An Attempt to Reduce the Number of Training Samples for Convolutional Neural Network

Yuwen Heng

Master of Science
Data Science, Technology, and Innovation
School of Informatics
University of Edinburgh
2020

Abstract

Training deep neural networks can be resources-consuming. The budget required is increasing with the size of the dataset. During the past few decades, many research is dedicated to developing training procedures to accelerate the convergence speed of deep learning. However, we still need the whole dataset to train the network and paying for a large dataset may not pay back well if we can use a smaller subset to achieve an acceptable performance. To solve this issue, we first adapted and evaluated three methods, Patterns by Ordered Projections (POP), Enhanced Global Density-based Instance Selection (EGDIS), and Curriculum Learning (CL), to reduce the size of two image datasets, CIFAR10 and CIFAR100, for the classification task. Based on the analysis, we present our two contributions: the Weighted Curriculum Learning (WCL) and a trade-off framework. The WCL outperforms POP and EGDIS in terms of both classification accuracy and time complexity. It achieves comparable performance compared with CL while keeping a portion of hard examples. The trade-off framework selects a subset of samples according to the acceptable relative accuracy and the dataset. In addition, the framework is also extended to predict the number of samples needed to achieve a particular accuracy with a given subset.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr Yang Cao, for offering me the opportunity to work with him on such an attracting and challenging project. His encouragement and valuable guidance helped me tackle the obstacles in my research path.

Furthermore, special thanks go to Professor Bob Fisher, Dr Pavlos Andreadis at the University of Edinburgh and Dr Jiacheng Ni at IBM for sharing me their knowledge about computer vision and deep learning. The programming skills and coursework experience that I learnt from them helped me to organise the experiments well.

Finally, I would like to send my love to my fiancee Danni Li for her accompany during the past three years . I wouldn't have the chance to study full-time without her full support.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

1.2 Research Goal

1.3 Significance

1.4 Beneficiaries

Chapter 2

Background Research

In this chapter, we begin with presenting the necessary background to understand the supervised learning and data reduction methods, as well as, other ideas required to understand our research method. We start with the structure of CNN and the training procedure. We then discuss the modern subset selection methods that can speed up the training procedure and outline their deficiencies. Next, we review the data reduction literature and present a CNN data reduction framework - use the network pre-trained on ImageNet to extract low-dimensional features and run the data reduction methods on extracted features. Furthermore, we cover the existing trade-off framework BlinkML [21] in the context of maximum-likelihood estimation machine learning algorithms and explain why it is not suitable for deep neural network. Finally, we present TAPAS [10], which is an accuracy predictor for deep neural network without training and has several properties that make it useful to build our trade-off framework.

2.1 Supervised Learning with CNN

CNN based supervised learning is a kind of machine learning task which learns the mapping between input visual data and output based on a set of well-labelled training samples. The visual data can be images, videos or even 3D models [24]. The output score after the softmax operation can be considered as the probability for a given image belongs to each class, $P(class|image, network)$. The CNN itself can be considered as a set of chained operations with trainable parameters. These parameters define the actual input-out mapping. For this reason, we use the symbol $f(x|\theta)$ to represent the output score predicted by the CNN which takes the input x with a particular parameter set θ. Figure 2.1 gives a basic CNN structure which is designed to classify images as cats

or dogs. It contains two convolutional (Conv) layers, one max pooling layer and one fully connected (FC) layer. The FC layer is actually a multi-class logistic regression model which maps the outputs of the max pooling layer to the class scores. From this perspective, we can divide the CNN structure into two parts: feature extraction part and logistic regression part. The feature extraction part performs as a blackbox which transforms the input images to points in a lower-dimensional, linearly separable space.

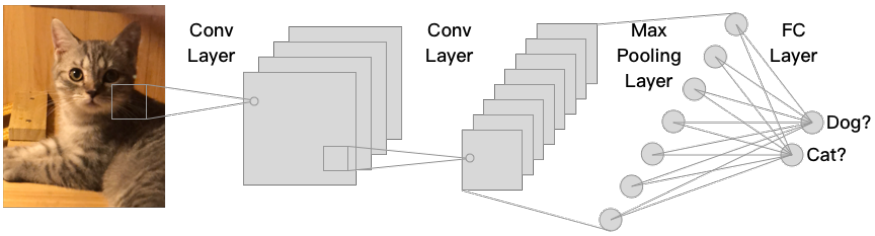


Figure 2.1: A basic CNN structure to classify images between cats and dogs. The outputs of the penultimate layer are extracted lower-dimensional features of the input images. These features should be linearly separable to achieve a high classification accuracy

If we use the symbol y to represent the ground truth of the input sample x, use $L(f(x|\theta),y)$ to represent the loss function which measures the difference between the predicted output and the ground truth label, then the training process is to find the parameter set θ^* which minimise the average loss of the whole training set as fellows:

$$\theta^* = \arg\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} L(f(x_i|\theta),y_i) \tag{2.1}$$

where the symbol N stands for the number of samples in the training set. This turns the training process into an optimisation problem. Different from machine learning algorithms like logistic regression and support vector machine, the equation 2.1 is non-convex thus cannot be solved analytically [8, p. 304]. A number of techniques have been developed to solve the problem with the requirement that the loss function $L(.,.)$ is continuous. The basic one to train on large dataset is called stochastic gradient descent (SGD) which updates the parameters with the partial derivatives of a randomly selected sample. At each step, the new parameter is calculated with

$$\theta_{t+1} = \theta_t - \eta\frac{\partial L(f(x|\theta),y)}{\partial\theta_t} \tag{2.2}$$

and η is the step size whose typical value is between 0.1 to 0.001. A simple variant of SGD is mini-batch gradient descent which divides the training set into disjoint subsets

and averages the gradients within the subset before updating the parameters:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{M} \sum_{i=1}^{M} \frac{\partial L(f(x_i|\theta), y_i)}{\partial \theta_t} \tag{2.3}$$

where M is the batch size of the subset. For CNN, batch size M is often smaller than the training set size N because it takes too much memory to fit the whole dataset. Usually we use 128 or 256 as the batch size.

2.2 Mini-batch Sampling

Since the mini-batch gradient method trains the network with a subset of samples at each step, how to select the samples becomes a problem in the deep learning literature. Instead of uniform sampling, many researchers proposed to rank the samples with importance score or classification difficulty and selects a mini-batch based on different criteria. According to [9], we can divide the sampling methods into two categories: **current hypothesis method** and **targer hypothesis method**. Current hypothesis method measures the samples based on the parameter set θ_t at step t while targer hypothesis method is based on the final parameter set θ^*.

2.2.1 Current Hypothesis Method

Different authors have proposed a variety of current hypothesis methods. Specifically, in self-paced learning [15, 16, 19], active bias learning [5], and hard example mining [23, 17], the scores are calculated based on the sample difficulty, which is proportional to the classification score of the true class, $f(x|\theta, y)$. For importance sampling methods, the scores are calculated based on the gradient norm for each sample, $|\frac{\partial L(f(x_i|\theta), y_i)}{\partial x_i}|$. We finish this section by briefly explaining the actual implementations of these approaches.

2.2.1.1 Difficulty Based Method

Self-paced learning method tends to select easy samples which have a high classification score by injecting a pace function into the optimisation target function 2.1:

$$\theta^* = \arg\min_{\theta, v} \sum_{i=1}^{N} v_i L(f(x_i|\theta), y_i) + \lambda \sum_{i=1}^{N} v_i \tag{2.4}$$

where v is the score calculated by the pace function. The pace function can be either a simple step function [15] or a more complicated dynamic function which changes with step t [16] as long as it can assign value 0 to samples. By minimising the target function 2.4, the method would zero out hard examples which have higher loss L thus keep only the easy samples. This makes the trained network more robust to outliers [19].

A potential problem of self-paced learning is that it would gradually increase the loss of hard examples [5]. The possible solution is to use the active bias learning method, which is designed to select the uncertain samples whose classification score vary near the decision threshold. Chang et al. proposed and evaluated many self-paced methods and the representative one is called SGD Sampled by Threshold Closeness (SGD-STC) [5]. It records the historical average classification probability $\bar{P}$ for each sample and the score is calculated with a equation that is proportional to $(1-\bar{P}) \times \bar{P}$. However, the problem is that we need extra space and computation to maintain the historical scores.

Hard example mining is yet another heuristic method aims at maximising the convergence speed by extending the self-paced learning method [23]. The algorithm proposed by [17] ranks the samples based on the latest computed classification score in descending order. At early training stages, the algorithm chooses easy samples just like self-paced learning. After a thorough exploitation process, the algorithm tends to select hard examples which have low classification scores.

2.2.1.2 Importance Based Method

Although published experiments in the cited resources above prove that difficulty based methods can surely speed up the training process and may achieve even higher accuracy, the lack of mathematical prove could lower the interests of researchers. In the contrary, importance based method raises from the profound mathematical demonstration [26] and is more reliable. Despite the elaborate derivation, the most important conclusion is that the optimal weight distribution is proportional to the per sample gradient norm.

