every iteration, training a new network using all labeled data to see what test accuracy is obtained. We call this network the evaluation network. The data to be labeled is selected by another network, called the selection network which is assisted by a query strategy.

With these two networks, a more fair comparison can be made for the incremental and cumulative selection method than was done in the *Query Strategies and Training Methods* substudy in Section 3.1.1. It can, for example, mitigate the potential benefits and disadvantages of training in a certain order, since the evaluation network uses the cumulative training method.

Another advantage of this approach is that the evaluation and selection model can be different. This makes it possible to answer questions like

- Can a network with few hidden neurons choose data such that a network with more hidden neurons learns efficiently?
- Can a network with many hidden neurons choose data such that a network with fewer hidden neurons learns efficiently?
- How does the number of epochs used for the selection model affect the evaluation model?

4.1.2 Sample Distributions

It has already been observed in the substudy on *Query Strategies and Training Methods*, see Section 3.4.1, that active learning often outperforms random sampling, at least with the margin query strategy. There are two hypotheses to why it outperforms random sampling. First is that the class distribution is good, for example meaning that active learning samples more data from informative classes. This is called a distribution between classes. The other hypothesis is that the distribution within classes is good, meaning that the data within one class is informative. This could, for example, be sampling data from close to the decision boundaries. This substudy aims to investigate the characteristics of the subsets selected by the margin query strategy and random sampling.

4.1.3 Random Start

In the *Query Strategies and Training Methods* substudy, see Section 3.4.1, it was hypothesised that the first batch selected by a query strategy, using a randomly initialised network, may be biased towards a certain class. This substudy will investigate if this bias exists and how initial random sampling, during the first couple of batches, affects the performance of the trained network. Moreover, if this bias exists, can the underlying cause behind it be found?

Another advantage with sampling randomly in the beginning is that the random sample could act as a test set since it has the same expected distribution as a test set. This is further investigated in the *Estimates of the Test Accuracy* substudy, see Section 4.1.4.

4.1.4 Estimates of the Test Accuracy

A difficulty associated with active learning is that it could be complicated to obtain a test accuracy if labeling data is expensive. This substudy will, therefore, investigate how to estimate the test accuracy without having a dedicated test set. The first batches, even though it is small, could be selected with random sampling and could, after the data has been labeled, act as a test set.

4.1.5 Course Plan Training

In the *Query Strategies and Training Methods* substudy, see Section 3.3.1, it was observed that the accuracy of the selection model is dependant on which training method was used. In the incremental case, the accuracy decreased at the end, while for the cumulative case the accuracy did not decrease. This was seen in Figure 3.1a, where there was a considerable difference in test accuracy when 100% of the data had been used for training. The only factor that could have caused this difference in performance is that the incremental training took the order of selecting data into consideration, while the cumulative did not. This difference in accuracy must thus solely depend on the order of data when training. It was discussed in Section 3.4.1 that the order which had been created, by incremental training, consisted of informative data in the beginning, leaving non-informative data in the end. If an order can be constructed that gives worse results than random, then maybe an order can be constructed which performs better than random.

4.1.6 Semi-supervised Learning

In the active learning setting, it is assumed that unlabeled data is abundant. So far, this thesis has only covered the scenario where the labeled data set is used for training the neural network. This substudy investigates if the labeled and unlabeled data sets together can increase the accuracy compared to only using the labeled data set. The method of using the network to label unlabeled data is called *pseudo-labeling* and is an efficient semi-supervised learning method for deep neural networks

[26]. In short, the network first trains using all labeled data then the trained network labels the data it is most confident about.

Here it is interesting to compare the benefit of assigning pseudo-labels after different query strategies. As mentioned, the data, sampled with the margin query strategy, helps the model discriminate among specific classes [9, p. 14], since it prioritises the data the model is conflicted about. Random sampling, on the other hand, samples uniformly from all data. It would be interesting to investigate how this distribution within classes affects the performance of semi-supervised learning.

4.2 Method

4.2.1 Evaluation of Sampled Data Sets

It is desirable to be able to sample two different subsets and compare them using the same predictive model. To do this two networks were used, namely a selection network and an evaluation network. The selection network uses active learning equivalently to all simulations in Chapter 3 and was solely responsible for selecting the data and its weights were discarded after labeling was done. This means that the weights of the selection network does not have any value, only the data it has sampled is of interest. The selection network can either be trained incrementally or cumulatively, using the procedure explained in Section 2.3.5 without any of the optional steps. The evaluation network trained on the data set selected by the selection model and evaluated the test accuracy. The evaluation network is meant to be viewed as the network which is to be used after all labeled data has been obtained. Three experiments were conducted. Each experiment used slightly different circumstances, such as a different number of hidden neurons or epochs.

The first experiment compared the test accuracy obtained from data sets selected by the incremental and cumulative training methods. $ANN1_{100}$ networks were used to test this. The selection network used 1 epoch for selection network while the evaluation network used either 1 or 5 epochs. What was tested is if incremental and cumulative training can sample data that give a similar evaluation accuracy, given that the selection network uses 1 epoch.

The second experiment investigated if a selection network could beneficially select points for an evaluation network when they differ in how many epochs they used for training. $ANN1_{100}$ networks were used as both the selection network and the evaluation network. Two data sets were sampled by letting the selection model either train with 1 or 5 epochs. The data sets were evaluated using an evaluation network was also trained with either 1 or 5 epochs. This amounted to four different simulations and two tests. Let x correspond to the number of epochs in the selection network and let y correspond to the number of epochs in the evaluation network. A simulation will be written as $[x, y]$. The first test investigates if it is appropriate to use a selection network with 1 epoch to select data for an evaluation network with 5 epochs. This means $[1, 5]$ will be compared to $[5, 5]$. The second test investigates

if it is appropriate to use a selection network with 5 epochs to select data for an evaluation network with 1 epoch. This means $[5, 1]$ will be compared to $[1, 1]$. This was done for both the incremental and cumulative training methods.

The third experiment conducted had a varying number of neurons, m, in the selection network instead of the number of epochs. A medium capacity network, $ANN1_{100}$, was consistently used as the evaluation network. The selection network was either a small capacity $ANN1_{10}$ network, a medium capacity $ANN1_{100}$ network, or a high capacity $ANN1_{1000}$ network. This allows us to test is if $m_{evaluation} > m_{selection}$, $m_{evaluation} < m_{selection}$ or $m_{evaluation} = m_{selection}$ is preferable.

4.2.2 Sample Distributions

To isolate why active learning achieves a higher test accuracy than random sampling, three types of subsets were sampled and compared. For each subset the distribution of labels and the obtained accuracy will be investigated. The three types of subsets were sampled in the following manner:

- Random sampling, which used the random query strategy.
- Margin sampled subset, which was sampled by the margin query strategy as done in previous substudies.
- Resampled subset, which was constructed such that the distribution of classes is the same as for the margin sampled subset. Random sampling was used to obtain the correct number of occurrences of each class.

The distribution between classes is defined as the number of occurrences belonging to a particular class in the subset. The margin sampled and resampled subsets are designed to have the same distribution between classes, but will not necessarily have the same distribution within one class.

A tSNE dimension reduction was performed for all data sets. This was to provide a basis for visually determining if the label distribution is appropriate. The theory behind it and an example of how to perform a tSNE dimension reduction can be read in [27].

For the Fashion-MNIST and MNIST data sets an $ANN1_{100}$ network has been used. For the CIFAR-10 data set a more advanced convolutional neural network, $CNN1$, was used, described in Section 2.3. For the Fashion-MNIST and MNIST data sets the experiment consisted of sampling 30000 images and for the CIFAR-10 data set 5000 images were sampled. To gather a randomly sampled subset and a margin sampled subsets, the algorithm from Section 2.3.5 was used with none of the optional steps.

4.2.3 Random Start

The goal for this substudy is to examine if there is any bias in the first batch selected by a query strategy. To do this, an $ANN1_{100}$ network was initialised with random weights, described in Section 2.3. The network, with randomly assigned weights, used a query strategy to obtain the most informative batch of size 1000, using 1 epoch for training. After training on the single batch, the performance of the network was assessed with the obtained test accuracy. This was repeated 200 times and was done for the random, margin, entropy, least confident, and least squares query strategies.

For the same simulation as described above the distribution of the classes of the first samples was also stored. This was done to see if there was any bias towards any specific class in the different query strategies. To do this, the number occurrences of each class were counted in the first sample selected by each query strategy, and then the mean and variance was calculated using all reruns. Since every sample will be of size 1000 this means that if one class is more abundant, that also means that the other classes are less abundant. A large variance for each class means that the sampling sometimes chooses many of that class, which implies that it sometimes chooses few from the other classes.

In the results from the *Query Strategies and Training Methods* substudy, see Section 3.3.1, it was observed that many query strategies started with low test accuracy. This could be due to a bias. A method to mitigate the problem, by sampling randomly for the first batches, was investigated. The experimental design, of selecting data with the random query strategy in the beginning, and switching to another query strategy, later on, will be referred to as *random start*. The *random start size* refers to the number of data points that are randomly sampled.

To investigate the long-term effect of the random start another experiment was set up where two query strategies were used, namely entropy and margin. Both of these either used a random start or no random start with an $ANN1_{100}$ network. The algorithm from Section 2.3.5 was used with the optional step of sampling randomly in the beginning. The aim is to see if there is an increase or decrease in the test accuracy even after random sampling ended. Two different random start sizes were chosen, 200 and 2000, to see what is most beneficial for the test accuracy.

Lastly, to find the underlying cause behind a potential bias, when using a randomly initialised network, an experiment was done to see if there is any bias towards dark images. The mean informativeness measure $\bar{I}_B^A$ was calculated, using different query strategies and a randomly initialised $ANN1_{100}$ network, for images with a resolution of 28×28 pixels. Each image had a certain brightness level associated with it. For a given image, $1 - \alpha$ percentage of pixels were bright, and α percentage of pixels were completely dark. The bright pixels had a brightness uniformly distributed between 0 and 1.

Table 4.1: The experimental setups used in the *Estimates of the Test Accuracy* substudy, see Section 4.2.4.

Data set	Random start size (percentage of all unlabeled data)	epochs
MNIST	3000 (5%)	1
MNIST	3000 (5%)	5
MNIST	600 (1%)	1
MNIST	600 (1%)	5
Fashion MNIST	3000 (5%)	1
Fashion MNIST	3000 (5%)	5
Fashion MNIST	600 (1%)	1
Fashion MNIST	600 (1%)	5

4.2.4 Estimates of the Test Accuracy

To estimate the test accuracy, an upper and lower bound was constructed using data from the *Random Start* substudy, see Section 4.1.3. This was done by using two evaluation networks, where each of them had trained on slightly different training sets. To create the training sets, a selection network was used. The selection network used a random start, as in Section 4.1.3, after which it used the margin query strategy. The algorithm from Section 2.3.5 was used with the optional step of sampling randomly in the beginning. Two different training sets, $\overline{L}$ and $\underline{L}$ were then created:

1. $\overline{L}$: containing all labeled data, including the random start.

2. $\underline{L}$: containing all labeled data, excluding the random start. Thus, this set contained only data sampled with the margin query strategy.

The accuracy associated with the two training set sets were calculated with two identical evaluation networks, $\overline{C}$ and $\underline{C}$, which used the random start set as a substitute for the lacking test set. Thus, $\overline{C}$'s accuracy was evaluated on data it had used for training, while $\underline{C}$ evaluated its accuracy on data it had not trained on. Therefore, $\overline{C}$ was biased in its evaluation.

It is possible that $\underline{C}$ may be slightly biased towards the test set, since the network had trained with the test set when choosing informative data during the active learning phase. The data sampled with the margin query strategy may therefore be better, or worse, at describing the test set than should normally happen with a dedicated test set.

The procedure, of estimating the test accuracy, was done for a varying number of epochs, data sets, and size of the random start to see what effects these had on the estimates. The parameters of the experiments are shown in Table 4.1. $ANN1_{100}$ networks were used in all experiments, both for selection and evaluation. The selection of data was stopped after 50% of all unlabeled data had been labeled, including the random start subset. All experiments were repeated 50 times.

4.2.5 Course Plan Training

The goal is to generate a good order in which to train. The hypothesis is that the network should start training with non-informative data, and continue with increasingly more informative data. This is the opposite order of training compared to the incremental case which resulted in a bad performance, Section 3.3.1. The algorithm in Section 2.3.5 was used without performing any optional steps, using only 1 epoch for selecting and training on data with the teacher network. To generate a training order, an $ANN1_{1000}$ network was trained incrementally, using either the margin or random query strategy to select points from the unlabeled data set to be labeled. This network is called the *teacher network*. The teacher network then evaluated the margin informativeness measure $I_{\vec{x}}^M$ on all labeled data points $\vec{x} \in L$. The data in L was then sorted in an increasing order with respect to $I_{\vec{x}}^M$. The sorted data was then split up into batches consisting of 32 samples each, which was used to train another $ANN1_{1000}$ network, called the *student network*. The test accuracy obtained by the student network will be compared to the test accuracy obtained by the teacher network. The created order of data and batches is referred to as the *course plan* for the student network.

There is a shortcoming with this method, namely that the teacher network may suggest that some classes are not as informative, such as classes of 0, 6, and 1 for the MNIST case or classes 1 and 8 for Fashion-MNIST. These classes were observed to be less informative in the results in the *Sampled Distributions* substudy, see Figures 4.5a and 4.6a. The teacher network may create an order where these classes are over-represented in the first batches. Efforts were made such that the class distribution would be homogeneous throughout the batches. To solve the aforementioned problem a different *course plan order* was constructed, as explained by the following steps:

1. A teacher model was trained with the cumulative training method, selecting either 320 or 9600 points from the unlabeled data set using the random query strategy.

2. The teacher model was asked to evaluate the margin informativeness measure $I_{\vec{x}}^M$ of all data points $\vec{x} \in L$. The data in L was sorted in ascending order with respect to $I_{\vec{x}}^M$.

3. The labeled data L was split up in its classes, while retaining the sorted order.

4. Batches, of size 32, were generated to have a similar class distribution as all labelled data in L. This ensured that each batch had a class distribution similar to the class frequencies seen in Figures 4.5a and 4.6a. The samples $\vec{x}$ with the smallest $I_{\vec{x}}^M$ were prioritised when added to a batch. This ensured that that the first batch has the smallest average informativeness measure, and the last batch has the highest. This is similar to stratified sampling [19] in that each batch is homogeneous regarding the class labels. It is, however, not a strict stratified sampling since each batch is inhomogeneous with respect

to the informativeness measures.

The most informative data is not necessarily representative of the test set. Therefore, another course plan order was constructed, which we named the *folded course plan order*. It was constructed by taking the course plan order and mixing the k'th most informative batch with the k'th least informative batch to construct the k'th and $k+1$'th batch. This sorting was done for all k from 1 to $k = n/2$, n being the total number of batches. The first batches was thus a mix of very informative data and not so informative data, according to the margin query strategy. The last batch used in training was the batch with the median informativeness measure for each class.

The batch size was set to 32 and the networks were trained with 1 epoch for both the student and teacher networks. Two different number of batches were considered to see how the amount of data changes the comparison between the student and teacher networks. 300 batches (9600 images) and 10 batches (320 images) were both tested. Both the random and margin query strategies were used when training the networks on 300 batches, while only the random query strategy was used when training the networks on 10 batches. The margin query strategy was omitted in this case due to the data being selected with a randomly initialised network and the distribution of the data was not motivated. More can be read about this in the *Random Start* substudy, see Section 4.1.3. Of most interest is the test accuracy of the student network, compared to the teacher network, after the student has trained on its last batch. The slope getting to the last accuracy is not relevant, since it is assumed that all data will be used for training.

4.2.6 Semi-supervised Learning

The proposed semi-supervised method assigns pseudo-labels to all unlabeled data that the model is certain about. It is similar to the pseudo-label algorithm described in [26]. The methodology for this substudy consists of three different parts. The first of which, in Section 4.2.6.1, is a description of the algorithm used to find the data that the network is most confident about, and how they are labeled by the network. This algorithm uses a parameter ξ that determines how confident the network needs to be to be allowed to assign pseudo-labels. The second part, in Section 4.2.6.2, is a description on how a suitable ξ can be found. The last part, in Section 4.2.6.3 is a description of how to compare the benefits of performing semi-supervised learning after using the query strategies margin and random.

4.2.6.1 Algorithm

Semi-supervised learning is performed after a network C has been trained using some labeled set L. For every point $\vec{x} \in U$, the probability estimate matrix $P(y = c|\vec{x}_i)$ is evaluated for every point $\vec{x} \in U$ and for every class c. As mentioned, in pseudo-labeling the network is supposed to label the data that it is most confident about. How confident the network needs to be to label the data is determined by the threshold parameter ξ.

The least confident informativeness measure, $I_{\vec{x}}^{LC}$, see Equation (2.4) was used to evaluate how confident the network is in its labeling of $\vec{x}$. This informativeness measure was calculated for all unlabeled data in U. The subset S_ξ, containing all data that the network is certain enough about, is created by selecting the data for which the informativeness measure is below the threshold:

$$S_\xi = \{\vec{x} \in U \mid I_{\vec{x}}^{LC} < \xi\}.$$

Every point $\vec{x} \in S_\xi$ is given a pseudo-label matching the predicted class $\arg\max_c \hat{y}_{ic}$ according to the network. The points $\vec{x} \in S_\xi$ are then added to the labeled set L and removed from the unlabeled data set U. The network is retrained with the new data set $L \cup S_\xi$ and the process is repeated until too few points are added to S_ξ.

Note that any of the informativeness measures from the suggested query strategies, in Section 2.2.1, could be used to determine what data the network is allowed to assign a psuedo-label. This is a refinement of previous research in [26] since we have a cutoff, meaning that data the model is uncertain about will not get a pseudo-label.

4.2.6.2 Finding a Suitable Threshold ξ^*

The threshold ξ could have a significant impact on the semi-supervised learning scheme. If the threshold ξ is too hard, the network will not be confident enough to label any of the data in the unlabeled data set U, which is equivalent to not performing semi-supervised learning. On the other hand, a soft threshold ξ allows the network to assign pseudo-labels to all remaining data in the unlabeled data set, which could be suboptimal. If the semi-supervised learning is performed in many steps the model could get more certain about the unlabeled data, and thus label data incorrectly less frequently.

Between the extreme of a soft and hard threshold, it is hypothesised that there exists an optimal threshold ξ^*. Let Ξ be a set of all candidate thresholds. For every candidate threshold $\xi \in \Xi$ the semi-supervised algorithm in Section 4.2.6.1 is applied to an $ANN1_{100}$ network trained with 1 epoch as well as an $ANN1_{1000}$ network trained with 5 epochs. The test accuracy is then reexamined for the networks to find what improvement threshold ξ resulted in. Let a_ξ denote the new test accuracy for the threshold ξ.

Given the thresholds and new test accuracies $\{(\xi, a_\xi) \mid \xi \in \Xi\}$, the trend of $a_\xi(\xi)$ can be approximated using regression. There will probably be a large variance after performing semi-supervised learning which is why a KNN-regression seems suitable to smooth out the noise. A KNN-regression $f(\xi)$ is a rolling average of the K closest thresholds, i.e

$$f(\xi) = \frac{1}{K} \sum_{\xi_i \in \mathcal{N}_\xi} a_{\xi_i},$$

where $\mathcal{N}_\xi$ is the set of the K closest thresholds to threshold ξ. A K of 30 was chosen to smooth out the noise. Given the regression, the heuristic approximation of the

optimal threshold ξ^* is given by the maximum of $f(\xi)$ with respect to ξ:

$$\xi^* = \arg\max_{\xi} f(\xi).$$

4.2.6.3 The Study

An $ANN1_n$ network was trained cumulatively on data selected and gathered by the query strategy A, using the algorithm from Section 2.3.5 without any of the optional steps. When a certain percentage p of the unlabeled data set had been assigned by an oracle through a query strategy, the optimal threshold ξ^* was found for $ANN1_{100}$ and $ANN1_{1000}$. The set of candidate thresholds Ξ consisted of 100 thresholds uniformly distributed between 0 and -1. The KNN-regression was done with $K = 30$, which was used to find the approximate optimal threshold ξ^*. It was investigated how beneficial the semi-supervised training was for different fractions p of the entire data set. This evaluation step was repeated 4 times.

The fractions p examined where equal to 5/600, 10/600, 25/600, 50/600, 100/600, 200/600, and 500/600. For one experiment the query strategy A was either the margin query strategy or the random query strategy. Two different types of networks were considered, namely $ANN1_{100}$ with 1 epoch and $ANN1_{1000}$ with 5 epochs used for training on the labeled and pseudo-labeled data. Both the MNIST and Fashion-MNIST data sets were examined.

4.3 Results

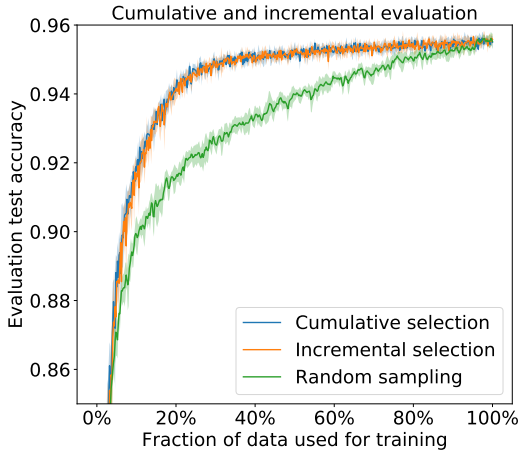
4.3.1 Evaluation of Sampled Data Sets

It can be observed in Figure 4.1 that both the cumulative and incremental selection networks chose data that produced almost identical evaluation accuracies. In Figure 4.2 and 4.3 it can be seen how the change of the number of epochs in the cumulatively trained selection and evaluation networks affects the test accuracy. The legends in the figures should be read to understand how many epochs were used in the selection and evaluation network. For example, $[x, y]$ means that x epochs were used by the selection network, and y epochs were used by the evaluation network.

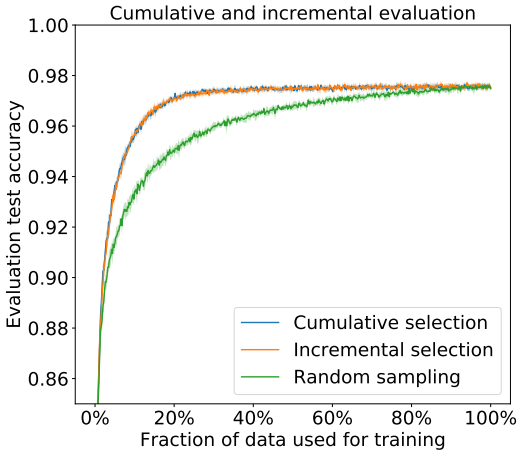
In Figures 4.2 and 4.3, the results show that the evaluation network reaches the highest test accuracy when the selection network is the same as the evaluation network.

In Figure 4.4 we see the results of doing incremental training with a changing number of epochs in the selection and evaluation networks. It can be observed from the figure that the test accuracy is highest when the selection network is the same as the evaluation network regarding the number of epochs and number of neurons.

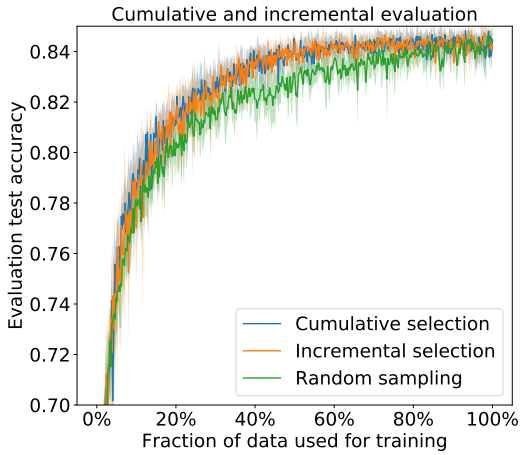
To summarise, there is a benefit for both the cumulative and incremental selection network to be the same as the evaluation network. Since incremental and cumulative



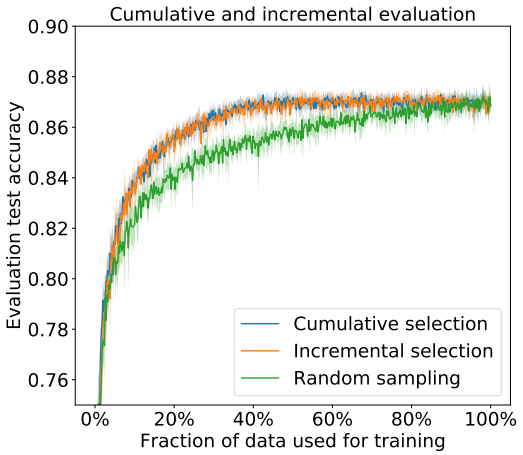
(a) MNIST. Evaluation network with 1 epoch.



(b) MNIST. Evaluation network with 5 epochs.



(c) Fashion-MNIST. Evaluation network with 1 epoch.

