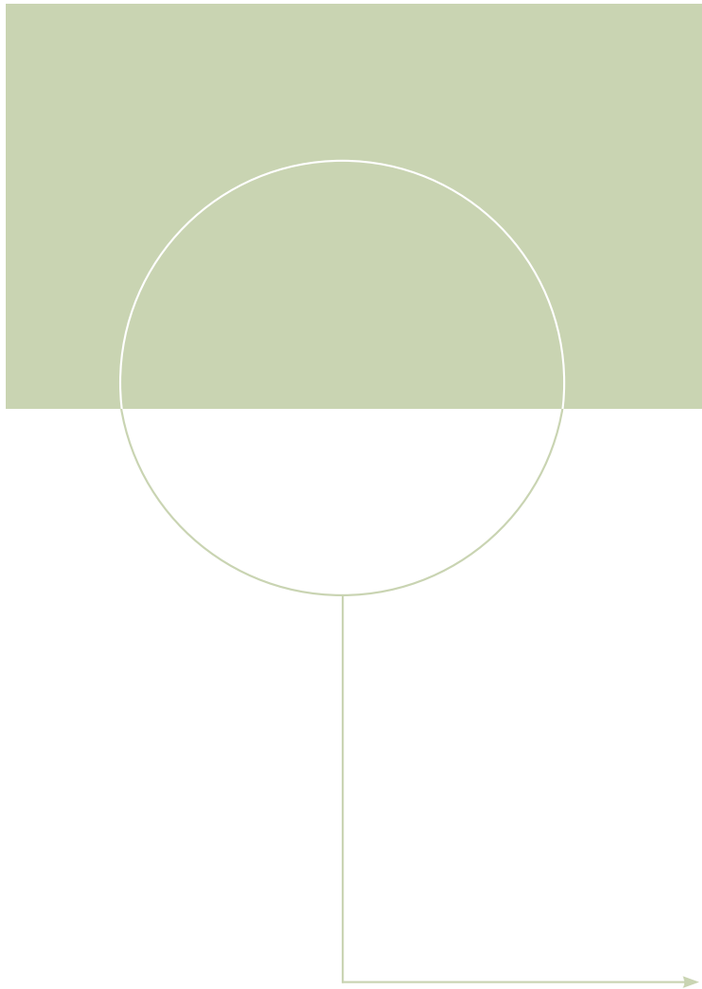


Master's thesis

2019

Johan Phan

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Engineering Cybernetics

Johan Phan

Dual Active Sampling

A batch-mode active learning method

June 2019

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU

Norwegian University of
Science and Technology

Dual Active Sampling

A batch-mode active learning method

Johan Phan

Cybernetics and Robotics
Submission date: June 2019
Supervisor: Massimiliano Ruocco
Co-supervisor: Francesco Scibilia

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

Recently, Convolutional Neural Networks (CNNs) have shown **unprecedented success** in the field of computer vision, especially on challenging image classification tasks by relying on a universal approach, i.e., training a deep model on a massive dataset of supervised examples. While unlabeled data are often an abundant resource, collecting a large set of labeled data, on the other hand, is very expensive and often requires considerable human efforts. One way to ease out this is to effectively select and label **highly informative data** from a pool of unlabeled data (i.e., active learning).
This thesis proposed a new method of batch-mode active learning, Dual Active Sampling(DAS), that has a **comparable performance** or **even slightly outperforms** existing **state of the art** approaches. Compared to other methods, DAS is considerably simpler in terms of implementation and computational complexity while being highly parallelizable. The method is based on a simple assumption, if two deep neural networks (DNNs) of the same structure and trained on the same dataset give significantly different output for a given sample, then that particular sample should be picked for additional training.

In the first part of the experiment, the performance of DAS in terms of accuracy per *number of labeled training data* was compared with different state of the art methods, where it slightly outperformed all of them on multiple datasets. Furthermore, DAS seems to have a smart selection behavior where it was able to select more samples from the more challenging classes and less from the easier one.

Sammendrag

Nylig har CNN vist enestående suksess innen datasyn, spesielt for utfordrende bildeklassifiseringsoppgaver ved å basere seg på en universal tilnærming, mao. trene en dyp model på et massivt datasett med veiledende eksempler. Mens "unlabeled" data finnes i store mengder, er det dyrt og ofte tidkrevende å samle store mengder med "labeled" data. En måte å optimalisere dette arbeidet er å effektivt velge og sortere de mest informative dataene fra en gruppe med usorterte data (mao. aktiv læring).

Denne oppgaven foreslår en ny metode innen Batch-mode aktiv læring, Dual Active sampling (DAS), som har en sammenlignbar ytelse, eller til og med utkonkurrer eksisterende state of art tilnærminger. Sammenlignet med andre methoder er DAS betraktelig enklere i implementering og beregningsmessig kompleksitet samtidig som den har høy paralleliserbarhet. Metoden baserer seg på en enkel antakelse, hvis to dype nevrale nettverk (DNN) av sammme struktur trenes på samme datasett, men gir vesentlige forskjellig output, så burde det bestemte datapunktet bli tatt ut for ytterligere trening.

I den første delen av eksperimentet ble ytelsen til DAS sammenlignet med forskjellige state of the art metoder, hvor den utkonkurrerte alle metodene på flere datasett. Videre klarer DAS å vise en smart utvelgelsesadferd hvor den velger flere datapunkter fra de mer utfordrende klassene og færre fra de enklere klassene.

Preface

This project would not have been possible without the guidance and help from my supervisor Massimiliano Ruocco and my co-supervisor Francesco Scibilia. Moreover, I would also like to thank the Norwegian Open Artificial Intelligence Lab for providing me with the GPU-server, which I have made intensive use during my experiment. Finally, I want to give a thank to the Norwegian A.I society who had provided me an opportunity to publish my work during this thesis.

Johan Phan

Trondheim, June 10, 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the last few years, Deep Convolutional Neural Networks(CNNs) have completely dominated the field of **image recognition** and proven itself to be a versatile and robust tool for achieving top performance on many tasks. However, as a data-driven method, it requires a considerable amount of labeled data in order to provide a good result. More importantly, the performance of CNNs are in most cases, better with more data. This **data-hungry** characteristics of CNNs has led to a constant need to collect more data, even though data labeling is a time consuming and expensive task.

Aiming at improving the performance of an existing model by incrementally selecting and labeling the most **suitable/informative** unlabeled samples, **Active Learning (AL)** has been well studied over recent decades, and most of the early work can be found in [1].

In the field of image recognition using CNNs, several attempts to develop an effective active learning strategy have been made, notably **core-set sampling** [2], where the active learner selects the samples that has the most representative features by treating the problem as a metric k-center problem [3]. However, considering the complex and unpredictable nature of DNNs, nearly all of the state of the art methods in this field are often depended on the extracted output information from the networks as the selecting criteria, e.g. [4], [5], [2]. While this approach has been proven to be effective, it has made the traditional **serial query AL**, i.e., queries and re-train one at a time, less desirable. One of the main reasons for this is that DNNs are often slow to train and computationally expensive. For this reason, **batch-mode AL**, i.e. agent queries multiple samples at once, has become a much more suitable approach.

Batch mode active learning allows multiple samples to be queried and labeled at once without waiting for the model to re-train on each sample, thus making it possible for parallel labeling on multiple workstations. However, one of the main challenges with this approach compared to "serial query AL" is the lack of constant feedback from the model on each selection, which often leads to overlapping of information between samples within the selected batch [1], i.e., the majority of them shares similar features.

This paper proposed a new method of batch-mode AL, **Dual Active Sampling (DAS)**, which is based on a simple assumption, if two DNNs of the same structure and trained on the same dataset give significantly different output for a given sample, then that particular sample should be picked for additional training. While other state of the art methods in this field typically require an intensive computational effort [2] or having a complicated structure with a lot of tunable parameter [4], the proposed method is simpler to implement, has shorter running time and, managed to get improved results on multiple datasets without the need to make any change on the classification model. On top of that, DAS does not require tuning in order to perform well and is compatible with other popular techniques in deep learning such as random drop-out, batch normalization, random cropping, and other data argumentation methods.

1.1 Background and Motivation

Studies around active learning have started even before the rise of deep learning. In [6], a basic framework of active learning on support vector machine(SVM) has been addressed and thoughtfully studied. However, during the last decade, following the success of deep learning and the abundance of unlabeled data, thanks to big data and the Internet, the interest in active learning has become larger than ever. Active Learning(AL) referred to algorithms that can effectively collect data by automatically make adjustments based on previously collected data and the current state of the model in order to accelerate machine learning.

Serial-query active learning [1] is one of the most common methods in active learning, where queries are selected one at a time. This method has been intensively studied in the previous decade and has achieved a high degree of success in several fields, most notably in natural language processing (NLP). However, in the case of image recognition problems, such a method is very impractical for real-world applications where human's labeling speed usually is much faster than the active learner query speed. This slow query speed is because modern deep learning models require typically a signification amount of time to train while the complex structure of deep learning force the active learner to continually make

adjustments based on the current state of the model. Therefore, batch-mode active learning has been proposed as a solution to this problem. In batch-mode active learning, the learner is allowed to query instances in groups. By doing so, the active learner would be able to keep up with the labeling speed of the oracle, thus reduce the annotation cost.

However, batch-mode active learning still has to deal with many difficult challenges. The most prominent challenge of this approach is to avoid the overlapping of information. In batch-mode active learning, the active learner often becomes unable to diversify its selection since it only wants to select the "best" instances in every selection, which usually turned out to be very similar to each other. One of the main reason for this problem is the lack of feedback on the effect of each selection on the performance of the model.

1.2 Goals and Research Questions

Goal: The goal of this thesis is to propose a new method in the field of active learning and study the strengths and limitations of the method. This consists of quantitative testing of the proposed method and compared the performance to other state of the art methods in the field. This goal is further divided into three research questions:

Research question 1 *Does the proposed method perform better than random-selection and other state of the art baseline?*

Research question 2 *Why does the proposed method achieve better results than other methods?*

Research question 3 *What is the best way to implement the method with regards to parameter choice?*

1.3 Report Structure

The rest of the report is organized as follows. Chapter 2 is an introduction to some of the basic concepts in deep learning and active learning. Chapter 3 is going to give a review of related works in active learning for image classification. Chapter 4 and 5 outlines what DAS is and how it was implemented in the experiments. In chapters 5 & 6, the methodology and experimental result will be shown, evaluated and discussed. Finally, the conclusion of this thesis can be found in chapter 7.

Chapter 2

Background Theory

In this chapter, different basic concepts in the field of deep learning and active learning will be introduced to make it easier for the reader to understand the proposed method. Furthermore, different techniques and methods that have been used during the experiment will also be mentioned in this chapter.

Since the main focus of this paper is about active learning, most of the basic knowledge related to deep learning will only be briefly explained . In order to fully understand this work, [7] is highly recommended to read. The book provided all basic knowledge related to deep learning in detail.

2.1 Introduction of deep learning

Deep learning (also known as deep structured learning or hierarchical learning) is a sub-field of machine learning, where the method solves problems by optimizing the weight of an Artificial Neural Network (ANN). While the idea of using ANNs in machine learning has been around since the end of the second world war, the true golden age of deep learning has just begun in the last decade. Among the classes of deep neural networks, Convolutional Neural Network is one of the most widely used methods in Deep Learning. With the introduction of CNNs, deep learning has become the dominant player in the field of object detection and image recognition.

CNNs

As one of the most significant contributing factors to the rise of Deep learning, Convolutional neural network(CNN) [8] are current the most widely used type of neural networks. The interest in CNN started with Alexnet(2012)[9], which scored a top-5 test-error of 15.3% in the ImageNet competition[10], 10.8 percentage points better than that of the runner up. Since then, its introduction, CNNs has evolved a lot, with deeper and more advanced architecture.

The basic concept of CNN can be shown in Figure 2.1.

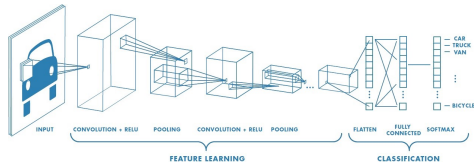


Figure 2.1: A basic convolutional neural network architecture [11]

To put it briefly, convolutional neural networks are a sub-set of neural networks. They are also made up of neurons that have learnable weights and biases. In CNNs, the networks will process the input train data through a series of convolution layers with filters (kernels) and pooling layer, then through multiple fully connected layers in order to perform the classification with the help of functions like softmax at the end.[9] The term convolution in this context refers to the mathematical combination of two functions to produce a third function.

Convolutional layers: The convolution layer comprises of a set of independent filters. Figures 2.2 shown how a 7×7 input passes get convolved by a 3x3 kernel filter. In this example, for the input pixel, there will be about 9 multiplications and 8 additions required to produce the corresponding output. In this way, the spatial relationship between pixels would be preserved, thus made it easier for feature extraction. Figure 2.3 shown how a combination of multiple convolutional layers was able to extract different features from an image, therefore make it become trainable and classifiable by a feed forward network.

Pooling layers: After the convolutional layer is the pooling layer which is used to reduce the numbers of parameters and prove to be very useful, for example, in cases when the input image is large. There many types of pooling layers

- Local pooling combines small clusters, typically 2 x 2.
- Global pooling acts on all the neurons of the layer.

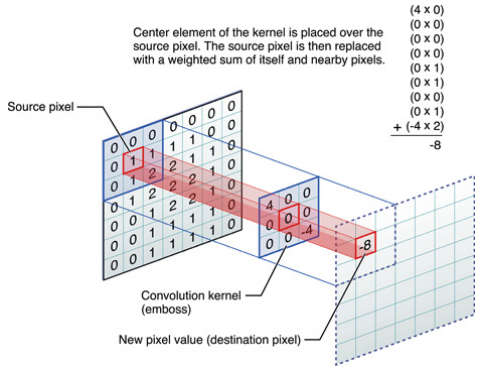


Figure 2.2: An illustration of a 2D convolution. [12]

- Max pooling uses the maximum value from each cluster of neurons at the prior layer.
- Average pooling, which is similar to max pooling but uses average value instead of max value.

Fully connected layers: is just a layer of regular neural nets, where each neuron is fully connected to all neurons in the previous layer. In fully connected layers, the neurons within a layer do not share any connection with the other. This layers are shown in the right part of Figure 2.4

Soft-max function: is an activation function that turns an array of numbers into an array of probabilities that sum to one. Soft-max is often used in deep learning, to convert the non-normalized output to a probability distribution over predicted output classes. [14]

VGG-net

The VGG-net is a deep convolutional neural network architecture for object recognition developed and trained by the Visual Geometry Group at Oxford.[15] "It was demonstrated that the representation depth is beneficial for the classification accuracy, and that state-of-the-art performance on the ImageNet challenge dataset can be achieved using a conventional CNN architecture with substantially increased depth. " (VGG-net authors[15]

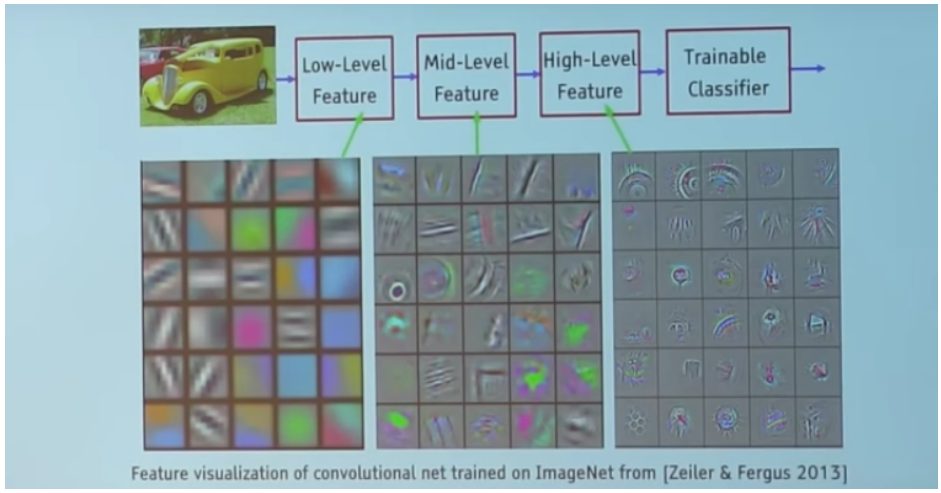


Figure 2.3: An illustration of feature extraction in CNNs. [13]

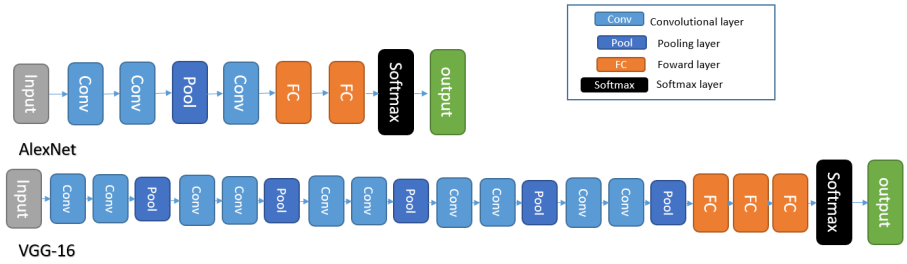


Figure 2.4: Architecture Comparition between Alexnet and VGG-16 [11]

Among the VGG-nets, VGG-16 is the most widely used version. Figure 2.4 illustrates the architecture of VGG-16. As the name implies, Vgg-16 consists of 16 layers in total. The original architecture takes three channels(RGB) as input, and the convolution layers use only 3×3 pixel window with stride 1 (the filter convolves 1 extra square around the input point) [7]). Max-pooling is performed over a 2×2 pixel window, with stride 2. Furthermore, there are three Fully-Connected (FC) layers follow the convolutional layers: the first two have 4096 channels each and the third contains 1000 output channels (one for each class). The final layer is the soft-max layer.