The challenge is that computing the per sample gradient norm $|\frac{\partial L(f(x_i|\theta),y_i)}{\partial x_i}|$ is intractable. In the past few years, many researchers have adapted their approximate methods to speed up the process. The most convincing one is proposed by Katharopoulos et al. which derives an upper bound of the gradient norm [11],

$$|\frac{\partial L(f(x_i|\theta), y_i)}{\partial x_i}| \leq |h(x_i)| \tag{2.5}$$

that $h(x_i)$ is the upper bound function depends on the last layer pre-activation outputs and time step t. With this equation, we can compute the largest sample gradient after a single forward propagation.

The benefits of current hypothesis methods is that the sample importance varies with time step thus the chosen samples at each step can reflect the current capacity of the network. However, because evaluating the whole training set is time-consuming, we often select a subset uniformly first and then select the samples within the subset. This would affect the optimal theory performance.

2.2.2 Target Hypothesis Method

Compared with current hypothesis method, target hypothesis method selects instances based on the possible final performance of the network thus the weights of the samples are pre-defined and won't change during the training process [3]. For this reason, target hypothesis methods are more suitable to reduce the size of the dataset. To our knowledge, Curriculum Learning (CL) is the only method with these properties.

Similar with hard example mining, CL trains the network with easy samples first then adds more difficult samples into the dataset and gradually the subset would contain all the training samples. The main difference is that the difficulties of the samples are measured in advance, whether with a pre-trained network or with a linear classifier like SVM [9].

We use extracted features from NasNetLarge to train a simple two FC layer network. The output score is recored and used to select samples in descending order.

Algorithm 1: CL

Data: compressed 128-D feature vectors M

Input: number of samples to select m, classification score for each sample *scores*, number of classes n

Output: selected sample index by CL

```
selected_idx_list = [] ;
foreach class label L do
    scores = all sample scores with label L ;
    idx_list = sort_by_value(scores) ;
    selected_idx_list.append(idx_list[: floor(m/n)]) ;
end
return selected_idx_list ;
```

2.3 Data Reduction Algorithm

Data reduction algorithm becomes famous with the increasing of dataset size and compute time. The most common one is called Principal Components Analysis (PCA) which projects the features onto the most important few eigenvectors to capture the most variants. The famous example eigenface proves that PCA is efficient even with image dataset [12]. Apart from PCA, there are other methods such as dimensionality reduction methods which reduce the size of the sample features and instance selection methods that can reduce the number of samples in the training set. We discuss some typical implementations in the next subsections.

2.3.1 Dimensionality Reduction Methods

As discussed above, PCA is the most common method for both structured dataset and unstructured dataset. In the context of deep CNN, however, the term feature extraction is more famous as discussed in section 2.1. Kornblith et al. evaluated the extracted features with pre-trained networks on many vision datasets and the results show that the logistic regression accuracy is linearly related to the ImageNet classification accuracy [13].

2.3.2 Instance Selection Methods

Most instance selection methods are designed to reduce the size of structured dataset for machine learning algorithms like SVM and logistic regression with the assumption that we can recover the decision boundary with fewer samples. According to the thorough review [20], the instance selection methods can be divided in to two categories: wrapper and filter. Wrapper methods select the subset samples based on the classification results. Often misclassified samples will be selected because they can contribute to the accuracy of classifier like k-NN [2]. Filter methods tend to select samples near the decision boundary because these samples can help the classifier to recover the original shape [22]. Although the wrapper methods could achieve higher accuracy because they are classifier dependent, they tend to cost too much compute time because they need to evaluate the accuracy multiple times. From this perspective, filter methods are more suitable for deep learning.

The POP algorithm is designed to select inner samples by projecting the samples onto each feature dimension.

Algorithm 2: POP for continuous features

Data: compressed 128-D feature vectors M

Input: weakness threshold wt, equal tolerance et

Output: selected sample index by POP

$weakness = zeros(len(M))$;
foreach *feature dimension* $F_j = M[:,\ j]$ **do**
    $idx\_list = sort\_by\_value(F_j)$;
    $idx\_list = resort\_by\_label(F_j,\ et,\ idx\_list)$;
    foreach *idx in idx_list* **do**
        if *M[idx, j] is not border* **then**
            $weakness[idx] += 1$;
        end
    end
end
return $argwhere(weakness < wt)$;

EGDIS selects the boundary samples with the 3-NN algorithm and the samples at the densest area.

Algorithm 3: EGDIS

Data: compressed 128-D feature vectors M

Input: number of neighborhoods k

Output: selected sample index by EGDIS

```
boundary_idx = [] ;
densest_idx = [] ;
neighbour_distance_list, neighbour_index_list = kneighbours(M, k) ;
for i in range(len(neighbour_index_list)) do
    neighbour_index = neighbour_index_list[i, :] ;
    irrelevance_score = irrelevance(neighbour_index) ;
    if irrelevance_score is greater or equal to floor(k/2) then
        boundary_idx.append(k) ;
    else
        if density(M[i]) is larger than density(M[neighbour_index]) then
            densest_idx.append(i)
        end
    end
end
return union1d(boundary_idx, densest_idx) ;
```

2.4 Trade-off Framework

2.5 Accuracy Predictor

Chapter 3

Adapted Data Reduction Methods

In this chapter, we begin by presenting the experimental datasets, CIFAR10 and CIFAR100 along with the image feature extraction process. Next we adapt three methods overviewed in Chapter 2 to reduce the size of image dataset, called the Patterns by Ordered Projections (POP) [22], Enhanced Global Density-based Instance Selection (EGDIS) [18], and Curriculum Learning (CL) [9]. Then we propose our edited data reduction method, called Weighted Curriculum Learning (WCL), based on CL scores and Boundary Based Curriculum Learning (BBCL), based on the EGDIS selected boundary instances. After that, our work is focused on the comprehensive evaluation of the methods. We illustrate the data reduction geometry patterns with three generated datasets, blobs, moons and circles. We then describe the model fitting procedure of the logistic regression algorithm. Finally, we extend the reduction pipeline to deep learning method with the particular network, DenseNet121. Figure 3.1 gives the pipeline overview of this project.

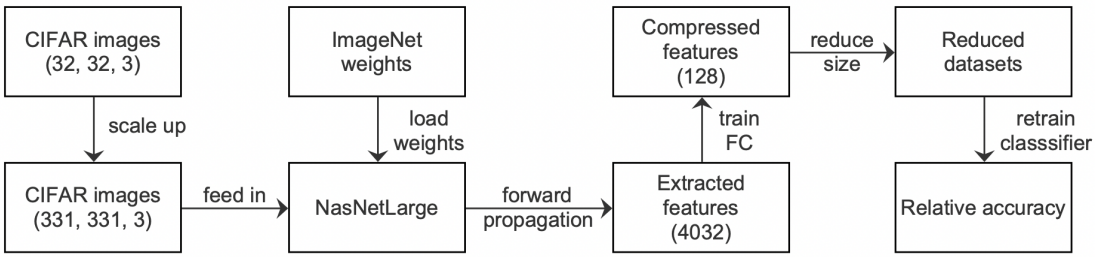


Figure 3.1: Overview of the data reduction pipeline.

3.1 Datasets and Image Feature Extraction

We choose to use CIFAR10 and CIFAR100 [14] as our experimental datasets which contain 6,000 and 600 tiny images of size 32×32 per class respectively. The advantage of CIFAR is that they are large in the number of images and small in the size of images. With CIFAR datasets, we could train the network faster thus explore the reduction rate for a wider scale within the required timetable. Another advantage is that CIFAR datasets can reflect the performance of data reduction algorithms for both simple dataset and hard dataset in term of classification accuracy. According to Kornblith et al. [13], the test set results indicate that CIFAR10 is very easy to classify and CIFAR100 is as difficult as other high resolution datasets such as the Describable Textures Dataset (DTD) [6], Food-101 [4]. These features could gain us thorough and representative evaluation results with limited compute resources.

After scaling up the image size to 331 and transforming the images into range 0 to 1 by dividing 255, we therefore performed the feature extraction task. The goal was to provide structural data for the data reduction algorithms to work with. We did this job with pre-trained NasNetLarge [27] because Kornblith et al. [13] have evaluated the quality of extracted features and the quality of extracted features is good enough. Their experiments show that the classification scores with simple logistic regression are very close to the state-of-art classification scores with CNN. In order to simplify the implementation, we chose to use the Keras implemented NasNetLarge network from TensorFlow Hub, which is designed to get feature vectors from images [1]. However, the original shape of the feature vector is 4032 and it would take longer time to run the reduction algorithms. To speed up the reduction process, we trained another network with two FC layers. The depth of the first FC layer is 128 and we took the outputs as compressed feature vectors. The test accuracy is also reported as the baseline performance. Figure 3.2 represents the network structure. We also trained a batch normalisation layer after the 128-D FC layer to limit the feature range. This ensured that the Euclidean distance between two vectors wouldn't be dominated by dimensions with wider range.