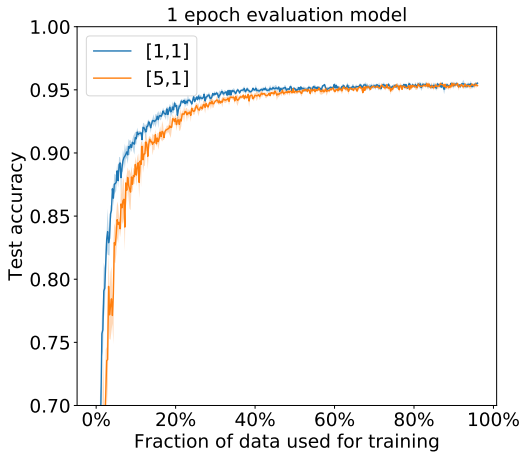


(d) Fashion-MNIST. Evaluation network with 5 epochs.

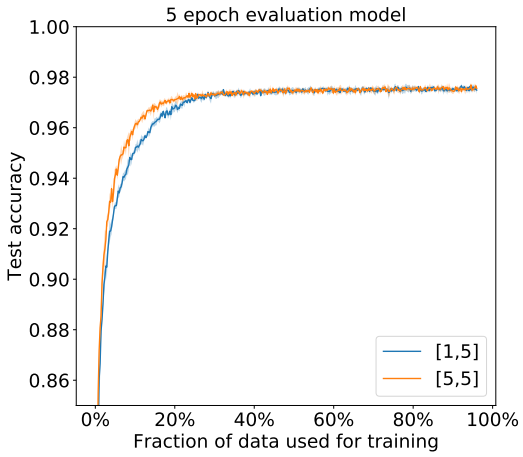
Figure 4.1: Evaluation model accuracy for selection networks sampling either incrementally or cumulatively. MNIST data set for which the selection and evaluation network both are $ANN1_{100}$ networks and the selection network only trained with 1 epoch, averaged over 6 reruns.

had similar performance when 1 epoch is used for selection, in Figure 4.1, it is clear that cumulative selection is better than incremental selection when 5 epochs are used. This can be confirmed by comparing the difference between the $[5, 5]$ and $[1, 5]$ lines in Figures 4.4a and 4.2b.

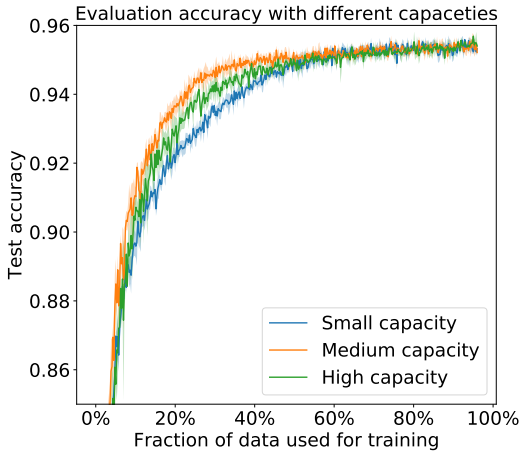
There may also be a large downside to having a selection and evaluation network which has not the same capacity for learning. It can be observed in Figures 4.2a, 4.2b, 4.2c and 4.3a that a higher test accuracy would be reached if the evaluation and selection model had the same number of epochs and neurons. In Figures 4.3c and 4.3b it is more difficult to distinguish the curves, but it can be observed that it is not disadvantageous to have the same number of neurons and epochs.



(a) A comparison if the selection network should use more epochs than the selection network.

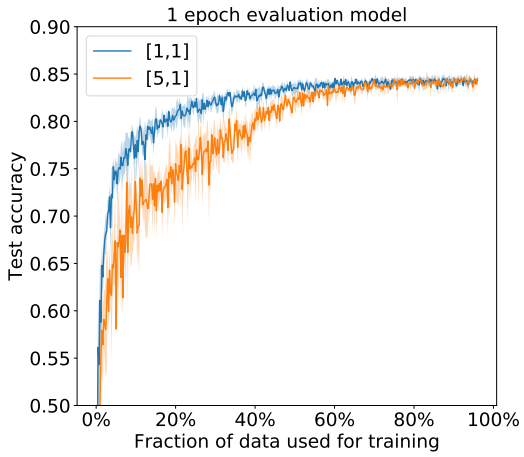


(b) A comparison if the selection network should use fewer epochs than the selection network.

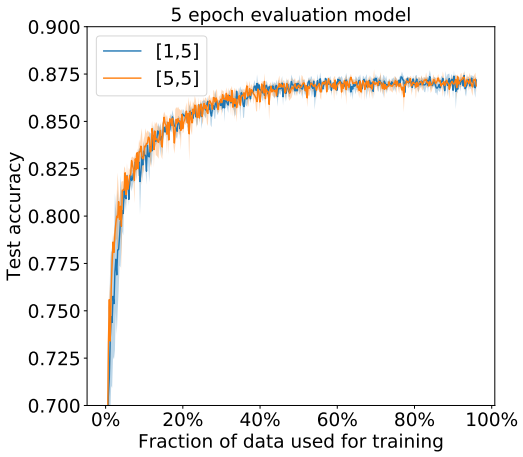


(c) The evaluation test accuracy for an $ANN1_{100}$ network. The selection network was either an $ANN1_{10}$, an $ANN1_{100}$ or an $ANN1_{1000}$.

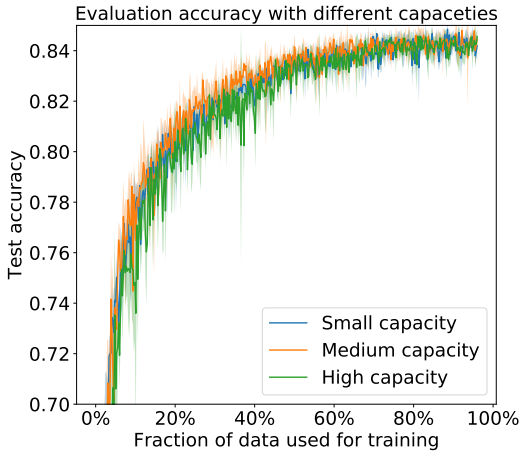
Figure 4.2: Varying network capacities for both the cumlatively trained selection and evaluation networks. MNIST data set averaged over 4 reruns.



(a) A comparison if the selection network should use more epochs than the selection network.

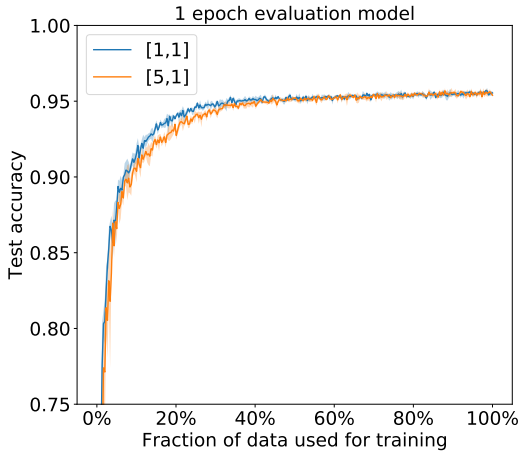


(b) A comparison if the selection network should use fewer epochs than the selection network.

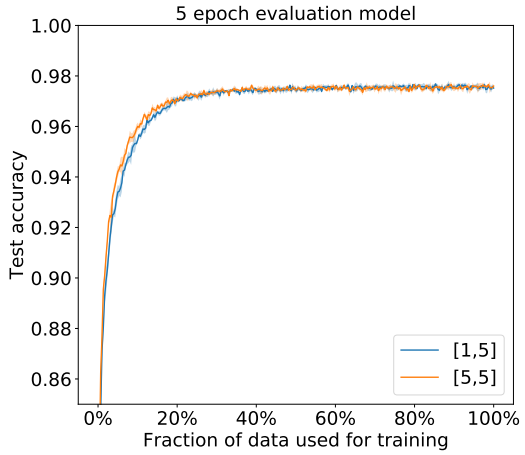


(c) The evaluation test accuracy for an $ANN1_{100}$ network. The selection network was either an $ANN1_{10}$, an $ANN1_{100}$ or an $ANN1_{1000}$.

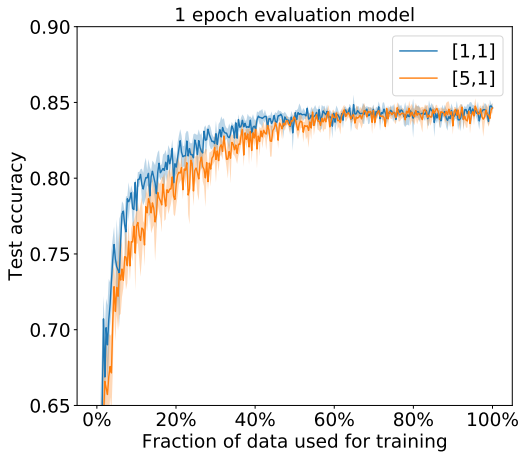
Figure 4.3: Varying network capacities for both the cumlatively trained selection and evaluation networks. Fashion-MNIST data set averaged over 4 reruns.



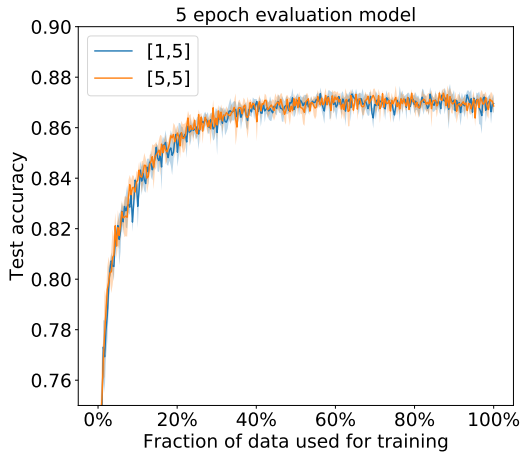
(a) A comparison if the incrementally trained selection network should use more epochs than the evaluation network. MNIST data set.



(b) A comparison if the incrementally trained selection network should use fewer epochs than the evaluation network. MNIST data set



(c) A comparison if the incrementally trained selection network should use more epochs than the evaluation network. Fashion-MNIST data set.



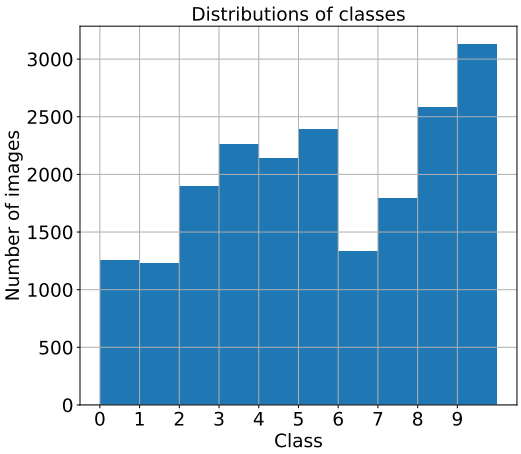
(d) A comparison if the incrementally trained selection network should use fewer epochs than the evaluation network. Fashion-MNIST data set

Figure 4.4: Varying number of epochs for selection and evaluation networks. Incremental training with an $ANN1_{100}$ network, averaged over 4 reruns.

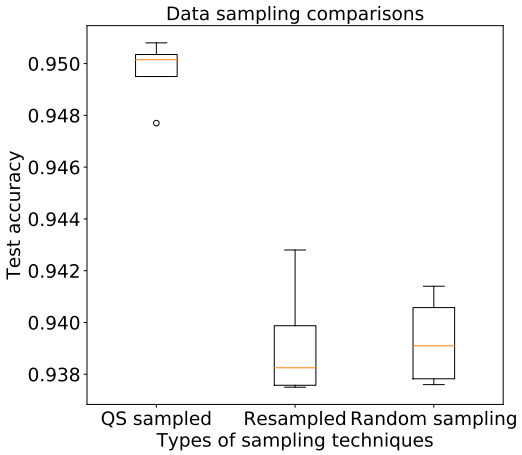
4.3.2 Sample Distributions

The margin subset, random subset and resampled subset were used to train a network. After training, the test accuracy of the network was assesed to compare the subsets. Keep in mind that all examined data sets, i.e. MNIST, Fashion-MNIST and CIFAR-10, have a close to uniform distribution between the classes, which can be seen in Figures 4.5d, 4.6d and 4.7d.

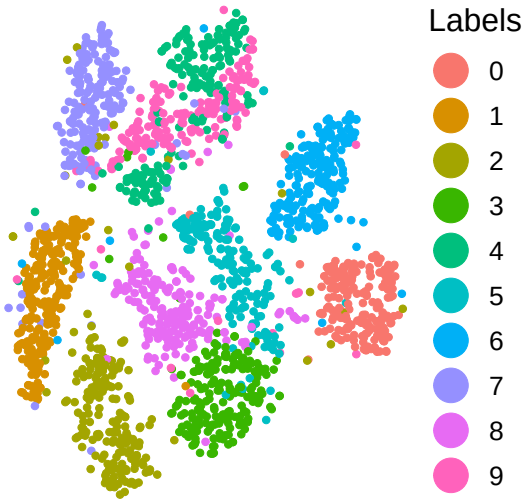
Considering the MNIST data set, four types of plots are shown in Figure 4.5. All types of plots were made with 20000 data points for MNIST and Fashion-MNIST. 5000 points were used for the CIFAR-10 data set. First type of plot is a histogram depicting the distribution of classes sampled by the margin query strategy, see Figure 4.5a. The second plot shows the test accuracy for the margin query strategy sampled subset, the resampled subset, and the randomly sampled subset, see Figure 4.5b. The third type of plot shows a tSNE dimension reduction performed on the MNIST data set, see Figure 4.5c. From this figure, a visual inspection can be made to determine how separable the clusters are. Similar figures can be seen in Figure 4.6 and Figure 4.7 for the Fashion MIST and CIFAR-10 data set, respectively. It is observed that the highest test accuracy is achieved with the subset sampled with the margin query strategy. The accuracy obtained with the resampled distribution is not significantly different compared to random sampling.



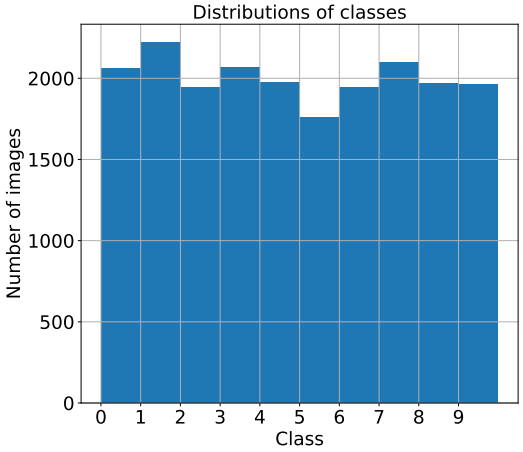
(a) Distribution of labels selected by the margin query strategy.



(b) Boxplot of the accuracy obtained with margin query strategy (QS sampled), the resampled subset and the random subset.

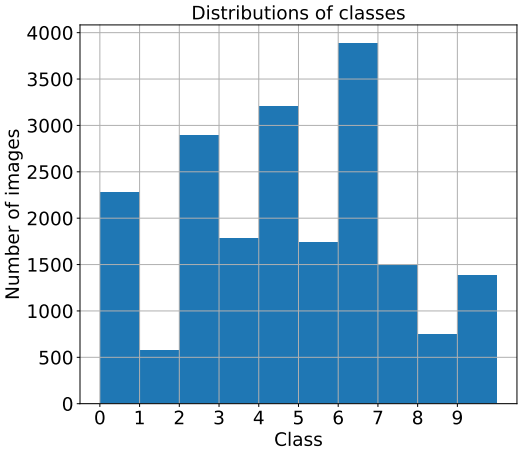


(c) The tSNE dimension reduction of the MNIST data set with true labels.

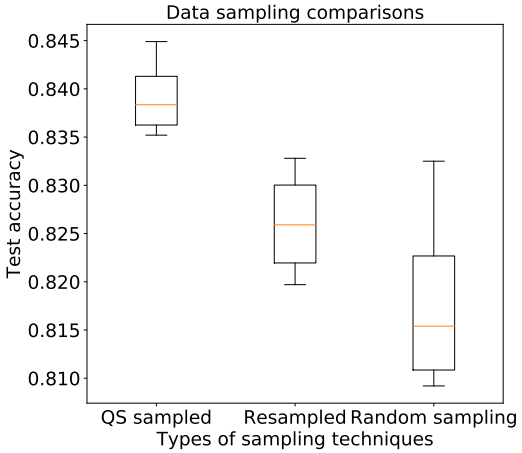


(d) Distribution of labels selected by the random query strategy.

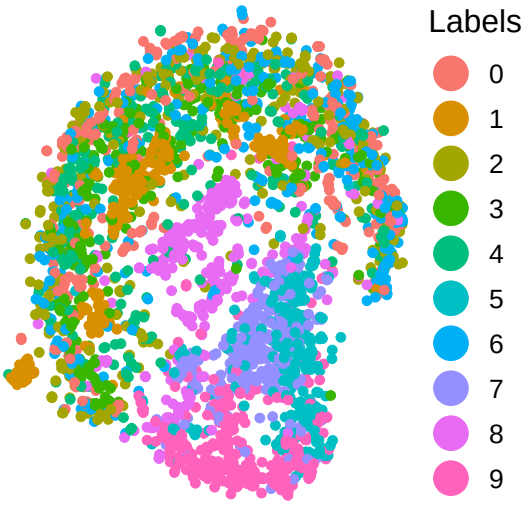
Figure 4.5: Characteristics of the subsets obtained by active learning for the MNIST data set, averaged over 50 reruns, choosing 20000 images.



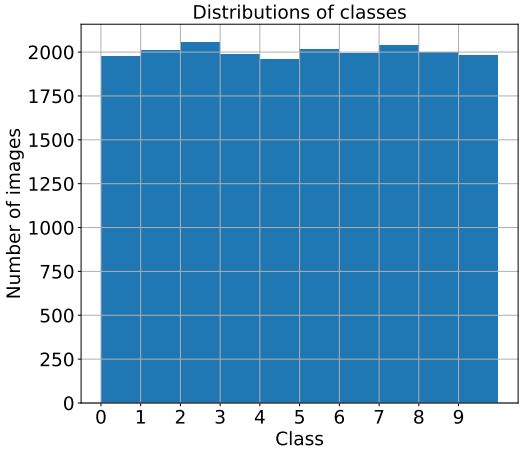
(a) Distribution of labels selected by the margin query strategy.



(b) Boxplot of the accuracy obtained with margin query strategy (QS sampled), the resampled subset and the random subset

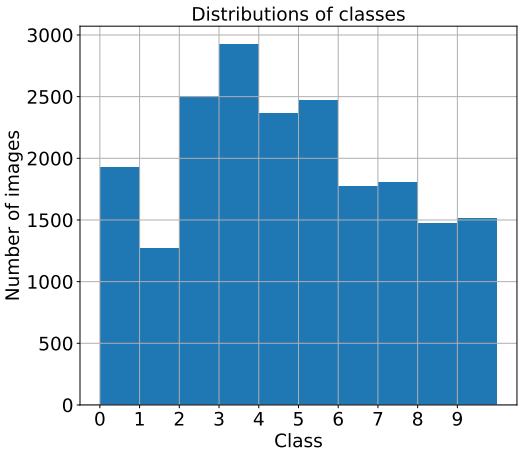


(c) The tSNE dimension reduction of the Fashion-MNIST data set with true labels.

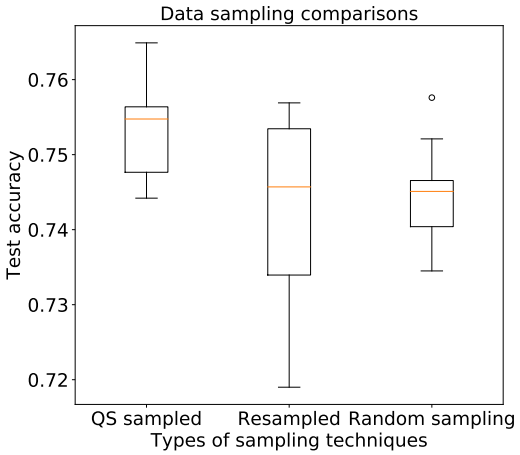


(d) Distribution of labels selected by the random query strategy.

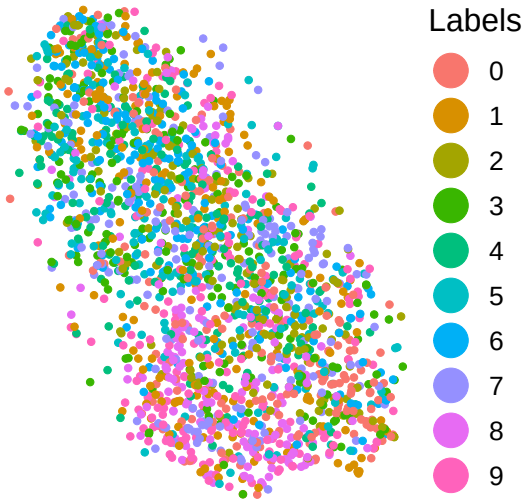
Figure 4.6: Characteristics of the subsets obtained by active learning for the Fashion-MNIST data set, averaged over 50 reruns, choosing 20000 images.



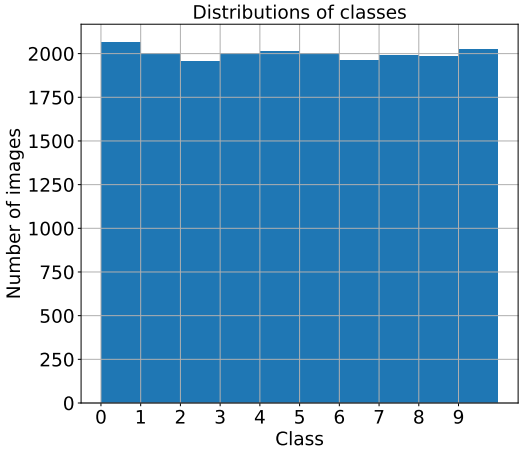
(a) Distribution of labels selected by the margin query strategy.



(b) Boxplot of the accuracy obtained with margin query strategy (QS sampled), the resampled subset and the random subset.



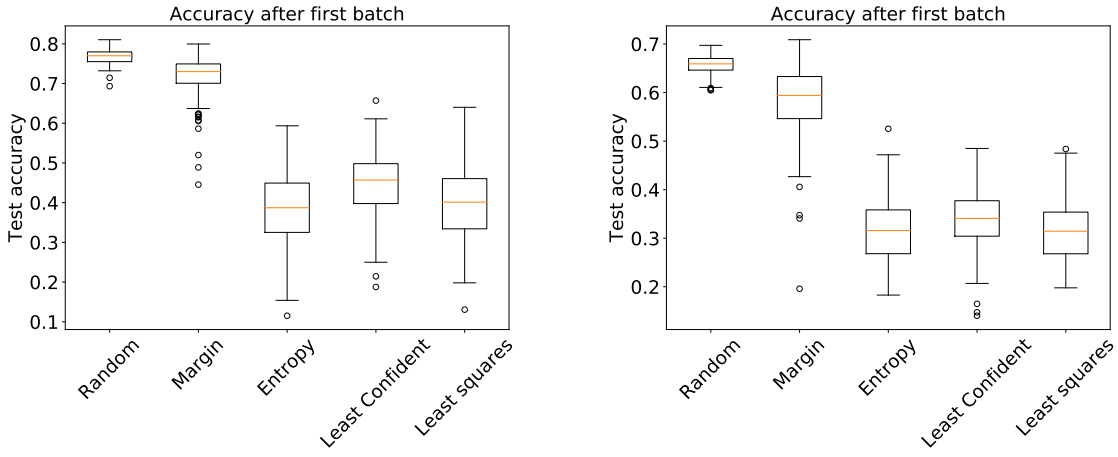
(c) The tSNE dimension reduction of the CIFAR-10 data set with true labels.



(d) Distribution of labels selected by the random query strategy.

Figure 4.7: Characteristics of the subsets obtained by active learning for the CIFAR-10 data set, averaged over 10 reruns, choosing 20000 images.

4.3.3 Random Start



(a) MNIST **(b)** Fashion-MNIST

Figure 4.8: The accuracy obtained by sampling the first 1000 points with a variety of query strategies. Averaged over 200 reruns with an $ANN1_{100}$ network.

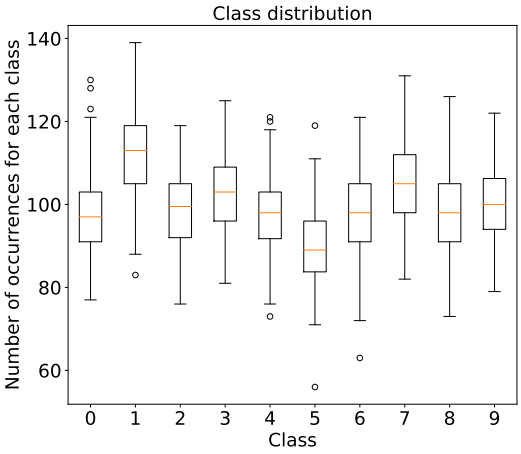
Figure 4.8 shows the the test accuracies of networks trained with a single batch selected by the different query strategies. It can be seen that the test accuracies are significantly lower for entropy, least confident and least squares, compared to random sampling. The mean accuracy from the margin query strategy seems to be slightly lower than random sampling. This holds true for both the MNIST and Fashion-MNIST data sets.