2.2 Gradient descent algorithm

2.2.1 Gradient descent

Since deep learning is about tuning every node in the network with the right weight, a good optimization strategy is always needed to effectively speed up the tuning process. The most common optimization strategy in the field of deep learning is gradient descent, where the goal of this is to minimize a loss function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. Equation 2.1, sums up the entire Gradient Descent algorithm in a single line. In this equation, Θ^0 and Θ^1 are the current and next position, α is often call the learning rate, while ∇ indicate the direction of the descent.

$$\Theta^1 = \Theta^0 - \alpha \nabla J(\Theta) \tag{2.1}$$

Loss function

The function that needs to be minimized or maximized by an optimizer is called the objective function or criterion. In many cases, the objective function can also be called: cost function, loss function, or error function. Figure 2.5 shows how a gradient descent algorithm works on a loss function.

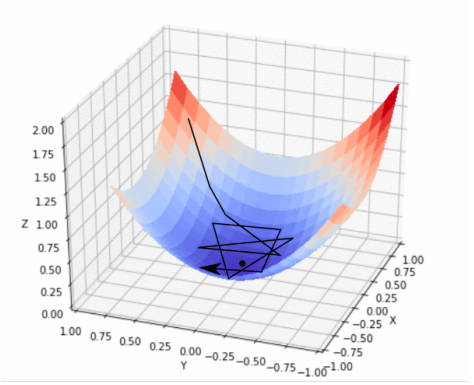


Figure 2.5: An illustration of gradient descent algorithm on a loss function. [12]

2.2.2 Stochastic gradient descent and Batch Gradient Descent

Stochastic Gradient Descent (SGD) is one of the most basic form of gradient descent algorithm. It randomly select a sample to evaluate the gradient and repeat the process. By doing so, it can estimate the a gradients of the big data set from a much smaller one. Since SGD uses only a single example, it works but is very noisy and takes long time to converge to the optima.

Opposed to SGD, **Batch Gradient Descent(BGD)** evaluate the gradient of the whole training-set at once. BGD is extremely computational expensive, especially on complex deep learning architecture with numerous of training data.

2.2.3 Mini-Batch Gradient Descent

Mini-batch gradient descent is a type of gradient descent algorithm that splits the training dataset into small batches. Thereafter, the cost-function and the gradient cost function (and therefore gradient) is averaged(or summed) over the batches and the learning process then happens on each mini-batch. The main advantage and disadvantage of this method is stated as follow:

Advantage

- The model update frequency is higher than batch gradient descent (gradient descent on the whole training data set at once) which allows for a more robust convergence, avoiding local minima.
- The batching improves the efficiency by not having all training data in memory and algorithm implementations.
- Updating by batches is a more computationally efficient process compared to stochastic gradient descent.

Disadvantage

- It requires one extra parameter "batch-size" to tune.
- It won't truly converge. On each iteration, the learning step may go back and forth due to the noise. Therefore, it wanders around the minimum region but never converges [16]

- It needs more training step compare to batch training

Figure 2.6 illustrates how SGD, BGD, and mini-batch gradient descent converges to a minimum point.

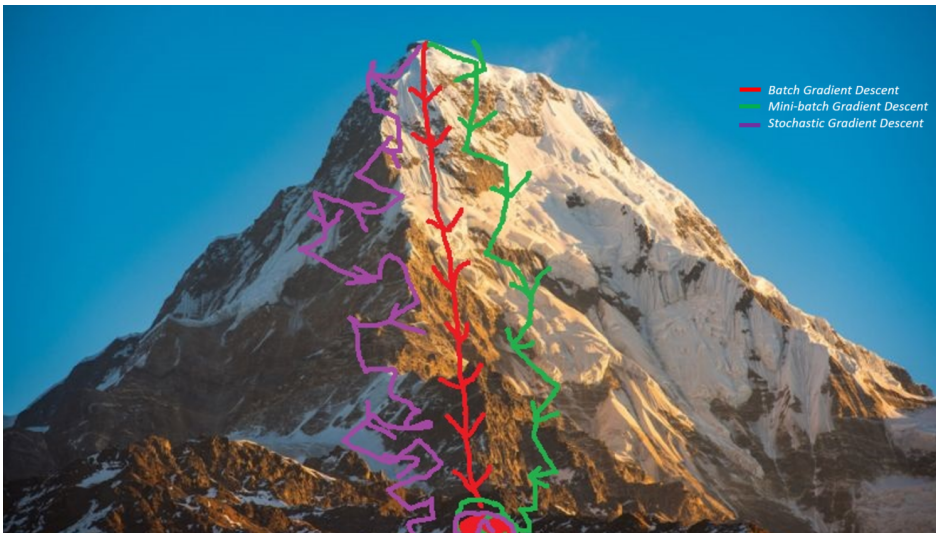


Figure 2.6: Trajectory towards local minimum [16]

2.2.4 Batch-normalization

Training deep neural networks with multiple layers is a challenging task. According to [17]: "Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities." Batch normalization is a technique that was designed to overcome this problem. It standardizes the inputs to a layer for each mini-batch, which lead to a more stable learning process and dramatically reducing the number of training epochs required.

Adam optimizer

One of the most widely used optimizers in deep learning is Adam.[18] The algorithm was introduced in 2015 and Adam is not an acronym and is not written as "ADAM". According to the authors, the attractive benefits of using Adam on non-convex optimization problems are stated as follows:

- Straightforward to implement.
- Computationally efficient.
- Little memory requirements.
- Invariant to diagonal re-scale of the gradients.
- Well suited for problems that are large in terms of data and/or parameters.
- Appropriate for non-stationary objectives.
- Appropriate for problems with very noisy/or sparse gradients.
- Hyper-parameters have intuitive interpretation and typically require little tuning.

Different from the classic Stochastic Gradient Descent(SGD) algorithm, which maintains a single learning rate for all weight updates and the learning rate does not change during training, Adam uses the squared gradients to scale the learning rate, and it takes advantage of momentum by using the moving average of the gradient instead of the gradient itself. In the scope of this work, Adam was chosen as the only optimizer since it converges faster, more stable, and requires less tuning than other optimizers.

2.3 Active Learning

2.3.1 Introduction of active learning

Active learning or query learning is a sub-field of machine learning and, more generally, artificial intelligence. It was developed as a mean to overcome the labeling bottleneck of machine learning problems by prioritizes, which data the model needs most to learn and requests labels for just those. In this way, the active learner aims to maximize the performance of the model using as few labeled instances as possible, thereby minimizing the labeling cost.

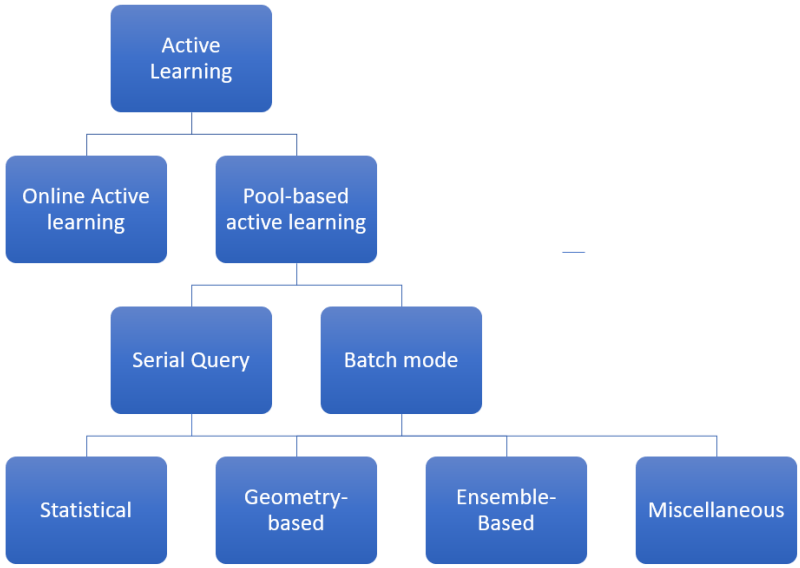


Figure 2.7: Categories of active learning

Active learning has been extensively applied in domains includes text classification, natural language processing, image classification, and filtering, etc. General speaking, active learning can be categorized as shown in Figure 2.7. At the highest level, active learning is divided into online active learning and Pool-based active learning. In **Online Active Learning**, the learner encounters the data points sequentially, and at each instant, the model has to decide whether to query for a label on this data point or not. The main challenge regarding this form of active learning is that the active learner has to make decisions without having access to the information from other unlabelled data.

As the more popular form of active learning, Pool-based active learning exposes the active learner to a pool of unlabeled data, where the learner can query a single example (**serial query**) or multiple examples from the data pool (**batch-mode** for manual annotation. While serial query has been widely used and well studied in the past few decades, batch mode active learning has been comparatively less explored. There are 3 main strategies for Pool-based active learning:

- **Statistical approaches**: The active learner selects data by optimizing a specific statistical criterion (variance, expected error rate, expected entropy, etc.). [19] [20]
- **Geometry-based approaches**: The active learner selects data by solving a combinatorial optimization problem in space. Alternatively, the active learner can also base the selection on distances of data points from the separating hyperplane(SVM approach). [2] [6] [21]
- **Ensemble-based approaches**: The active learner uses the ensemble output of the classifiers and the selection criteria are based on the degree of disagreement among the learners, or the uncertainty of the classifiers. [22] [1]

2.3.2 Batch-mode active learning

While serial query based active learning has been widely used in many fields, batch mode active learning has been comparatively less explored. [1]. Batch-mode active learning allows the learner to query instances in groups, which prove to be more effective in a parallel labeling setting with multiple human annotators. Furthermore, adding only one instance at a time slows the overall learning process down, which has a serious impact on the speed of the active learning process on DNNs. Another advantage of batch-mode active learning is that for a large model, a single point will not have a statistically significant impact on the model due to local optimization algorithms. However, in order to design a prac-

tical acquisition function, i.e., selection criteria, for this type of active learning, there are many significant challenges that one has to overcome. [23] discussed that a proper scoring function for batch-mode active learning has to incorporate the following measures: **informativeness, representativeness**, and **diversity** when selecting batches of example for the pool of unlabeled data. The measure of informativeness often relates to the uncertainty of an instance, while representativeness determined if an instance shares multiple similar features to a cluster of instances, which often seen in geometric approaches.

Finally, diversity measures the information overlapping among instances within the selected batch and often gets estimated in some form of variance calculation. In the case of image recognition using CNNs, popular uses of batch-normalization techniques have significantly reduced the effect of a single data-point on the performance of the model. Batch-normalization, combined with the demanding computational nature of DNNs has made batch-mode active learning the most viable form of active learning in this field. However, since the uses of CNNs often require massive datasets and the networks often have a complicated structure, it is nearly impossible for the active learner to find the optimal query set with the current state of computing technology. Additionally, most of the traditional active learning strategies also fail to consider the overlap in information content among the "best" instances in this field of machine learning. There has been several research studies to challenge this problem such as [24], [2]. For the most part, these approaches have shown better performances than random sampling.

2.4 Incremental learning

In the field of machine learning, incremental learning is a technique in which the training data get continuously expanded to extend the existing model's knowledge.e. to further train the model. The aim of incremental learning is for the learning model to adapt to new data without the need to re-train the model from scratch. Incremental learning is beneficial when it comes to applying active learning in a deep learning setting. Since deep learning model often expensive to train, re-train the model from the beginning on every newly retrieved labeled training data is not a good option. Therefore incremental learning is the most practical way to apply active learning on a deep learning setting, especially on complex neural network architecture.

Chapter 3

Related Work

3.1 Active Learning Literature Study

When it comes to mapping the landscape of active learning, the work, Active Learning Literature Survey [1] has to be mentioned. This paper has addressed all of the basic and advanced methods in active learning up until 2010. However, the landscape of active learning has changed a lot in the last few years. Followed by the rise of deep learning, most of the focus in active learning has switched to supporting DNNs. Some of the notable works that focus in batch-mode active learning is, [4] and [25], where the former addressed the use of pseudo labeling and the latter gave rise to how batch-mode active learning could combine meta-learning. In this work, 3 of the recently state of the art methods in batch-mode active learning which are Active deep model adaptation method (ADMA), [24], Ensembles Active Learning(ENS) [5] and Active learning using Core set approach, [2] were implemented. These methods were published in 2018 and together with Adversarial Active Learning [26] are the current state of the art methods in the field of image classification.

3.2 Uncertainty base Active Learning

One of the most basic and simplest method in active learning is to query base on the uncertainty of the classifier. The are 2 selection criteria that was used in this thesis is : least confidence (LC) and margin sampling (MS) [19].

3.2.1 Least Confidence

This method queries instances one by one by selecting an unlabeled instance x_i from an unlabeled data pool X^U, where x_i is the point that has the lowest score given by the function $max(p(y_j|x_i))$, where $p(y_j|x_i)$ is the predicting probability of the class y_j given the input instance x_i.

$$x_i^{LC} = \underset{x \in X^U}{argmin}(max(p(y_j|x_i))) \tag{3.1}$$

3.2.2 Margin Sampling

This method pick the sample with the smallest probability difference of the top two class predictions y_1 and y_2.

$$x_i^{MS} = \underset{x \in X^U}{argmin}((p(y_1|x_i) - p(y_2|x_i))) \tag{3.2}$$

3.3 Query by Committee

Query by Committee (QBC)is an active learning algorithm that was proposed in 1992 [27] . In QBC, a committee of students is trained on the same data set and the next query is chosen according to the principle of maximal disagreement. In this method, the active learner receives a stream of unlabeled samples as input and decides for each of them whether to query for its label or not. In this case, the active learner constructs a "committee" which consist of two or more classifiers. Each committee member then classifies the candidate sample and the learner measures the degree of disagreement among the committee members. The sample is then selected for labeling, which based on the degree of disagreement between the classifiers. QBC often uses Kullback-Leibler divergence (KL-d) [28] or the entropy of the distribution of classification to measure the disagreement. However, at the time of its introduction,QBC showed a bad performance on the machine learning architectures at that time. After that, QBC has become unpopular among the active learning methods. In this thesis, the proposed method is based on the same idea as QBC, however, there are some additional modification that make it more suitable to deep learning.

3.4 Deep Bayesian Active Learning

Deep Bayesian Active Learning(BALD) [29], is one of the more sophisticated uncertainty base method designed for deep CNNs. The method makes use of the Bayesian equivalent of CNNs, [30], which is CNNs with prior probability distribution placed over a set of model parameters $\omega = W_1, ..., W_L$. In order to achieve this, stochastic regularization techniques such as dropout can be applied, where dropout can be interpreted as a variational Bayesian approximation. The prediction probabilities of an instance in this case will be:

$$p(y_j|x_i) = \frac{1}{T}\sum_{t=1}^{T} p(y_j|x_i, \hat{\omega}_t) \tag{3.3}$$

with T is the number of different outcomes of the neural network affected by $\hat{\omega}_t$ with $\hat{\omega}_t \sim q^*_\theta(\omega)$ where $q^*_\theta(\omega)$ is the dropout distribution. The acquisition function of this method can be written as follows:

$$x_i^{BALD} = \underset{x \in X^U}{argmin}(-log(p(y_j|x_i)) + \frac{1}{T}\sum_{t=1}^{T} p(y_j|x_i, \hat{\omega}_t) log(p(y_j|x_i, \hat{\omega}_t))) \tag{3.4}$$

where $p(y_j|x_i, \hat{\omega}_t)$ is the prediction probability of each CNN with different weight parameters affected by dropout.