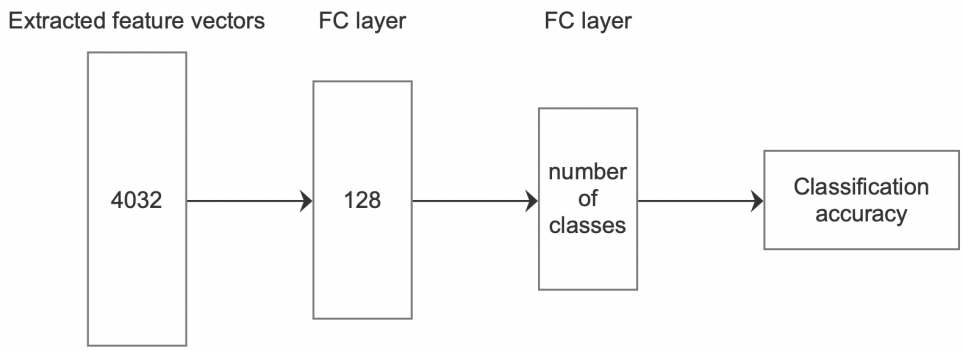


Figure 3.2: Network structure to compress the extracted feature vectors.

3.2 Difficulty Tuneable Algorithms

Before presenting the evaluation plans, it should be noted that the algorithms described in 2.2.1.2 are not perfect. First of all, POP and EGDIS are not deep learning based algorithms so the CNN may not work well with selected samples. Also, although CL is target hypothesis based algorithm, by keeping easy samples only may limit the highest performance that the network could achieve. Therefore, our first contribution is to enhance CL with the ability to contain a tuneable proportion of relatively difficult samples. We proposed two variations:

1. Weighted Curriculum Learning, which select samples according to classification scores.

2. Boundary Enhanced Weighted Curriculum Learning, which selects a proportion of the EGDIS boundary samples first then fill in the subset with easier samples.

We followed the requirement in [9] to select balanced subsets.

3.2.1 Weighted Curriculum Learning

Instead of selecting the top N samples after sorting the samples in ascending order based on classification scores, we normalised the scores as the survival probability. By dividing the sample score to the sum of all scores, the sum of all normalised scores will be 1 so that we can tread them as probability. In this way, sample with higher classification score would have higher probability to be selected. Therefore, not only the easy samples are selected, some hard examples are also selected. Because we only select the subset once, this behaviour should be able to achieve higher accuracy if

the dataset is harder to classify and the network is powerful to learn from these hard examples. However, if the network is not capable of handling these hard examples, then the test accuracy may decay.

Algorithm 4: WCL

Data: compressed 128-D feature vectors M

Input: number of samples to select m, classification score for each sample *scores*, number of classes n

Output: selected sample index by WCL

```
selected_idx_list = [] ;
foreach class label L do
    scores = all sample scores with label L ;
    scores = scores / sum(scores) ;
    idx_list = choose floor(m/n) samples based on scores ;
    selected_idx_list.append(idx_list)) ;
end
return selected_idx_list ;
```

3.2.2 Boundary Based Weighted Curriculum Learning

While WCL can reflect the difficulty of the datasets, it cannot guarantee to select enough hard examples for the network to mine the pattern. Therefore, we proposed the Boundary Based Weighted Curriculum Learning method to tune the amount of difficult samples in the selected subsets.

Algorithm 5: BWCL

Data: compressed 128-D feature vectors M

Input: number of samples to select m, classification score for each sample *scores*, number of classes n, EGDIS boundary sample index list *egdis_boundary_index*, percent of boundary to select p

Output: selected sample index by BWCL

```
selected_idx_list = [] ;
foreach class label L do
    scores = all non-EGDIS sample scores with label L ;
    egdis_boundary_index_L = all EGDIS boundary sample index with label L ;
    egdis_idx = choose floor(m/n × p) samples from egdis_boundary_index_L ;
    selected_idx_list.append(egdis_idx)) ;
    scores = scores / sum(scores) ;
    idx_list = choose floor(m/n× (1-p)) samples based on scores ;
    selected_idx_list.append(idx_list)) ;
end
return selected_idx_list ;
```

3.3 Evaluation Designs

We have the following few experiments to evaluate the performance of the chosen three and the two proposed algorithms:

1. We extracted the feature vectors for CIFAR10 and CIFAR100. We visualised the extracted features with t-SNE.

2. We use CIFAR10 to explore the intrinsic behaviour of these methods. We reported the extraction time for both CIFAR10 and CIFAR100.

3. We use logistic regression to test all five methods. We also use the experiment results to decide how to select the CIFAR subsets.

4. We used these four datasets to evaluate them CNN, DenseNet121.

Chapter 4

Data Reduction Evaluations

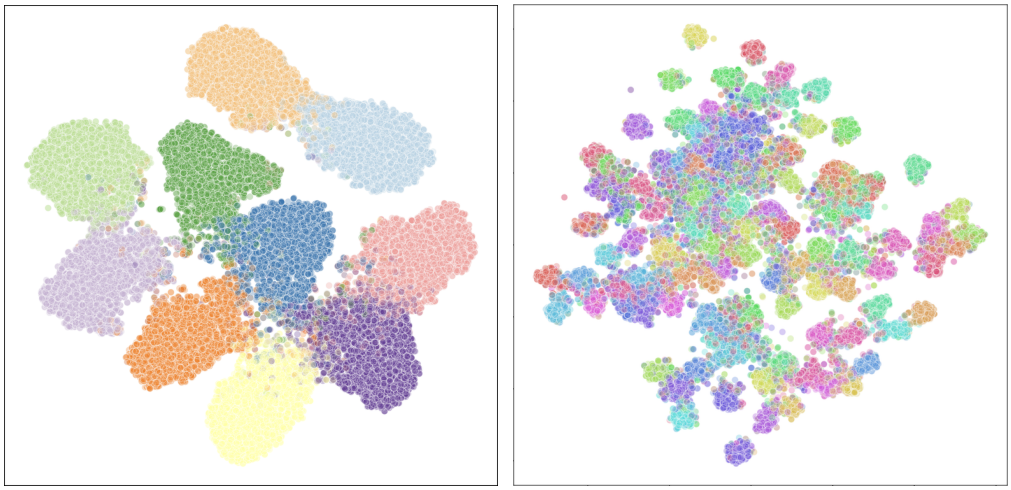
4.1 Experiment 1: Feature Extraction

The python version CIFAR10 and CIFAR100 datasets are downloaded from the official website with pre-defined train and test sets. The train set were split into train and validation sets with the ratio 8:2 randomly. We got 40,000 train samples, 10,000 validation samples and 10,000 test samples for both datasets. In order to speed up the experiments, we started by extracting the image feature vectors with a single forward propagation before training the FC layers.

All trainable parameters of the two FC layers were initialised by the Xavier uniform method [7]. The training process was monitored by the validation accuracy. After each epoch, we reported the validation accuracy and kept a record of the model parameters which achieved the best validation accuracy so far. We scheduled two training stages to approach the optimal classification accuracy. First we performed 500 training epochs with step size 0.01 then decreased it to 0.001 for another 500 epochs. After each training stage, the recorded parameters were loaded to prevent overfitting. The final held-out test set accuracy for CIFAR10 is 0.9258 and for CIFAR100 is 0.7444. Finally we compressed the extracted 4032-D feature vectors with the 128-D FC layer.

We further took a closer look at the compressed features by projecting them down to two dimensional vectors with t-SNE [25] and maintains the relative distance between samples. The plots are shown in Figure 4.1. Different colours represent different classes of samples. First of all, the network transformed the images into blobs where ideally samples from the same class should be closer to each other. This is true especially for CIFAR10. The quality of these compact blobs reflects the classification accuracy. Strays far from the cluster would be misclassified by the logistic regression

layer thus lower the classification accuracy. Furthermore, the boundaries between different classes are not always clear. Blobs are close to each other near the boundary thus the classification accuracy is sensitive to slight variations of the boundary. The effect is more obvious if there are more samples for each class. This partly explains the rapid vibration of the CIFAR10 validation accuracy curve shown in Figure 4.2(a). In fact, we can widen the distance between adjacent blobs by fine-tuning the pre-trained feature extraction network and achieve a higher classification score [13]. However, since we want to minimise the pre-processing time required, we omitted this step and stayed with the sub-optimal features.