Figure 4.9 shows boxplots of how many times each class was selected by respective query strategy using the randomly initialised network, over the different reruns. The equivalent results for least squares and least confident can be found in Figure A.4, where it can be noted that they behave very similarly to the entropy query strategy. Sampling randomly gives a sample distribution which has a similar class distribution to the entire training set. Moreover, the margin and the entropy query strategies are observed to select samples where one class is over-represented.

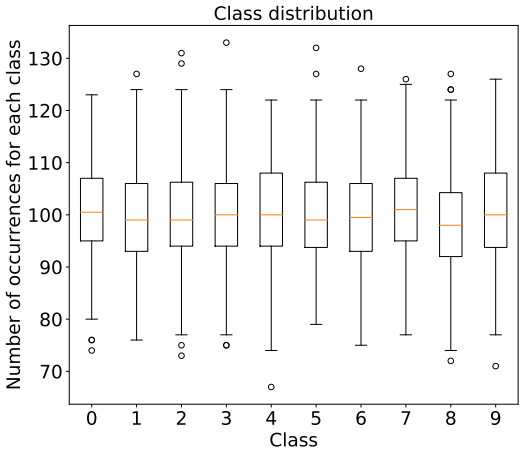
The Figures 4.10d and 4.11d shows the result for Fashion-MNIST and the entropy query strategy. The figures show a significant increase in test accuracy given by the usage of a random start. The effect that a random start size of 200 has on the margin query strategy can be seen in Figures 4.11a and 4.11b, for the MNIST and Fashion-MNIST data sets, respectively. Similar plots for a random start size 2000 can be seen in Figures 4.10a and 4.10b. All figures show that the random start has a negligible effect on the margin query strategy, independently of the examined random start sizes.

Lastly, Figure 4.12 shows the informativeness measures for the different examined query strategies with the randomly initialised networks given the value of α. α determines the fraction of dark pixels in 28×28 randomly generated image. The figure shows that the mean informativeness measure, averaged over the 100 reruns,

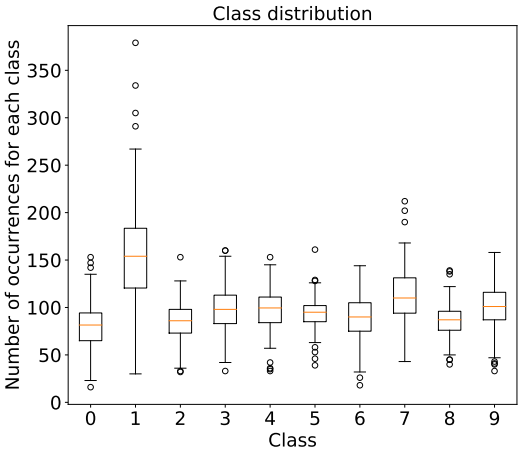
increases with α. Note that all correlation looks similar for all the query strategies. However, the margin query strategy shows the largest relative variance over the reruns. All query strategies examined, except for random sampling, showed preferences for dark images. We also note in Figure 4.12 that the query strategies obtain their maximum possible values at $\alpha = 100\%$ of $\ln(10)$ for entropy, 0 for least squares and margin, and $1/10$ for least confident.



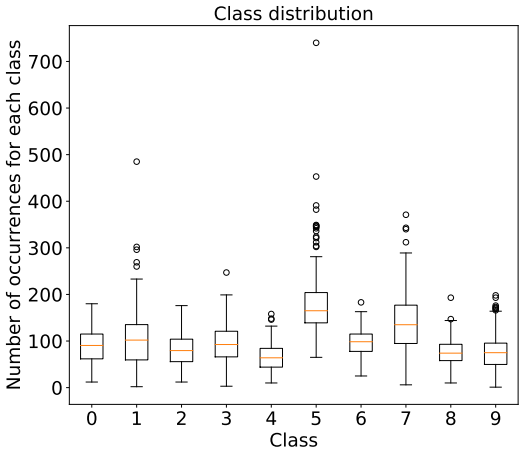
(a) Sampled with random query strategy using the data set MNIST.



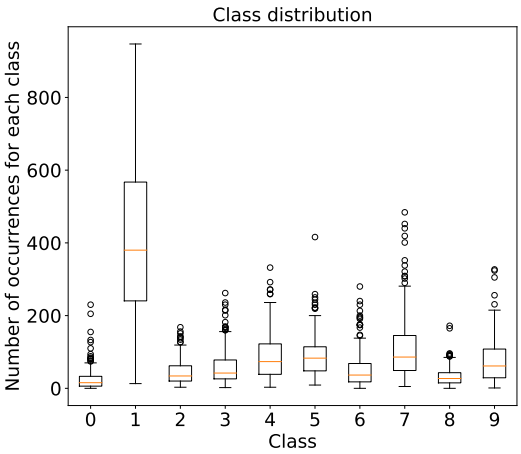
(b) Sampled with random query strategy using the data set Fashion-MNIST.



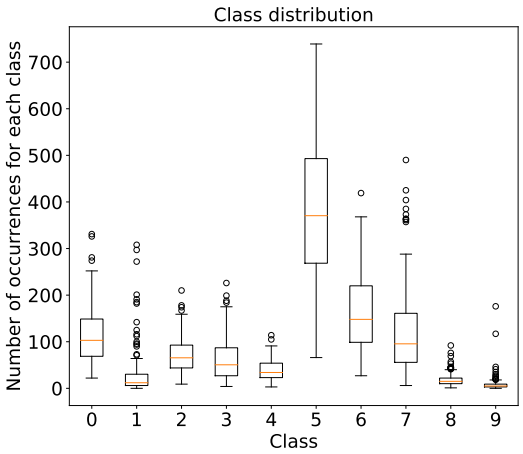
(c) Sampled with margin query strategy using the data set MNIST.



(d) Sampled with margin query strategy using the data set Fashion-MNIST.

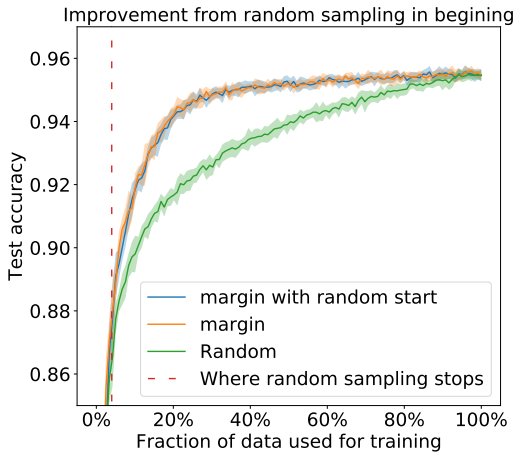


(e) Sampled with entropy query strategy using the data set MNIST

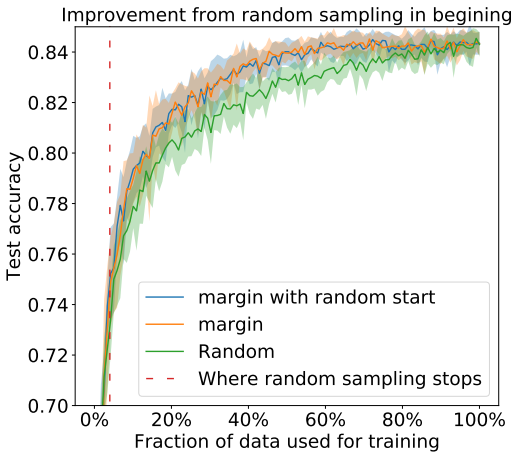


(f) Sampled with entropy query strategy using the data set Fashion-MNIST.

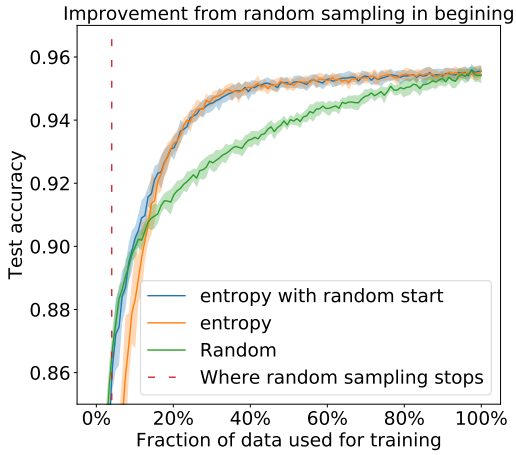
Figure 4.9: The label distribution obtained by sampling the first 1000 points with a variety of query strategies. 200 reruns.



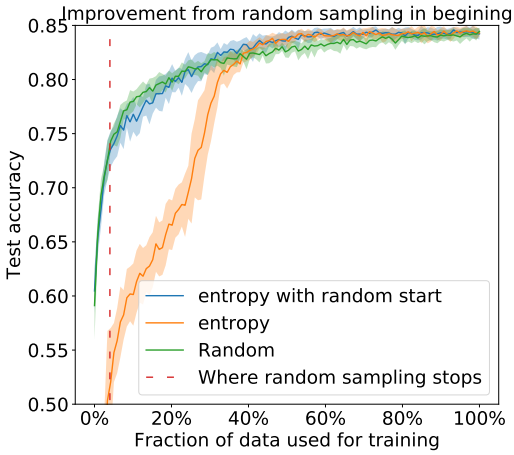
(a) MNIST



(b) Fashion-MNIST

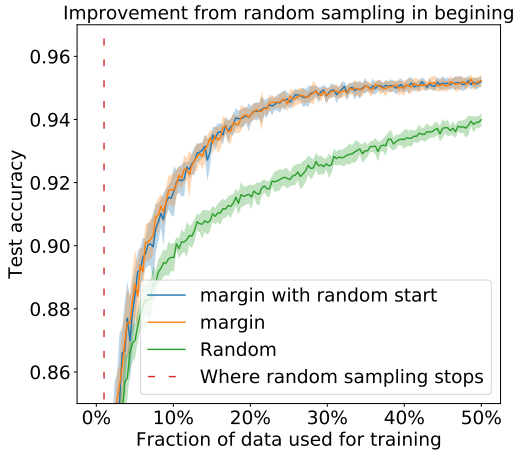


(c) MNIST

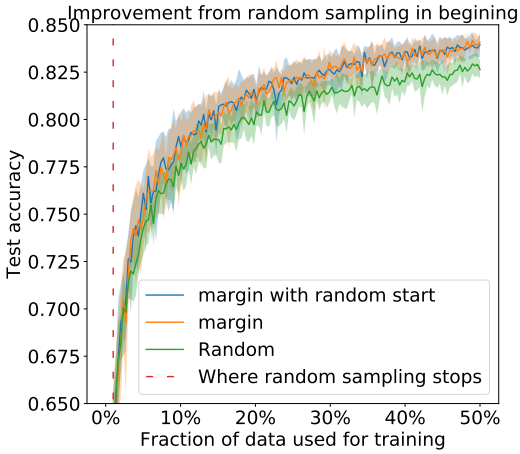


(d) Fashion-MNIST

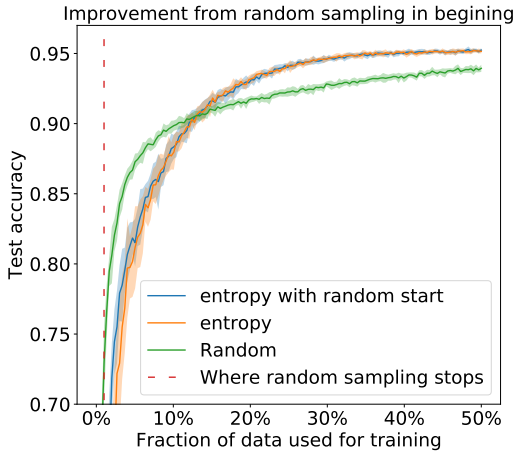
Figure 4.10: The prolonging effect of sampling with random data in the beginning for two different query strategies, namely Margin and Entropy. 2000 data points were randomly sampled in the beginning, averaged over 15 reruns.



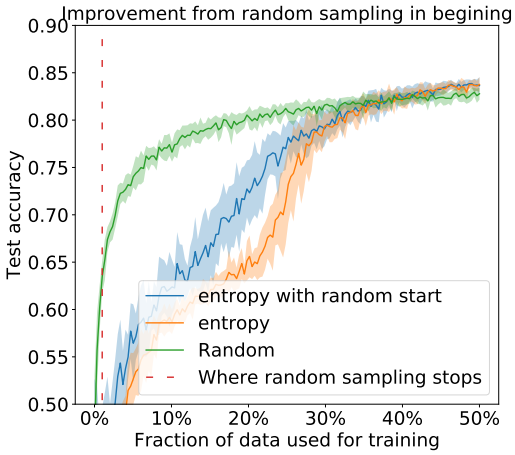
(a) MNIST



(b) Fashion-MNIST

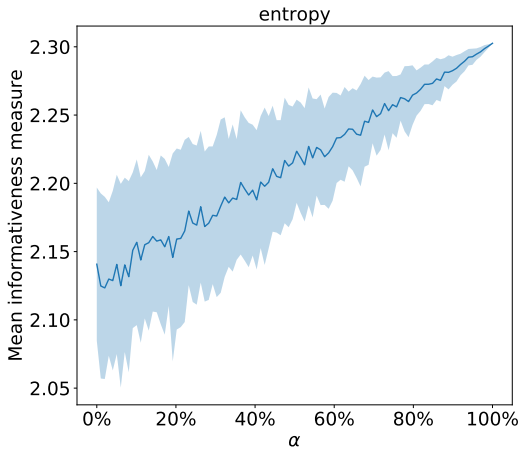


(c) MNIST

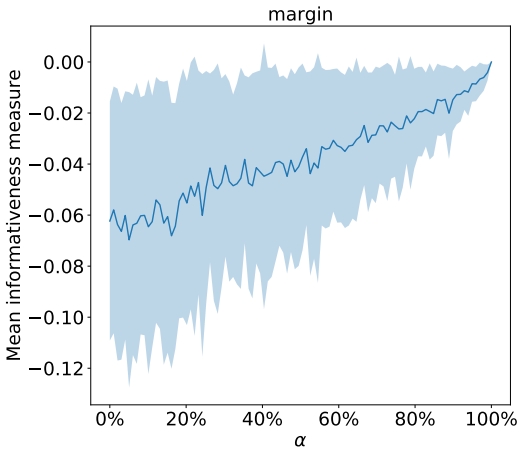


(d) Fashion-MNIST

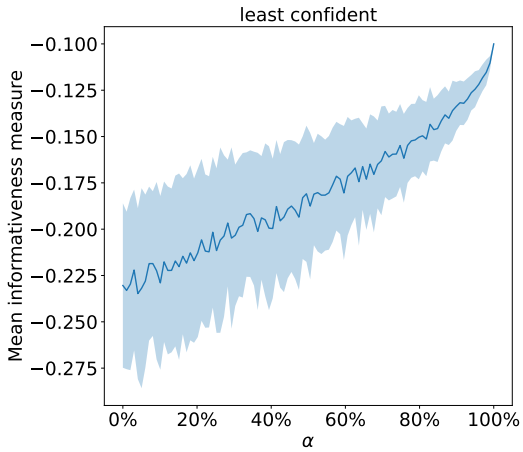
Figure 4.11: The prolonging effect of sampling with random data in the beginning for two different query strategies, namely Margin and Entropy. 200 data points were randomly sampled in the beginning, averaged over 15 reruns.



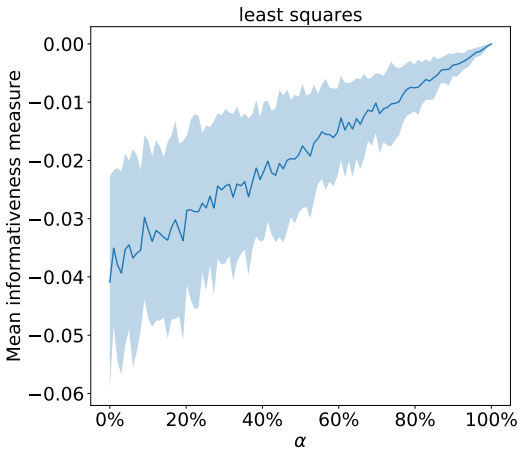
(a) Entropy query strategy.



(b) Margin query strategy



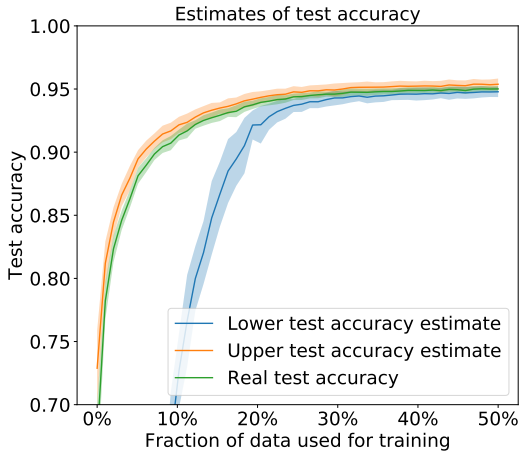
(c) Least confident strategy



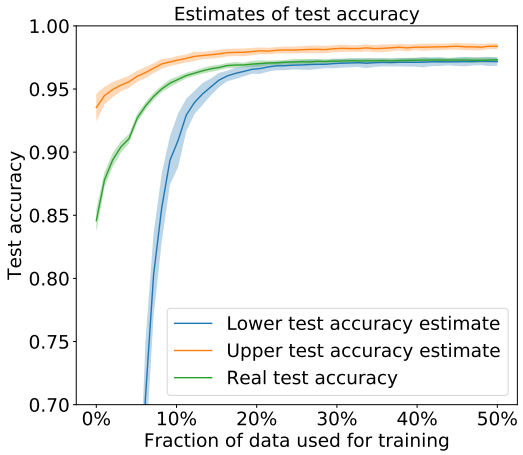
(d) Least squares strategy

Figure 4.12: Shows the informativeness measure as a function of how dark the image is, averaged over 100 reruns.

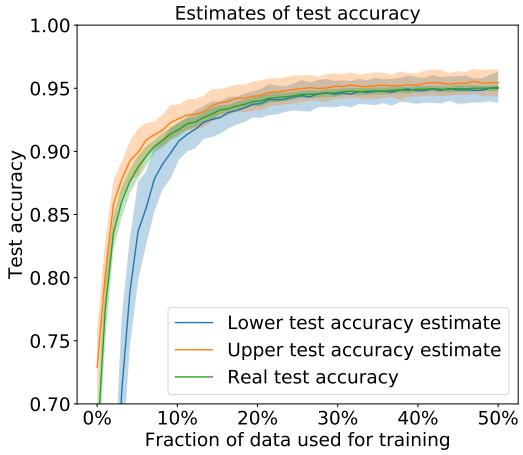
4.3.4 Estimates of the Test Accuracy



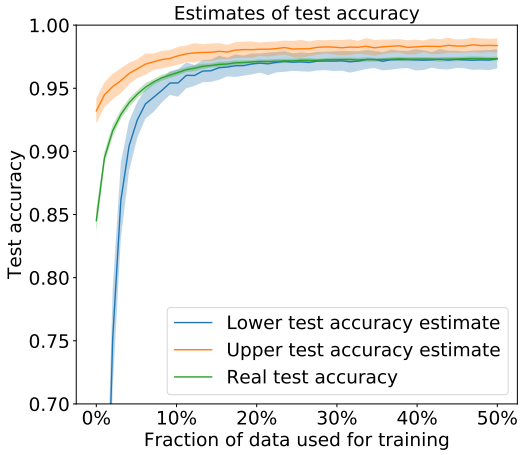
(a) 5% of the data, or 3000 images, made the random start. 1 epoch for selection and evaluation. Averaged over 50 reruns.



(b) 5% of the data, or 3000 images, made the random start. 5 epoch for selection and evaluation. Averaged over 50 reruns.



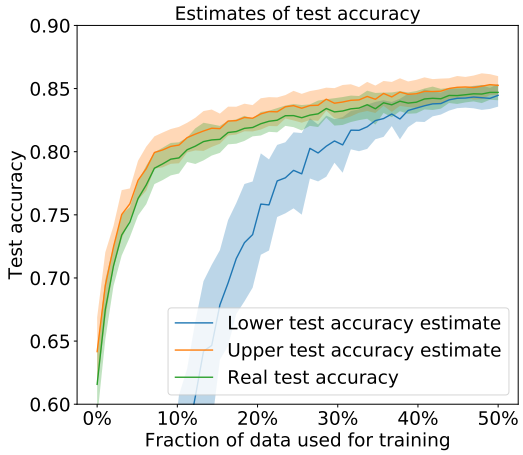
(c) 1% of the data, or 600 images, made the random start. 1 epoch for selection and evaluation. Averaged over 50 reruns.



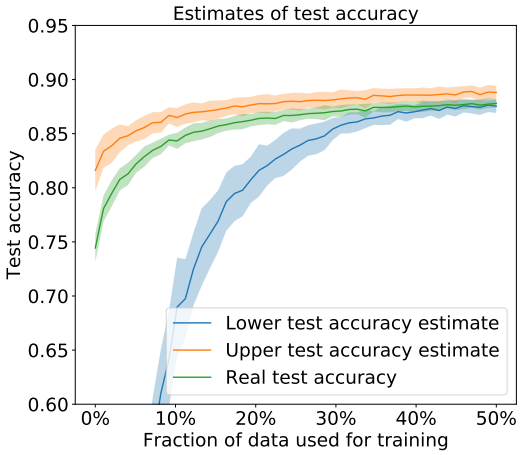
(d) 1% of the data, or 600 images, made the random start. 5 epoch for selection and evaluation. Averaged over 50 reruns.

Figure 4.13: Upper and lower test accuracy estimates for the test accuracy, for a variation of random start size and epochs for the MNIST data set.

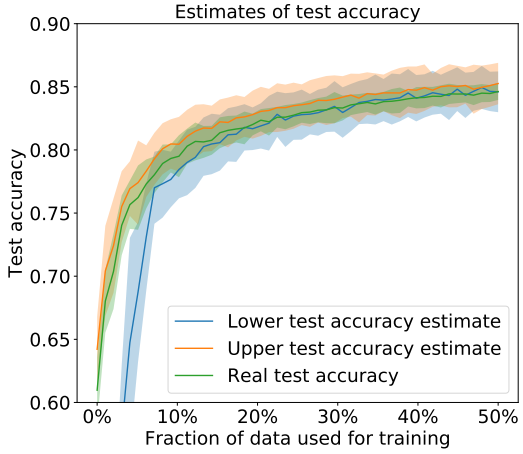
Figures 4.13 and 4.14 show the estimates of the test accuracy given different number of randomly sampled images. It can be observed that the random start, consisting of 1% the data, gave a more noisy estimate of the test accuracy than the test set consisting of 5%. It can also be observed that the upper estimate based on 5 epochs is worse than 1 epoch since with more epochs, a higher train accuracy is achieved. Having a larger random sample makes the lower estimate take a longer time to converge, needing up to 50% of the data to be labeled in Figures 4.14c and 4.14d.



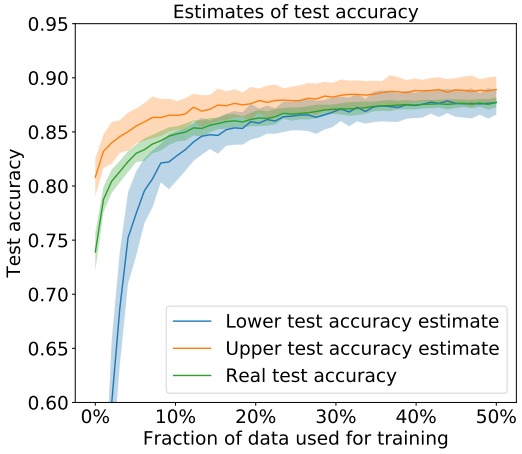
(a) 5% of the data, or 3000 images, made the random start. 1 epoch for selection and evaluation. Averaged over 50 reruns.



(b) 5% of the data, or 3000 images, made the random start. 5 epoch for selection and evaluation. Averaged over 50 reruns.



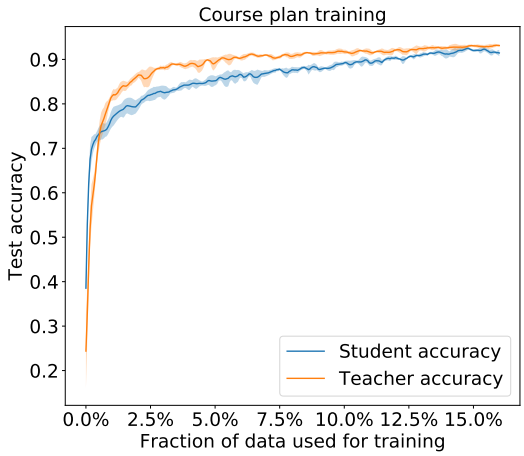
(c) 1% of the data, or 600 images, made the random start. 1 epoch for selection and evaluation. Averaged over 50 reruns.



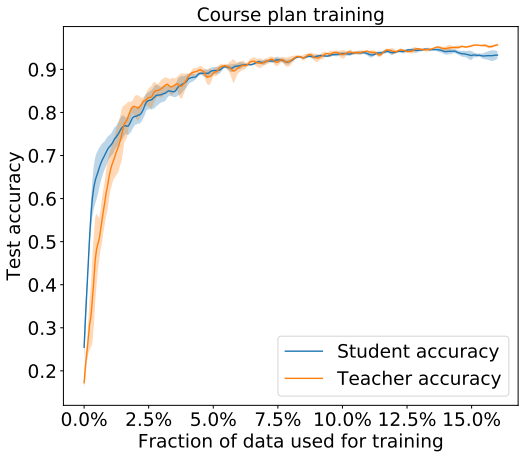
(d) 1% of the data, or 600 images, made the random start. 5 epoch for selection and evaluation. Averaged over 50 reruns.