3.5 Active Deep Model Adaptation Method

The ADMA (active deep model adaptation)[24] is one of the first active learning methods that was designed specifically for models that make use of transfer learning. This method has two selection criteria, *distinctiveness* and *uncertainty*, where *distinctiveness* is to based on the output layers of the pre-trained model. The main idea behind distinctiveness is: Since the pre-trained model is already optimized for the original task, if one wants to optimize it to fit a new task, the train-set that is chosen should capture the unique properties of this new task to make it more efficient to fine-tune the model. Such capacity of the train-set is called distinctiveness since it can distinguish the original task and the current task. Figure 3.1 illustrates how ADMA works. In order to calculate the distinctiveness, the active learner selects one representative instance per class from the **original** task to form the *center set* $C = c_1, c_2, c_3...c_k$. The representative instances can be acquired as follows: For each class k, the mean of all instances, z in the class is:

$$\bar{z}_k = \frac{1}{|\Omega_k| \sum_{z in \omega_k} z} \tag{3.5}$$

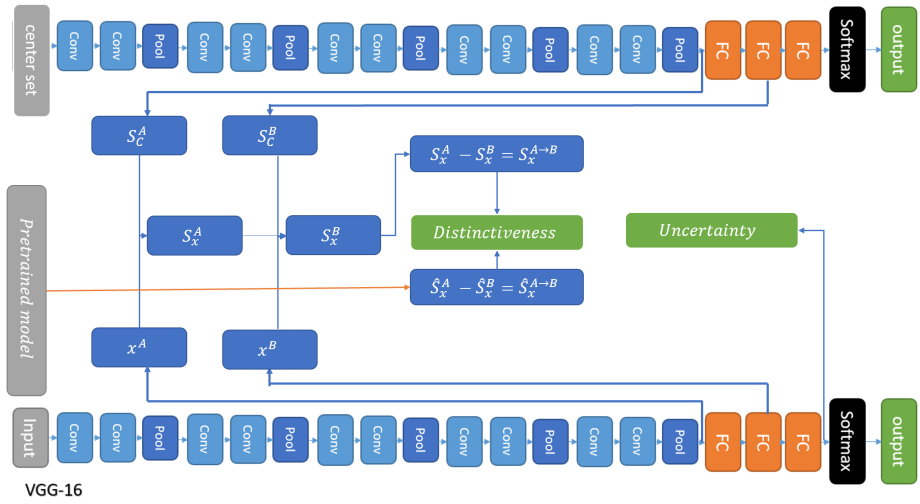


Figure 3.1: Active deep model adaptation method

where Ω_k is the set of all instances belong to the k-class and $|\Omega_k|$ is the set size. The representative instance c_k can then be calculate as:

$$c_k = \underset{z \in \omega_k}{argmin} ||z - \bar{z}||^2 \tag{3.6}$$

Figure 3.1 shows how this methods works, in this report, the last output of the convolutional layer is called A and the second forward connected(FC) layer is called B for convenience. S_C^A and S_C^B is defined as follows:

$$S_C^A = [S_{c_1}^A, S_{c_2}^A .... S_{c_k}^A] = \begin{bmatrix} ||c_1^A - c_1^A||^2 & ||c_2^A - c_1^A||^2 & ... & ||c_K^A - c_1^A||^2 \\ ||c_1^A - c_2^A||^2 & ||c_2^A - c_2^A||^2 & ... & ||c_K^A - c_2^A||^2 \\ ... & ... & ... & ... \\ ||c_1^A - c_K^A||^2 & ||c_2^A - c_K^A||^2 & ... & ||c_K^A - c_K^A||^2 \end{bmatrix} \tag{3.7}$$

$$S_C^B = [S_{c_1}^B, S_{c_2}^B .... S_{c_k}^B] = \begin{bmatrix} ||c_1^B - c_1^B||^2 & ||c_2^B - c_1^B||^2 & ... & ||c_K^B - c_1^B||^2 \\ ||c_1^B - c_2^B||^2 & ||c_2^B - c_2^B||^2 & ... & ||c_K^B - c_2^B||^2 \\ ... & ... & ... & ... \\ ||c_1^B - c_K^B||^2 & ||c_2^B - c_K^B||^2 & ... & ||c_K^B - c_K^B||^2 \end{bmatrix} \tag{3.8}$$

and S_x^A, S_x^B as:

$$S_x^A = \begin{bmatrix} ||x - c_1^A||^2 \\ ||x - c_2^A||^2 \\ ||x - c_3^A||^2 \\ ... \\ ||x - c_4^A||^2 \end{bmatrix} \quad , S_x^B = \begin{bmatrix} ||x - c_1^B||^2 \\ ||x - c_2^B||^2 \\ ||x - c_3^B||^2 \\ ... \\ ||x - c_4^A||^2 \end{bmatrix} \tag{3.9}$$

The next step is to calculate the feature transformation pattern from layer A to layer B, which is $S_x^{A\to B} = S_x^A - S_x^B$ and the approximated transformation pattern by assuming that the data is from the original task, $\hat{S}_x^{A\to B} = \sum_{k=1}^{K} p(M^0(x) = k) S_{c_k}^{A\to B}$. Where M^0 denotes the predicted class of x by the pre-trained model. Finally, the distinctiveness can be calculated by using this formula:

$$Distinctiveness(x) = \frac{1 - \tau(S_x^{A\to B}, \hat{S}_x^{A\to B})}{2} \tag{3.10}$$

where τ is the Kendall's tau correlation coefficient [31] that uses to estimate the difference. The *uncertainty* in this method is an uncertain base estimator defined as:

$$Uncertainty(x) = -\sum_j p(y_j|x)(1 - p(y_j|x)) \tag{3.11}$$

where $p(y_j|x)$ is the prediction probability of x belong to class y_j.

Dynamic trade-off

In this method, the distinctiveness measures the contribution of an instance on improving the representation capability of the network, while the uncertainty measures the prediction output improvement of the classifier. In order to select the most useful set of instances, distinctiveness and uncertainty need to be simultaneously considered. It is intuitive that distinctiveness is more important in the beginning than uncertainty since the network needs to adapt as quick as possible to the new task. Based on this motivation, the active learner will select instances with the highest score, which is defined as follows:

$$score(x) = (1 - \lambda t) \cdot distinctiveness(x) + \lambda t \cdot uncertainty(x) \tag{3.12}$$

where t is the iteration step and λ, is the selection constant.

3.6 Core-set

One of the main problems with batch-mode active learning is the information overlapping between query instances for large query size. With this problem in mind, the core-set strategy was proposed by [2] as an effective method when the active learner has to query for a considerable large set of points for labeling at each step. In this strategy, the active learner will try to minimize the core set loss, which is the average empirical loss difference between the labeled data-set and the unlabeled data-set. The core set loss is defined as follow:

$$L_{coreset} = \left| \frac{1}{|n|} \sum_{i \in n} l(\mathbf{x_i}, y_i) - \frac{1}{|s|} \sum_{j \in s} l(\mathbf{x_j}, y_j) \right| \tag{3.13}$$

where n is the set of unlabeled data and s is the set of labeled data. Since the goal is to find the best set of s for the label query, this loss is not directly computable because the access to the labels in the labeled data set is available at this point. Hence the paper [2] has reformulated and proved that this loss minimization problem is equal to an optimization problem where the goal is to minimize δ_s which is the cover radius of every labeled data points, i.e., the longest distance between a single unlabeled and labeled point. Figure 3.2 shows such a solution for this formulation, where the labeled data points(blue) with radius δ_s cover all the unlabeled data points(red)

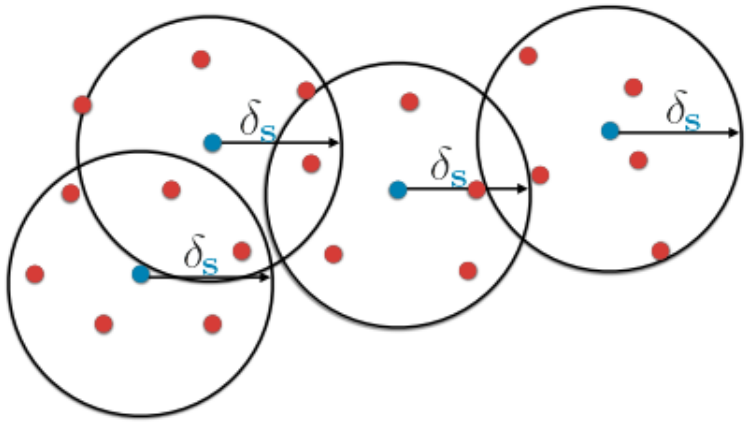


Figure 3.2: Visualization of the solution for Core-set problem δ_s [2]

This formulation is shown to be equivalent to the k-Center problem which can be defined as follows: given a set of points with specified distances; form a set of b center point where the maximum distance between a data point and its closest center is minimized. This problem can be formally defined as follows:

$$\min_{s^1:|s^1|\leq b} \max_{i} \min_{j\in s^1\cup s^0} \Delta(x_i, x_j) \tag{3.14}$$

where s^0 is the already labeled instances and s^1 is the going to be label instances. While metric k-center is a well-known NP-hard problem [32] it is still possible to obtain a good enough solution for the problem by using greedy methods or by redefining the problem as mixed integer program (MIP) by a third party program, e.g. Gurobi [33]. The core-set algorithm can be found on the author's Github. [34].

3.7 Ensembles Active Learning

The idea of combining predictions of an ensemble of neural networks for improving the performance of a model goes back many years [5]. However, besides increasing task accuracy, ensembles techniques can also be used to estimate the prediction uncertainty of deep neural networks. Based on this idea, the paper, "The power of ensembles for active learning in image classification" [35] has proposed a method that uses an ensembles various uncertainty based active learner coupled with variation ratio approach and geometric approaches(a simplified version of Core-set in this case) to achieved an impressive result on several dataset like MNIST, Cifar-10, ImageNet. Similar to the proposed method of DAS, ensembles active learning (ENS) uses five or more different ensembles networks that are trained with the same dataset and uses the same network architecture but with different random weight initializations. However, in ENS, instead of calculating the distance between the softmax output of the networks, it used several acquisition functions from different active learning methods on the averaged softmax vectors of each ensemble member to estimate the uncertainty and maximize the variance within the selected batch. Furthermore, the paper introduced a variant of the method, ENS-Var, which included the variance of the softmax output vectors within the ensemble of the sub-sample data-pool and the previously selected samples to boost the performance of the active learner. Generally speaking, ENS is a powerful but complex and compute-intensive method.

Chapter 4

Method

4.1 Method description

4.1.1 Dual Active Sampling

In this thesis, the proposed method ,DAS, was implemented using **2 CNNs models** with an **identical layout**. While both of them share the same training dataset, which increases in size over time, their training process was separated and could be implemented parallel if desired. Apart from sharing the dataset and structure, the models in this method are unrelated in terms of weights, and optimizers at any given time. Figure 4.1 illustrates how DAS works. As shown in the figure, on every step, the model selects a batch of unlabelled sub-sample data (R) from the unlabelled dataset (U). Then the unlabelled sub-sample batch get processed by the 2 models (with drop-out on) and the models outputs get normalized by using equation 4.1.

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{4.1}$$

The distance between the outputs of each data-point is then calculated using Euclidean distance, and the data point that has largest distance within the sub-sample batch get added to the query batch.

Then, the active learner selects a new sub-sample batch and repeats the same process until n items have been selected (n = query batch size). After that, the active learner retrieves the labels for every item within the batch and add these

items to the training data-set. In the experiment, the models get re-trained on this new training data-set for **m epochs**, and when the training is done, the active learner repeats the whole process until the maximum number of steps (N) is reached.

In section 6.20, the different way to measure the distance between outputs was calculated using Scipy's Distance computations tools (scipy.spatial.distance)m [36].

Due to the bad performance of DAS on models with high uncertainty (unconverged DNNs), it is recommended to use other sampling method on the first few steps to maximize the performance of DAS. In the experiment, the active learner used random sampling instead of DAS on the first steps (N = 1) in section 6.1.4 and on the first two steps (N = 2) for the rest of the experiments. Furthermore, in the experiment, it is important to note that only one of the models used validation. This is to improve the performance speed of the methods since using validation on 2 models is more time consuming and has no significant impact on the selection process.

Acquisition function of DAS

The acquisition function of DAS using Euclidean distance is shown in Equation 4.2 where Output1 and Output2 is the normalized output array from the models.

$$\textbf{Pick the item with index } = argmax(\sum_{i=1}^{S}(Output1_i - Output2_i)^2)) \quad (4.2)$$

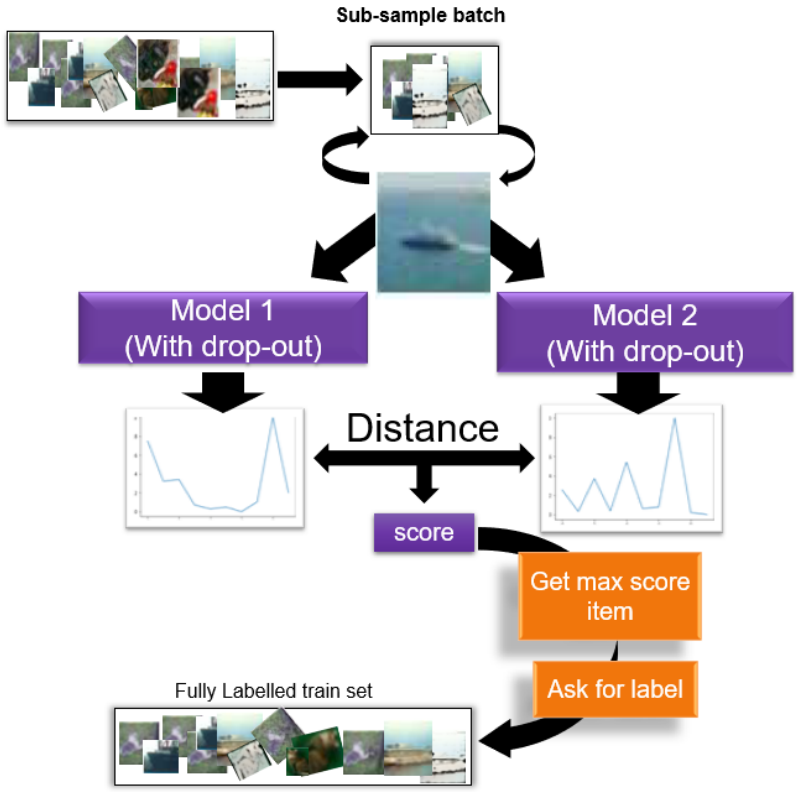


Figure 4.1: Illustration of Dual Active Sampling (DAS)

4.1.2 Multiple networks sampling

Similar to the method of dual learning, in section 6.2.2, the active learner used more than just two identical models and extended the distance calculation from between the output of 2 models to several models. In this experiment, the acquisitions function is based on the sum of Euclidean distances between each model's outputs.

4.2 Algorithm

The algorithm of DAS can be found in Algorithm 1 and the original implementation is available on the GitHub: `https://github.com/JohanPhan/Dual_active_sampling_Active_learning`

In Algorithm 1. N is the number of selection steps, which is the labeling budget divided by the size of the query batch (n). M is the number of steps that use random sampling, and m is the number of epochs that needed to re-train the model on every selection step.

Algorithm 1 Dual Active Sampling(Simple)

```
Initialize model1 and model2 with pre-trained weight
Initialize an empty Train set, S and unlabelled dataset, U
for step = 1 to step = N do
  if step < M then
    Query and Update S with n randomly selected samples from U
  else
    for i = 1 to n do
      random pick a batch of sample, R from U
      index = argmax(distance(model1(R),model2(R))
      get label for R[index] and add to S
    end for
  end if
  for epoch = 1 to epoch = m do
    train(model1)
    train(model2)
    validate model1
  end for
end for
```

Chapter 5

Experimental Setup

5.1 Dataset

Our experiments was conducted on 4 different datasets, Cifar10, Cifar100, SVHN and Indoor. These datasets are one of the most popular datasets that has been intensively used in the field of image classification in recent years.

5.1.1 MNIST

MNIST(Modified National Institute of Standards and Technology) is a dataset of handwritten digits. It has a training set of 60,000 images, and a test set of 10,000 images. The images used gray scaling technique and were centered and normalized in a 28x28 frame by computing the center of mass of the pixels. [37] Since its release in 1999, this dataset of has served as the basis for bench-marking classification algorithms and still remains one of the most popular datasets in this field.

5.1.2 Cifar-10

The main part experiments was conducted on Cifar-10 [38], the second most widely used dataset in deep learning research (after MNIST) [39] , which consists of 60000 32x32 colour images from 10 different classes: *airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.* The dataset is further splitted into

3 sub-sets: train-set(48000 instances) , validation set (2000 instances) and test set(10000 instances).

5.1.3 Cifar-100

CIFAR-100 is a similar dataset to CIFAR-10, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. This is a challenging dataset and we use this dataset to validate the performance of our method in a highly multi-class environment.

5.1.4 SVHN

SVHN is a real-world image dataset which can see to be similar in flavor to MNIST (which contains cropped single digits numbers), but incorporates an order of magnitude more labeled data (over 600,000 digit images) and comes from a significantly harder, unsolved, real world problem (recognizing digits and numbers in natural scene images). The dataset is obtained from house numbers in Google Street View.[40] Figure 5.1, figure 5.3, 5.2, 5.4, shown the example images of all the dataset that has been used in this thesis.