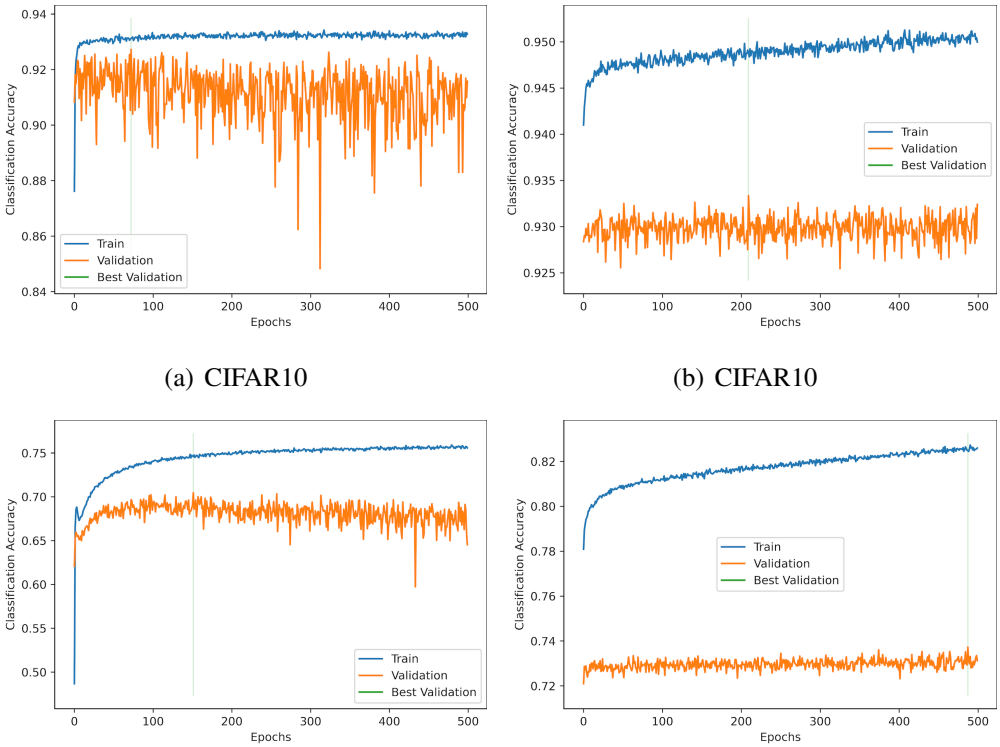
(a) CIFAR10 (b) CIFAR100

Figure 4.1: Visualisation of extracted features with the t-SNE algorithm.

In addition, we compared the training process of the two datasets in Figure 4.2. Across the two training stages, CIFAR10 converges faster than CIFAR100 but it starts to overfit earlier. The green line indicates that we could reduce the number of epochs in the first training stage by half and leave the mining job at the second stage.

Finally, we also visualised the classification distributions in Figure 4.3. As observed from the quality of blobs plotted in Figure 4.1, CIFAR10 samples tend to achieve higher classification scores around 0.9 while CIFAR100 samples are spread across the range below score 0.9. Samples from CIFAR100 are harder to classify correctly with lower scores. Based on the sample selection procedures described in Chapter 2, we may need more samples for CIFAR100 to achieve a similar relative accuracy compared with CIFAR10. This suggests that data reduction algorithms may be less efficient for datasets with more classes and lower classification accuracy. We will

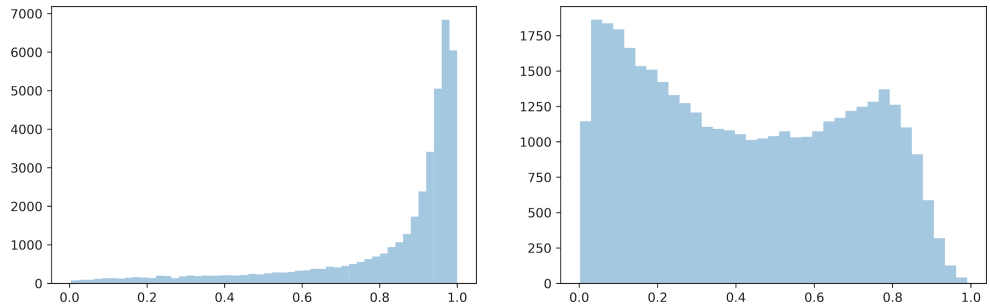
discuss the algorithm performance further in Section 4.3 and 4.4.



(a) CIFAR10 (b) CIFAR10

(c) CIFAR100 (d) CIFAR100

Figure 4.2: The training history of CIFAR10 and CIFAR100. The left column is the first 500 epochs and the right column is the second 500 columns. The green line represents the best validation epoch.



(a) CIFAR10 (b) CIFAR100

Figure 4.3: Classification score distributions

4.2 Experiment 2: Intrinsic Behaviour

Obtaining the discriminative t-SNE plots, we assumed the FC layers to have learnt high quality compressed features. In this experiment, we tried to get an understanding of the intrinsic behaviours of these data reduction algorithms by visualising the selected CIFAR10 samples with red points. We managed to explain the behaviours with the answers to two questions:

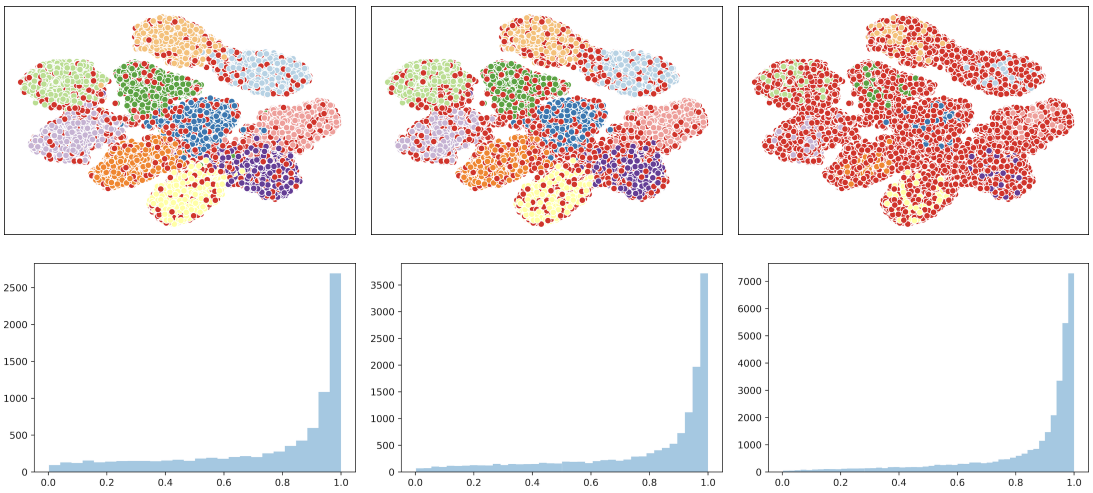
1. What are the geometrical distributions of the selected samples?

2. What are the classification score distributions of the selected samples?

For each algorithm, the selection behaviours were further explored by tuning the hyper-parameters. We chose to use CIFAR10 for the reason that the quality of t-SNE projected blobs are more discriminative than CIFAR100 thus we can understand the behaviours easily.

First of all, we present the POP selected vsamples by adjusting the equal tolerance of the *resort_by_label* function described in Algorithm 2. For each tolerance choice, we removed only pure inner samples whose weakness values are 128. Figure 4.4 depicts that with proper threshold setting such as $et = 1$, we can select most samples along the blob contours as desired. Also, from Figure 4.4(a) and 4.4(b), we observe a trend to select samples closer to the boundary between two adjacent blobs rather than cover the whole contours. This behaviour can guide linear classifiers to build the proper decision boundaries. Another advantage of POP is the fast compute speed. It took us 33.27 seconds on average to get the result. However, POP was designed to process integer features. It is difficult to choose the suitable tolerance value and keep just enough samples. Our experiments indicate that the number of samples far from the boundary decreases much faster than expected by increasing the tolerance value. One possible reason is that we normalised the compressed features so the value range is too compact. This makes POP hard to use for image datasets.