Figure 4.14: Upper and lower test accuracy estimates for the test accuracy, for a variation of random start size and epochs for the Fashion-MNIST data set.

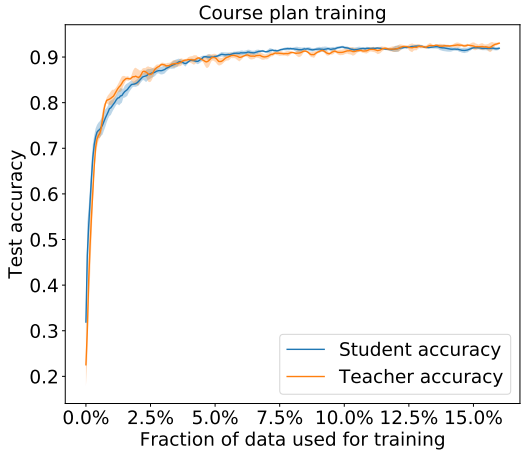
4.3.5 Course Plan Training



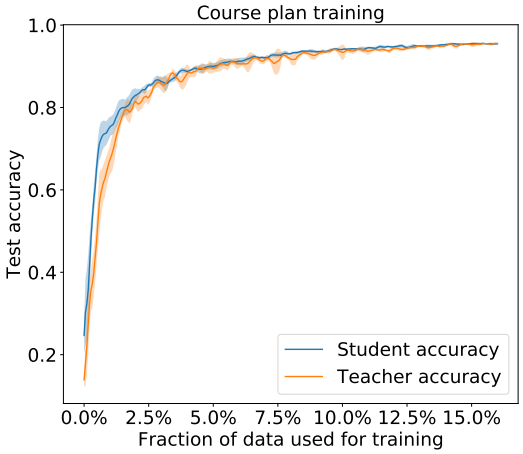
(a) Course plan order and random sampling.



(b) Course plan order and margin sampling.



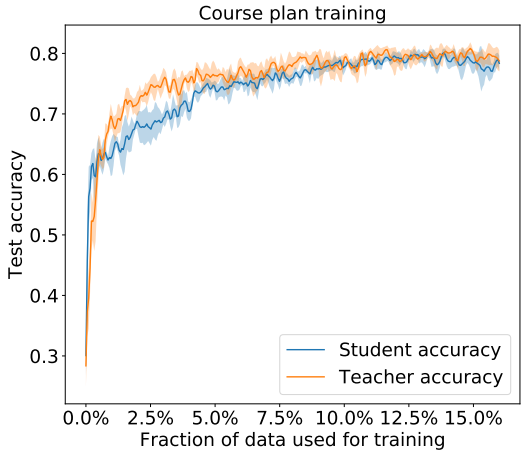
(c) Folded course plan order and random sampling.



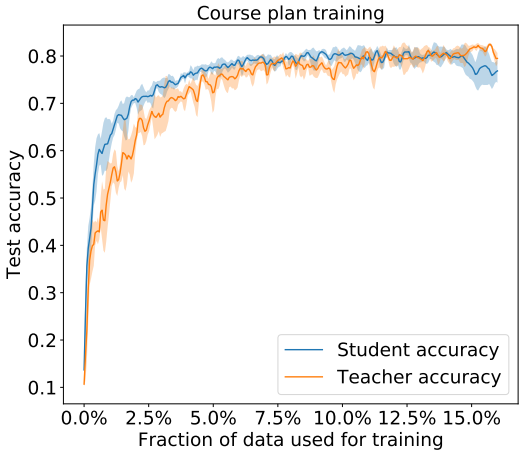
(d) Folded course plan order and margin sampling.

Figure 4.15: Accuracy of the teacher network and student's course plan training. MNIST with batch size 32 and 300 batches, averaged over 5 reruns.

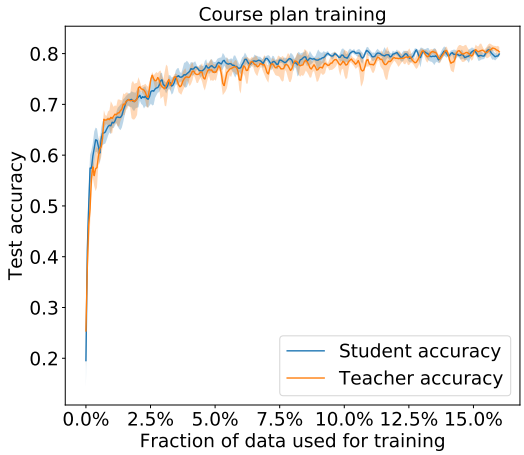
In order to compare the course plan orders, described in 4.2.5, the same subset will be used, but where the order is varying. Either the folded course plan, the course plan or random order will be used. The teacher network always used a random order for training on the labeled training set, which was selected by training incrementally with the use of a query strategy. The Figures 4.15 and 4.16 show the test accuracy of the student and teacher network after training on 300 batches (9600 images) on the MNIST and Fashion MNIST data sets, respectively. The student networks trained incrementally on batches, containing 32 images each, created and ordered by the teacher network. Of most interest is the test accuracy for the last batch. Training the student with the course plan training often causes the student network to have a lower test accuracy compared to the teacher network at its last batch, seen in Figures 4.15 and 4.16 for the MNIST and Fashion MNIST data sets, respectively.



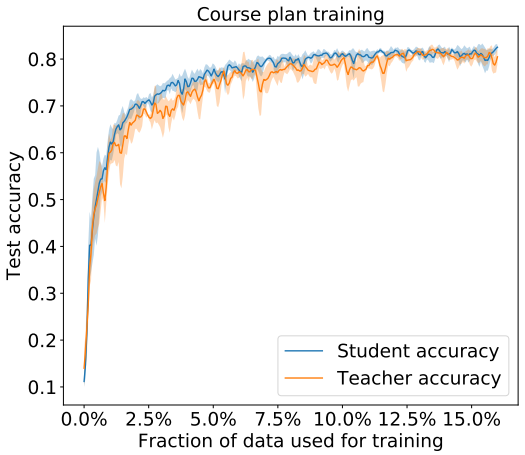
(a) Course plan order and random sampling.



(b) Course plan order and margin sampling



(c) Folded course plan order and Random sampling.

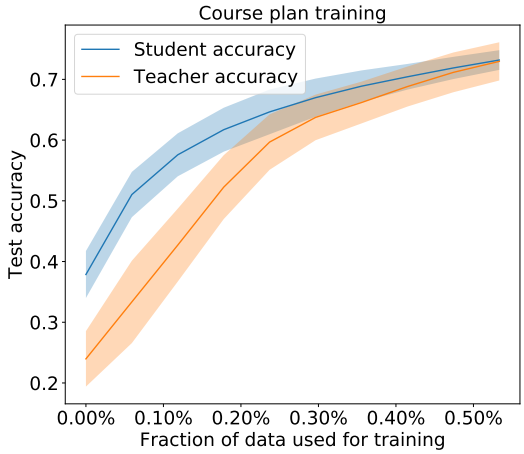


(d) Folded course plan order and margin sampling.

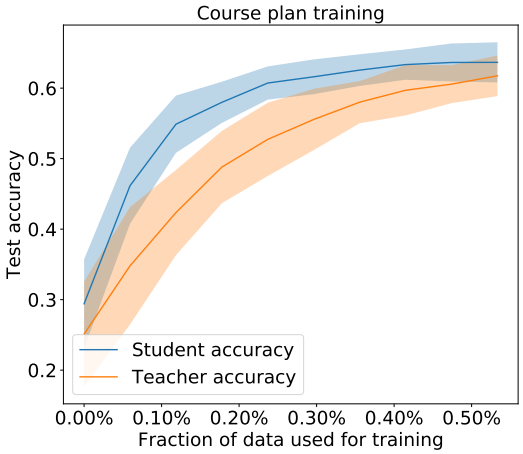
Figure 4.16: Accuracy of teacher network and student's course plan training. Fashion-MNIST with batch size 32 and 300 batches, averaged over 5 reruns.

This happens whether or not the teacher network was trained with the margin query strategy or the random query strategy. Training the student with the folded course plan order made the student perform similarly to the teacher network on the last batch, see Figures 4.15c, 4.15d, 4.16d and 4.16c. Again, the results are similar whether or not the teacher network is trained with the margin query strategy or not.

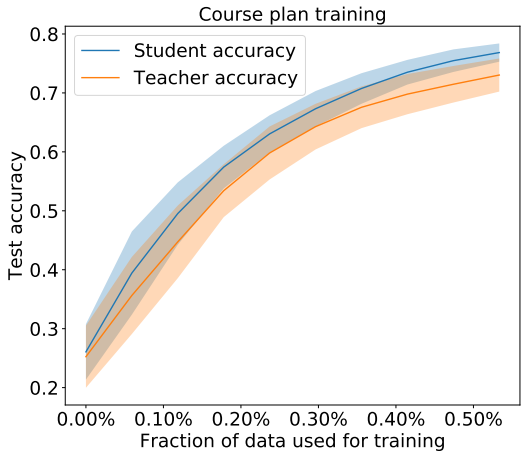
Figure 4.17 shows the test accuracy of the student and teacher network after training on 10 batches (320 images) on the MNIST and Fashion-MNIST data sets. For the results in this figure, the teacher network was only trained with the random query strategy. The figures show that the student networks has not gained performance using course plan training, see Figure 4.17a. However, by using the folded course plan training, the student does achieve a higher test accuracy compared to the



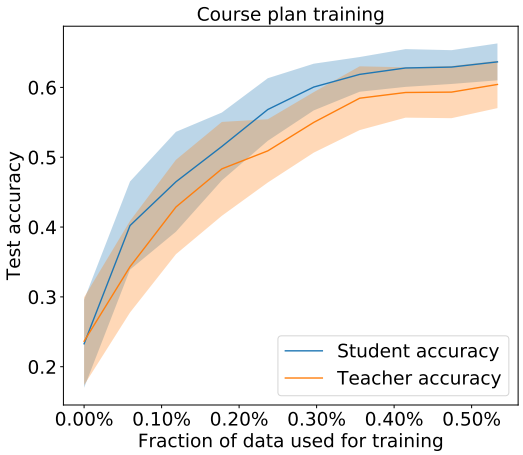
(a) Course plan order on the MNIST data set.



(b) Course plan order on the Fashion MNIST data set.



(c) Folded course plan order and random sampling on the MNIST data set.

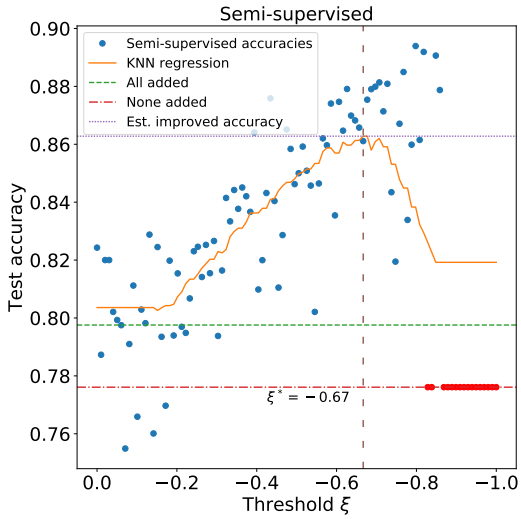


(d) Folded course plan order on the Fashion MNIST data set.

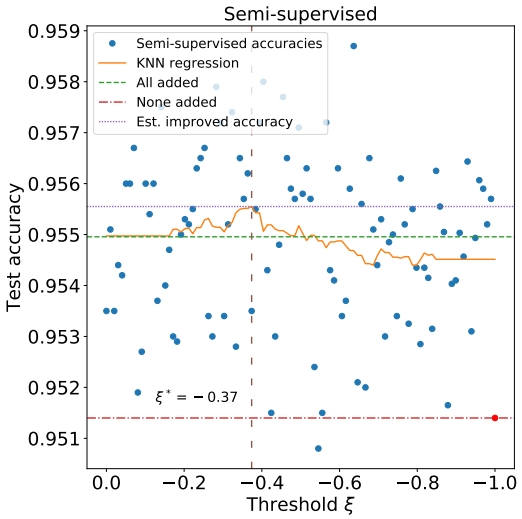
Figure 4.17: Accuracy of teacher network and student's course plan training using the random query strategy and the MNIST and Fashion-MNIST data sets with batch size 32 and 10 batches, averaged over 25 reruns.

teacher network on the last batch, see Figures 4.17c and 4.17d.

4.3.6 Semi-supervised Learning



(a) 1000 points, or fraction $\frac{1}{60}$, labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the MNIST data set.

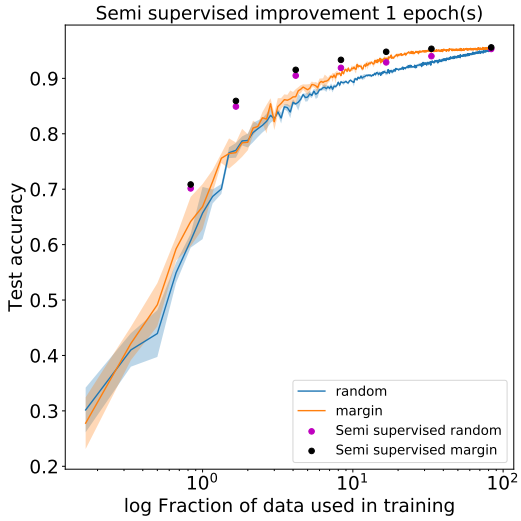


(b) 20000, or fraction $\frac{1}{3}$ points labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the MNIST data set.

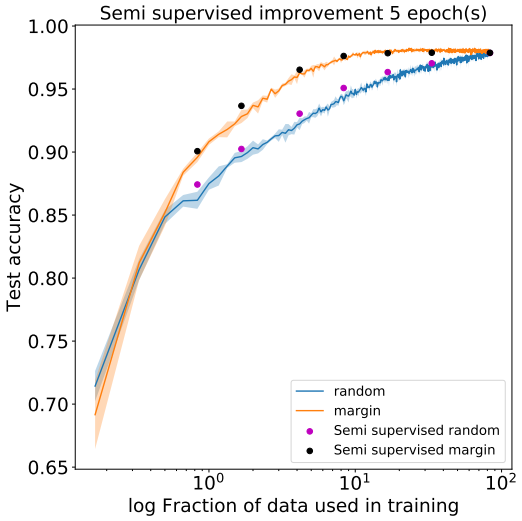
Figure 4.18: Shows the KNN-regression for two different fractions p of the data used for labeling. Note that there is a correlation between high threshold ξ and high test accuracy in 4.18a, while there is no correlation in 4.18b. Only 1 rerun of was performed for every fraction p.

Figure 4.18 shows how an optimal threshold ξ^* is found. It shows the improvement in test accuracy for semi-supervised learning using different candidate thresholds $\xi \in \Xi$ for an $ANN1_{100}$ network trained with the margin query strategy and 1 epoch. There is a correlation between a hard threshold (that is, a threshold ξ close to -1) and an improvement in test accuracy for the case when the network has trained on 1000 samples. When the network was trained with more data, 20000 samples, no correlation is observed. For the network trained with 1000 samples and 20000 samples, the approximate optimal thresholds are $\xi^* = -0.67$ and $\xi^* = -0.37$, respectively. The line corresponding to *All added*, in Figure 4.18, was averaged for all runs where all data was assigned a pseudo-label at the first iteration. This line corresponds to the way pseudo-labels was assigned in [26]. All corresponding figures for other p, data sets and query strategies can be found in Appendix, see Section A.5.

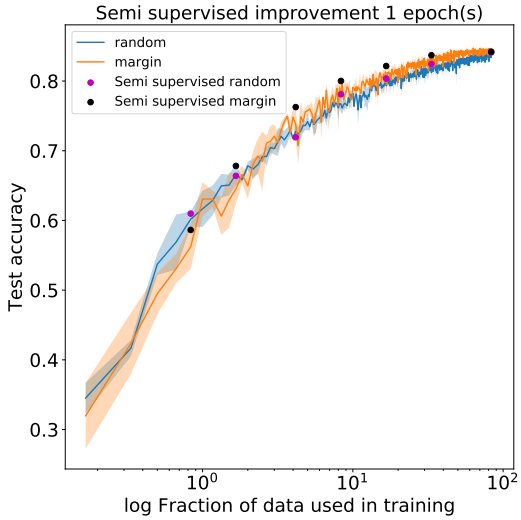
Figure 4.19 shows the test accuracy before and after pseudo-labeling was applied. It can be noted that the semi-supervised test accuracy a_ξ^* is higher after using the margin query strategy compared to the random query strategy. Also, note that the absolute and relative improvement was larger for the MNIST data set than for the Fashion-MNIST data set. Moreover, the absolute improvement is larger for the semi-supervised algorithm using an $ANN1_{100}$ network trained with 1 epoch, compared to an $ANN1_{1000}$ network trained with 5 epochs. The increase in accuracy is also



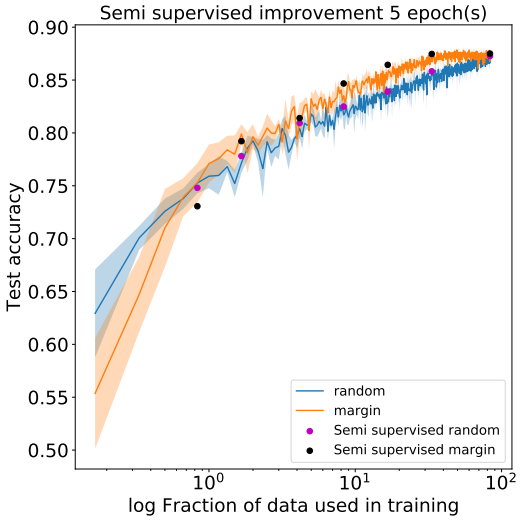
(a) MNIST for an $ANN1_{100}$ network with 1 epochs.



(b) MNIST for an $ANN1_{1000}$ network with 5 epochs.



(c) Fashion-MNIST for an $ANN1_{100}$ network with 1 epochs.



(d) Fashion-MNIST for an $ANN1_{1000}$ network with 5 epochs.

Figure 4.19: Shows the test accuracy before and after applying the semi-supervised learning. The optimal semi-supervised accuracies a_{ξ^*} are shown as dots, averaged over 4 reruns.

small when the fraction p of data is large.

4.4 Discussion

4.4.1 Evaluation of Sampled Data Sets

The results in Figure 4.1 show that having an incremental or cumulative selection model trained with 1 epoch is equally valid. However, by examining Figures 4.1, 4.2 and 4.4, we see that when the selection network trains with 5 epochs, then incremental is inferior in evaluation accuracy. We thus know that the accuracy of $[1, 5]$ in Figure 4.4b and the accuracy of $[1, 5]$ in Figure 4.2b are overlapping. This tells us that the accuracy of $[5, 5]$ in Figure 4.4b is smaller than the accuracy of $[5, 5]$ in Figure 4.2b, meaning that incremental selection with 5 epochs achieves a lower test accuracy than cumulative selection. For the Fashion-MNIST data set, where equivalent figures can be seen in Figures 4.1, 4.3b and 4.4d the same conclusion can not be drawn since the $[1, 5]$ and $[5, 5]$ networks yield similar test accuracies. In Figures 4.2a and 4.3a the evaluation network uses 1 epoch for training while the selection network uses 1 or 5 epochs for training. Given this, we have only observed that it is preferable to have a selection network with an equal number of epochs as the evaluation network. Selecting data with more epochs decreases test accuracy. Intuitively, this is probably because the network trained with 5 epochs is selecting points that are too difficult for the network with 1 epoch to fully learn from. In this setting, it is thus preferable to use a selection network that trains on the data with the same number of epochs as the evaluation network.

In Figures 4.2b and 4.3b the evaluation network uses 5 epochs for training while the selection network uses 1 or 5 epochs for training. Given this, we can see that it is preferable to have a selection network with an equal number of epochs as the evaluation network. Selecting data with fewer epochs decreases test accuracy. Intuitively, this is probably because the network trained with 1 epoch can not select data that is difficult enough for a network with more epochs to maximally learn from. It is choosing too simple points in some sense. In this setting, it is thus also preferable to use the same number of epochs for the selection and evaluation network.

The same conclusion can be made for Figure 4.4. The figure shows the same scenario, except that the incremental method is used to train the selection network, compared to the previous results where cumulative training was used. As previously seen, there is a decrease in accuracy when having a different number of epochs for the selection network compared to the evaluation network, except for Figure 4.4d, where both $[1, 5]$ and $[5, 5]$ perform equally well. A difference between the incremental and cumulative training methods is that there is overall a smaller difference between the accuracies corresponding to $[1, 5]$ and $[5, 5]$. This is probably because training many epochs on a single batch does not make the model learn efficiently.

Looking at Figures 4.2 and 4.3 we see that the case where the selection and evaluation model has the same number of neurons outperforms cases when there are more or fewer neurons in the selection network.

Regarding the evaluation networks all tested networks have benefited from using 5 epochs, compared to 1 epoch. It is, therefore, a good idea to use an appropriate number of epochs for the evaluation network. We also concluded that the selection network should use the same number of epochs as the evaluation network. It is therefore important that the evaluation model is determined before labeled data is selected using active learning. When the selection and evaluation networks are the same, and the selection network uses cumulative training, then having no evaluation network is needed.

Using the evaluation accuracy instead of the selection accuracy seems to have both an advantage and disadvantage. An advantage is that it answers more precisely how informative the selected data set is, given which network will use the data. For example the incremental selection accuracy can be seen in Figure 3.2a and the incremental evaluation accuracy can be seen in Figure 4.1. The disadvantage is that an evaluation network is not needed since it should be the same as the selection network.

4.4.2 Sample Distributions

A visual comparison between the histogram of the sampled classes in Figure 4.5a and the tSNE clustering in Figure 4.5c suggest that the label distribution is appropriate since the classes that are more difficult to separate in the tSNE clustering correspond to larger bins the histogram. For example, classes 4 and 9 seems to be difficult to separate. The same conclusion can be suggested by comparing the corresponding Figures 4.6a and 4.6c, even though the visual inspection of Figure 4.6c is more difficult. For example the Fashion-MNIST classes $1, 7, 8$ and 9 seem to be easier to separate than classes $0, 2, 4$ and 6. The classes $0, 2, 4$ and 6 correspond to *T-shirt/top, Pullover, Coat* and *Shirt*, respectively. Classes $1, 7, 8$ and 9, which are easier to separate, correspond to *Trousers, Sneaker, Bag* and *Ankle boot*, respectively. See Figure 2.2b for examples of the images. This supports the hypothesis that the margin query strategy samples a good distribution between classes. However, it is difficult to evaluate how appropriate a tSNE dimension reduction is. Another measure, such as silhouette width, would be needed to more thoroughly determine the separability of clusters.

The CIFAR-10 margin query strategy sampled class distribution in Figures 4.7a and 4.7c is more difficult to analyse since the tSNE dimension reduction is not as easy to examine visually. To get an intuition for if the tSNE dimension reduction is appropriate for the CIFAR-10 data set, Table 2.1 can be used to translate a class number to what the image is depicting. This is to help the reader decide if the class distribution looks appropriate.

Active learning, using margin query strategy, chose a more informative data set, even compared with a sample with the same distribution of labels. This can be observed in Figures 4.5b, 4.6b and 4.7b. The margin sampled subset has a higher average test accuracy, compared to the resampled subset, despite them having the same

distribution of labels. This means that two subsets with the same label distribution are not equally good for training. There is a distribution within each class which seems to be essential for training networks optimally. The margin query strategy samples from close to the borders of clusters [9, p. 14]. In Figures 4.5b and 4.6b it can be observed that the resampled accuracy is approximately the same as the random sampling. Data that is selected by the margin query strategy thus seems to be sampled with a good distribution within classes. The distribution between classes has not been concluded to affect the test accuracy at a large extent.

To further investigate this topic more reruns could be performed to determine with more confidence the relation between the accuracies obtained by the different samples. Also, a more advanced measure of how separable two clusters are could be used to bring more insight into if the histograms in 4.5a, 4.6a and 4.7a are appropriate. Finally, it would be interesting to investigate the performance of a sample with the same between class distribution as random, but with the same within class distribution as the sample selected with the margin query strategy.

4.4.3 Random Start

Figure 4.8 shows test accuracy achieved by sampling with different query strategies using a randomly initialised network. It can be seen that the way entropy, least squares, and least confident samples data makes the network achieve significantly lower test accuracy compared to a network trained with the same amount of randomly sampled data. The test accuracy is only slightly lower for the margin query strategy. However, the figure clearly shows that sampling randomly could be beneficial early in training. The underlying cause behind why we observe a low test accuracy for the entropy, the least squares, and the least confident query strategies, could be due to the distribution between classes or within classes, analogous to the *Sample Distributions* substudy, in Section 4.1.2. It was earlier hypothesised, in Section 3.4.1, that a query strategy could be biased when used for sampling with a randomly initialised network. Figures 4.9 and A.4 indicate that this is the case. The figures show the number of samples belonging to respective class for the first batch sampled, over several reruns. A total of 1000 images were sampled for every rerun, meaning that if one class was more abundant, then the remaining classes were less abundant. The quantiles shown in the boxplot helps us understand the variation of the number of sampled classes between reruns. For example, the maximum number of samples of class “1” was almost 900 in Figure 4.9e, meaning that only 100 images contained all of the remaining 9 classes. For random sampling, in Figure A.4a, the maximum of any class was 140, for comparison. Least squares, least confident and entropy all show preferences for class 1 in the MNIST data set and class 5 in the Fashion-MNIST data set, corresponding to a handwritten “1” and “Sandal”, respectively. The margin query strategy also has a bias for the same classes but to a lower extent.