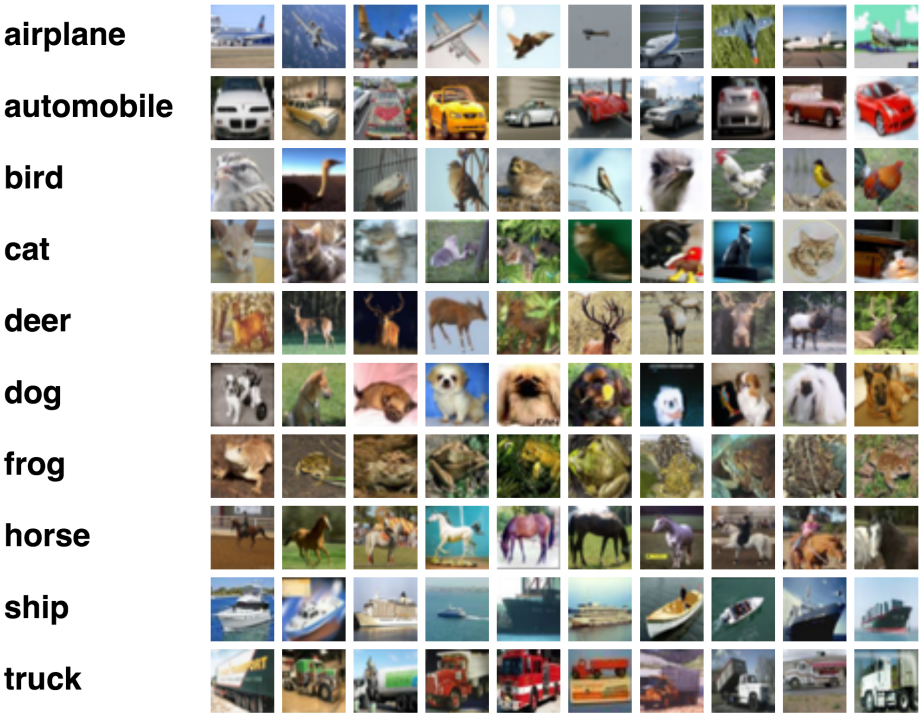


Figure 5.1: Example images from CIFAR-10 [38]

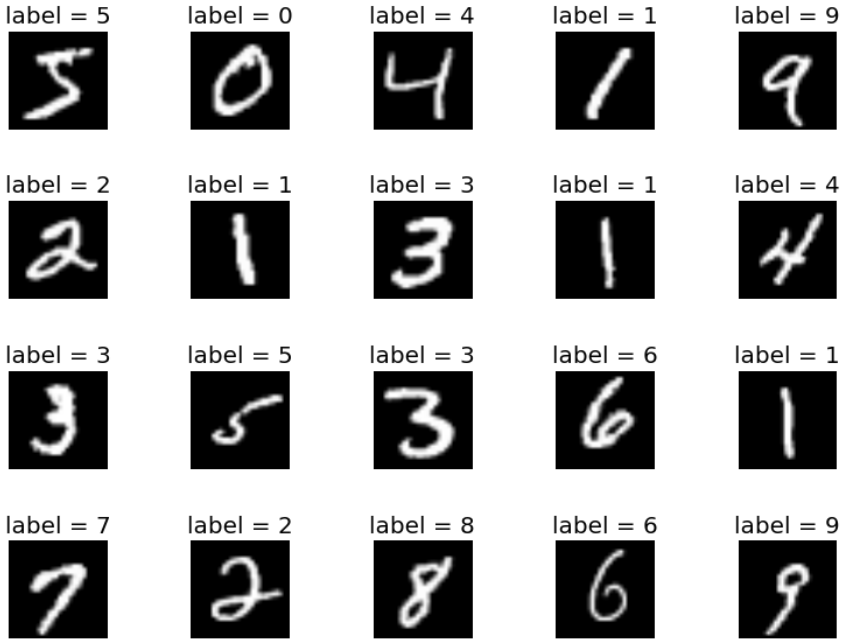


Figure 5.2: Example images from MNIST [37]

Figure 5.3: Example images from SVHN [40]

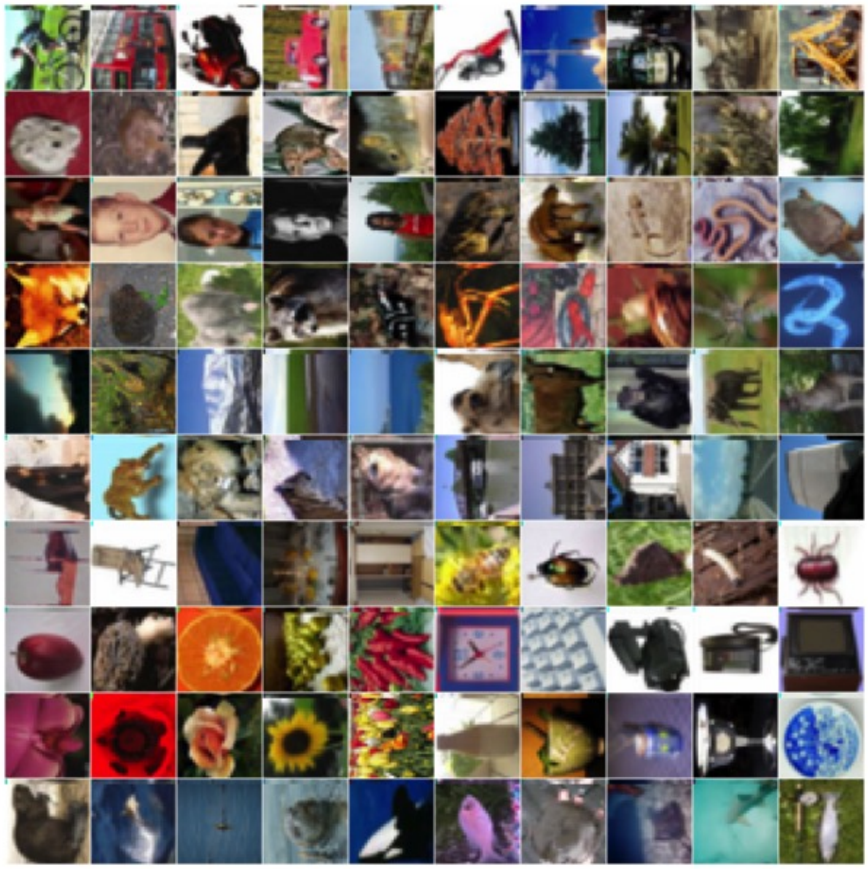


Figure 5.4: Example images from CIFAR-100 [38]

In this thesis, the training process will be implemented in an incremental way. This means that on every query step, the training dataset will get expanded by a fixed size of labeled data while the model and all of the training parameters from the previous step will remain unchanged. The mean accuracy from each step was plotted using Matplotlib, which is a Python 2D plotting library.

5.2 Model

The experiments were performed on Cifar-10, Cifar-100, SVHN using a VGG16 model initialized with pre-trained weights on ImageNet [18]. While most of the experiments uses model validation to avoid overfitting, the first experiment on Cifar-10 (Cifar-10 no val. in Table 5.1) was conducted without validation. This is to reuse the results of other comparison base line from the previous project where the experiments didn't use model validation. Moreover, DAS was also tested on a full run where the sampling process would not stop until it has selected all available data points.

The experiment on MNIST used the implementation of CNN on Keras instead of VGG-16 [41]. This simple CNN has 2 convolutional layers and 2 feed-forward dense layers. Table 5.1 shows how different parameters were selected on each experiment.

Dataset	Model	Learning rate	Batch size(n)	Data size Train/ val/ test	sub-sample pool size (R)	Max budget	Initial train size
Cifar-10 no val.	VGG-16	0.0001	100	50k/0/10k	200	10k	200
Cifar-10	VGG-16	0.001	100	48k/2k/10k	200	10k	200
Cifar-10 full	VGG-16	0.0001	100	58k/2k/10k	200	10k	200
MNIST	k-CNN	0.0003	20	58k/2k/10k	2000	1k	40
SVHN	VGG-16	0.0003	100	73k/2k/10k	200	10k	100
SHN-balanced	VGG-16	0.0003	100	40k/2k/10k	200	10k	100
Cifar-100	VGG-16	0.001	100	48k/2k/10k	200	10k	200

Table 5.1: Experimental setup

It is important to note that the training data in this experiments uses some minor data augmentation where the image got randomly flipped horizontally and cropped by random size between 0.8-1.0. The cropped sections then resized for training.

5.3 Implementation

Framework and hardware

All of the works in this thesis was conducted on PyTorch 4.0, which is one of the most popular open source deep learning framework, that built to be flexible and modular for research. It has a vibrant ecosystem of tools and libraries that support areas from computer vision to reinforcement learning. The experiment's code was written in Python programming language (version 3.4) on Jupiter Notebook and ran on a GPU server equipped with NVIDIA Tesla P100.

5.3.1 Sampling process

In section 6.1, the proposed method was tested on different datasets and compared mainly to the performance of **random sampling**. The reason for this is because batch-mode active learning on CNNs is a new field of study and the number of methods that was designed only for this field are few. On the other hand, most of the traditional active learning method has inferior or similar performance compared to random selection on this case [24]. In the first part of the experiment, on every step, a batch of $\mathbf{n = 100}$ images will be selected from the unlabelled pool, then the model will get trained for $\mathbf{m = 10}$ epochs before proceeding to the next step. The total acquisition size, i.e., the maximum size of the training dataset was set to be 10,000 on every dataset except MNIST. For random sampling, on every step, the active learner request label for $\mathbf{n = 100}$ images which are randomly selected from the data pool. As for DAS, the active learner was allowed to select and request label for one image from the randomly selected pool of $\mathbf{R = 150}$ images and then repeat the process until it reached the required batch-size, i.e., 100 images. Furthermore, since DAS has bad performance in the beginning, the $\mathbf{M = 2}$ first steps of the sampling process uses random sampling instead of DAS.

5.3.2 Experimental setup on the experiment on MNIST

While most of the active learning algorithms mentioned in this thesis are self-implemented or re-implemented from the available codes published by the authors on GitHub, it was not possible to implement ENS due to the lack of detail from the original paper [5]. In the paper, the author only showed the **plotted results** of their method and stated that they use four acquisition functions on five different models without providing detail information about the acquisition functions they are used.

In order to compare the performance DAS to ENS/ENS-Var, the plotted data point from the graphs in the original paper has been recovered and extracted using Engauge Digitizer [42]. Furthermore, the paper also provided a very detail description of the experimental setup, which included the choice of optimizer and learning rate, link to the CNNs model and how the selection process was performed. Based on this information, a new experiment was constructed, where the performance of random sampling has been used as the point of reference to the original experiment. By tuning the unmentioned parameter on random sampling, the same performance as the result from the paper was achieved. After that, DAS was implemented and run on the same experimental setting so that its

performance can be compared to random sampling and the plotted results from the paper.

5.3.3 The effect of parameters selection on DAS

The goal of the second part of the experiment is to study how the change in different parameters affect the performance of the proposed methods. These parameters included: choice of distance function, sub-sample pool size, and the number of models.

- **Choice of distance function:** For most of the experiments, Euclidean distance had been chosen as the default function to measure the difference between outputs. However, since Euclidean distance is one of the most straightforward and most simple forms to measure the difference between two array/distribution, there are possibilities that other types of distances or different measurement could outperform Euclidean distance in the case of DAS.
- **Sub-sample pool size:** the size of the randomly selected pool of unlabeled data where the active learner can pick from. In the experiment, the performance of DAS with different sub-sample pool sizes were tested. The goal of this experiment is to observe the relation between the number of choices and the performance of the method.
- **Number of models:** While the name of the proposed method is Dual Active Sampling; there is possible to use more than just 2 models in the selection process. Therefore, the goal of this experiment is to study if the use of the distance between more models would result in a better performance compared to the distance from just two models.

Chapter 6

Experimental Results and Evaluation

In this chapter, the results of the quantitative experiments will be presented in two parts. The goal of the first part of the experiment is to validate the performance of dual active sampling. Several methods in active learning have been implemented in this part on MNIST and Cifar-10 in order to make a good comparison. However, since nearly every "state of the art" methods in active learning is very time consuming to run, especially in the case of incremental training, the time constraint of this project has limited any further comparison on the SVHN and Cifar-100. In the second part of this chapter, the effect of choice in **distance measurement method, number of models** and **the size of the sub-sample pool size** will be studied, and the experiment results will be presented on different datasets. It is important to note that in the figures below, Dual Active Sampling will be called Dual Sampling, and the modified version of Dual Active Sampling that uses 3 networks will be called Tri Sampling or Tri Active Sampling (TAS).

6.1 Performance of DAS on different dataset

6.1.1 Results and Evaluation of experiments on Cifar-10 without using model validation

Most of the experimental results in Figure 6.1 are from the specialization project conducted by the author in Autumn 2018 [43]. In the mentioned project, several state of the art methods in active learning was implemented and their average test results on Cifar-10 were compared and discussed. In contrast to other experiments presented this thesis, the experiments in this chapter were done without the use of model validation [7]. The experimental results in the figures below, 6.1, is the **average results** on **5 different** runs.

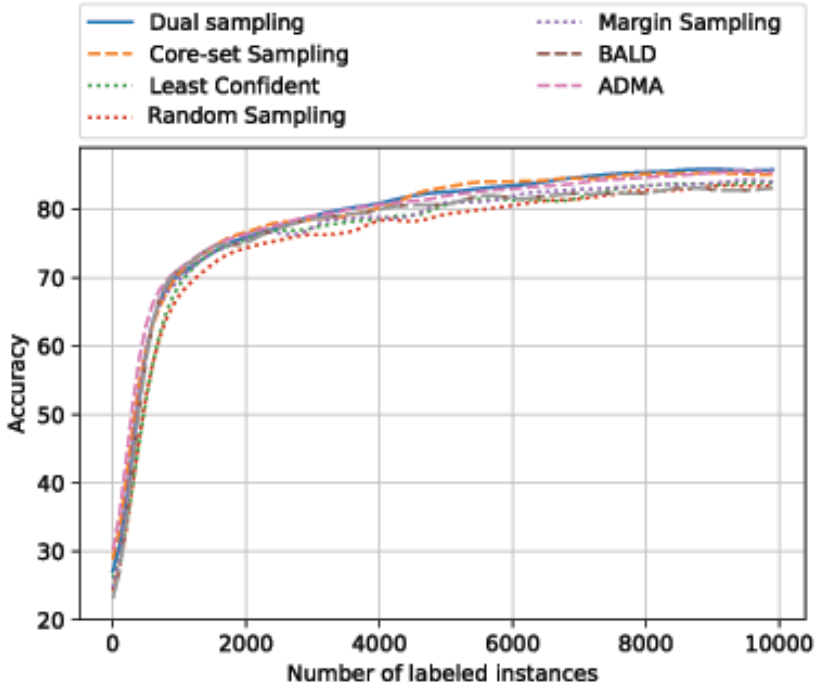
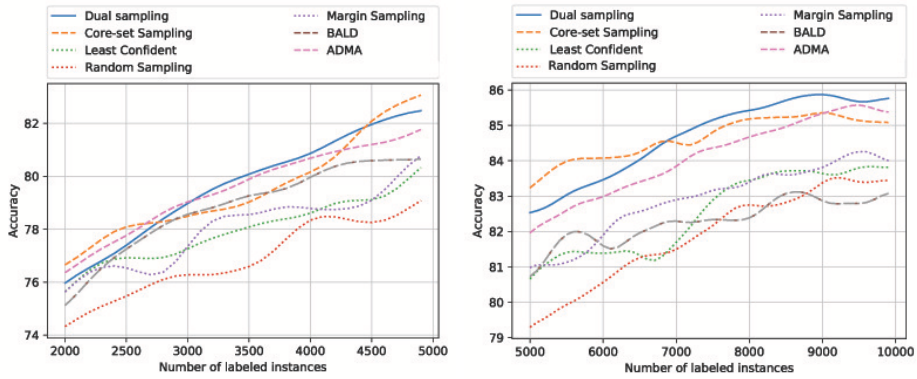


Figure 6.1: Test result on Cifar-10 using pre-trained VGG-16 without model validation



(a) Zoomed result from step 20 to step 50 (b) Zoomed result from step 50 to step 100

Figure 6.2: "Zoomed in image" of the plot in Figure 6.1

Method	2000	5000	8000	10000
DAS	76±2.1%	82.5±0.5%	**85.4**±0.3%	**85.7**±0.2%
Core-set	**76.8**±1.8%	**83.2**±0.4%	85.2±0.2%	85.4±0.1%
Least Confident	75.7±2.2%	82.0±0.4%	83.5±0.3%	83.8±0.2%
Random Sampling	74.5±2.5%	79.3±0.4%	82.8±0.4%	83.4±0.3%
Margin Sampling	75.7±2.3%	81.0±0.5%	83.4±0.2%	84.0±0.2%
BALD	75.2±2.0%	80.7±0.5%	82.4±0.2%	83.1±0.2%
ADMA	76.5±1.6%	82.0±0.5%	84.6±0.3%	85.3±0.3%

Table 6.1: Test result on Cifar-10 (pre-trained VGG-16 without model validation)

Evaluation

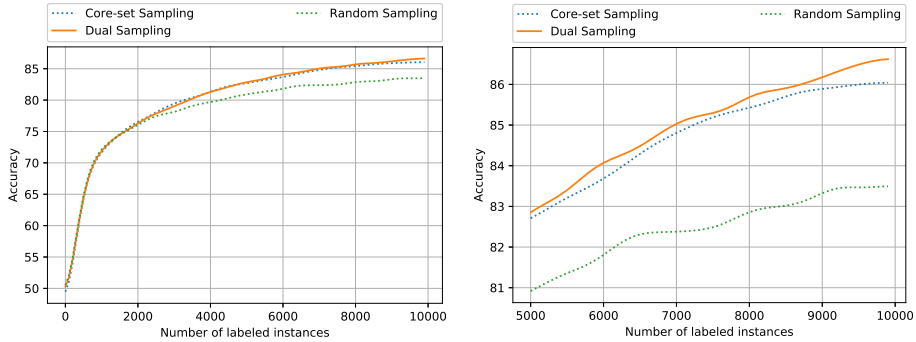
The results from Figure 6.1 and Figure 6.2 shown that DAS has comparable performance to other state of the art methods like Core-set sampling and ADMA. However, in terms of computational complexity, Das is much less demanding than the mentioned methods. In this experiment, the time it took to finish 1 run using Core-set sampling is approximately **3 days** since the selection mechanism of core-set is based on solving an NP-hard problem. In the case of ADMA, the method has a similar computation complexity compared to DAS but has a more complicated structure and more CPU-intensive (as opposed to DAS which

is GPU intensive). Another weak point of ADMA is that the method only work on pre-trained models [24]. Both ADMA and DAS needs between 2-3 hours to finish a run on Cifar-10 without validation while random sampling needs only **1 hours** on this incremental experimental setting.