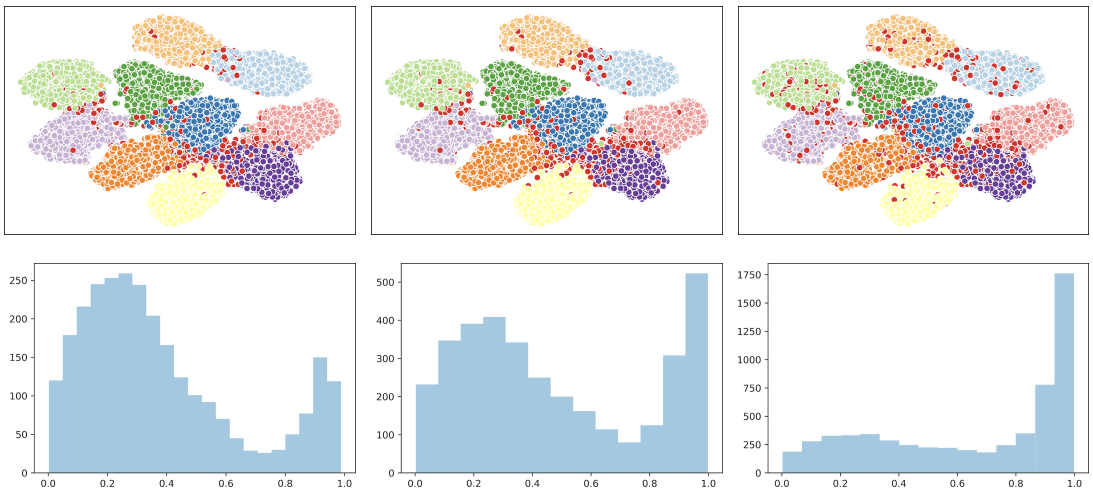
The second algorithm we explored is EGDIS by tuning the hyper-parameter k of the *knn* classifier. Figure 4.5 presents how the selection pattern varies with k values 3, 5, and 7. Our first discovery is that EGDIS tends to select all misclassified samples together with samples surrounding them. Compared with Figure A.1, only a few samples still exist with different colours. Also, by increasing the value of k, fewer inner samples and boundary are selected. To explain these results, we need to know how EGDIS works with hyper-parameter k. In short, higher k values require boundary



(a) T=1, 8701 samples, 33.3s (b) T=0.5, 14111 samples, 29.7s (c) T=0.1, 31966 samples, 36.8s

Figure 4.4: POP tune threshold

samples to have more neighbours from different classes. Based on this mechanism, we can infer that most misplaced samples are alone and surrounded by correctly classified samples. For these isolated samples, they would be considered as boundary samples and have enough number of neighbours to be selected. For samples near the boundary, fewer samples will be qualified and only those closely contacted with other blobs would be selected. For dense samples near isolated misplaced points, some of them may be categorised as boundary samples and wouldn't be selected anymore.

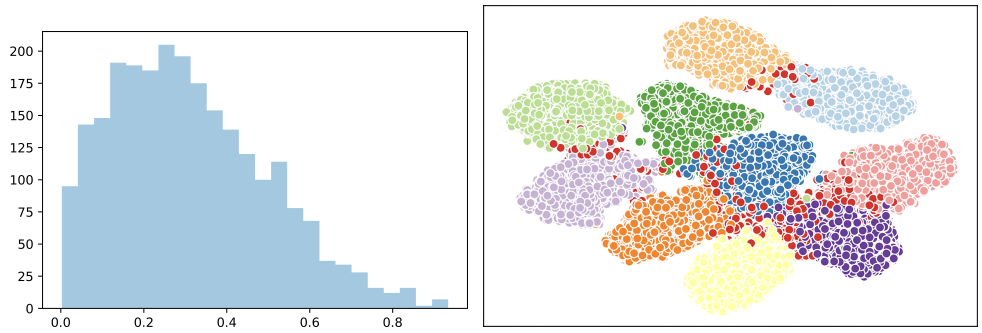


(a) k=7, 2799 samples, 14min 27s (b) k=5, 3483 samples, 15min 7s (c) k=3, 5966 samples, 14min 56s

Figure 4.5: EGDIS tune kNN

The drawback of EGDIS is the global density calculation which computes the dis-

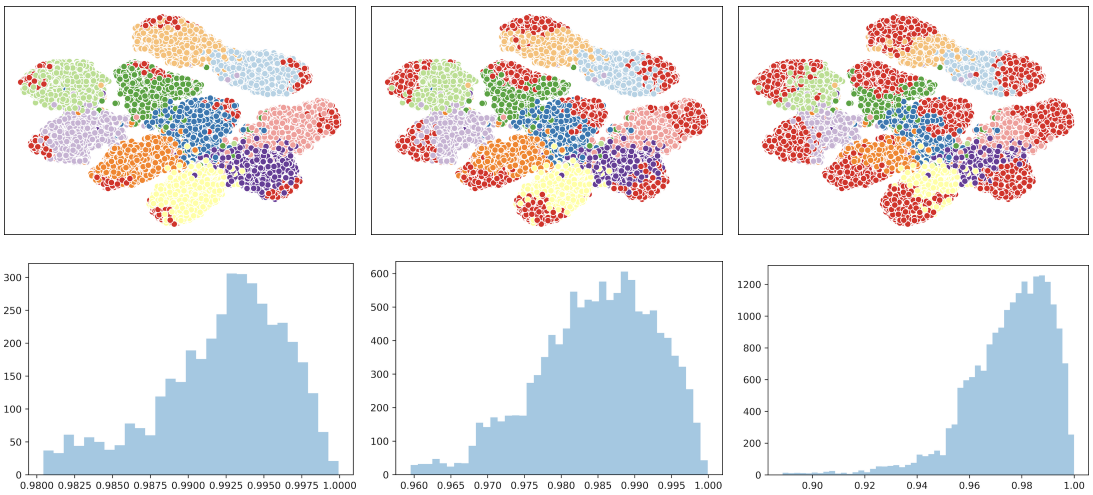
tance between all training samples. In our experiments, it took us on average 14 minutes 50 seconds to get all the selected samples and we only need about 12 seconds to select the boundary samples. Figure 4.6 shows the score distribution of EDGIS selected boundary samples with $k = 3$. Compared with the second column of Figure 4.5(c), the main difference is the lack of high score samples. Therefore, if we choose to select higher score samples from the score list, we can combine them with EGDIS selected boundary samples and get a similar EGDIS score distribution within about 20 seconds. We will mention this topic in the next few paragraphs.



(a) Scores (b) Plots

Figure 4.6: EGDIS boundary score distribution

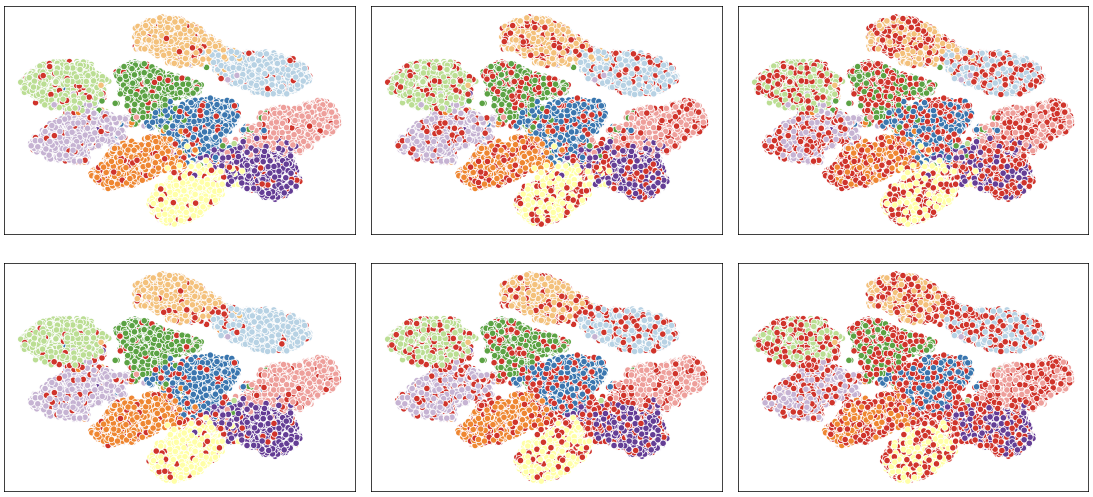
Next, we visualised the selection preference of CL in Figure 4.7. We selected different percentages of samples and the selection pattern is very obvious. We can see that CL tends to select only high score samples, and these samples are lying in the opposite direction as the EGDIS boundary samples. This is what we expected because samples far from the decision boundary would have high scores if the classifier. From the score distributions, we can see that all selected samples are easy to classify for deep learning. This reminds us to combine these high score samples with EGDIS boundary samples. We can construct a limit of the blobs. However, the downside is that no inner samples are selected and the limit borders are not complete. We can not capture the whole shape of the blobs and the results are more like two isolated blobs for each class. With linear regression model, the boundary cannot be recovered precisely thus we may get lower accuracy.



(a) P=10%, 4000 samples (b) P=30%, 12000 samples (c) P=50%, 20000 samples

Figure 4.7: CL tune selection percentage

Rather than selecting only the top score samples, we proposed WCL to randomly select samples within each class based on the sample score. The results are shown in the first column of Figure 4.8. Although most samples are selected from the high score region, there are some sample selected from the lower score regions. The problem is that less samples near the EGDIS boundary are selected. The boundary between close blobs are blurry thus we cannot guarantee the performance.