Looking at Figure 2.2, where an example of all classes are shown, and using intuition for how the number “1” and a “Sandal” usually look, we suggest that the query

strategies had a preference for classes with few bright pixels. In Figure 4.12 it can be seen that a randomly initialised network, with the query strategies entropy, least squares, least confident and margin, is biased toward choosing images with a lot of dark pixels. For the margin query strategy, the same trend can be observed, but there is a larger relative variance, which was probably the reason for why it sampled more appropriately than entropy in Figure 4.9. The images, which were modeled without any inherent structure, are still representative of any image since the weights between any pair of neurons in two layers had the same Glorot distributed weights. We believe that the bias is mostly due to the bias terms $\theta = 0$ in the initialisation of the networks. A graphical intuition to why would be to note that $\theta = 0$ makes all decision boundaries, of each artificial neuron, intersect the origin. The darker images are closer to the origin which makes the network conflicted about how to do the classification.

What is more important than the accuracy after the first batch is the accuracy obtained after more than one batch has been labeled. Figures 4.10 and 4.11 show the test accuracy of random sampling, the entropy query strategy and the entropy query strategy with random sampling in the beginning. It can be seen in Figures 4.10d, 4.10c and 4.11d that there is a lasting benefit from performing random sampling during the first couple of batches for the entropy query strategy. This can be confirmed by noting that all test accuracies attributed to entropy without random start are lower or equal to entropy with random start. It is hypothesised that the same conclusion could be drawn for the least confident and the least squares query strategies since they behave similarly to entropy based on the results from Section 3.3.1 as well as in Figures 4.9 and A.4. This is however needed to be tested in order to be concluded. Sampling randomly in the beginning does not decrease or improve the test accuracy for the margin query strategy, as can be seen in Figures 4.10a, 4.10b, 4.11a and 4.11b. In other words, using random start has a negligible effect on the margin query strategy.

The difference between Figure 4.10 and 4.11 is that the random start sample size is 200 and 2000, respectively. Comparing these shows us that the test accuracy of entropy with random start is higher when the random start sample was larger. This probably shows that a random start sample size of 200 is insufficient to train on, due to the variety of data needed for this particular problem, while a sample size of 2000 is. This shows that the sample size is important and needs to be sufficiently large. No general conclusion can be drawn on how to determine the optimal random start sample size if this methodology is used in practice.

As a last remark, we would like to mention that knowledge[1] about the data probably could be used with a more advanced sampling technique to make a good initial sample for training [19, p. 249]. However, if nothing is known about the data then

[1]For example, if it is known that the data contains classes *Fish*, *Dolphines*, *Baby sharks* and *Sea stars* more data could be sampled that has a lot of blue pixels in them, initially. This is reasonable since the sea star is probably the easiest class to predict correctly, and it should also have less blue pixels in the image.

random sampling, which is equivalent to a uniform prior in Bayesian statistics, is appropriate. This ensures that the model has trained on a variety of data.

Two conclusions have been made. First is that the query strategies tested with a randomly initialised network are biased towards dark images. The second is that some query strategies perform worse than random until a lot of the data has been labeled and trained on. This effect can be somewhat reduced by sampling random in the beginning. These two conclusions are presumably linked. Our main hypothesis is that for the entropy, least squares and least confident query strategies the network needs to have some understanding of the whole data set before their corresponding informativeness measures can measure the benefit of labeling specific data points for the network. Random start seems to improve the performance of active learning in two ways: (1) using a random start may improve the performance of a query strategy, (2) by using a random start, the network will have seen a variety of data, without introducing any bias towards any specific class. We believe that this helps the network to choose data that is informative, which adds an additional performance benefit later in the training.

4.4.4 Estimates of the Test Accuracy

Looking at Figures 4.13 and 4.14 it can be seen that when using 1% of the data, or 600 images, as the test set, there is a larger variance compared to when 5% of the data is used. This means that just by chance the lower bound of the test accuracy could be larger than the real test accuracy. In order to be more certain that the lower test accuracy estimate is appropriate, it is, therefore, important to ensure that the random start size is sufficiently large. The estimates given by a usage of a random start set size corresponding to 5% of the unlabeled data gives a worse upper bound since a lot of the training data is in the test set. It also gives a worse lower bound since a lot of data has been excluded to make up the random start. Thus a compromise exists between how much data should be in the random start and in the training set.

A reasonable solution for a better lower bound estimate would be a cross validation algorithm which always includes the training set, but only includes parts of the random start for the training, and evaluating on the remainder of the randomly sampled data. A future extension would be to apply a Leave-One-Out Cross-Validation algorithm [28], with the modification that only one point from the randomly sampled data is used for evaluation, and for the training data the remaining randomly sampled points are added to the query strategy sampled data. We believe that this would provide for a good estimate of the test accuracy.

All images in Figures 4.13 and 4.14 show that the upper and lower estimates are better when a lot of data has been used. This is intuitive since if a lot of data has been labeled, then the randomly sampled subset will constitute a small part of the entire data set. It is also natural that the upper estimate improves since it is more difficult to model the noise in the data when there are more data in total. This argument is

supported by observing the training accuracy in Figure 3.9a, where the CIFAR-10 data set uses 50 epochs for training, and the training accuracy is practically 1 in the beginning but decreases when more data is labeled. When using few epochs, in Figures 4.13a, 4.13c, 4.14a and 4.14c and the upper estimate is close to the real test accuracy. When using many epochs, in Figures 4.13b, 4.13d, 4.14b and 4.14d the upper estimate is considerably higher. This is natural since the more epochs are used, the more can be learned about the training data. This directly affects the upper bound, since the data is shared between the test set and the training set.

The estimates work reasonably well but one has to obtain an intuition to make a good estimate of the test accuracy based on these. Things to consider are:

- If there are a lot of epochs, then the upper estimate is probably not useful.
- If there is a lot of data, then the lower and upper estimates will be a better approximation.
- If the test set substitute is small, that is a small random start was used, the result will be subjected to more noise.

4.4.5 Course Plan Training

In all results using a teacher network trained with 300 batches (9600 images), the student increases in accuracy quicker than, or as quick, as the teacher, see Figures 4.15 and 4.16. Towards the end of the training, the student declines in accuracy, and is outperformed by the teacher. As mentioned earlier, the only accuracy that is relevant in this case is the test accuracy obtained at the end. Thus, the faster learning rate in the beginning of the training is not an argument for using the course plan order or the folded course plan order. For the results corresponding to the training on 10 batches, see Figure 4.17, there might be a potential benefit in using the folded course plan training. Remember, it was seen in Section 3.3.1 that the order in which data is trained plays an important role in the performance of the network. Training incrementally with query strategies tended to create an order for which the end performance of the network was worse compared to networks trained with the random query strategy. In Figure 3.3 it could be seen that the final accuracy was higher for the random query strategy than the margin query strategy. Since they had used the same data the only difference between the performance is the order in which they were used for training. Thus, there could be a possibility that there exists an order, i.e. a course plan, that would imply a performance increase, compared to random order, for the student network. However, so far, we have not observed a reliable way of harnessing the usefulness of an ordered data set. For the special case where only 10 batches were used, we have seen a suggestion that the folded course plan training order may imply a performance increase. This indicates that there may be a way to reliably order to the data such that learning is maximised. We encourage further research into the topic, especially in combination with training with more than 1 epoch.

4.4.6 Semi-supervised Learning

The results in Figure 4.18 show the semi-supervised test accuracy given a threshold ξ. The difference between Figures 4.18a and 4.18b is that they train using 1000 and 20000 labeled images before applying semi-supervised learning, respectively. Figure 4.18b shows no correlation between the semi-supervised test accuracy and a hard threshold ξ. On the other hand, Figure 4.18a suggests a positive correlation. Figure 4.18a thus seems to indicate that the optimal threshold ξ^* is the strictest possible, that still allows the network to assign pseudo-labels, i.e. a threshold that makes S_ξ non-empty. Remember, a too strict threshold of ξ allows for no pseudo-labels to be assigned by the network which will result in an early stop of the algorithm. The interpretation of why there is an increase in accuracy, associated with performing the pseudo-labeling in many steps, is not known. It is possible that by labeling non-informative data first makes the incorrect labels uncommon. What happens in the next step is that the same pseudo-labeling occurs, but with a network that has trained on more data. It is therefore hypothesised that there are fewer incorrect labels when the threshold ξ is harder. In article [26] the equivalent of setting $\xi = 0$ was performed. Another choice has thus been contributed to further improve the effectiveness of semi-supervised learning.

A probable explanation as to why there is no correlation in Figure 4.18b is that the network is more certain in its labeling when it has been trained on more data. Almost all candidate thresholds in Ξ triggered the early stopping criterion of the semi-supervised algorithm due to the entire unlabeled set were given pseudo-labels at the first iteration. This means that the semi-supervised test accuracy in Figure 4.18b is merely random noise around the accuracy for labeling all data. Thus, there is no correlation between threshold and accuracy at the later stages of training, i.e. when a large portion of all unlabeled data has been labeled with active learning. In this case, there can be a benefit in increasing the unlabeled data set if this is possibility exists. This could make the network learn more about the data. Looking at Figure 4.18b, where 20000 images had been labeled, we can see that accuracy given by performing semi-supervised learning ranges between 0.951 and 0.959. Training with the entire data set with 1 epoch and an $ANN1_{100}$ network usually gives an accuracy of approximately 0.95 which can be seen Figure 4.2a. Even if semi-supervised predicted the labels of all remaining unlabeled data correctly, the accuracy would not increase significantly.

We can conclude that no validation set is necessary to find an optimal threshold. The hardest possible threshold with the condition that the S_ξ is non-empty gave one of the highest semi-supervised test accuracy, independently of how much data was manually labeled. This observation allows the procedure described in Section 4.2.6.2 to be disregarded for finding a suitable threshold when applying the algorithm in a real-world scenario. However, we did observe a negative correlation between good test accuracy improvement and hard threshold in a few experiment. This can be seen for the figures with 500 labeled data images in Section A.5. These cases were rare, and was not further investigated.

The results in Figure 4.19 allows for a comparison between the margin and the random query strategies. For the MNIST data set, looking at the improvement in Figures 4.19a and 4.19b, it can be seen that the semi-supervised accuracy a_{ξ^*} is higher after having used the margin query strategy compared to the random query strategy. However, this is not necessarily an indication that it is more beneficial to perform semi-supervised learning after the margin query strategy. It must be taken into consideration that the margin query strategy had a higher test accuracy before the semi-supervised learning was performed. To compare these it must first be defined how to compare improvements, which is not trivial. Is it for example better to increase from 0.89% to 0.92% compared to 0.85% 0.90%? The latter has a higher increase in accuracy while the former increases from an high accuracy to an even higher - which could be seen as more difficult. Thus, we cannot concluded if semi-supervised learning benefits from the distribution within classes associated with the margin query strategy. However, we can conclude that it is possible to use semi-supervised learning after usage of a query strategy.

This study has also shown that semi-supervised learning can be beneficial in the scenario where the only a small fraction of all data can be labeled. In the MNIST case, in Figure 4.18a, the test accuracy increased from 78% to 88% by applying semi-supervised learning, while only labeling 1000 images, of the total unlabeled data set consisting of 60000 images. An extension would be to compare more advanced semi-supervised learning algorithms to potentially increase the test accuracy even more. Another extension would be to perform semi-supervised learning during the active learning step, making the model more certain about which data to query next.

4.5 Conclusion

4.5.1 Evaluation of Sampled Data Sets

- Incremental and cumulative training select equally informative data, as long as the selection network only trains with one epoch.
- Many epochs can be used by the selection model beneficially if cumulative training is used. It is not as beneficial when incremental training is used.
- If many epochs will be used for the evaluation network, then many epochs should be used for the selection network as well.
- If networks with many neurons will be used for the evaluation, then networks with many neurons should be used for selection.

4.5.2 Sample Distributions

- The main contributor to the improved performance of the margin query strategy compared the random query strategy seems to the selected within class distribution. The selected between class distribution does not seem to be

equally important.

- A margin query strategy seems to sample data proportional to how inseparable clusters are, according to a tSNE dimension reduction.

4.5.3 Random Start

- All query strategies in Section 2.2.1, except for the random query, have been observed to not sample uniformly when used with randomly initialized networks.

- The margin query strategy, with a randomly initialized network, achieves a slightly lower test accuracy than random sampling on the first batch. Other query strategies achieve significantly lower test accuracy.

- Sampling randomly in the beginning of training is beneficial for the test accuracy when using query strategies such as entropy, least squares, and least confident. The increased test accuracy associated with performing random start propagates throughout the training.

- The margin query strategy test accuracy seem to be unaffected by sampling randomly in the beginning.

- All tested query strategies, using a randomly initialized network, are biased toward choosing darker images.

4.5.4 Estimates of the Test Accuracy

- Given that the training is initialized by sampling randomly, one can use the random samples to create a lower and upper estimate of the test accuracy. This can be beneficial in the absence of a test set.

4.5.5 Course Plan Training

- The suggested approach has not been observed to perform better than training on a random order.

4.5.6 Semi-supervised Learning

- Assigning pseudo-labels gradually, after both the random query strategy and the margin query strategy, usually yields a significant increase in test accuracy.

- In the case that there exists a correlation between a strict threshold ξ and increased test accuracy, the correlation is positive.

- To find a good threshold ξ to use in the semi-supervised algorithm, there are two possibilities:

- To use a randomly sampled subset, like in the *Random Start* substudy, in Section 4.1.3.

- To use the hardest possible threshold such that the network makes predictions of the labels in the unlabeled data set.

4.6 Future Work

In order to estimate the test accuracy, as in Section 4.1.3, a better lower estimate could be constructed. A technique similar to cross-validation could be used, where some part of the random sample is in the training set, while the rest is in the test set. What part of the data is in the test and train set alternates and after all randomly sampled data has been in the test set once, then the average accuracy is taken from all simulations. This has the benefit of using more data when estimating the test accuracy, and we believe that this would be a good improvement of the estimates, especially in the early stages of training.

In Section 4.4.2 it was mentioned that a more appropriate way of showing the distance between clusters should be used. Right now only a tSNE dimension reduction has been used, which can not be used more than for intuition. It would be interesting to use different distance measures, for example the silhouette width, determine if the between class distribution seems appropriate.

In Section 4.1.3 we used a randomly sampled set with two intentions. The first was to make sure the model had been trained on a variety of data, thus acting as a start seed for the selection network. The second intention was to use this sample as an approximate test set to estimate the accuracy of the network. It was shown that the estimates worked satisfactory, but this strategy could introduce some bias in the network for the test set substitute. For example, this random sample has been used to train a selection network which with a query strategy determines what data should be labeled next. It is thus possible that the data sampled is biased towards the randomly sampled subset. This could happen if the following samples are sampled in the close neighborhood to the random samples. A more theoretical analysis of this would be interesting.

It was shown in Section 4.3.5 that a beneficial order in which to train can potentially be constructed. It was only found to work for small randomly sampled subsets and using only one epoch. A future extension would be to come up with alternative data orders and investigate when they work. How does the order change depending on the number of epochs? What order should be used depending on the amount of data? How is the order affected by sampling with the random query strategy in comparison to the margin query strategy?

The work in this thesis can be extended by also considering other query strategies. Query strategies that are not based on uncertainty could be considered. For example, several networks could be trained, all initialized with different randomly assigned

weights. Unlabeled data could then be selected based on the disagreement of the networks. Data, for which the output of the networks differs, could be deemed as informative. Another possible query strategy would be to select the data for which the loss improves the most, instead of using stochastic gradient descent. However, to make the output of a neural network match closer to the true label, the gradients of the loss, with respect to the weights and biases, need to be estimated. This requires access to the true labels, which is a problem. This can maybe be partially solved by some of the methods introduced in the Semi-supervised Learning substudy: gradually assigned pseudo-labels can be used to label data. This method could potentially suffer if the network has bad prediction accuracy, but it is unknown how the would compare to the random query strategy in this case. This could potentially be an interesting query strategy to consider.

Two easier suggestions for future work would simply be to test to see if the results are reproducible for other data sets and types of classifiers. Bayesian Neural Networks would be interesting to investigate, since these networks would give a real probability of the different predicted classes given the data, unlike the networks examined in this thesis which output can merely be interpreted as probability estimates. This could potentially improve uncertainty based query strategies significantly. Other interesting models to consider would be recurrent neural networks.

We have observed smaller benefits from performing active learning with the CIFAR-10 data set. The reason why is not clear. It would, therefore, be valuable to use the same methods on different data sets in order to see how the results generalize. This holds true both for this chapter, as well as previous Chapter 3.

Much more can be researched about the topic of active learning, and we strongly encourage anyone to implement the suggested *Future Work*.

Bibliography

[1] Miroslav Kubat, Robert C. Holte, and Stan Matwin. "Machine learning for the detection of oil spills in satellite radar images". English (US). In: *Machine Learning* 30.2-3 (Dec. 1998), pp. 195–215. ISSN: 0885-6125.

[2] Aaron E. Maxwell, Timothy A. Warner, and Fang Fang. *Implementation of machine-learning classification in remote sensing: An applied review*. May 2018. DOI: 10.1080/01431161.2018.1433343.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[4] Rishab Gargeya and Theodore Leng. "Automated Identification of Diabetic Retinopathy Using Deep Learning". In: *Ophthalmology* 124.7 (2017), pp. 962–969. ISSN: 0161-6420. DOI: https://doi.org/10.1016/j.ophtha.2017.02.008. URL: http://www.sciencedirect.com/science/article/pii/S0161642016317742.

[5] Sharada P. Mohanty, David P. Hughes, and Marcel Salathé. "Using Deep Learning for Image-Based Plant Disease Detection". In: *Frontiers in Plant Science* 7 (2016), p. 1419. ISSN: 1664-462X. DOI: 10.3389/fpls.2016.01419. URL: https://www.frontiersin.org/article/10.3389/fpls.2016.01419.

[6] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. "Active Learning with Statistical Models". In: *CoRR* cs.AI/9603104 (1996). URL: https://arxiv.org/abs/cs/9603104.

[7] James T. Wilson, Frank Hutter, and Marc Peter Deisenroth. *Maximizing acquisition functions for Bayesian optimization*. 2018. arXiv: 1805.10196 [stat.ML].

[8] Ozan Sener and Silvio Savarese. "Active Learning for Convolutional Neural Networks: A Core-Set Approach". In: (Aug. 2017). URL: http://arxiv.org/abs/1708.00489.

[9] Burr Settles. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison, 2009.

[10] Neil Houlsby et al. "Bayesian Active Learning for Classification and Preference Learning". In: (Dec. 2011). URL: http://arxiv.org/abs/1112.5745.

[11] Daniel Gissin and Shai Shalev-Shwartz. "Discriminative Active Learning". In: (July 2019). URL: http://arxiv.org/abs/1907.06347.

[12] Angelos Katharopoulos and François Fleuret. “Not All Samples Are Created Equal: Deep Learning with Importance Sampling”. In: (Mar. 2018). URL: http://arxiv.org/abs/1803.00942.

[13] X. Zhu et al. “Active Learning from Data Streams”. In: *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. 2007, pp. 757–762.

[14] Saad Mohamad, M. Sayed Mouchaweh, and Hamid Bouchachia. “Active learning for classifying data streams with unknown number of classes”. In: *Neural Networks* 98 (Oct. 2017). DOI: 10.1016/j.neunet.2017.10.004.

[15] Yann LeCun and Corinna Cortes. “MNIST handwritten digit database”. In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[16] Han Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms”. In: *CoRR* abs/1708.07747 (2017). arXiv: 1708.07747. URL: http://arxiv.org/abs/1708.07747.

[17] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. In: *University of Toronto* (May 2012).

[18] Léon Bottou. “Stochastic gradient learning in neural networks”. In: *In Proceedings of Neuro-Nîmes. EC2*. 1991.

[19] William G. Cochran. *Sampling Techniques, 3rd Edition*. John Wiley and Sons, 1977. ISBN: 978-0-471-16240-7.

[20] *tf.keras.layers.Dense | TensorFlow Core v2.2.0*. June 2020. URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense.

[21] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: (Dec. 2014). URL: http://arxiv.org/abs/1412.6980.

[22] Martn Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[23] Sanjeev Arora, Nadav Cohen, and Elad Hazan. “On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization”. In: *CoRR* abs/1802.06509 (2018). arXiv: 1802.06509. URL: http://arxiv.org/abs/1802.06509.

[24] Christine Körner and Stefan Wrobel. “Multi-class Ensemble-Based Active Learning”. In: *Machine Learning: ECML 2006*. Ed. by Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 687–694. ISBN: 978-3-540-46056-5.

[25] Gareth James et al. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. URL: https://faculty.marshall.usc.edu/gareth-james/ISL/.

[26] Dong-Hyun Lee. “Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks”. In: *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)* (July 2013).

[27] Laurens Van Der Maaten and Geoffrey Hinton. *Visualizing Data using t-SNE*. Tech. rep. 2008, pp. 2579–2605.

[28] "Leave-One-Out Cross-Validation". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 600–601. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_469. URL: https://doi.org/10.1007/978-0-387-30164-8_469.

[29] *tf.keras.Model | TensorFlow Core v2.2.0*. June 2020. URL: https://www.tensorflow.org/api_docs/python/tf/keras/Model.