Moreover, DAS seems to scale better with the size of labeled data compared to other methods, as shown in table 6.1 where the performance of DAS with 8,000 images is better than or equal to any other methods with 10,000 images.

6.1.2 Results and Evaluation of experiments on Cifar-10 using VGG-16 with model validation

From this section, all of the following experiments used model validation. The results on Figure 6.3 shows average test accuracy of random sampling, core-set and DAS on **10 runs**. Similar to the experiment in 6.1.1, on every step, the active learner selected and labeled 100 images, then trained the model for **10 epochs** and repeated the process. In the case of DAS, the training data in the **first two steps** were randomly selected.



(a) Result from step 0 to step 100 (b) Zoomed result from step 50 to step 100

Figure 6.3: Test result on Cifar-10 using pre-trained VGG-16

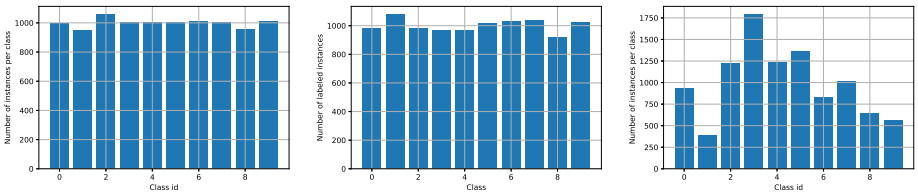
Method	2000	5000	8000	10000
DAS	76.9±1.7%	**82.9**±0.5%	**85.6**±0.3%	**86.7**±0.2%
Core-set	**77.0**±1.5%	82.8±0.4%	85.4±0.2%	86.0±0.2%
Random Sampling	76.8±1.7%	80.9±0.4%	82.8±0.4%	83.5±0.3%

Table 6.2: Test result on Cifar-10 using pre-trained VGG-16

Evaluation

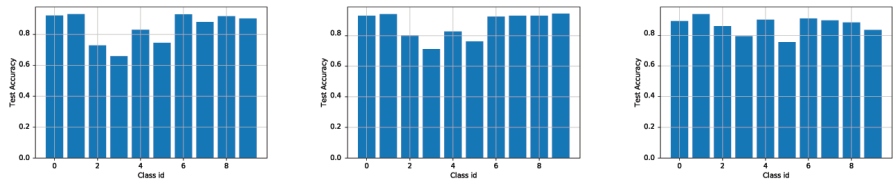
In Figure 6.3, DAS was able to give a significant accuracy improvement compared to random selection. It is important to note that the models that used active learning were able to reach the test accuracy of around 83.5 % using less than 6000

labeled instances as training data, while the model that used random sampling needed 10000 labeled instances to achieve the same level of performance (Table 6.2). Although it is hard to draw any clear conclusion on the accuracy different between core-set and DAS in this experiment, the final result on Figure 6.3b and Table 6.2 shown that DAS was able to achieve 0.7% higher test accuracy compared to core-set and consistently outperformed core-set after step 50. In order to study the reason behind the performance of DAS, the distribution of labels in training data at step 100 from a single run was plotted in Figure 6.4. In figure 6.4c, DAS showed a significant difference in label distribution compared to other sampling methods. Since Cifar-10 is a perfectly balanced data-set where each class contains 10,000 images, this is obvious that the distribution of labels in the training data in the case of random sampling(Figure 6.4a) is around 1000 images per class. Additionally, in Figure 6.4b, core-set sampling also shown a somewhat balanced distribution. This is understandable due to the fact that core-set is a method that selects base on the representativeness level of each image within different clusters and the classes in Cifar-10 are unrelated to each other. Combined with the information from Figure 6.5, it is easy to see that the four classes in the middle (class 2 to class 5) are harder to classify than other classes in Cifar-10. It must be pointed out that according to Figure 6.4c, the fourth class (class 3) got selected by DAS more often than other classes(1750 times) while the second class (class 1) got selected less often(300 times). In term of accuracy per class, despite the fact that DAS selected less than **a third** of instances from class 1 compared to other methods, the accuracy on this class according to Figure 6.5 is similar to other methods. On the other hand, by having nearly twice the size of training data on class 3, DAS was able to achieve higher accuracy on this class compared to other methods. Finally, it is worth noting that in Cifar-10, **class 1** represents **image of ship** and **class 3** represents **image of cat**. While the object "ship" in Cifar-10 is always surrounded by a body of water and having a similar shape, the object "cat" in Cifar-10 appeared on different backgrounds with different shapes (Figure 6.6).



(a) Random Sampling (b) Core-set (c) DAS

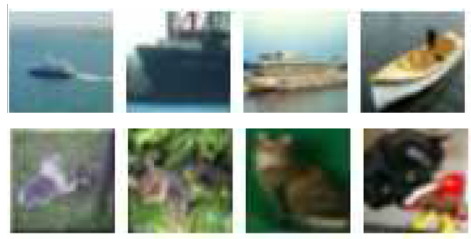
Figure 6.4: Distribution of items/class on Cifar-10 for 10000 samples. (VGG16-pretrained)



(a) Random Sampling (b) Core-set (c) DAS

Figure 6.5: Test accuracy per class on Cifar-10. (VGG-16 pretrained)

Figure 6.6: Some example images of cat and ship in Cifar-10



6.1.3 Results and Evaluation of experiments on Cifar-10 on the whole dataset

In this experiment, most of the chosen hyper-parameters were similar to those in 6.1.2. However, since the experiment consists of 500 steps, i.e., 5000 training epochs, the learning rate was turned down to 0.0001 instead of 0.001 like other experiments on Cifar-10 in order to maintain the stability of the model. This decision comes from the fact that Adam optimizer tends to be unstable when it gets trained for too long. The reason for this when the gradient becomes and stays very close zero, this will make the squares of the gradient become so low that they either have large rounding errors or are effectively zero. Further detail on this behavior of Adam or other adaptive optimizers can be found in the following discussion on [44]. Since this is a very time consuming experiment, the presented results are the filtered average result from only 2 run.

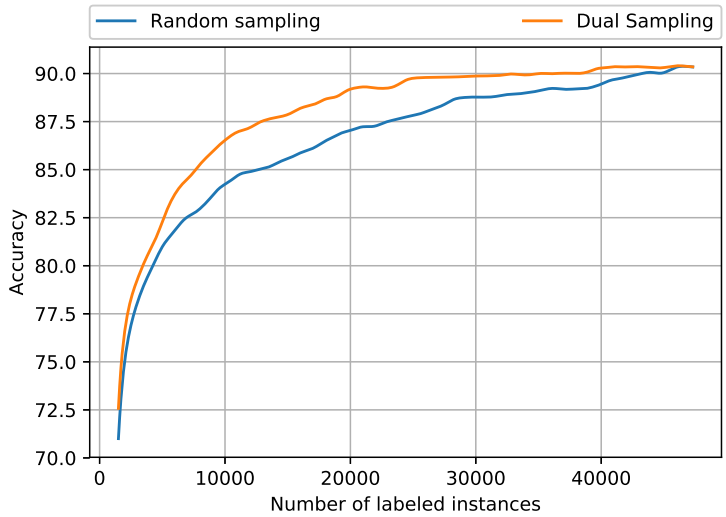


Figure 6.7: Test result on Cifar-10 using pre-trained VGG-16 on the whole dataset (from step 10 to 480)

Method	10000	20000	30000	40000	48000
DAS	**86.5**±0.4%	**88.9**±0.3%	**89.9**±0.2%	**90.4**±0.1%	**90.5**±0.1%
Random Sampling	83.8±0.4%	87.0±0.2%	88.7±0.2%	89.1±0.1%	**90.5**±0.1%

Table 6.3: Test result on Cifar-10 using pre-trained VGG-16 on the whole dataset

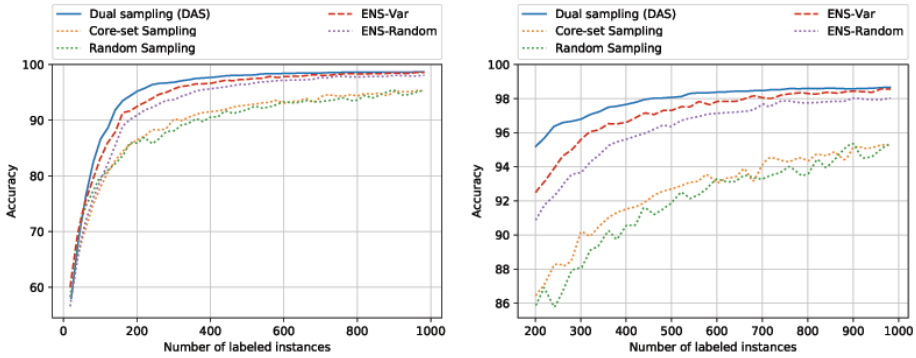
Evaluation

Figure 6.7 and Table 6.3 shown a significant improvement in test accuracy between a model that uses DAS and a model that uses random sampling. In the table, the models that used DAS needs only 20,000 labeled training data to achieve a comparable performance to the models that used random sampling and trained on 40,000 labeled training data. According to the figure, the test accuracy of DAS grew rapidly in the first half of the training process and slowed down in the second half while the growth rate of test accuracy for Random Sampling is more evenly. It is safe to say that most of the difficult instances of Cifar-10 have been selected by DAS in the first 25000 selection, thus led to the slow improvement in accuracy for the rest of the selection.

There are two theories regarding the significant accuracy improvement of DAS around 40000 labeled instances. The first theory is that since there has been little to no improvement from between 25000 instances and 40000 instances, i.e.step 250 to step 400, the improvement around step 400 is just a normal behavior when the model gets a better validation result after having a significant amount of extra data. However, there is a second theory that could explain this behavior. Since the acquisition function of DAS is based on the distance between the output of two models, there will be cases where both of them produced similar output with high certainty on a miss-classified instance, thus reject the selection of it. Therefore, at the later stage of the sampling process, these instances got finally selected and result in a significant improvement of the test accuracy of the model. However, due to the limited amount of experiments in this part, it is hard to draw any true conclusion.

6.1.4 Results and Evaluation of experiments on MNIST

This experiment showed the performance of DAS on MNIST where the results got compared with other states of the art AL methods on this dataset. As mentioned in section 5.3.2, the result of DAS is plotted together with the recovered datapoint of the graphs from the original paper. This experiment was performed on the Keras's CNNs implementation on MNIST. (S-CNN) [41]



(a) Result from step 0 to step 50 (b) Zoomed result from step 10 to step 50

Figure 6.8: Test result on MNIST using S-CNN

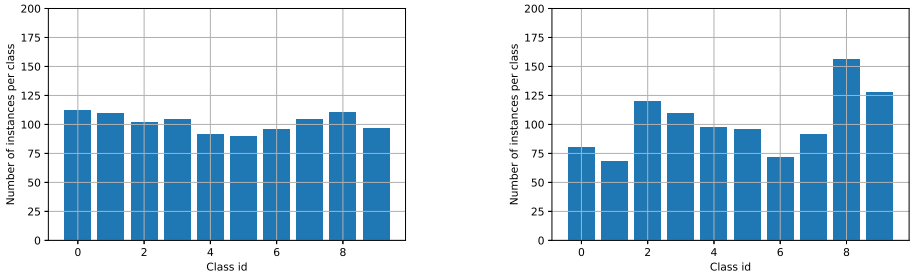
Method	200	500	800	1000
DAS	**95.2**±0.9%	**98.0**±0.5%	**98.7**±0.2%	**98.8**±0.1%
ENS-Var	92.5±0.5%	97.2±0.2%	98.3±0.2%	98.7±0.1%
ENS-Random	90.9±0.7%	96.3±0.3%	97.7±0.2%	98.0±0.1%
Core-set	85.9±0.8%	92.7±0.4%	94.5±0.6%	95.2±0.4%
Random Sampling	86.7±1.5%	91.9±1.1%	93.7±0.7%	95.2±0.6%

Table 6.4: Test result on MNIST using S-CNN

Evaluation

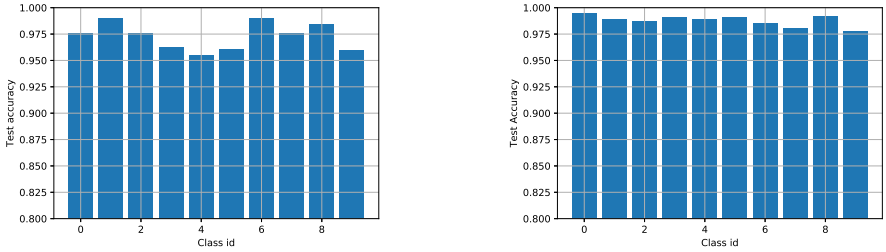
Figure 6.8 shown the test accuracy of different methods on MNIST. On average, DAS is the most effective method on this experiment, greatly outperformed ENS-VAR, ENS, and Core-set. However, it is worth noting that ENS-Var was able to catch up with the performance of DAS on the last step. Additionally, it

is noticeable that core-set performed very badly in their experiment, which could happen because the paper used the simplified version of core-set (k-greedy) [2]. In Table 6.4, DAS significantly outperformed other methods, especially in the beginning where it needed only 200 labeled instances to achieve the same performance as Random Sampling. Regarding the distribution of labels in Figure 6.9, it is noticeable that the class 8 and 9, got selected more often than other classes, while class 1, 0 and 6 got selected slightly less than other classes. However when compared to Figure 6.10, while selecting more item from class 9 resulted in a slight improvement on the test accuracy of this class, the class 8, which has the largest training-data gave a similar performance compared to the respective result of random sampling. In the case of class 4, 5, 6, which is amongst the lowest accuracy in the case of random sampling, dual sampling gives a significant improvement on test accuracy despite the fact it selected a similar number of instances on these classes compared to random selection. Moreover, this should be noticed that the classes in MNIST are closely related to each other(hand-written numbers), this could lead to a situation where a large number of training data on class 8 result in the improvement of test accuracy on class 0.



(a) Random sampling (b) DAS

Figure 6.9: Distribution of items/class on MNIST for 1000 samples.

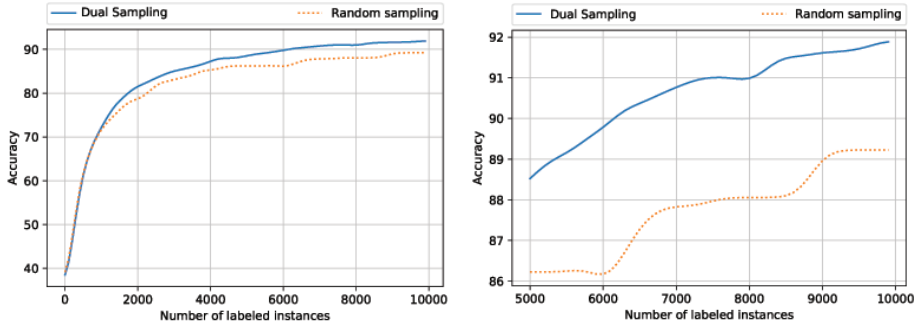


(a) ENS-random (b) DAS

Figure 6.10: Test accuracy per class on MNIST

6.1.5 Results and Evaluation of experiments on SVHN

Because of time limitation, the following results is compared between Figure 6.11 shown the average results of 5 runs on the SVHN dataset. The hyperparameters is relative similar to the hyperparameters used in 6.1.2 with a little change in learning rate(lr = 0.0003).



(a) Result from step 0 to step 100 (b) Zoomed result from step 50 to step 100

Figure 6.11: Test result SVHN on VGG-16(pre-trained)

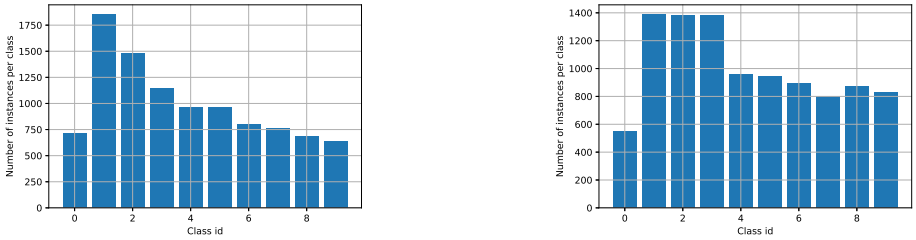
Method	2000	5000	8000	10000
DAS	**82.1**±1.9%	**88.6**±1.2%	**91.0**±0.6%	**91.8**±0.3%
Random Sampling	78.3±2.1%	86.3±1.8%	88.0±1.5%	89.2±1.0%

Table 6.5: Test result on SVHN using pre-trained VGG-16

Evaluation

It is important to note that SVHN is a **unbalanced dataset** and it can be seen in Figure 6.12a, where the distribution of items per class in random sampling closely represented the true distribution of the dataset. In Figure 6.11 and Table 6.5, it is easy to see that DAS outperformed random sampling by a large margin. As an example, DAS was able to achieve the test accuracy of 89% using less than 5,500 labeled images while random sampling needs around 9,000 labeled

images in order to achieve the same performance. In Figure 6.12b, DAS was able to form a more balanced distribution of items per class compared to random sampling, especially in the last 3 classes. In Figure 6.13, it is hard to draw any conclusion based on the test accuracy per class. The only noticeable thing is that while selecting less sample from class 1 compared to random selection, the test accuracy on class 1 in the figures 6.13 a & b are identical.