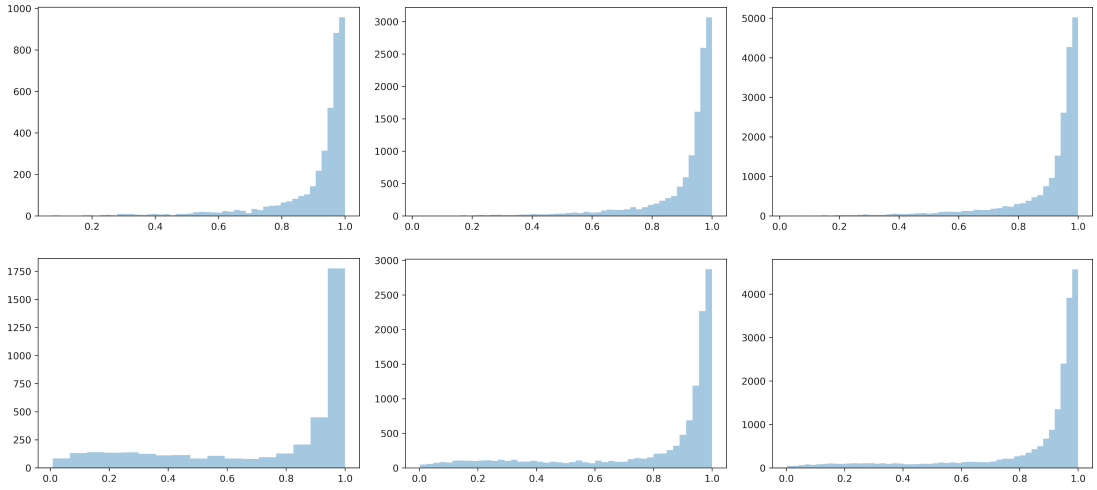


(a) P=10%, 4000 samples (b) P=30%, 12000 samples (c) P=50%, 20000 samples

Figure 4.8: WCL tune selection percentage. The first row we use weighted sampling method, the second row we use EGDIS based method.

Therefore, we combined both WCL and EGDIS selected boundary samples and

proposed the method BWCL. Compared with EGDIS, our BWCL can both get the similar score distribution as well as recover the shape of blobs. However, due to randomness during selection, we cannot guarantee to select the same datasets each time. This makes the classification accuracy unstable for machine learning models. We plotted the score distributions in Figure 4.9. We found that the score distribution of WCL is similar with the score distribution of the whole dataset in Figure 4.3 while by selecting similar amount of samples, BWCL is similar with EGDIS in Figure 4.5(c).



(a) P=10%, 4000 samples (b) P=30%, 12000 samples (c) P=50%, 20000 samples

Figure 4.9: WCL tune selection percentage. The first row we use weighted sampling method, the second row we use EGDIS based method.

4.3 Experiment 3: Logistic Regression

For more comprehensive evaluation results and inspired by [10], we manually synthesised two CIFAR100 subsets and fill in the classification gap between CIFAR10 and CIFAR100. Between 10 classes to 90 classes with gap 10, we choose to build 9 subsets with by selecting the samples from required number of classes. For each required classes, we randomly do the job 5 times and trained them to convergence with logistic regression model. The boxplot is shown in Figure 4.13. We choose 40 classes and 20 classes as the number of our extra synthesised datasets, with LR test accuracy: 0.78925 and 0.853. They have 8,028 samples and 16042 samples. Then we repeated the feature compression process described before to fine-tuning the selected subsets and compress the extracted features. The final test accuracy is 0.8065 and 0.8745 respectively. In our reported accuracy below, we use CIFAR20 and CIFAR40 to refer to these synthesised

datasets.

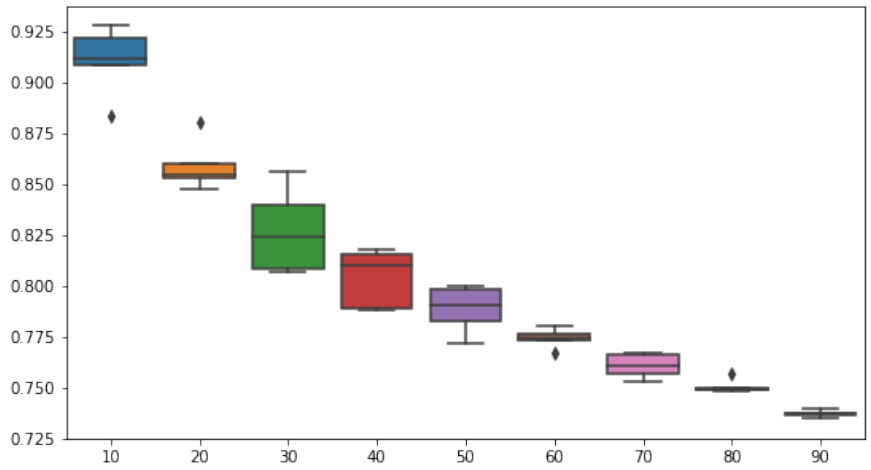
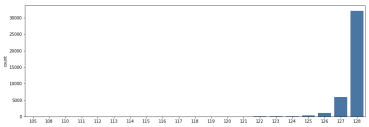


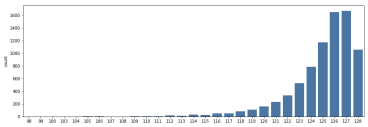
Figure 4.10: The test set accuracies of CIFAR100 subsets. Horizontal axis is the number of classes selected. Vertical axis is the accuracy score. For each selection, we randomly choose the classes for five times and report the accuracy with logistic regression model.

Our first discovery is that it is hard to select right amount of subsets with POP because the number of samples with weakness 128 is very low for all datasets except CIFAR10. Even for threshold 1, the number of pure inner is still low. Therefore, it is not good to choose samples with weakness ¡ 128. From our experiment, we found that the reduction rate for EGDIS is good enough. Therefore, we make POP, CL, WCL and BWCL to select the same amount of samples as EGDIS. For POP, we ranked samples based on weakness and select from low to high. Therefore, we can have a fair comparison of their classification performance. The POP weakness distributions are shown in Figure 4.11.

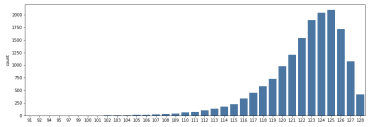
For the five algorithms, we performed the selection process 12 times and reported the average accuracy with logistic regression model. The relative accuracy is recorded in Table 4.1. We can see that the retention rate is increasing with the classification difficulty of the datasets. POP and EGDIS perform better with simpler dataset such as CIFAR10. For EGDIS, it even achieved an accuracy increase. We think this is because with selected boundary and dense samples only, the overfitting problem is reduced. However, the performances of WCL and BWCL are more stable than EGDIS. Their relative accuracy decreases slowly just like POP. Among all three classification score based algorithms, BWCL is more capable of dealing with harder datasets. We believe the reason is that BWCL can maintain the EGDIS selected boundary and the self-



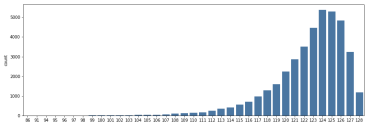
(a) CIFAR10, 40000 samples



(b) CIFAR20, 8028 samples



(c) CIFAR40, 16042 samples



(d) CIFAR100, 40000 samples

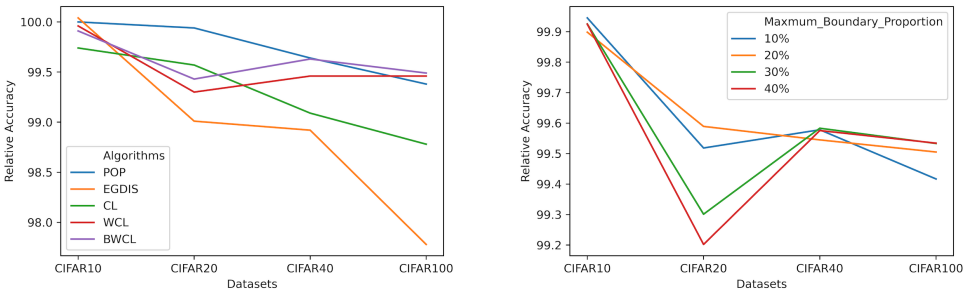
Figure 4.11: POP with 4 datasets.

adaptive selection manner of WCL.

Datasets	Retention Rate	POP	EGDIS	CL	WCL	BWCL
CIFAR10	14.915%	100.00%	**100.04**%	99.74%	99.96%	99.91%
CIFAR20	16.67%	**99.94**%	99.01%	99.57%	99.31%	99.43%
CIFAR40	21.09%	**99.64**%	98.92%	99.09%	99.46%	99.63%
CIFAR100	31.48%	99.38%	97.78%	98.78%	99.46%	**99.49**%

Table 4.1: Logistic Regression test set relative accuracy by averaging 12 runs

We should notice that the average relative accuracy of BWCL is not the best value. We tuned the maximum proportion of boundary samples from 0.1 to 0.4 with step 0.1 and trained them 3 times each. In Figure 4.12(b), we reported the relative accuracy in plot. It is clear that for simpler datasets, we should choose less hard examples. For harder datasets, we should choose more boundary samples.