A

Appendix 1

A.1 Code Describing the Neural Network Architecture, and Parameters

A.1.1 $ANN1_m$ Type of Networks

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape = (28,28)),
    tf.keras.layers.Dense(m, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])
model.compile(optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])
```

A.1.2 $CNN1$ Type of Network

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
    kernel_initializer='he_uniform', padding='same',
    input_shape=(32, 32, 3)))
model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
    kernel_initializer='he_uniform', padding='same'))
model.add(tf.keras.layers.MaxPooling2D((2, 2)))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu',
    kernel_initializer='he_uniform', padding='same'))
model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu',
    kernel_initializer='he_uniform', padding='same'))
model.add(tf.keras.layers.MaxPooling2D((2, 2)))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu',
    kernel_initializer='he_uniform', padding='same'))
model.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu',
    kernel_initializer='he_uniform', padding='same'))
model.add(tf.keras.layers.MaxPooling2D((2, 2)))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(128, activation='relu',
```

```
    kernel_initializer='he_uniform'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(10, activation='softmax'))
opt = tf.keras.optimizers.SGD(lr=0.001, momentum=0.9)
model.compile(
    optimizer=opt,
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

A.1.3 Parameters for Training and Evaluation

Tensorflow's standard settings for the hyperparameters were used when training and evaluating the network, for the `model.fit()` function [29]:

```
batch_size=32
validation_split = 0
validation_data=None
Shuffle = True
class_weight=None
sample_weight=None
initial_epoch=0
steps_per_epoch=None
validation_steps=None
validation_freq=1
max_queue_size=10
workers=1
use_multiprocessing=False
```

Tensorflow's standard settings for evaluation [29], for the `model.evaluate()` function:

```
x=None
y=None
batch_size=None
verbose=1
sample_weight=None
steps=None
callbacks=None
max_queue_size=10
workers=1
use_multiprocessing=False
```

A.2 Plots for Section 3.3.1

Some plots in the substudy were not included, since they showed the same results when a different amount of hidden neurons were used in the networks. Figures A.1 and A.2 shows the train accuracy, train loss and mean informativeness measure $\bar{I}_B^M$ while training incrementally and cumulatively. The Figure shows these metrics for the $ANN1_{100}$ and $ANN1_{1000}$ networks. The corresponding Figure in the report

is Figure 3.4, which shows the development of the metrics during training for the $ANN1_{10}$ network.

A.3 Comparison Between the Incremental and Cumulative Method With the Random Query Strategy

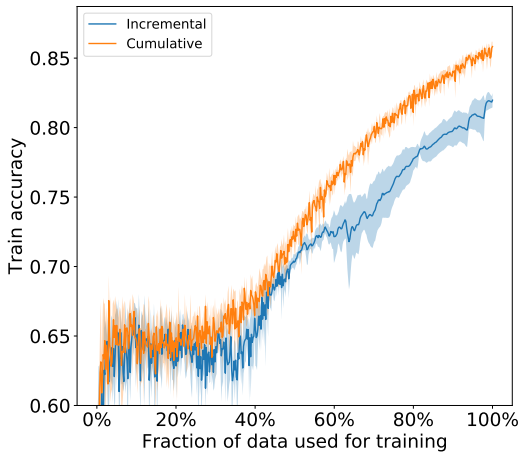
In Section 3.3.1 we claimed that networks trained with the incremental method and the cumulative method perform similarly if the data is randomly sampled. This is shown in Figure A.3. The figure shows that the performance of the networks trained with the different training methods are similar. There might be a small benefit to using the incremental training method when using neural networks with 10 hidden neurons, but we believe this difference would be smaller given more reruns. The reason for incremental and cumulative being different should mainly be due to how Tensorflow implements stochastic gradient descent.

A.4 Plots for Section 4.3.3

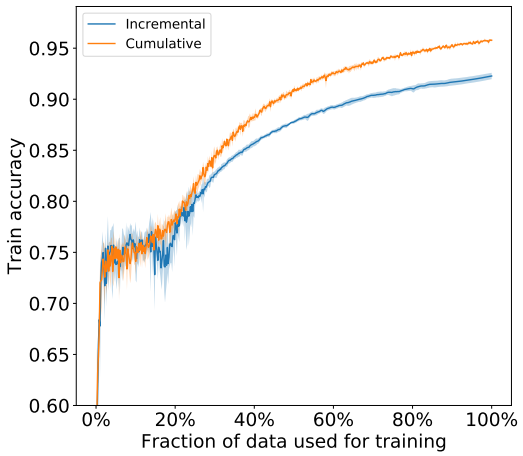
Data on what classes the different query strategies selected for a randomly initialised network can be seen in Figure A.4.

A.5 Plots for Section 4.3.6

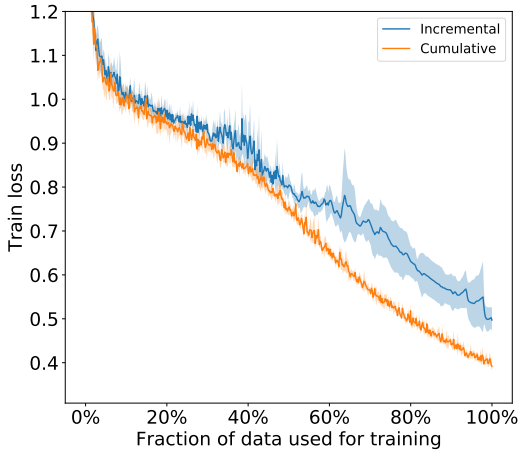
Plots for the improved inproved performance, with gradual pseudo-labeling, for different thresholds ξ, for different fractions p, for the two data sets can be seen in the Figures A.5 and onward.



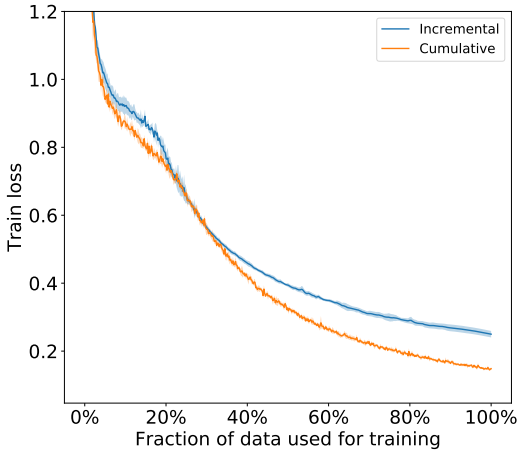
(a) Train accuracy Fashion-MNIST



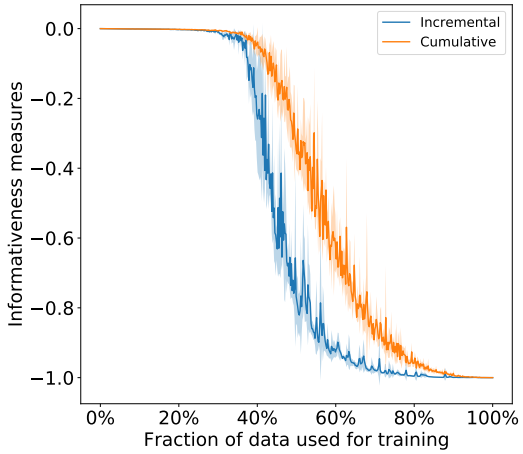
(b) Train accuracy MNIST



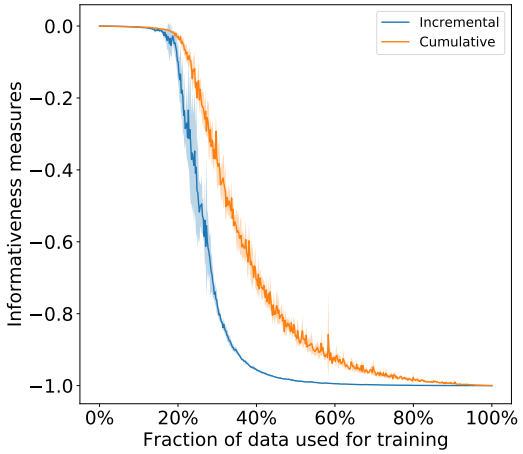
(c) Train loss Fashion-MNIST



(d) Train loss MNIST

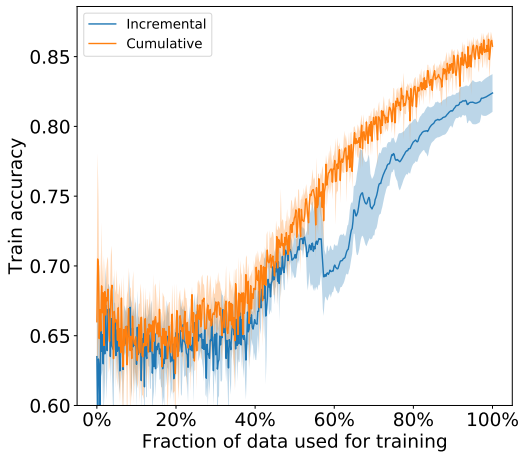


(e) Evolution of the mean informativeness measures Fashion-MNIST

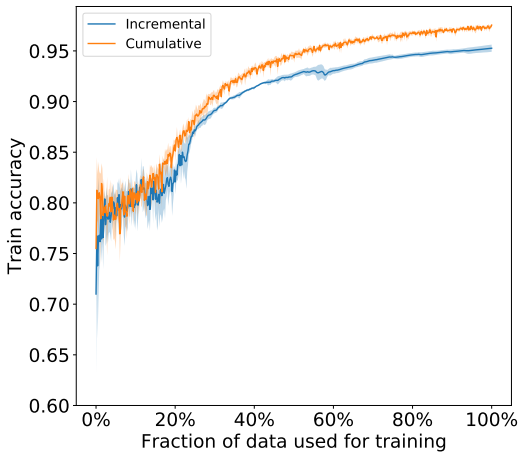


(f) Evolution of the mean informativeness measures MNIST 5 reruns.

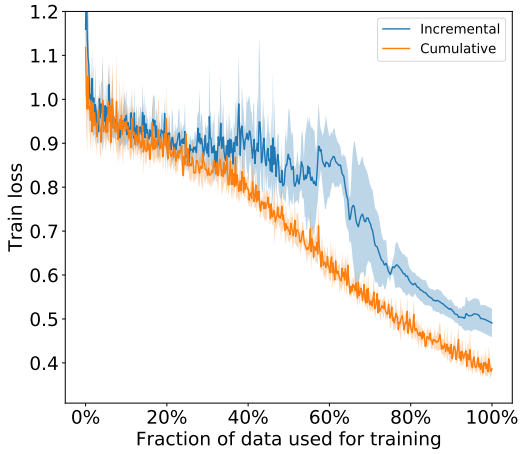
Figure A.1: Shows different metrics for the two training methods using an $ANN1_{100}$ network with the margin query strategy for both the MNIST and Fashion-MNIST data sets. The corresponding test accuracy can be seen in Figure 3.2.



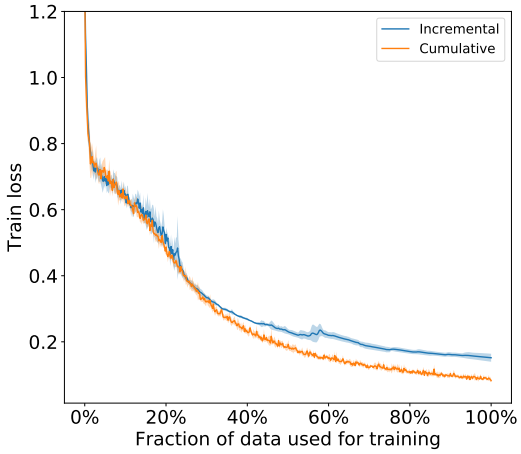
(a) Train accuracy Fashion-MNIST



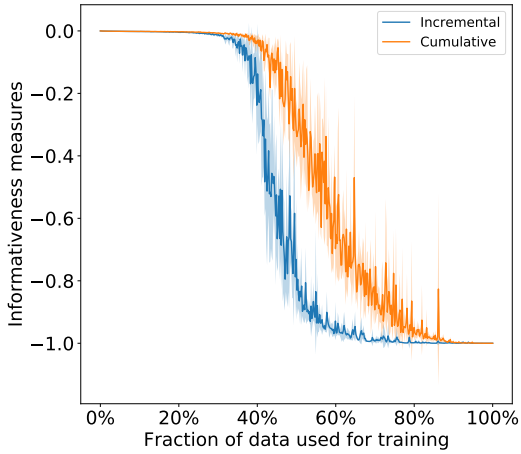
(b) Train accuracy MNIST



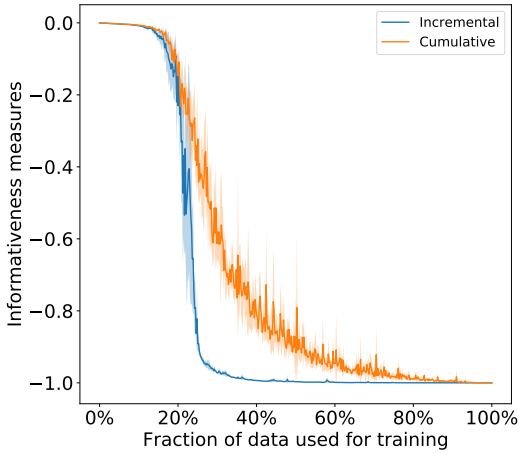
(c) Train loss Fashion-MNIST



(d) Train loss MNIST

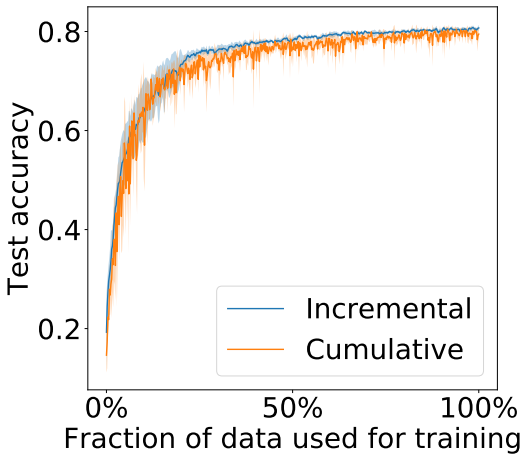


(e) Evolution of the mean informativeness measures Fashion-MNIST

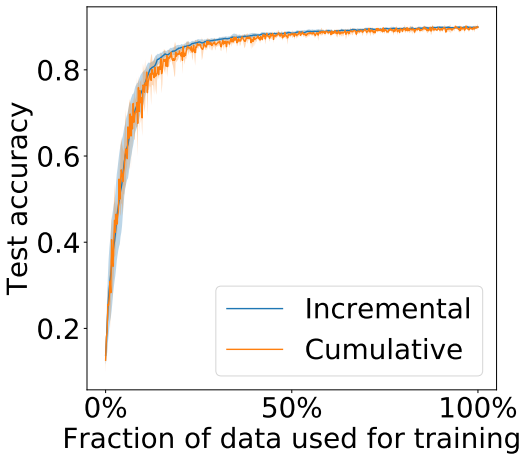


(f) Evolution of the mean informativeness measures MNIST 5 reruns.

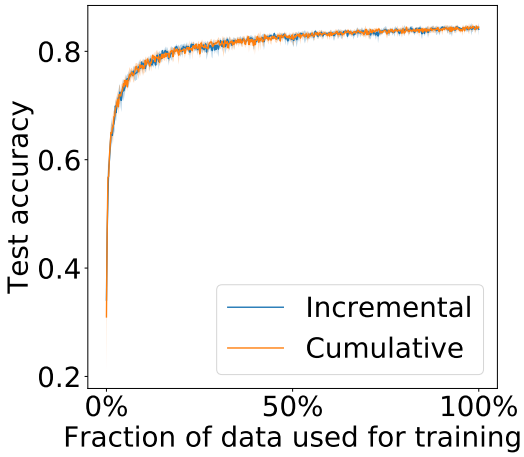
Figure A.2: Shows different metrics for the two training methods using an $ANN1_{1000}$ network with the margin query strategy for both the MNIST and Fashion-MNIST data sets. The corresponding test accuracy can be seen in Figure 3.3.



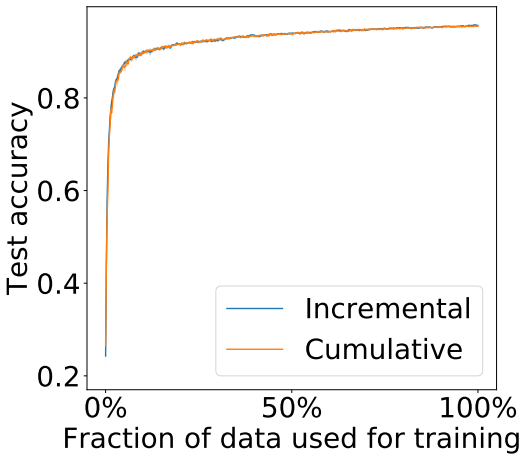
(a) 10 hidden neurons, Fashion MNIST



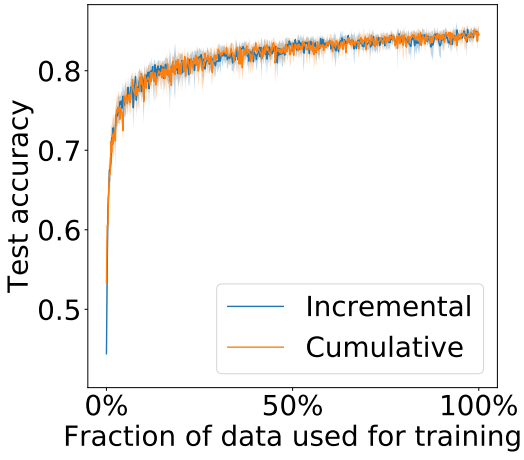
(b) 10 hidden neurons, MNIST



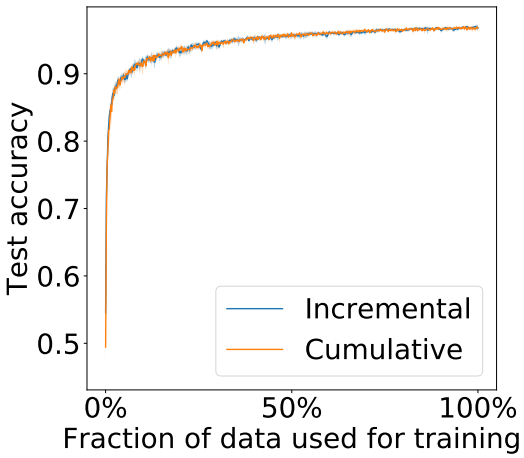
(c) 100 hidden neurons, Fashion MNIST



(d) 100 hidden neurons, MNIST

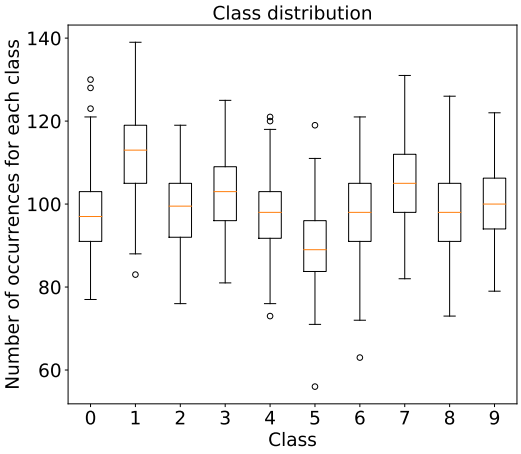


(e) 1000 hidden neurons, Fashion MNIST

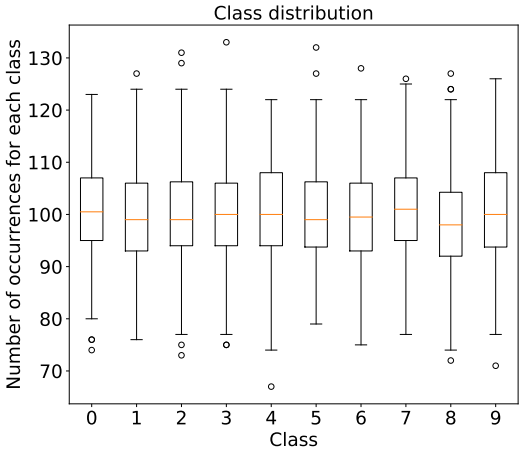


(f) 1000 hidden neurons, MNIST

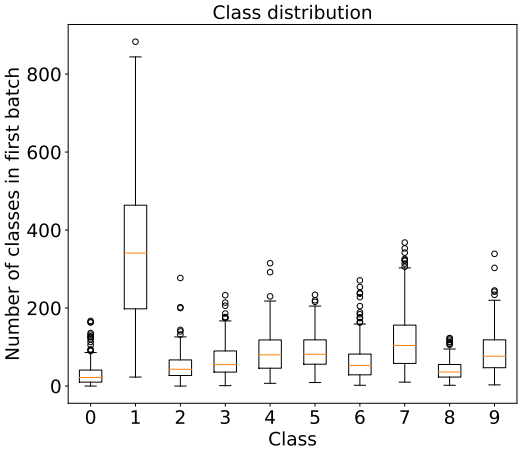
Figure A.3: Shows a comparison in performance of networks trained with the cumulative method to networks trained with the incremental method. The samples are selected with the random query strategy. The results shown are averaged over 5 reruns. There is no notable difference in performance between the training methods.



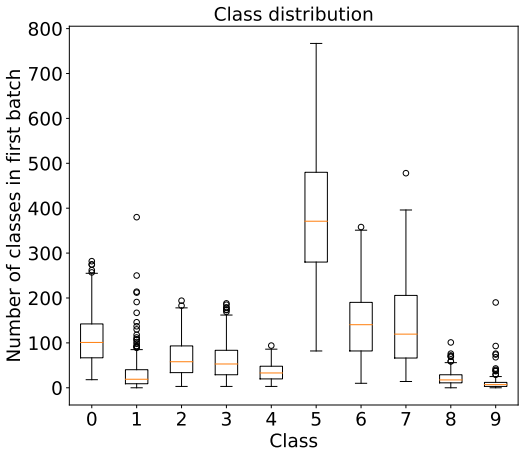
(a) Sampled with random query strategy using the data set MNIST



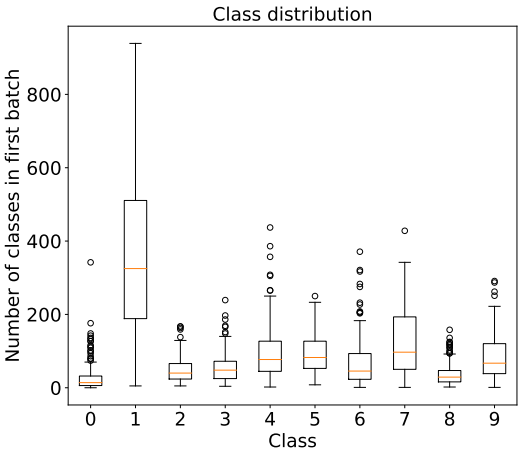
(b) Sampled with random query strategy using the data set Fashion-MNIST



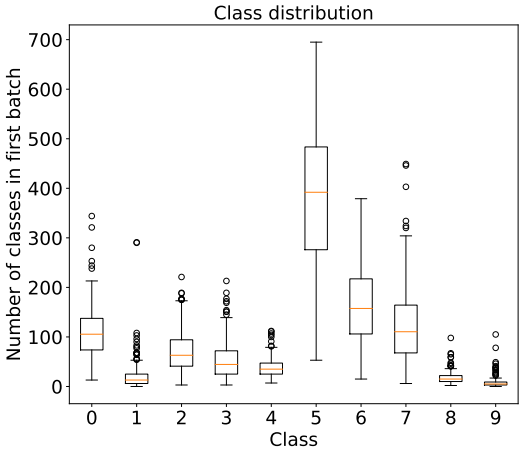
(c) Sampled with least confident query strategy using the data set MNIST



(d) Sampled with least confident query strategy using the data set Fashion-MNIST

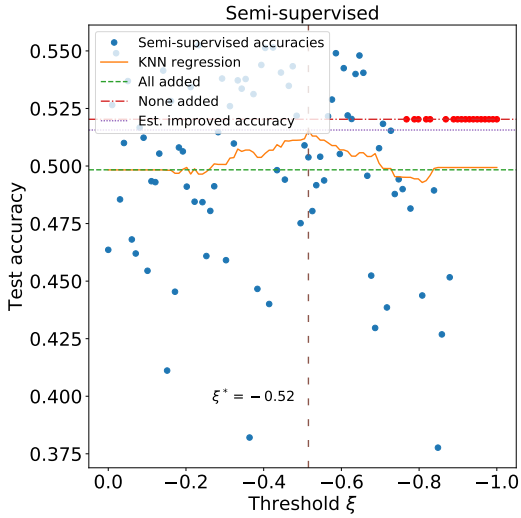


(e) Sampled with least squares query strategy using the data set MNIST

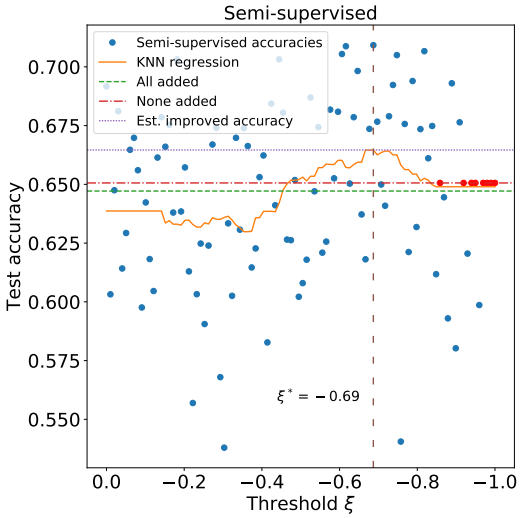


(f) Sampled with least squares query strategy using the data set Fashion-MNIST

Figure A.4: The label distribution obtained by sampling the first 1000 points with a a variety of query strategies. 200 reruns

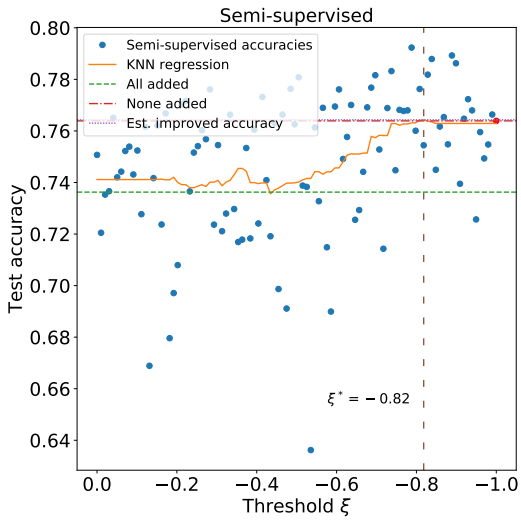


500 ($\frac{5}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the Fashion-MNIST data set.

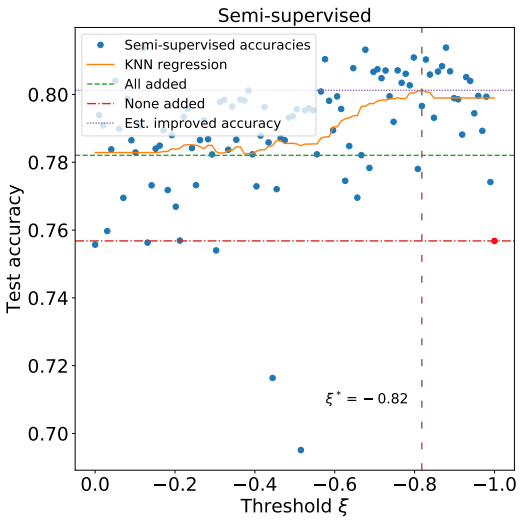


1000 ($\frac{10}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the Fashion-MNIST data set.

Figure A.5: Shows the KNN-regression for two different fractions p of the data used for labeling. 1 rerun of was performed for every fraction p.

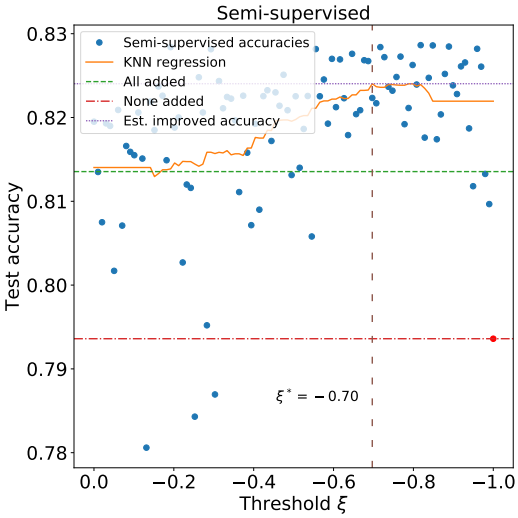


2500 ($\frac{25}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the Fashion-MNIST data set.

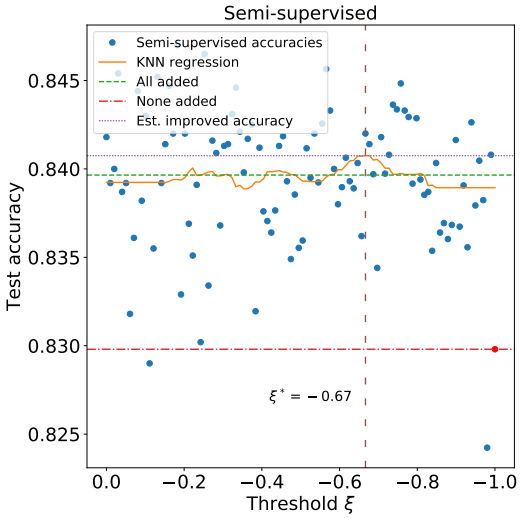


5000 ($\frac{50}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the Fashion-MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. 1 rerun of was performed for every fraction p.

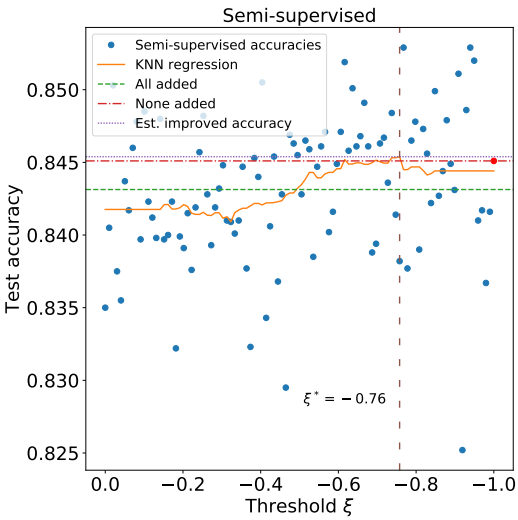


10000 $(\frac{100}{600})$ points labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the Fashion-MNIST data set.

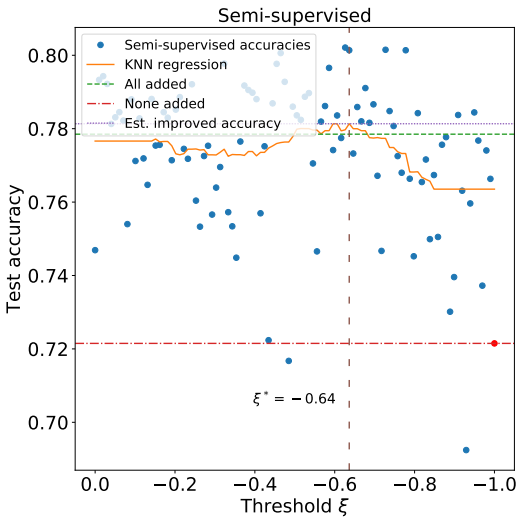


20000 $(\frac{200}{600})$ points labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the Fashion-MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. 1 rerun of was performed for every fraction p.

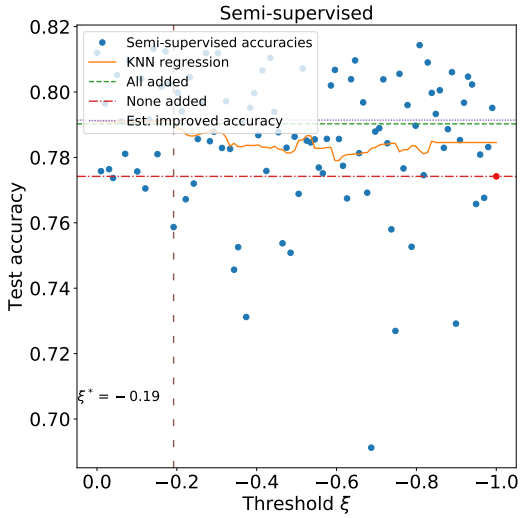


50000 $(\frac{500}{600})$ points labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the Fashion-MNIST data set.

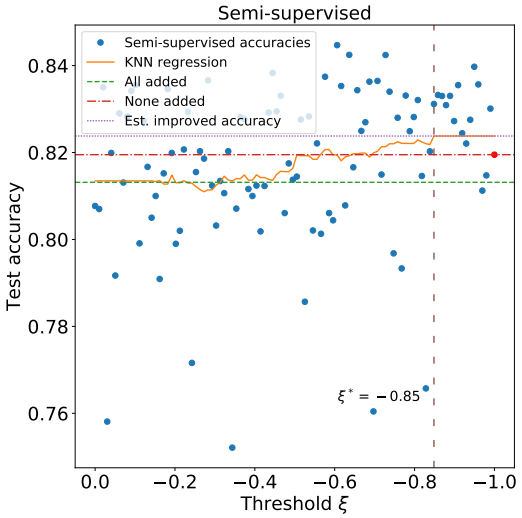


500 $(\frac{5}{600})$ points labeled using 5 epochs, an $ANN1_{1000}$ network with the margin query strategy and the Fashion-MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. 1 rerun of was performed for every fraction p.

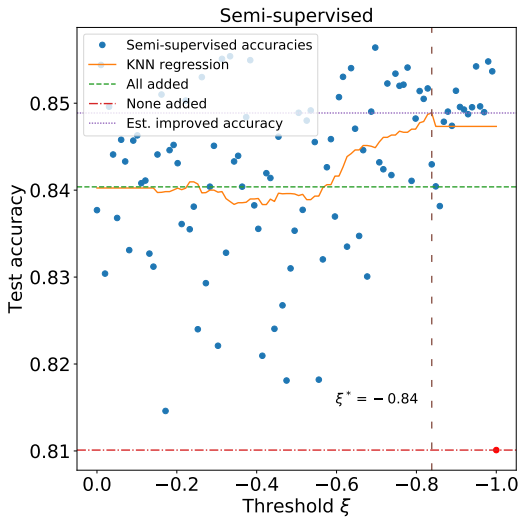


1000 ($\frac{10}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the margin query strategy and the Fashion-MNIST data set.

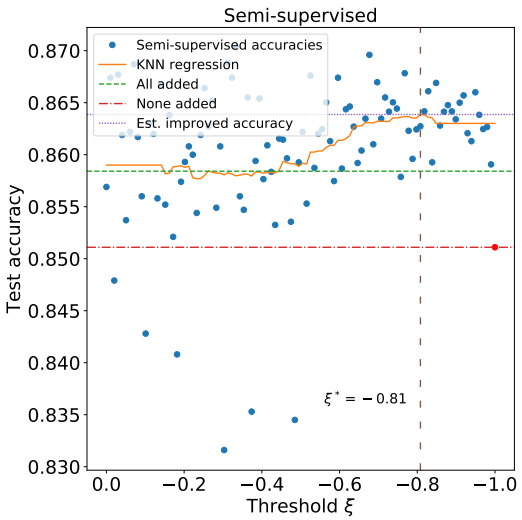


2500 ($\frac{25}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the margin query strategy and the Fashion-MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. 1 rerun of was performed for every fraction p.

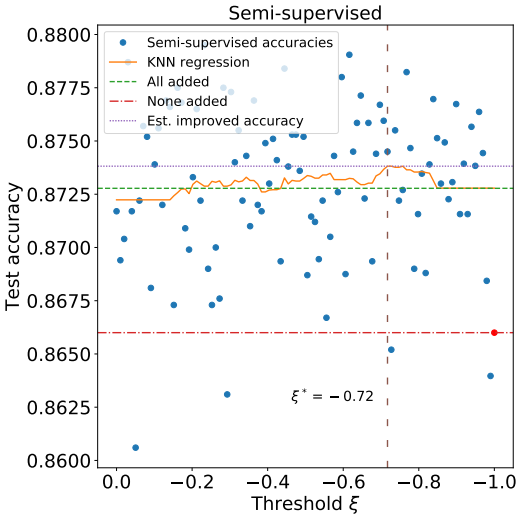


5000 ($\frac{50}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the margin query strategy and the Fashion-MNIST data set.

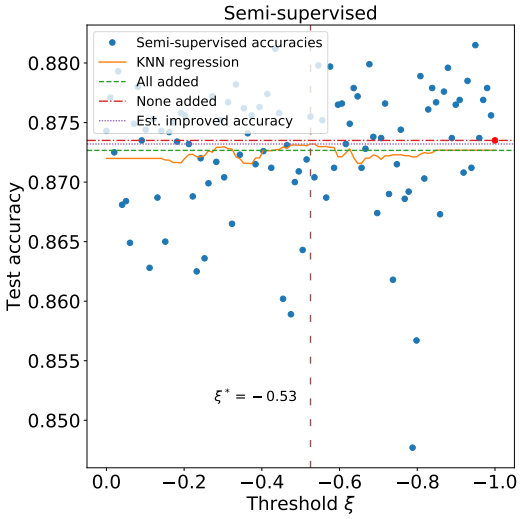


10000 ($\frac{100}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the margin query strategy and the Fashion-MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. 1 rerun of was performed for every fraction p.

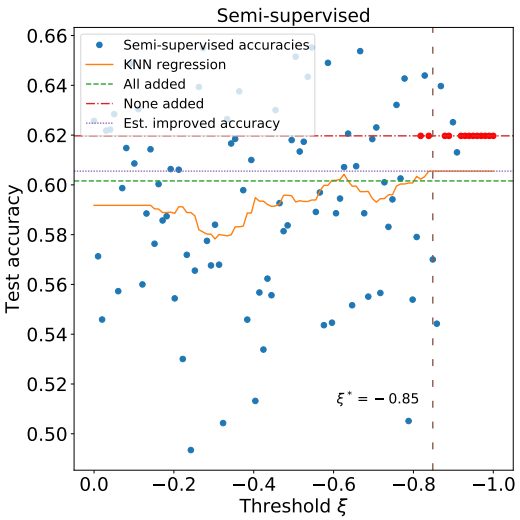


20000 ($\frac{200}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the margin query strategy and the Fashion-MNIST data set.

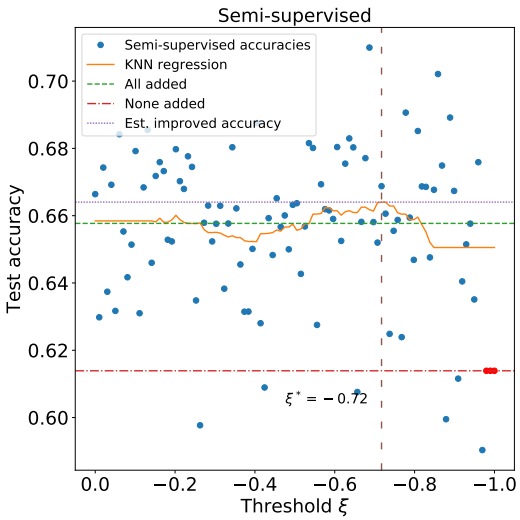


50000 ($\frac{500}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the margin query strategy and the Fashion-MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. 1 rerun of was performed for every fraction p.

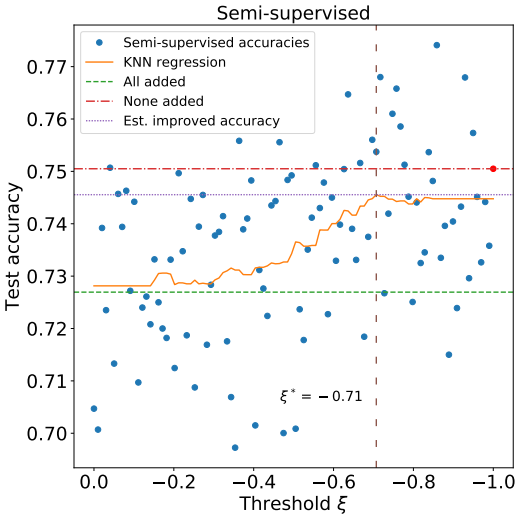


500 ($\frac{5}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the random query strategy and the Fashion-MNIST data set.

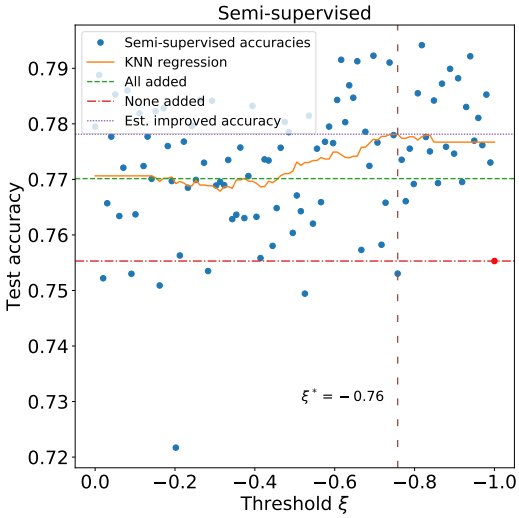