(a) Random sampling (b) DAS

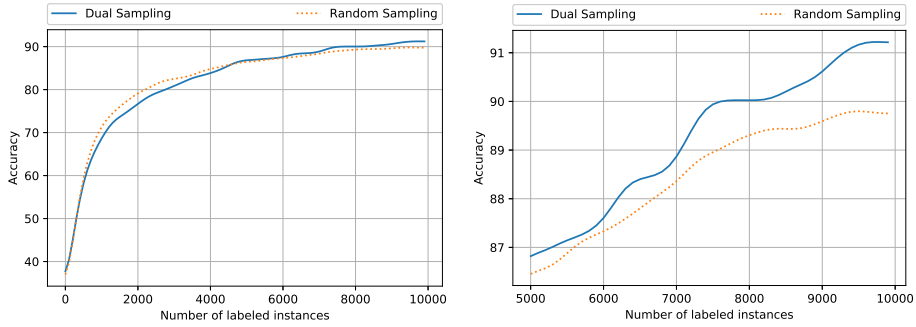
Figure 6.12: Distribution of items/class on SVHN for 1000 samples.

(a) Random sampling (b) DAS

Figure 6.13: Test accuracy per class SVHN

6.1.6 Result on SVHN-balanced

The experiments conducted in this experiment reused the hyper-parameters from the previous experiment on SVHN. However, in this part, the mentioned unbalanced SVHN dataset got balanced out by selecting 4,000 images from each class and removed the rest, effectively reduce the total size of this dataset from 73,257 images to 40,000. Figure 6.14 shown the average results of 5 runs on the balanced SVHN dataset.



(a) Result from step 0 to step 100 (b) Zoomed result from step 50 to step 100

Figure 6.14: Test result SVHN on pre-trained VGG-16

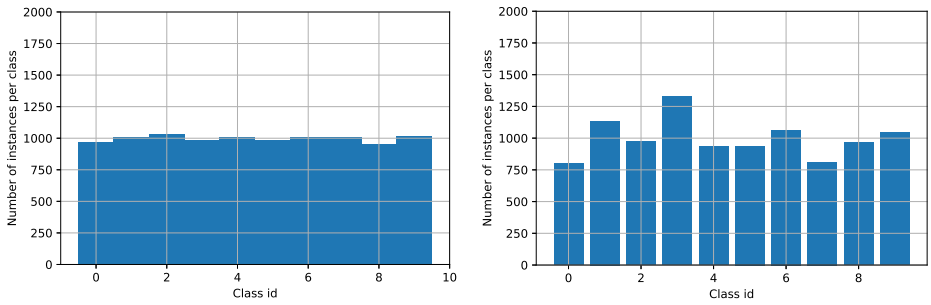
Method	2000	5000	8000	10000
DAS	76.9±2%	**86.7**±1.6%	**90.0**±1.0%	**91.3**±0.6%
Random Sampling	**78.9**±2.3%	86.4±1.7%	89.4±1.6%	89.7±1.1%

Table 6.6: Test result on SVHN-balanced using pre-trained VGG-16

Evaluation

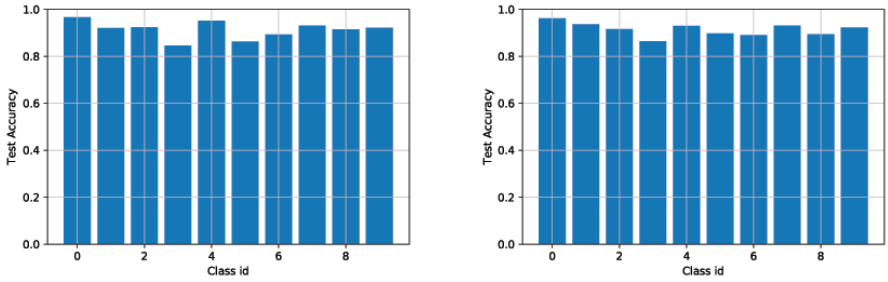
Because of time constraint, only the performance DAS and Random-sampling got compared in this experiment and the following experiments. By comparing the result from Table 6.5 and Table 6.6, it is surprising to note that while the performance of Random Sampling got improved on the balanced dataset, the performance of DAS decreased significantly . With 5000 labeled instances, the test accuracy of DAS on balanced SVHN is nearly 2% worse than on an unbalanced SVHN. With 10000 labeled instanced, this difference was reduced to 0.7%, while the performance of random sampling got improved by nearly 0.5%.

By balancing the data, the number of options has been reduced with nearly a half. This could explain why DAS performed worse in this example. In Figure 6.15, it is also worth noting that the class 4, which is the most challenging class in both of the experiments, got selected most often. However, the effect of this selection on the accuracy per class in Figure 6.16 is limited. Otherwise, contradicting to what was expected from this experiment, the distribution of classes in the class of DAS 6.15 was quite balanced.



(a) Random sampling (b) DAS

Figure 6.15: Distribution of items/class on SVHN for 1000 samples.

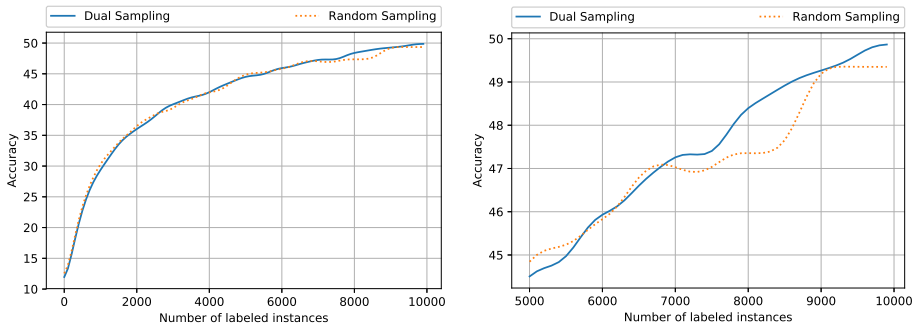


(a) Random sampling (b) DAS

Figure 6.16: Test accuracy per class SVHN-balanced

6.1.7 Results and Evaluation of experiments on Cifar-100

This experiment used the hyper-parameters as the experiment in 6.1.2. However, the VGG-16 model in this case with slightly modified to have 100 outputs instead of 10.



(a) Result from step 6 to step 100 (b) Zoomed result from step 0 to step 100

Figure 6.17: Test result on Cifar-100 using pre-trained VGG-16

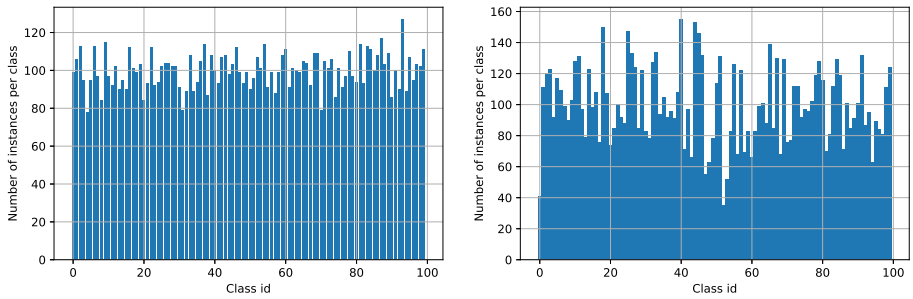
Method	2000	5000	8000	10000
DAS	36.5±2.8%	44.4±1.9%	**48.4**±1.0%	**49.8**±0.3%
Random Sampling	**36.7**±2.3%	**44.8**±1.8%	47.4±0.9%	49.4±0.4%

Table 6.7: Test result on Cifar-100 using pre-trained VGG-16

Evaluation

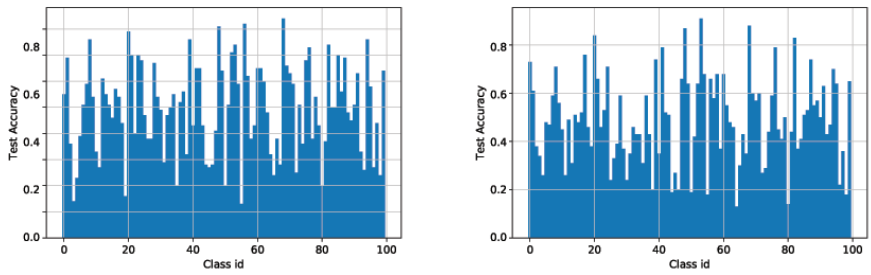
It is not a surprise that DAS was not able to perform well in Cifar-100, as seen in Table 6.7 and Figure 6.17. Since Cifar-100 has 100 of classes and DAS is depended on the Euclidean distance between the outputs, the acquisition function would be very noisy in this case. Furthermore, since the accuracy of the model in this experiment is very low, the outputs of the two models were also expected to be very noisy and unpredictable. This could lead to the performance of DAS compared to random sampling become similar or even worse. The results in Figures 6.18 and 6.19 is very noisy and hard to draw any comparison. However,

it is worth noting that the distribution of labels per class in the case of DAS, Figure 6.18b, is also very unbalanced.



(a) Random Sampling (b) DAS

Figure 6.18: Distribution of items/class on Cifar-10 for 10000 samples. (VGG16-pretrained)



(a) Random Sampling (b) DAS

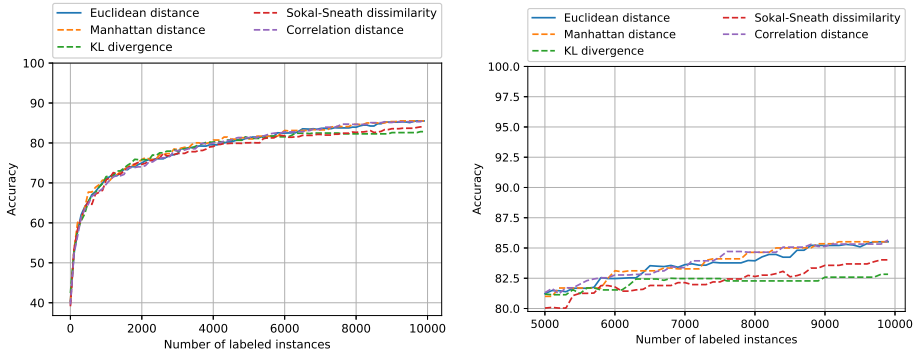
Figure 6.19: Test accuracy per class on Cifar-10. (VGG-16 pretrained)

6.2 The effect of various parameters on the proposed method

In this section, the effect of parameter choice in DAS and its variant will be tested and discussed.

6.2.1 Comparison of performance between different measurements method in DAS

In this experiment, there are 5 different output comparison methods that were chosen and compared on Cifar-10 and MNIST using the same experimental setting as in section 6.1.2 and section 6.1.4. Figure 6.20 shown the average results of 3 runs.



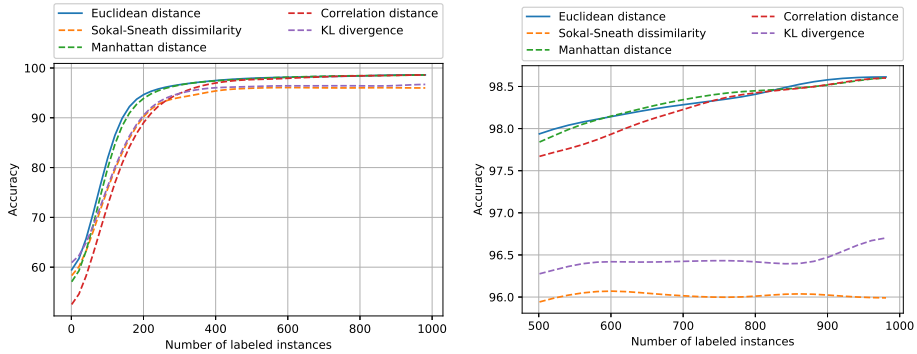
(a) Result from step 0 to step 100

(b) Zoomed result from step 50 to step 100

Figure 6.20: Test accuracy of different distance measurements in DAS on Cifar-10

Measurement method	2000	5000	8000	10000
Euclidean distance	75.1±1.5%	**81.4**±0.4%	83.7±0.2%	**85.6**±0.2%
Sokal-Sneath dissimilarity	74.8±1.4%	80.1±0.6%	82.6±0.3%	83.2±0.2%
Manhattan distance	**76.0**±1.6%	81.2±0.5%	**84.7**±0.2%	**85.6**±0.3%
Correlation distance	74.5±1.4%	81.3±0.4%	**84.7**±0.2%	**85.6**±0.2%
KL divergence	75.9±1.7%	81.5±0.4%	82.3±0.4%	82.7±0.3%

Table 6.8: Test accuracy of different distance measurements on Cifar-10



(a) Result from step 0 to step 100

(b) Zoomed result from step 25 to step 50

Figure 6.21: Test accuracy of different distance measurements on MNIST

Measurement method	200	500	800	1000
Euclidean distance	**94.9**±0.6%	**98.0**±0.3%	**98.4**±0.1%	**98.6**±0.1%
Sokal-Sneath dissimilarity	91.7±3.4%	96.0±0.4%	96.0±0.5%	96.0±0.5%
Manhattan distance	94.7±1.2%	97.8±0.33%	**98.4**±0.1%	**98.6**±0.2%
Correlation distance	89.4±5%	97.7±0.3%	**98.4**±0.2%	**98.6**±0.2%
KL divergence	91±3.3%	96.3±0.8%	96.4±0.5%	96.8±0.5%

Table 6.9: Test accuracy of different distance measurements on MNIST

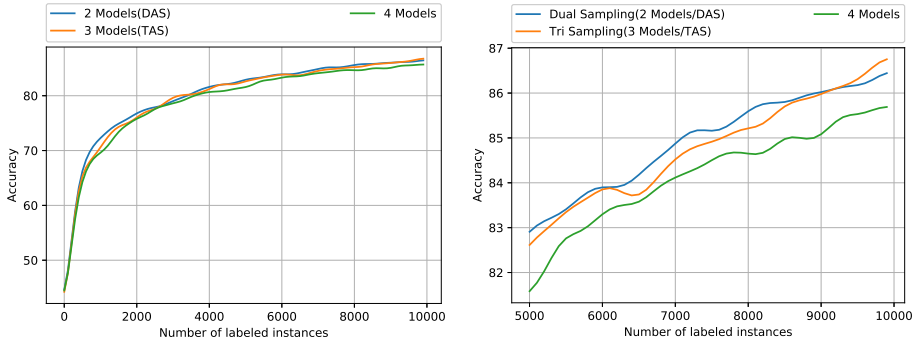
Evaluation

Figure 6.20 and Figure 6.21 shown that all of the distance measurements gave similar results compared to each other while KL divergence [28] and Sokal-Snearth dissimilarity [45] gave worse results. It is worth noting that Euclidean distance,

Manhattan distance and Correlation distance [46] gave the same result at the end in both cases, as shown in Table 6.8 and Table 6.9.

6.2.2 Effect of number of models

The following experimental results are about the effect of the number of models on the performance of the main model. 3 models sampling or Tri sampling/TAS (Tri Active Sampling) uses the measured distance between 3 models instead of 2 as the acquisition function. The same happens with 4, 5, 6 models sampling.