(a) Relative Accuracy

(b) Relative Accuracy of BWCL by tuning maximum boundary proportion

Figure 4.12: Relative Accuracy of data reduction algorithms

We took a further analysis for BWCL, by varying the proportion of samples selected relative to the number of samples selected by EGDIS. We fixed the maximum boundary proportion to 0.4. The relative accuracy is reported in Figure 4.13. We found that the quality of the extracted features are so good that the test set samples are classified easily. This means that the results from logistic regression may not transfer to CNN experiments well but this provide us a good start.

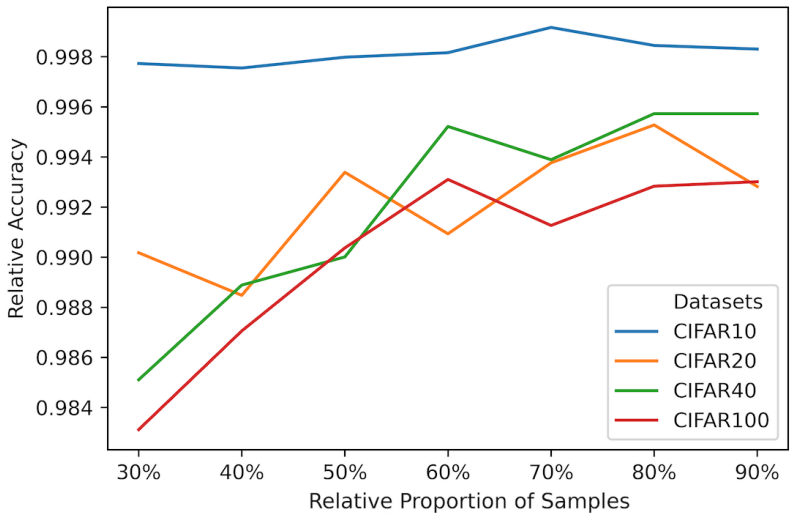


Figure 4.13: The BWCL test set accuracies. Averaged with 3 individual runs. The threshold is 10%. Horizontal axis is the percentage of samples selected, in term of EDGIS selected samples. Vertical axis is the accuracy score.

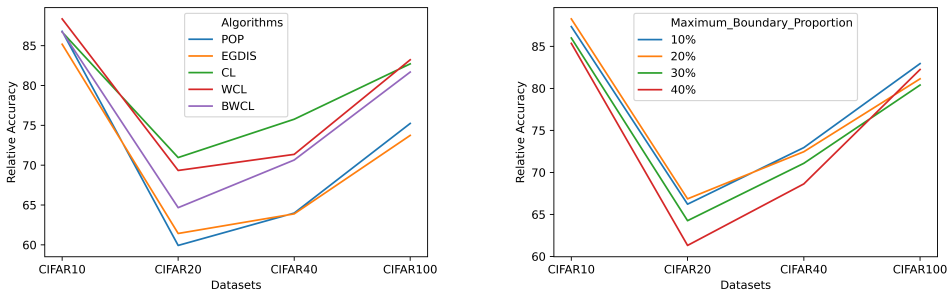
4.4 Experiment 4: Data Reduction for CNN

We trained the network densenet101, three times, with learning rate 0.1 (150 epochs), 0.01 (100 epochs) and 0.001 (100 epochs). First we trained the network with the same selection configuration as logistic regression and reported the classification accuracy in Table 4.2.

Datasets	Retention Rate	POP	EGDIS	CL	WCL	BWCL
CIFAR10	14.915%	86.80%	85.18%	86.71%	**88.36%**	86.75%
CIFAR20	16.67%	59.93%	61.43%	**70.96%**	69.34%	64.67%
CIFAR40	21.09%	63.99%	63.89%	**75.77%**	71.36%	70.65%
CIFAR100	31.48%	75.24%	73.74%	82.72%	**83.23%**	81.69%

Table 4.2: CNN test set relative accuracy

First of all, we found that POP and EGDIS selected samples are not suitable for neural network. The network cannot extract good features from these images thus the relative accuracy is much lower than expected. Second, classification score based algorithms can achieve relatively higher accuracy. In particular, for datasets with less per-class samples, CL is better. For datasets with more per-class samples, WCL is better. However, BWCL performs worse than expected. We believe that this is because we evaluated the classification scores with NasNetLarge, who has better extraction power than DenseNet. We just contains too much hard examples so the network cannot handle the training samples well. We proposed another hypothesis that if the network could learn these hard samples well, then it should be able to achieve higher classification accuracy based on the hard-example mining method described before.

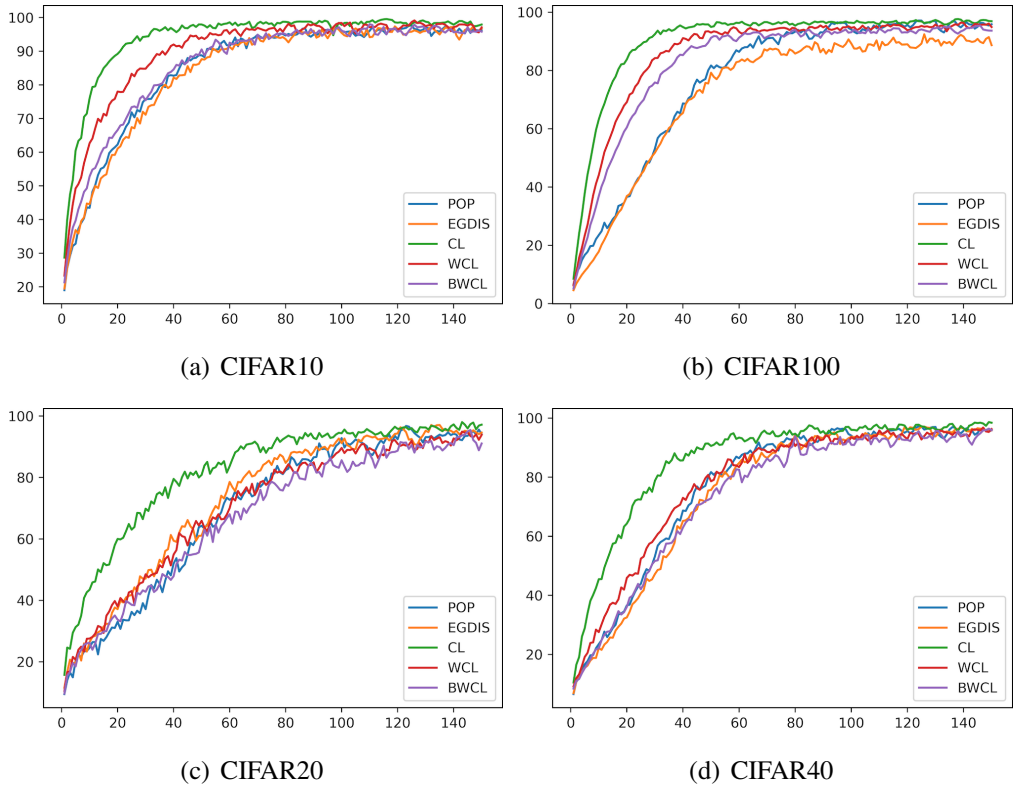


(a) Relative Accuracy

(b) Relative Accuracy of BWCL by tuning maximum boundary proportion

Figure 4.14: Relative Accuracy of data reduction algorithms

We took a closer look at the training curve of the first training stage to analysis the convergence speed.



(a) CIFAR10 (b) CIFAR100

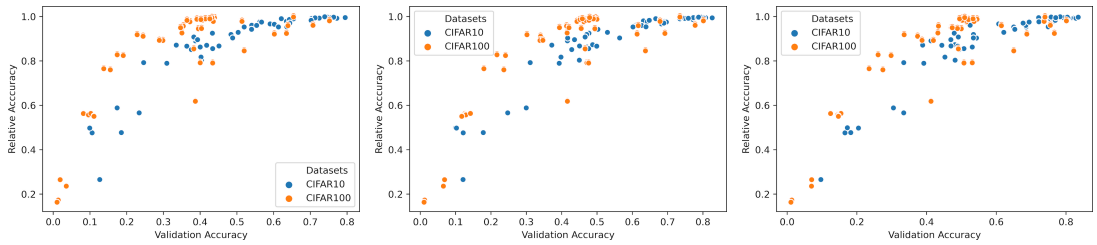
(c) CIFAR20 (d) CIFAR40

Figure 4.15: The training trend of four datasets by selecting the same amount of samples as EGDIS, of the first 150 epochs