1000 ($\frac{10}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the random query strategy and the Fashion-MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. 1 rerun of was performed for every fraction p.

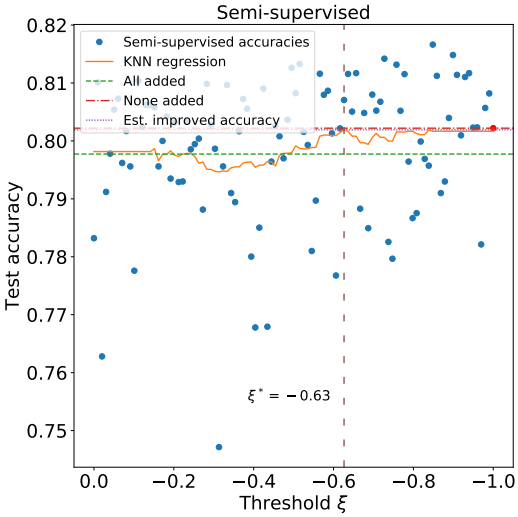


2500 ($\frac{25}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the random query strategy and the Fashion-MNIST data set.

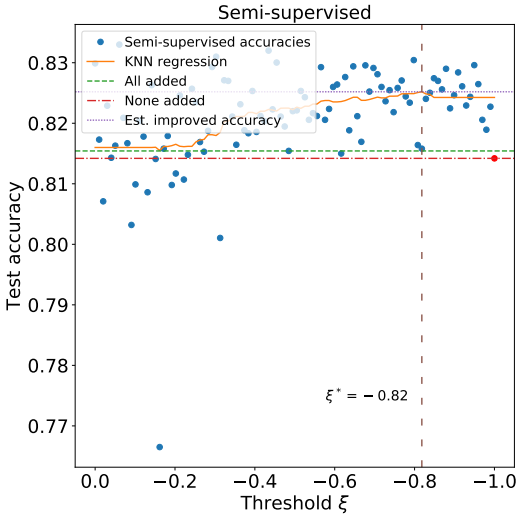


5000 ($\frac{50}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the random query strategy and the Fashion-MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. 1 rerun of was performed for every fraction p.

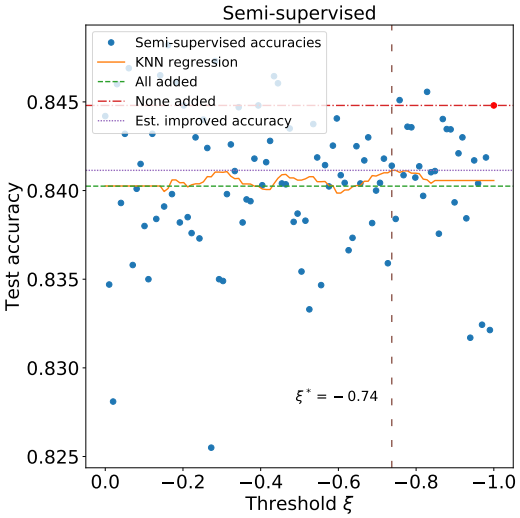


10000 ($\frac{100}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the random query strategy and the Fashion-MNIST data set.

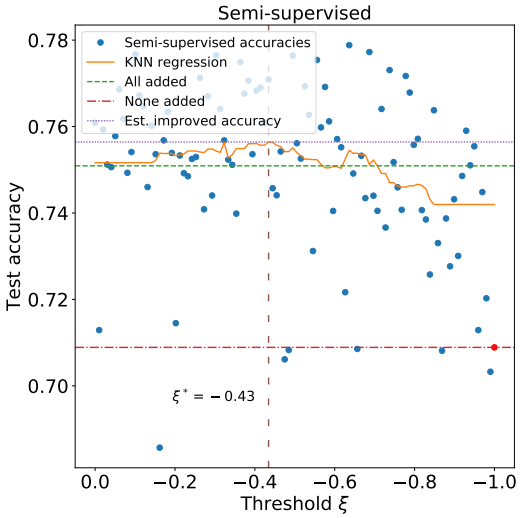


20000 ($\frac{200}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the random query strategy and the Fashion-MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. 1 rerun of was performed for every fraction p.

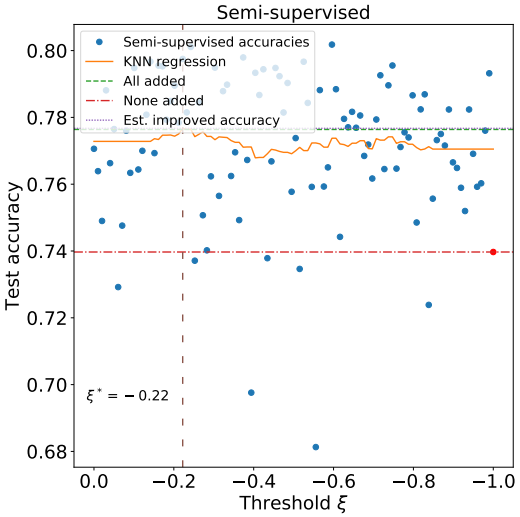


50000 ($\frac{500}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the random query strategy and the Fashion-MNIST data set.

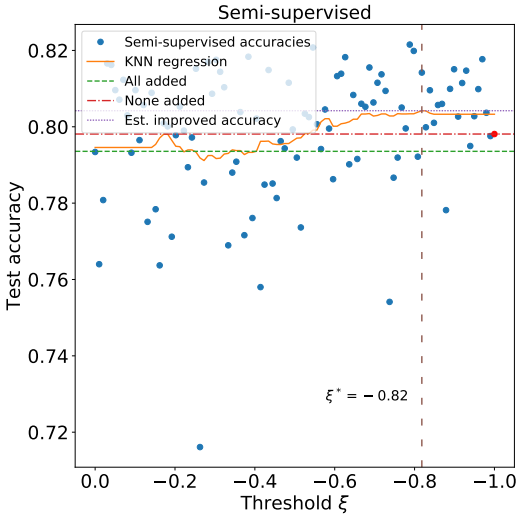


500 ($\frac{5}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the random query strategy and the Fashion-MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. 1 rerun of was performed for every fraction p.

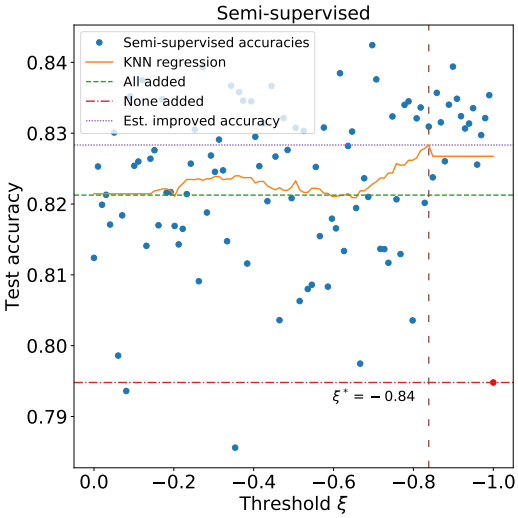


1000 ($\frac{10}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the random query strategy and the Fashion-MNIST data set.

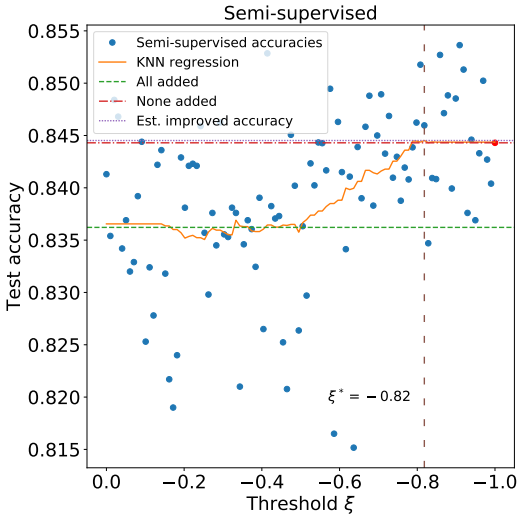


2500 ($\frac{25}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the random query strategy and the Fashion-MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. 1 rerun of was performed for every fraction p.

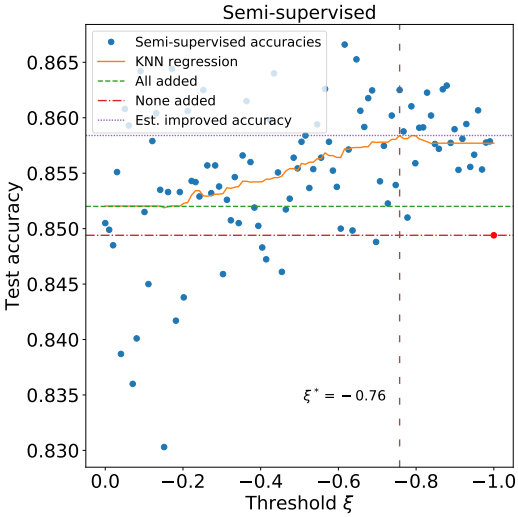


5000 ($\frac{50}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the random query strategy and the Fashion-MNIST data set.

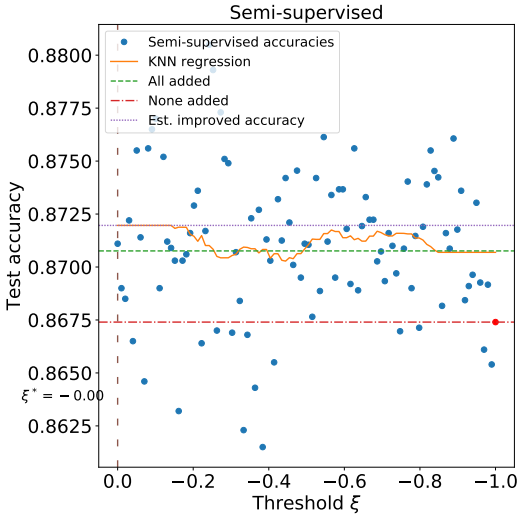


10000 ($\frac{100}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the random query strategy and the Fashion-MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. 1 rerun of was performed for every fraction p.

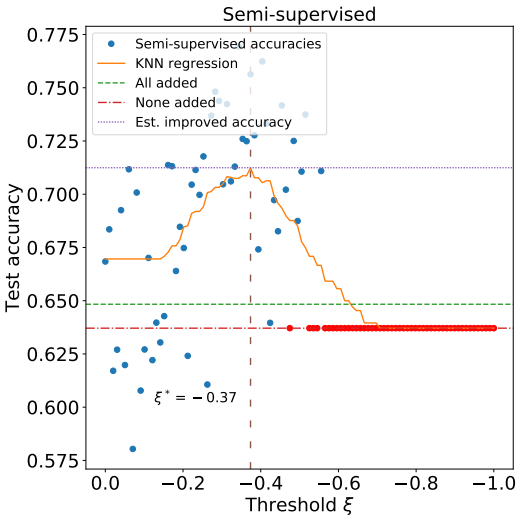


20000 ($\frac{200}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the random query strategy and the Fashion-MNIST data set.

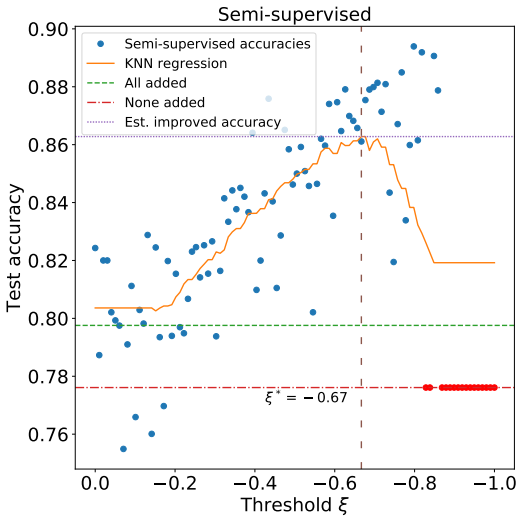


50000 ($\frac{500}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the random query strategy and the Fashion-MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. 1 rerun of was performed for every fraction p.

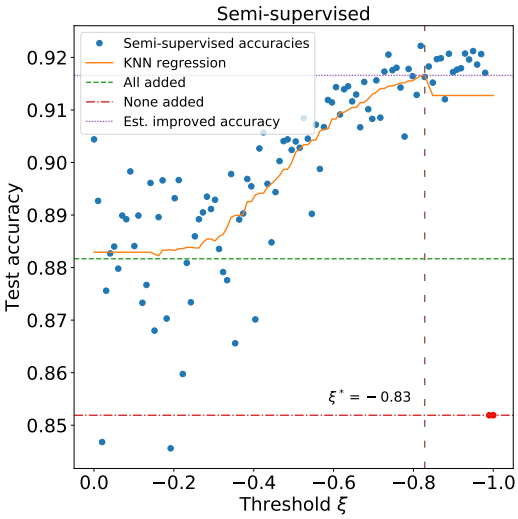


500 ($\frac{5}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the MNIST data set.

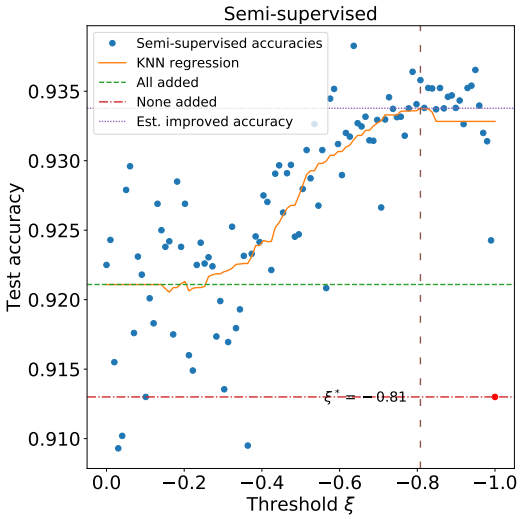


1000 points labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. Note that of was performed for every fraction p.

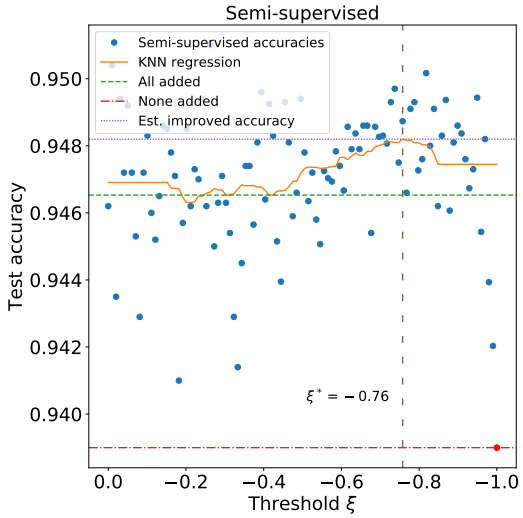


2500 ($\frac{25}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the MNIST data set.

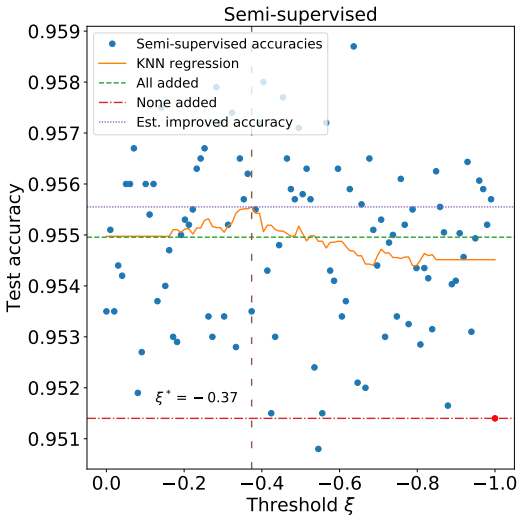


5000 points labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. Note that of was performed for every fraction p.

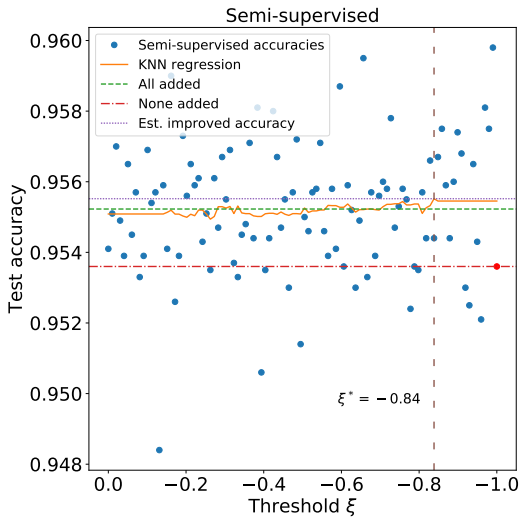


10000 ($\frac{100}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the MNIST data set.

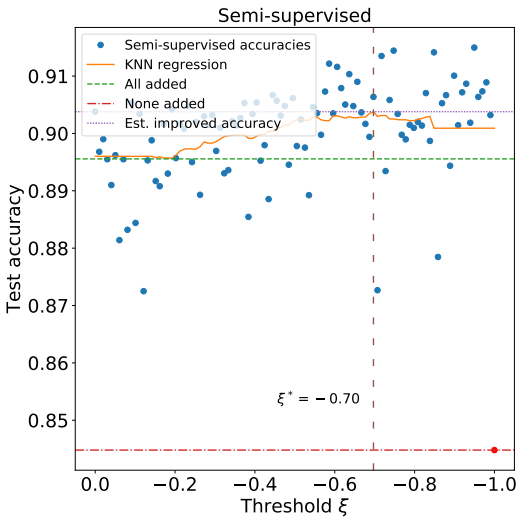


20000 points labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. Note that of was performed for every fraction p.

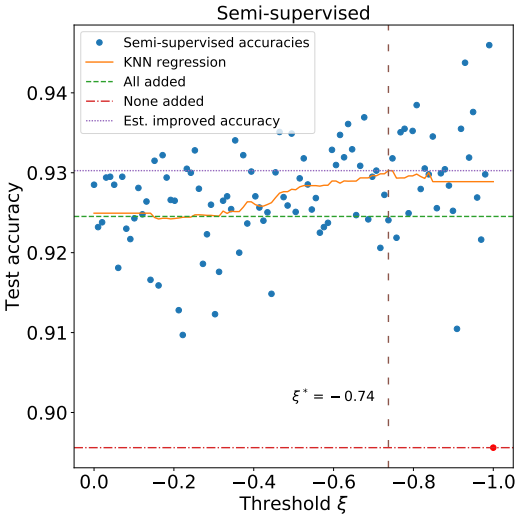


50000 ($\frac{500}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the margin query strategy and the MNIST data set.

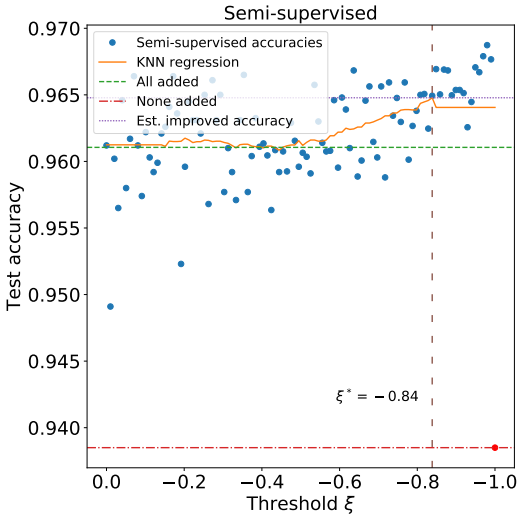


500 points labeled using 5 epoch, an $ANN1_{100}$ network swith the margin 0query strategy and the MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. Note that of was performed for every fraction p.

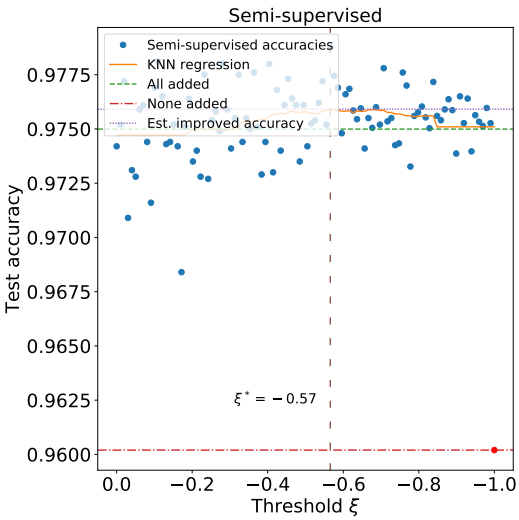


1000 ($\frac{10}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the margin query strategy and the MNIST data set.

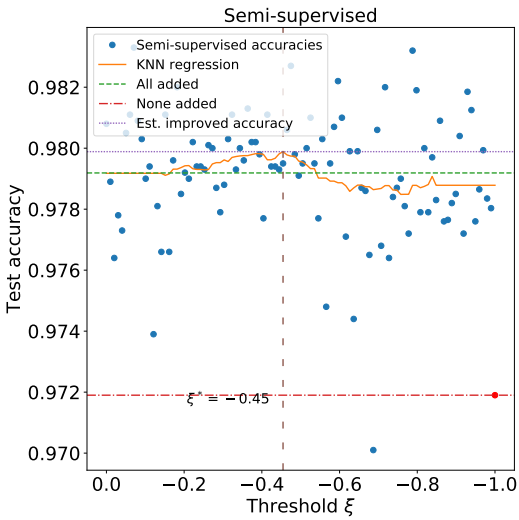


2500 points labeled using 5 epoch, an $ANN1_{100}$ network swith the margin 0query strategy and the MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. Note that of was performed for every fraction p.

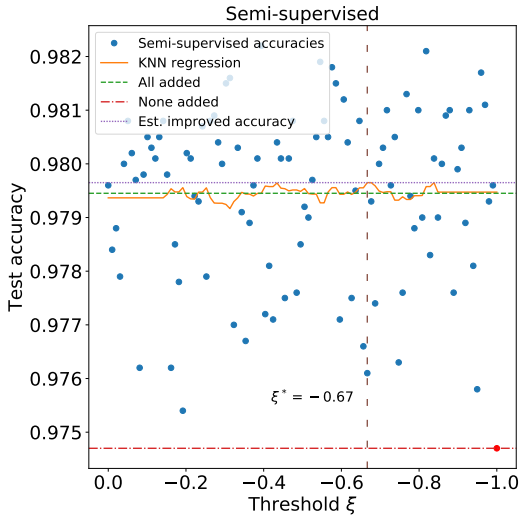


5000 ($\frac{50}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the margin query strategy and the MNIST data set.

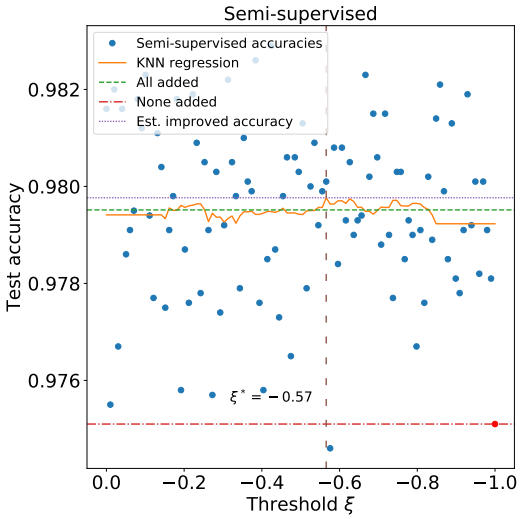


10000 points labeled using 5 epoch, an $ANN1_{100}$ network swith the margin 0query strategy and the MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. Note that of was performed for every fraction p.

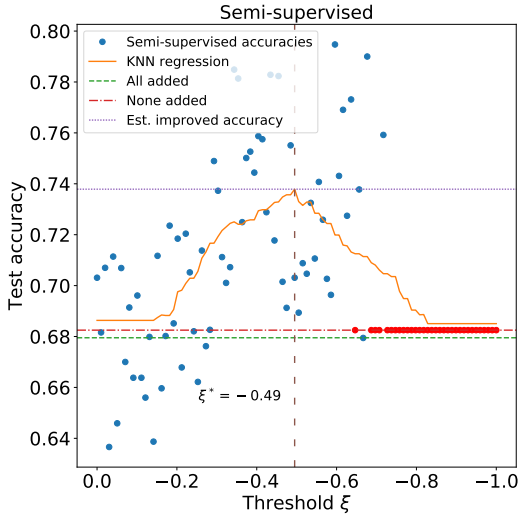


20000 $(\frac{200}{600})$ points labeled using 5 epochs, an $ANN1_{1000}$ network with the margin query strategy and the MNIST data set.

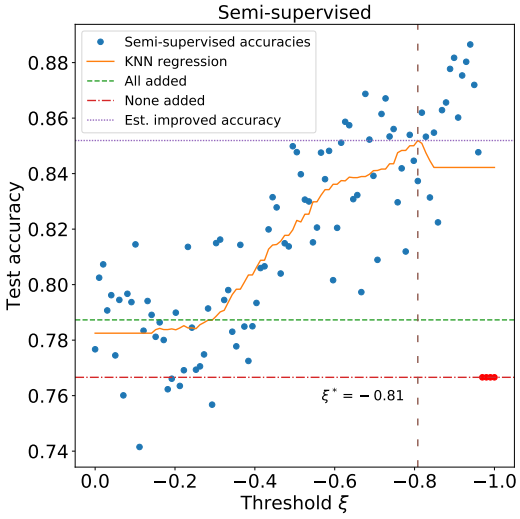


50000 points labeled using 5 epoch, an $ANN1_{100}$ network swith the margin 0query strategy and the MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. Note that of was performed for every fraction p.

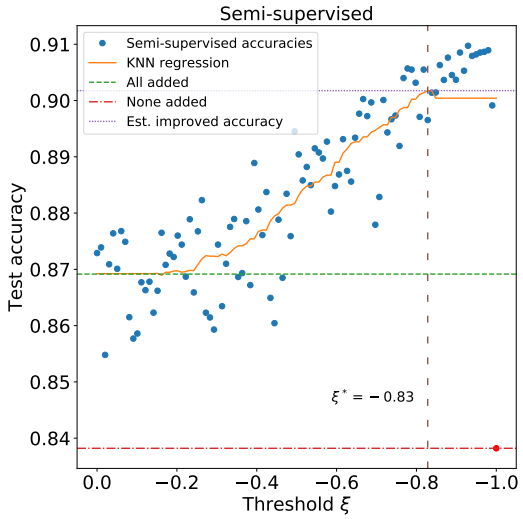


500 $(\frac{5}{600})$ points labeled using 1 epoch, an $ANN1_{100}$ network with the random query strategy and the MNIST data set.

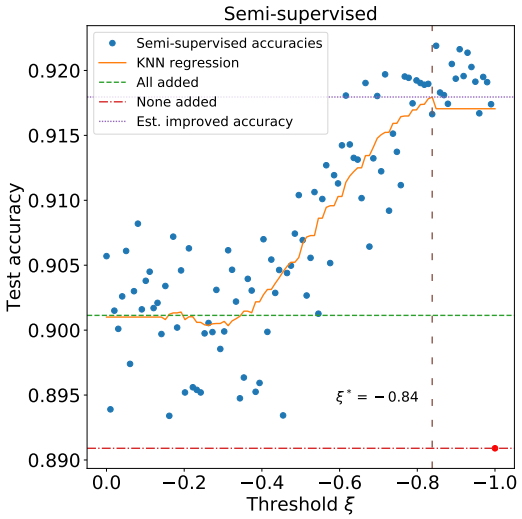


1000 points labeled using 1 epoch, an $ANN1_{100}$ network with the random query strategy and the MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. Note that of was performed for every fraction p.

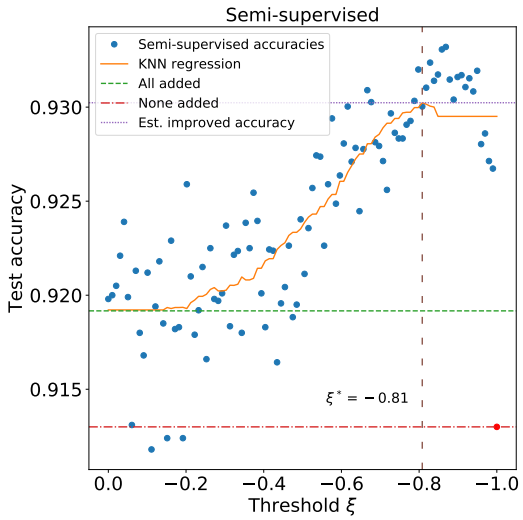


2500 ($\frac{25}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the random query strategy and the MNIST data set.

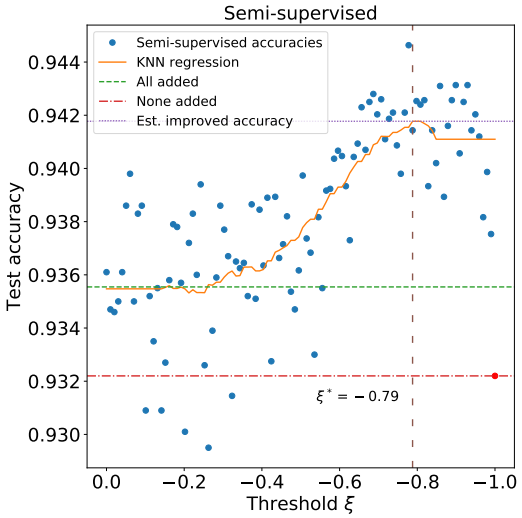


5000 points labeled using 1 epoch, an $ANN1_{100}$ network with the random query strategy and the MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. Note that of was performed for every fraction p.

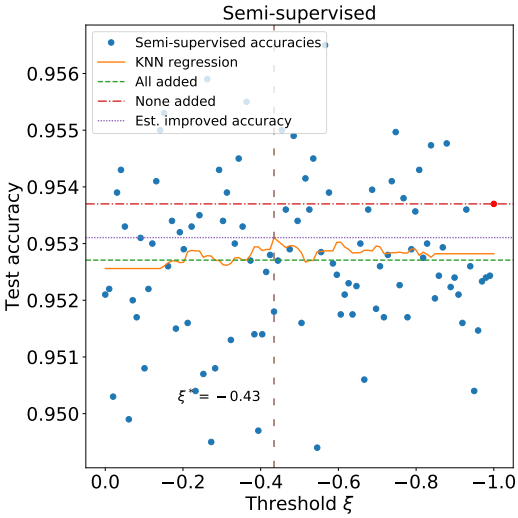


10000 ($\frac{100}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the random query strategy and the MNIST data set.

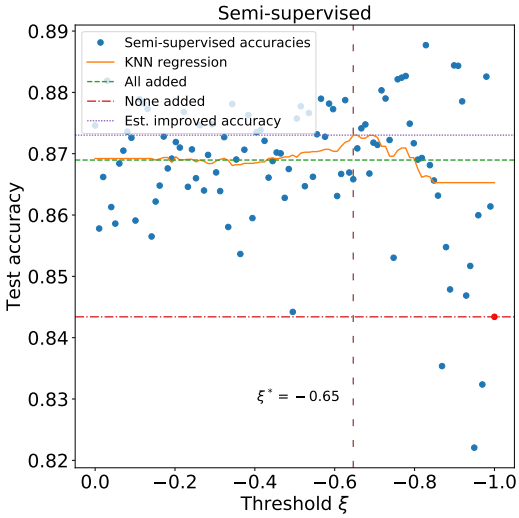


20000 points labeled using 1 epoch, an $ANN1_{100}$ network with the random query strategy and the MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. Note that of was performed for every fraction p.

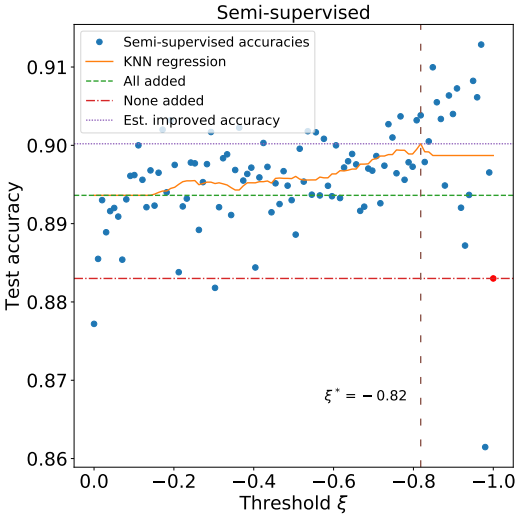


50000 ($\frac{500}{600}$) points labeled using 1 epoch, an $ANN1_{100}$ network with the random query strategy and the MNIST data set.

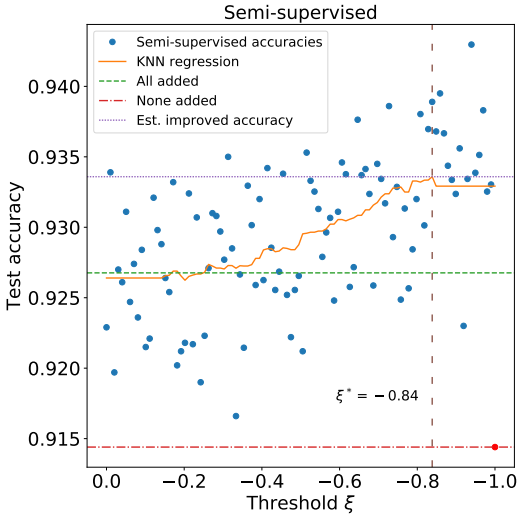


500 points labeled using 5 epoch, an $ANN1_{100}$ network swith the random 0query strategy and the MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. Note that of was performed for every fraction p.

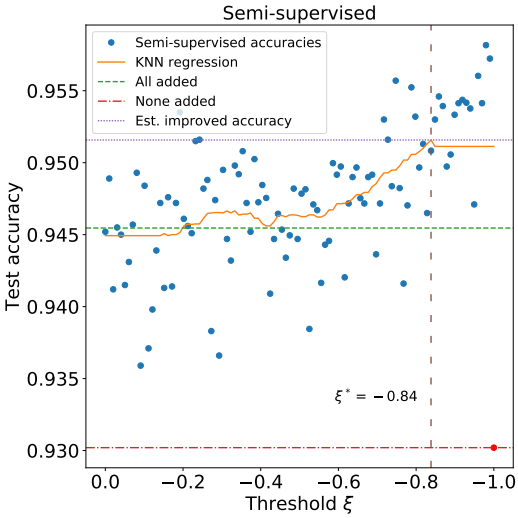


1000 ($\frac{10}{600}$) points labeled using 5 epochs, an $ANN1_{1000}$ network with the random query strategy and the MNIST data set.

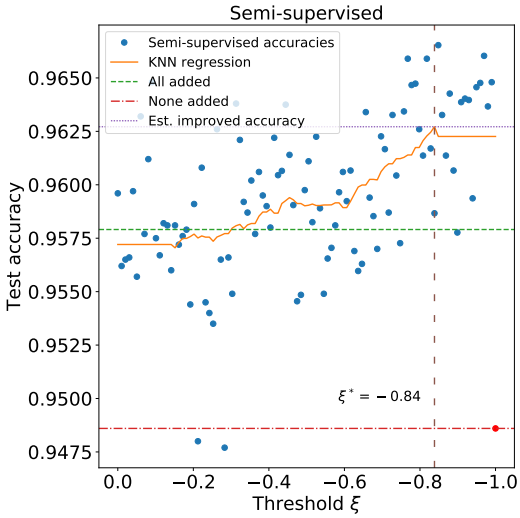


2500 points labeled using 5 epoch, an $ANN1_{100}$ network swith the random 0query strategy and the MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. Note that of was performed for every fraction p.

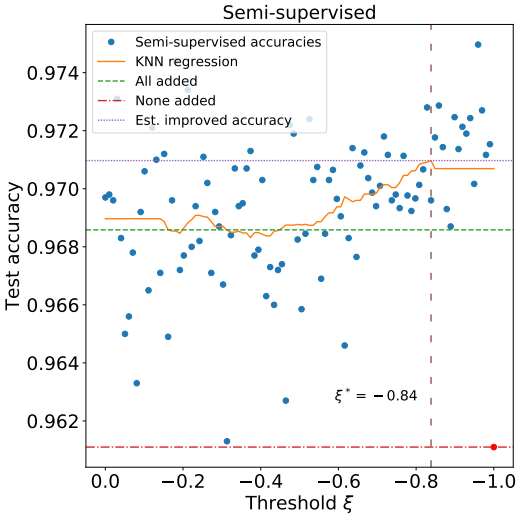


5000 $(\frac{50}{600})$ points labeled using 5 epochs, an $ANN1_{1000}$ network with the random query strategy and the MNIST data set.

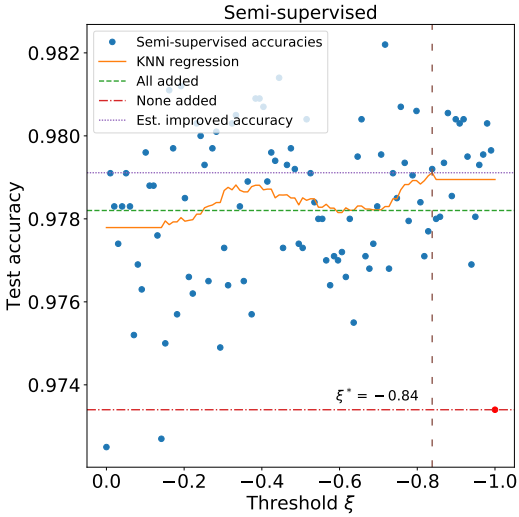


10000 points labeled using 5 epoch, an $ANN1_{100}$ network swith the random 0query strategy and the MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. Note that of was performed for every fraction p.



20000 $(\frac{200}{600})$ points labeled using 5 epochs, an $ANN1_{1000}$ network with the random query strategy and the MNIST data set.



50000 points labeled using 5 epoch, an $ANN1_{100}$ network swith the random 0query strategy and the MNIST data set.

Shows the KNN-regression for two different fractions p of the data used for labeling. Note that of was performed for every fraction p.