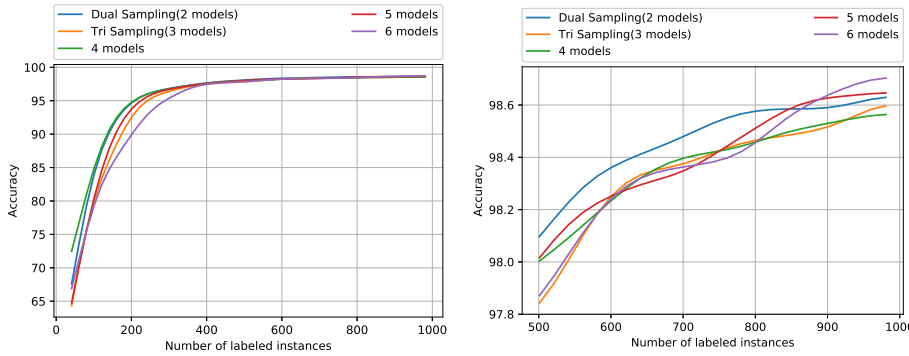
(a) Result from step 4 to step 50

(b) Zoomed result from step 25 to step 50

Figure 6.22: Result on Cifar-10 using pre-trained VGG-16

Method	2000	5000	8000	10000
2 models (DAS)	**76.4**±1.4%	**82.9**±0.4%	**85.6**±0.3%	86.5±0.2%
3 models	75.8±1.4%	82.6±0.5%	85.2±0.2%	**86.7**±0.5%
4 models	75.6±1.3%	81.6±0.4%	85.6±0.2%	85.7±0.4%

Table 6.10: Result on Cifar-10

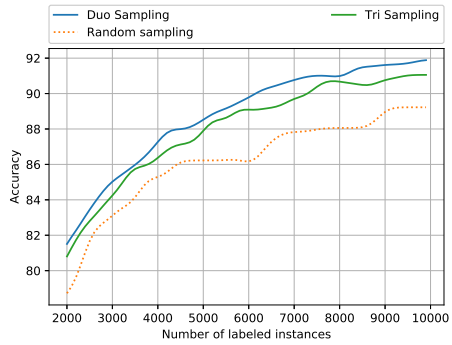


(a) Result from step 6 to step 100 (b) Zoomed result from step 50 to step 100

Figure 6.23: Test result on MNIST using k-CNN

Method	200	500	800	1000
2 models (DAS)	**94.8**±0.9%	**98.1**±0.5%	**98.6**±0.2%	98.6±0.1%
3 models	92.3±0.5%	97.9±0.2%	98.5±0.2%	98.6±0.1%
4 models	94.7±0.7%	98.0±0.3%	98.5±0.2%	98.6±0.1%
5 models	93.1±0.8%	98.0±0.4%	98.5±0.1%	98.6±0.1%
6 models	89.9±0.7%	97.9±0.3%	98.5±0.1%	**98.7**±0.1%

Table 6.11: Result on MNIST

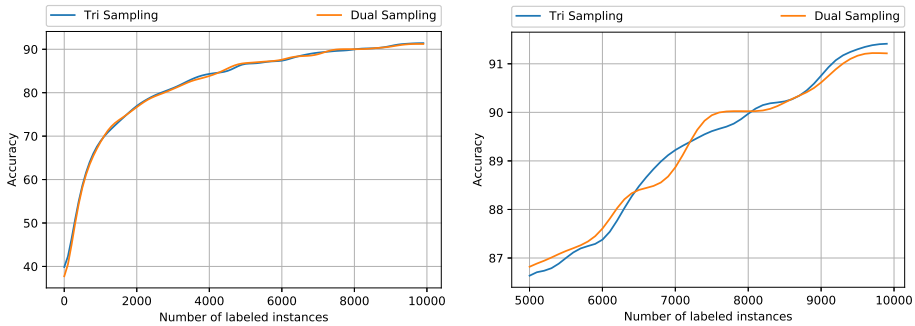


(a) Zoomed result from step 20 to step 100

Figure 6.24: Test result on SVHN using pre-trained VGG-16

Method	2000	5000	8000	10000
DAS	**81.5**±1.9%	**88.6**±1.2%	**91.0**±0.6%	**91.8**±0.3%
Tri Sampling (3 models)	80.9±1.8%	87.8±1.0%	90.7±0.8%	91.1±0.2%
Random Sampling	78.3±2.1%	86.3±1.8%	88.0±1.5%	89.2±1.0%

Table 6.12: Result on SVHN



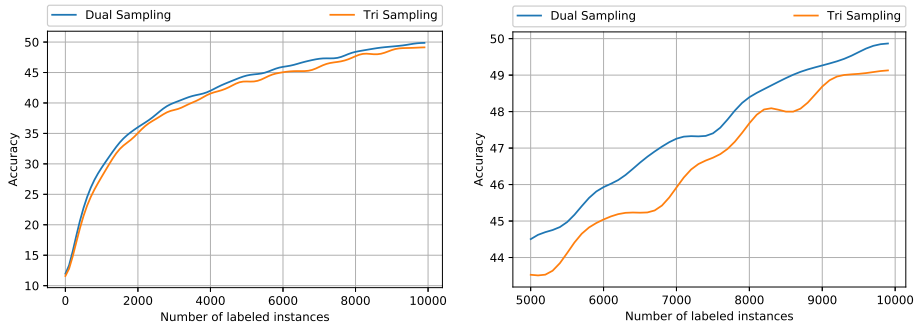
(a) Result from step 0 to step 100

(b) Zoomed result from step 50 to step 100

Figure 6.25: Test result on SVHN-balanced using pre-trained VGG-16

Method	2000	5000	8000	10000
DAS	76.9±1.3%	**86.7**±1.6%	**90.0**±1.0%	91.3±0.6%
Tri Sampling (3 models)	**77.0**±1.4%	86.5±1.3%	89.9±0.9%	**91.6**±0.6%

Table 6.13: Result on SVHN-balanced



(a) Result from step 0 to step 100 (b) Zoomed result from step 50 to step 100

Figure 6.26: Test result on Cifar-100 using pre-trained VGG-16

Method	2000	5000	8000	10000
DAS	**36.5**±2.8%	**44.4**±1.9%	**48.4**±1.0%	**49.8**±0.3%
Tri Sampling (DAS with 3 DNNs)	35.1±2.4%	43.6±1.8%	47.6±0.9%	49.2±0.4%

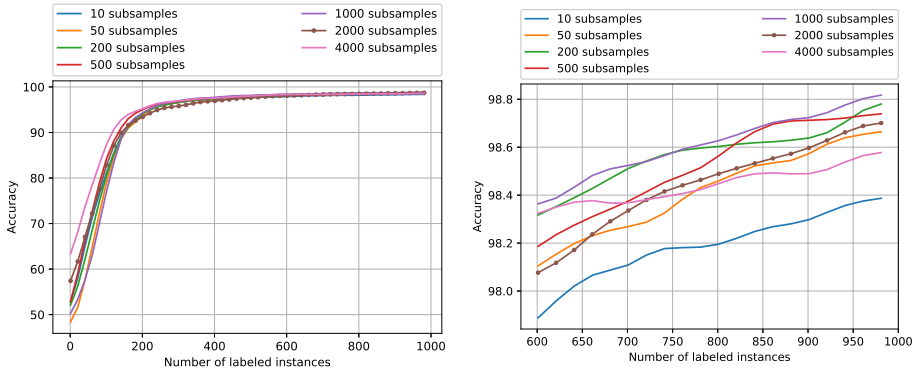
Table 6.14: Result on Cifar-100

Evaluation

Figures 6.23, 6.22, 6.24, 6.25, 6.26 while using the measured distance from an extra network, the result of tri sampling does not get improved, and even worse in some cases(MNIST, SVHN, CIFAR100). The result from 6.23 and 6.22 also shown that, measuring distance from more models damage the performance of the method, not only in term of computational time but also accuracy of the model. Furthermore, based on the information from the tables 6.10, 6.11 6.12, 6.13 6.14, the performance of 2 models (DAS) seems to perform better or at least equal to the performance of more models in most of the cases.

6.2.3 Sub-sample Pool size

Because of time limitation, the last experiment was only conducted on MNIST using k-CNN. In this experiment, different size of the randomly selected sub-sample pool where chosen and their effect on the average test accuracy on 5 different runs is shown in Figure 6.27. It is worth to take a note that during the experiment on MNIST, the pool-size of n = 2000 was chosen.



(a) Result from step 6 to step 100

(b) Zoomed result from step 50 to step 100

Figure 6.27: Test accuracy of DAS on MNIST using different sub-sample size

Pool size	800	1000
10 subsamples	97.9±0.4%	98.4±0.3%
50 subsamples	98.1±0.2%	98.7±0.2%
200 subsamples	98.3±0.2%	98.7±0.2%
500 subsamples	98.2±0.2%	98.7±0.1%
1000 subsamples	**98.4**±0.2%	**98.8**±0.2%
2000 subsamples	98.1±0.2%	98.7±0.2%
4000 subsamples	98.3±0.2%	98.6±0.1%

Table 6.15: Test accuracy of DAS on MNIST using different sub-sample size

Evaluation

The result in Figure 6.27 and Table 6.15 gives an indication that if the number of options is too small, 10 or 50, the performance of DAS will get worse. However, if the number of options is too large, i.e., 4000, the result will also become not as good as the medium size of the sub-sample pool. However, the effect of different sub-sample size on the performance of DAS is insignificant as long as the sub-sample is large enough. According to Table 6.15, amongst the available options, the pool size of 1000 and 200 works best on MNIST.

Chapter 7

Discussion

7.1 Method evaluation

7.1.1 Approach Discussion

Based on the experimental results, the proposed idea of using pairwise distances between the outputs of two similar models as the **acquisition function** in batch-mode active learning seems to have great potential in the field of image recognition. The results in the Figures 6.2, 6.7, 6.24 have shown that DAS needs less labeled data to the achieve the same level of performance as random sampling. In term of test accuracy, DAS gives a slightly better performance compared to other state of the art methods, especially with a large amount of labeled data. Furthermore, in Figure 6.4, DAS was able to show a smart behavior where it selected more items from the difficult classes and fewer items from the easier without any prior knowledge of their true label.

Method explanation

In order to understand the reason behind the well performance of a simple method like DAS, one need to understand how the acquisition function really work. In DAS, the acquisition function is based on the dissimilarity between the outputs of models that has the same structure and trained on the same dataset. This dissimilarity between outputs is also similar to one the of the often encountered problems in deep learning, where DNNs always gives a slightly different result on

different runs. The four main factors that cause this **inconsistency property** of deep learning are, **mini-batch gradient descent, weight initialization, random drop-out, and training data augmentation**.

In **mini-batch gradient decent**, the algorithm randomly splits the training dataset into smaller batches, and uses these batches to train the model. Since this algorithm takes the sum the gradient over the mini-batch or takes the average of the gradient to further reduces the variance of the gradient, the way these mini-batches was randomize selected directly affect the change in weight of the model during the training process.

The second contributing factor is the way the weights of the model was randomly initialized at the start of the training process. Even when using a pre-trained model, the last layers often gets initialize with random weights.

The third factor is **random dropout**. This factor plays an instrumental role in the performance of DAS and will be further discussed later in this section.

The last factor is the effect of different data augmentation techniques on the training data. In the scope of this thesis, "random cropping and horizontal flipping" [47] was used in the experiment in order to boost the performance of the classification model on the datasets. These techniques apply random changes on every single item in the training data on every epoch and play an essential role in the mentioned output different.

When it comes to DAS, this inconsistency property of deep learning has been exploited and used as a fundamental part of the sampling strategy. These mentioned four factors made the models in DAS to converges to different local optimums during the training process, even when they are sharing the same dataset and parameters. Given the dimension complexity of deep learning algorithm, most of the optimization algorithm like ADAM often converge to local optima instead of global optima, thus making it nearly impossible for the two models to become the same at any point of the training process. [48]

Informativeness and Representativeness

In section 2.3.2, the 3 measurements, informativeness, representativeness, and diversity, in batch-mode active learning was mentioned as a way to evaluate the quality of a sampling strategy. When it comes to **informativeness**, DAS has shown that the method is capable of selecting the more difficult/harder images, as is shown in Figure 6.4. Furthermore, the way DAS works is also based on the uncertainty of the models (when the two models are most disagree on a sample), and the informativeness measurement in the case of deep learning is

often equivalent to uncertainty. Therefore it is reasonable to say that DAS fulfilled the informativeness requirement of a strong active learning strategy. In the case of representativeness, it is no evidence that DAS fulfills any requirement of this measurement. Consider that the method's acquisition function is based solely on the distances between outputs, which is unrelated to how representative a sample is, it is unlikely that DAS performs well when on this measurement. Since representativeness is more important early on, especially when there are few labeled data in the training set [1], and DAS strategy does not consider the representativeness in its selection strategy, it is understandable to see that DAS does not perform so well in the early part of the sampling process.

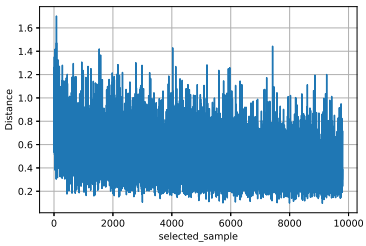
The importance of diversity in batch-mode active learning

Diversity measurement is often regarded as the most important measurement when it comes to data selection in deep learning [2]. Since CNNs often require a large amount of training data, the more diverse the data points are, the less likely for the model to become overfitted. There are two types of diversity in batch-mode active learning, diversity within the selected batch, and diversity among all of the training data . While the diversity among the selected item often appear in the form of uncertainty (the classifier struggle to recognize what it has not seen before), the diversity of the selected batch is much harder to measure.

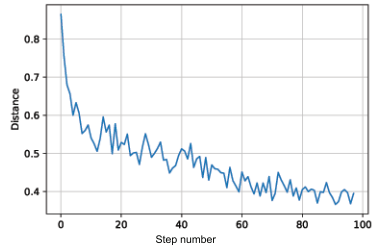
In the case of image recognition using deep learning, it is hard to define a meaningful measurement of diversity, especially when the diversity in this case is also based on the current knowledge of the model about the problem. There are methods such as variance ratio [49], where the variance between a certain network layer of the selected items get calculated and the goal is to maximize this variance. However, most of the variance based methods do not incorporate the uncertainty of the sample, therefore perform bad in general. [1]

The selection behavior of DAS

Figure 7.1 showed the distance between models on every selected image during the whole training process with DAS. The most important thing to notice in this Figure is while the average distance decrease over time, the different of distance between the selections is still very high at any given point of the sampling process. Since the figure only showed the distance between models of the selected items, which is the item with the highest distance between models within the sub-sample batch, such high noise can indicate how diverse the selection is, even when the sampling method is based on such a simple measurement.



(a) Distance value of all selected samples at the time of its selection



(b) Average distance value of the selected batch on every step

Figure 7.1: Distance between outputs of the selected images by DAS on Cifar-10 with batch size and sub-sample size of 100

Figure 7.3 showed some of the examples from the outputs of the model. The plotted values are from items that got selected by DAS during the sampling process, which means that these plotted values are from the items that have the highest distance value within their respective sub-sample pool. By looking at the figures, there are **four different cases** where an item get selected for labeling:

- **Case 1**: When both of the models are uncertain (Figure 7.3a and Figure 7.3b).
- **Case 2**: When one of the models is certain and the other is uncertain (Figure 7.3c and Figure 7.3d)
- **Case 3**: When both of models are certain and but one is more certain than the other. (Figure 7.3e)
- **Case 4**: When both of the models are certain but on different way.(Figure 7.3f)

More detailed analysis of how DAS selects the data point is shown in Figure 7.2. According to the figure, among the 10,000 selected items during the whole training process, case 3 is the most common selection case in DAS. Furthermore, case 1 and case 2 happened more often in ealier steps than later steps while case 4 is a rare case overall. This is an interesting behavior since DAS is also an uncertainty in nature, and unlike other uncertainty based methods, DAS selects more items from the certain cases. The main reason for the small number of items on case 4 is because the size of the sub-sample pool in this experiment is relatively small (S = 200), and case 4 is a rare case, especially on converged models.

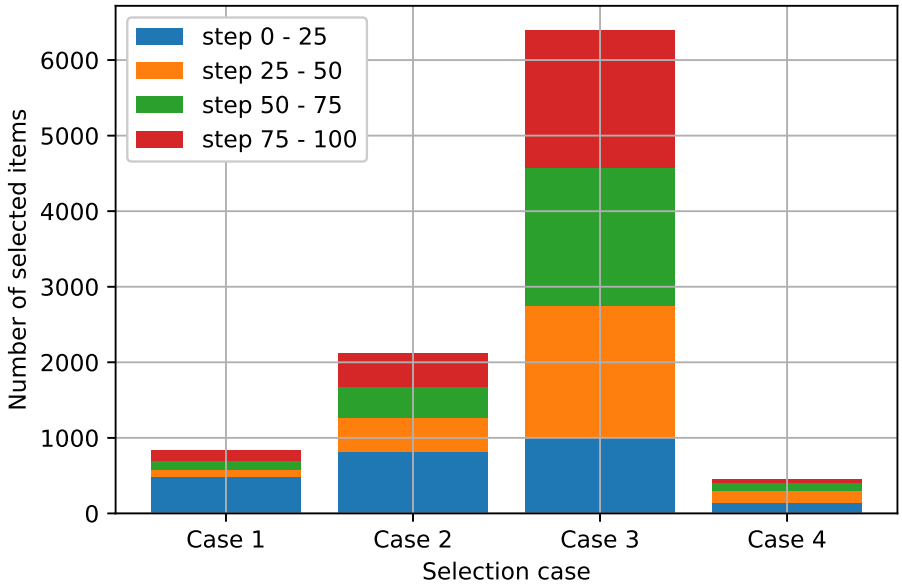
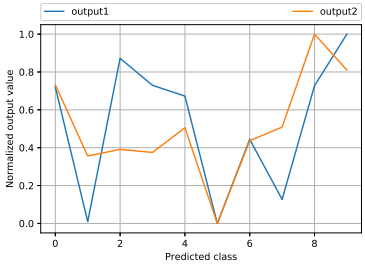
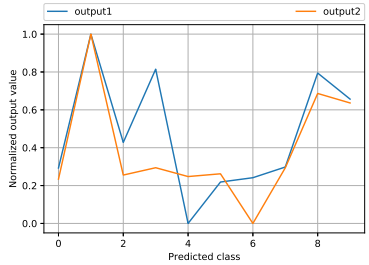


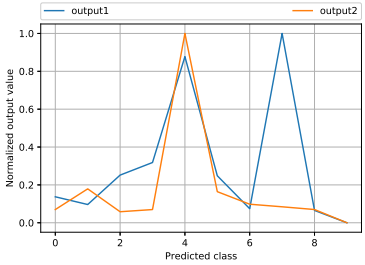
Figure 7.2: Number of selected item per case from an example run on Cifar-10 using DAS with n = 100



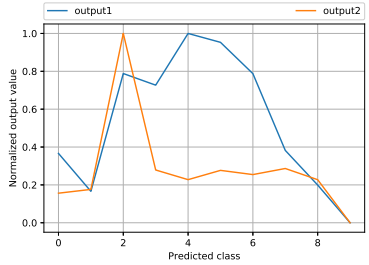
(a) Distance value of 0.872



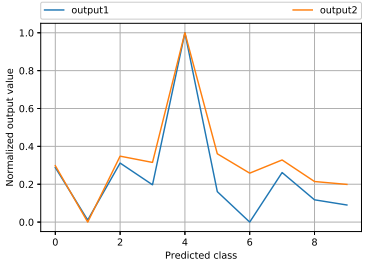
(b) Distance value of 0.662



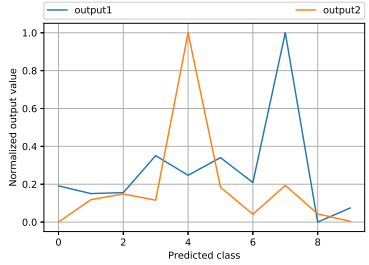
(c) Distance value of 0,986



(d) Distance value of 1,28



(e) Distance value of 0,384



(f) Distance value of 1,170

Figure 7.3: Example of outputs from the 2 models in DAS and the respective euclidean distance value between them

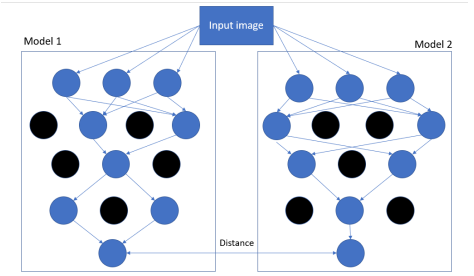


Figure 7.4: The role of drop-out in DAS

The pitfall of uncertainty

While other active learning strategies often focused on select the most uncertain data-point, this type of selection does not guarantee the best outcome. There are 2 types of uncertain data-points in deep learning, the data points that the model should learn, and the data points that the model should not learn. Since the goal of deep learning is the create a classifier that can generalize well, there are cases where some data points could prove to be harmful for this generalization. The reason for this could be anything between miss labeling to bad quality samples. While uncertainty could be a good indication of the quality of a data point for training, relying too much on uncertainty based sampling could lead to overfitting, where the training data-set is full of extreme cases. Therefore, with this in mind, the way DAS selects more item which belongs to case 3 as shown in Figure 7.2 could be a reason while it was able to perform well in the field where other uncertainty-based methods have failed to work.

The role of randomized sub-sample pool

Two main factors that have a direct contribution to maintain the diversity within the selected batch are *"randomized sub-sampling"* and *"random Drop-out"*. The effect of randomized sub-sampling can be seen in Figure 6.27. While it was evident that a small sub-sample size would decrease the effectiveness of the method, a huge sub-sample size also significantly undermined the effectiveness of the method. As for a large sub-sample size, the chance for the active learner to repeatedly select samples from only one or two classes is higher. Since the method selects the "best" item from every sub-sample pool, with a large pool size, the chance of these selections become similar to each other are very high.

The role of random drop-out

When it comes to random drop-out, Figure 7.4 demonstrated how drop-out work in DAS. It is crucial to keep in mind that a model using CNNs usually has a lot of layers and nodes. Therefore, even in the case of a simple network like k-CNN, which has only 1 layer that uses drop-out, i.e 128 nodes, even with the low drop-out rate of 0.2, the number of possible node's combinations could be up to 1×10^{27} which is more than enough to give a significant output different between models. In the case of VGG-16, the number of combinations is even much higher since it has several of layers that has drop-out. The effect of drop-out is also one of the main reasons for the result in section 6.2.2, whereby using more models, the effect of drop-out on the output got severely reduced.

Diversity in DAS

Based on all the mentioned factor, it is possible to conclude that there is some degree of diversity in DAS. As shown in Figure 7.2 and Figure 7.3, the way the method selects item is quite varied even it is based on only one simple selection criteria, Euclidean distance between outputs.

7.2 Advantage and Disadvantage of DAS

7.2.1 Advantages

Simplicity:

Before evaluating other advantages of the proposed method compared to other batch-mode AL methods, one should look at the biggest strength of DAS, which is the simplicity of its implementation. As shown in Algorithm 1, DAS needs only a few lines of code to be implemented and is also highly modular since the implementation does not need any modification on other parts of the program. Regarding the computational complexity of DAS, there are only two aspects of the method that demands a significant amount of additional computation efforts. The first and most visible aspect is the need to train an additional model. Without using parallelization, this additional model should always lead to at least double the time required to run the method. However, when compared to other methods like Core-set that has an exponential computational complexity or ENS, which uses a least five different models, such an increase in computational complexity is acceptable. Moreover, there is no need to use validation on the second model

since it would reduce the difference between outputs of the main and the second model, which, in turn, might decrease the effectiveness of the method as a whole. Finally, it should be remembered that the second model could also be trained parallel, on the same GPU or another GPU, thus removes the extra time needed.

Another computationally expensive aspect of the method is the selection process. This is closely related to the following factors, *size of the sub-sample pool*, *size of the selected batch* and *choice of distance measurement*. In most of the experiments, DAS worked well with a sub-sample pool size of 150-200 images. In the case of Cifar-10, this is equivalent to 1.5% or 2% of the needed effort to test the models on the test set(10,000 images). Even when the active learner selects 100 of instances on every step (batch-size 100), the computational time of the entire selection process is only equivalent to the time needed to test the models twice, which in many cases, insignificant. Finally, regarding the choice of distance function, Figure 6.20 shown that Euclidean distance was able to give a strong performance compares to other distance methods while being very simple and fast to calculate.

In conclusion, apart from the time that needs to train an additional model, which can be easily parallelized, the time it takes on the selection process of DAS is only depended on the choice of model and is a constant during the entire sampling process.

Stable performance:

Apart from the experiment on Cifar-100 in 6.1.7, DAS was able to deliver a favorable performance compared to random sampling and other mentioned state of the art methods on all the given datasets, even on different architecture(k-CNN and MNIST).

No need for parameter tuning:

In the second part of the experiment 6.2, it has been shown that DAS works best on 2 models, using a pairwise distance function as the distance measurement. While only the randomly selected sub-sample pool size is worth tuning, unless the chosen pool size is too small or too large, the effect of this parameter on the performance of the method is more or less insignificant. (Figure 6.27)

7.2.2 Disadvantages

Bad performance on model with low accuracy:

Because of its dependency on the current state of the model, DAS does not perform well on model with low accuracy. However this is rather hard to see in the presented experiment results. The first part of Figure 6.1 and Figure 6.2 shown that before reaching 77% accuracy (with less than 2500 labeled instances), DAS showed a significantly worse performance compared to other methods. The similar problems can be seen in the first part of in Figure 5.1a, Figure 6.23a, Figure6.24a. Moreover, in Figure 6.25 and Figure 6.17, DAS was only managed to give a similar performance compared to random sampling when the test accuracy is low. This weakness of DAS when the models have low accuracy is understandable, since on uncertain models, the respective outputs of them also become unstable and noisy. This greatly undermines the effectiveness of the acquisition function of DAS which is based on the distance between these outputs. In the case of such noisy model outputs, the performance of DAS will effectively be turned into that of random sampling. However, there are many other representativeness based active learning method that are designed to work well in these early states such as Core-set and ADMA.

Bad performance on Data-set that has a large number of classes:

Similar to outputs of models that have high uncertainty, in the case of classification tasks that have a large number of classes like Cifar-100, the outputs also tend to be very noisy and unreliable. Since DAS uses Euclidean distance to measure the difference between outputs, the sum of the measured pairwise distance between hundreds of items is often a poor choice of acquisition function. There are some ways to combat this drawback such as reduce the dimension of the output function (merge several classes into a superclass) or finding a more suitable way to measure the distances. However, this is still very unlikely that any modified version of DAS would be able to outperform other states of the art method like core-set or ADMA in this case.

Chapter 8

Conclusion

8.1 Answering the Research Questions

This section is to answer the following the research questions from the introduction chapter.

Research question 1 *Does the proposed method perform better than random-selection and other state of the art baseline?*

Based on the evaluation of the experimental results, apart from Cifar-100, **DAS** were able to outperform most on the current baseline in term of accuracy on different well-known dataset. Furthermore, since DAS is a simpler method, it also performs better than other baseline in term of computational complexity and implementation difficulty.

Research question 2 *Why does the proposed method achieve better results than other methods?*

There are many possible answers to this question. Based on the distribution of selected data in Figure 6.4, DAS showed that it was able to select more items from the more difficult classes and fewer items from the easier classes. More interestingly, in the case of Cifar-10, the models that used DAS needs only 25% of the training data from the "ship" class to reach the same class accuracy compared to the model that used random sampling. Furthermore, by having some degree of randomness in its selection strategy and model structure(Drop-out), DAS was able to diversify its selection while maintaining the uncertainty based focus using a single acquisition function. Another explanation of why DAS performs better

than other uncertainty based methods is possibly because DAS does not focus only on output uncertainty in its selection, which might turned out to benefit the learning process as discussed in the previous section.

Research question 3 *What is the best way to implement the method with regards to parameter choice?*

In section 6.2, the experimental results have shown that a simple version of DAS with 2 models, Euclidean distance and a reasonable sub-sample size between 200-1000 is the best way to implement the method in term of cost-effectiveness.

8.2 Conclusion

This thesis proposed a new method that works well in the challenging field of batch-mode active learning for deep image classification. The proposed method, dual active sampling, was able to give good results on the datasets like MNIST, Cifar-10, and SVHN(Google street view house number). Furthermore, the method also demonstrated the capability to diversify its selection while maintaining the sampling focus on more difficult classes. While the method showed several weaknesses including an indifferent performance on the dataset that has too many classes, its simplicity and its favorable computational complexity gives it a sufficient competitive advantage compared to any "state of the art" methods in batch-mode active learning.

This thesis also studied the effect of different parameter choice when implementing DAS. Based on the available results from the tables 6.8, 6.21), it is possible to draw a conclusion that it is best to use pairwise distance to measure the difference between output in DAS. In the case of sub-sample size, a sub-sample between 200-1000 images is possibly the best choice as shown in section 6.2.3.

While it is possible to increase the complexity of DAS by increase the number of models, section 6.2.2 shows that such approach is not a good idea and might reduce the performance of the method. In the discussion, the selection behavior of DAS was analyzed and most notably, in Figure 7.2, the plotted data showed that for most of the time, DAS selects item from the cases where both of the models are certain. However, it is still unclear how this selection behaviour affects the performance of the method.

8.3 Contributions

This thesis proposed a simple, fast and effective active learning method that works well in batch-mode active learning settings. The performance of the proposed method and its variation was demonstrated in many experiments, and the results indicate that it has a comparable if not better performance in terms of accuracy per number labeled data than most of the current active learning methods in the field of image classification. Furthermore, by studying the reason behind the good performance of DAS, different factors that could contribute to the success of a batch-mode active learning method in the field of image recognition using deep learning has been analyzed. More importantly, because of its simplicity, DAS could be implemented by anyone with little effort therefore, makes it easy for practical use in the field of image recognition compared to other methods. By using DAS instead of random sampling, one could save up to 50% labeling effort without any significant effort.

While DAS might work as a completed active learning method on its own, the true goal of this thesis is to show that the distance between two identical networks that trained on the same dataset from the beginning would be a good selection criterion for a combined acquisition function in active learning. DAS can also, in many cases, be used as a simple critics function to verify the performance of other active learning methods. While the complexity of active learning methods kept rising in the recent years since deep learning problems has become harder and harder to predict and analyze, the impressive performance of a simple and naive method like DAS has shown that active learning in deep image recognition task should be treated differently compared to classical active learning theory. Instead of trying to reduce the black box property of deep learning when doing active sampling, there might be better to accept it and utilize its randomness in favor of the user.

8.4 Future Work

Since DAS is a simple and naive method, there are many possible ways to improve its performance. The first way is to incorporate some kind of representativeness measurement in the acquisition function, such as variance ratio or something similar to core-set. The second way to use different models with different structures instead of 2 identical ones, which should increase the average output distances between the models. The third way, which focuses more on performance speed is to allow the method to select multiple samples from a sub-sample pool at once,

thus significantly speed up the selection process without having any significant consequences on the overall performance.

The uses of DAS as the critic or as a part of the Q function in a reinforcement learning based active learning strategy should also be considered since DAS is a fast and simple method. Moreover, as a new concept in Batch-mode active learning for deep CNNs, DAS should be further tested on different experiments to discover the boundary of the method. Experiments on how the number of classes affects the performance of DAS compares to other active sampling methods should also need to be considered since it is not known yet if DAS would work on binary classification problems. Finally, the use of active learning methods which based on similar assumptions like DAS should be considered in other fields of deep learning such as language recognition or object detection in 3d.

Bibliography

[1] Burr Settles. *Active learning*. Morgan & Claypool Publishers, 2012.

[2] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *ICLR 2018 Conference*, 2018.

[3] Pankaj K Agarwal and Micha Sharir. Planar geometric location problems. *Algorithmica*, 11(2):185–195, 1994.

[4] Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. Cost-effective active learning for deep image classification. *CoRR*, abs/1701.03551, 2017. URL `http://arxiv.org/abs/1701.03551`.

[5] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, Oct 1990. ISSN 0162-8828.

[6] Simon Tong. *Active learning: theory and applications*, volume 1. Stanford University USA, 2001.

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

[8] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[10] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual

Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[11] Johanna Pingel. Introduction to deep learning what are convolutional neural networks. https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks–1489512765771.html [Accessed: 26.11.2018].

[12] westworld. An illustration of a 2D convolution. http://www.westworld.be/page/2/ [Accessed: 26.11.2018].

[13] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. URL `http://arxiv.org/abs/1311.2901`.

[14] John S. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. pages 211–217, 1990. URL `http://papers.nips.cc/paper/195-training-stochastic-model-recognition-algorithms-as-networks-can-lead-pdf`.

[15] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv e-prints*, September 2014.

[16] Imad Dabbura. Gradient descent algorithm and its variants, 2017. URL `https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3`.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL `http://arxiv.org/abs/1502.03167`.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

[19] Nicholas Roy and Andrew McCallum. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, pages 441–448, 2001.

[20] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

[21] Pabitra Mitra, CA Murthy, and Sankar K Pal. A probabilistic active support vector learning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):413–418, 2004.

[22] Yoav Freund, H Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28 (2-3):133–168, 1997.

[23] Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. Multi-criteria-based active learning for named entity recognition. page 589, 2004.

[24] S.-J. Huang, J.-W. Zhao, and Z.-Y. Liu. Cost-Effective Training of Deep CNNs with Active Model Adaptation. *ArXiv e-prints*, February 2018.

[25] Sachin Ravi and Hugo Larochelle. Meta-learning for batch mode active learning, 2018. URL `https://openreview.net/forum?id=r1PsGFJPz`.

[26] Melanie Ducoffe and Frédéric Precioso. Adversarial active learning for deep networks: a margin based approach. *CoRR*, abs/1802.09841, 2018. URL `http://arxiv.org/abs/1802.09841`.

[27] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. pages 287–294, 1992. URL `http://doi.acm.org/10.1145/130385.130417`.

[28] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statistics*, 22:79–86, 1951. ISSN 0003-4851. doi: 10.1214/aoms/1177729694. URL `https://doi.org/10.1214/aoms/1177729694`.

[29] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. *CoRR*, abs/1703.02910, 2017. URL `http://arxiv.org/abs/1703.02910`.

[30] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015.

[31] William R Knight. A computer method for calculating kendall's tau with ungrouped data. *Journal of the American Statistical Association*, 61(314): 436–439, 1966.

[32] William J Cook, WH Cunningham, WR Pulleyblank, and A Schrijver. Combinatorial optimization. *Oberwolfach Reports*, 5(4):2875–2942, 2009.

[33] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2018. URL `http://www.gurobi.com`.

[34] Ozan Sener. Active learning via core-sets. `https://github.com/ozansener/active_learning_coreset`, 2018.

[35] William H. Beluch, Tim Genewein, Andreas Nurnberger, and Jan M. Kohler. The power of ensembles for active learning in image classification. *The IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[36] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`. [Online; accessed 6.10.2019].

[37] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

[38] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). *CVPR 2009*, 2009. URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

[39] Popular datasets over time, 2019. URL `https://www.kaggle.com/benhamner/popular-datasets-over-time/comments`. Accessed: 05.07.2019.

[40] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011. URL `http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf`.

[41] Keras Team. Mnist cnn, 2018–. URL `"https://keras.io/examples/mnist_cnn/"`. [Online; accessed 6.10.2019].

[42] Mark Mitchell. The engauge digitizer tool, version 11.2. `https://markummitchell.github.io/engauge-digitizer/`, 5/1/2019.

[43] Massimiliano Ruocco Johan Phan and Francesco Scibilia. Batch-incremental method in activelearning for image classification. *Specialization Project,Department of Cybernetics and Robotics, NTNU*, 2018.

[44] Bai. Strange behavior with adam optimizer when training for too long, 2017–. URL `"https://datascience.stackexchange.com/questions/25024/strange-behavior-with-adam-optimizer-when-training-for-too-long"`. [Online; accessed 6.10.2019].

[45] R. R. SOKAL and P. H. A. SNEATH. Principles of numerical taxonomy. 1963.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

[47] J. Shijie, W. Ping, J. Peiyi, and H. Siping. Research on data augmentation for image classification based on convolution neural networks. pages 4165–4170, Oct 2017. doi: 10.1109/CAC.2017.8243510.

[48] Kenji Kawaguchi and Leslie Pack Kaelbling. Every local minimum is a global minimum of an induced model. *arXiv preprint arXiv:1904.03673*, 2019.

[49] Yazhou Yang and Marco Loog. A variance maximization criterion for active learning. *Pattern Recognition*, 78:358 – 370, 2018. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2018.01.017. URL `http://www.sciencedirect.com/science/article/pii/S0031320318300256`.