Chapter 5

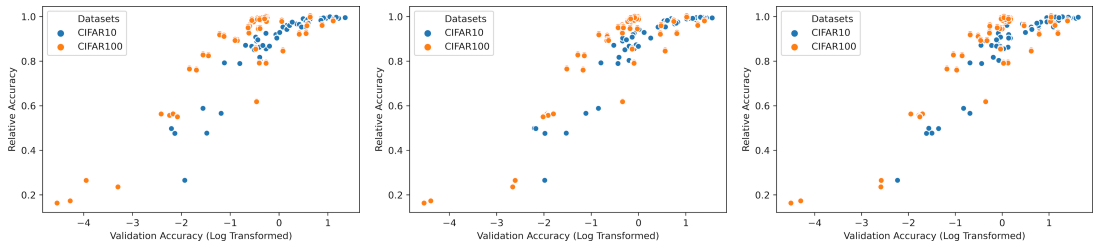
Trade-off Framework

5.1 Subset Selection Framework



(a) Epoch 5 (b) Epoch 7 (c) Epoch 9

Figure 5.1: The linear relationship between the validation accuracy and the final relative accuracy achieved

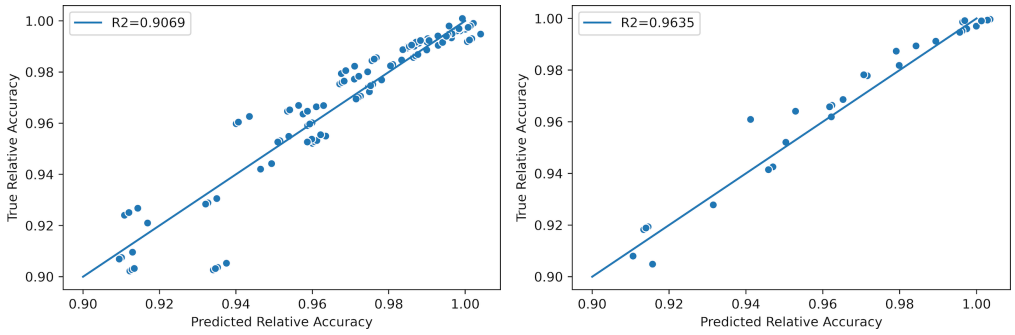


(a) Epoch 5 (b) Epoch 7 (c) Epoch 9

Figure 5.2: The linear relationship between the validation accuracy and the final relative accuracy achieved

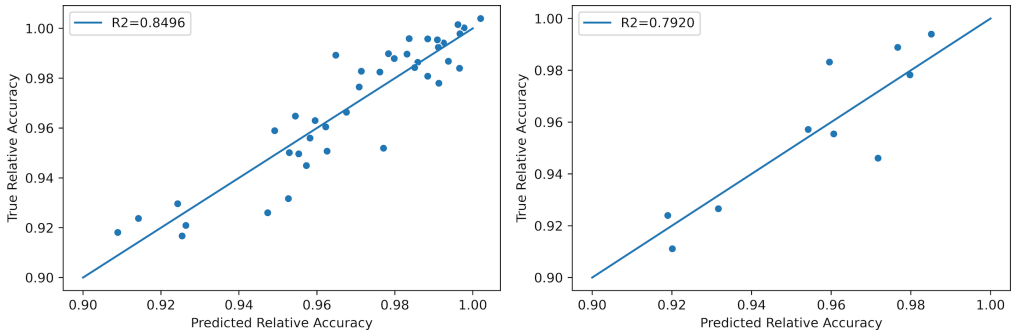
Chapter 6

Trade-off Evaluation



(a) CIFAR10 train

(b) CIFAR10 test



(c) CIFAR100 train

(d) CIFAR100 test

Figure 6.1: The linear model for CIFAR10 and CIFAR100 to predict the relative accuracy

Chapter 7

Conclusion and Future Work

Bibliography

[1] TensorFlow Hub. NasNetLarge.

[2] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, jan 1991.

[3] Yoshua Bengio, Jérome Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ACM International Conference Proceeding Series*, volume 382, pages 1–8, New York, New York, USA, 2009. ACM Press.

[4] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 - Mining discriminative components with random forests. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8694 LNCS, pages 446–461. Springer Verlag, 2014.

[5] Haw Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 1003–1013, 2017.

[6] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3606–3613. IEEE Computer Society, sep 2014.

[7] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Journal of Machine Learning Research*, volume 9, pages 249–256, 2010.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. 2016.

[9] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *36th International Conference on Machine Learning, ICML 2019*, volume 2019-June, pages 4483–4496, 2019.

[10] R. Istrate, F. Scheidegger, G. Mariani, D. Nikolopoulos, C. Bekas, and A. C. I. Malossi. TAPAS: Train-Less Accuracy Predictor for Architecture Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:3927–3934, jun 2019.

[11] Angelos Katharopoulos and François Fleuret. Not All Samples Are Created Equal: Deep Learning with Importance Sampling. *35th International Conference on Machine Learning, ICML 2018*, 6:3936–3949, mar 2018.

[12] Ramandeep Kaur and Er Himanshi. Face recognition using Principal Component Analysis. In *Souvenir of the 2015 IEEE International Advance Computing Conference, IACC 2015*, pages 585–589, 2015.

[13] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, pages 2656–2666, may 2019.

[14] Alex Krizhevsky. Learning multiple layers of features from tiny images. Tech. rep., CIFAR-10 (Canadian Institute for Advanced Research), 2009.

[15] M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010, NIPS 2010*, pages 1189–1197, 2010.

[16] Hao Li and Maoguo Gong. Self-paced Convolutional Neural Networks. Technical report, 2017.

[17] Ilya Loshchilov and Frank Hutter. Online Batch Selection for Faster Training of Neural Networks. nov 2015.

[18] Mohamed Malhat, Mohamed El Menshawy, Hamdy Mousa, and Ashraf El Sisi. A new approach for instance selection: Algorithms, evaluation, and comparisons. *Expert Systems with Applications*, 149:113297, jul 2020.

[19] Deyu Meng, Qian Zhao, and Lu Jiang. What Objective Does Self-paced Learning Indeed Optimize? 2015.

[20] J Arturo Olvera-López, J. Ariel Carrasco-Ochoa, J. Francisco Martínez-Trinidad, and Josef Kittler. A review of instance selection methods, 2010.

[21] Yongjoo Park, Jingyi Qing, Xiaoyang Shen, and Barzan Mozafari. BlinkML: Efficient maximum likelihood estimation with probabilistic guarantees. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1135–1152, New York, New York, USA, jun 2019. Association for Computing Machinery.

[22] José C. Riquelme, Jesús S. Aguilar-Ruiz, and Miguel Toro. Finding representative patterns with ordered projections. *Pattern Recognition*, 36(4):1009–1018, apr 2003.

[23] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 761–769, 2016.

[24] Wei Song, Lingfeng Zhang, Yifei Tian, Simon Fong, Jinming Liu, and Amanda Gozho. CNN-based 3D object classification using Hough space of LiDAR point clouds. *Human-centric Computing and Information Sciences*, 10(1):1–14, dec 2020.

[25] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

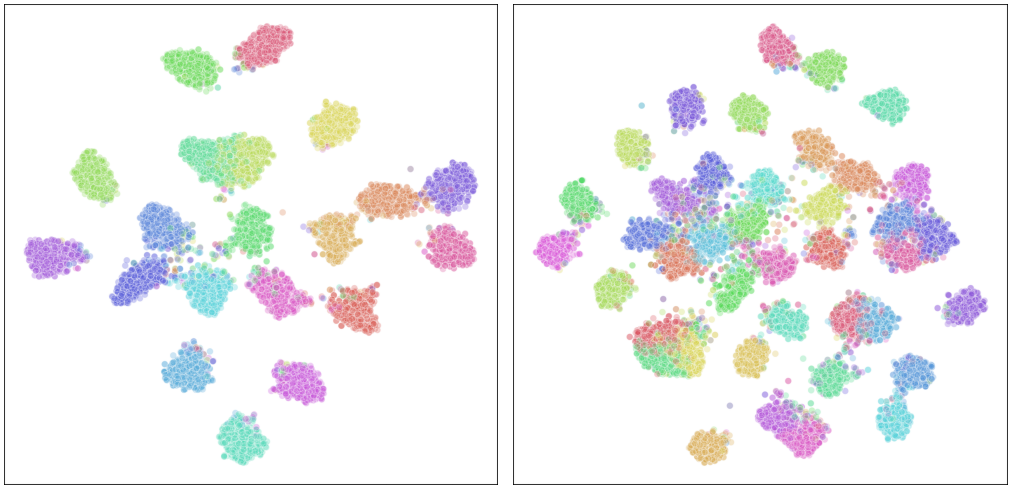
[26] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *32nd International Conference on Machine Learning, ICML 2015*, volume 1, pages 1–9, 2015.

[27] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning Transferable Architectures for Scalable Image Recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8697–8710. IEEE Computer Society, jul 2018.

Appendix A

Training History

A.1 Feature Extraction for CIFAR20 and CIFAR40



(a) CIFAR20 (b) CIFAR40

Figure A.1: Extracted features visualisation with t-SNE algorithm

A.2 Logistic Regression Original History

Datasets	Retention Rate	Whole	POP	EGDIS	CL	WCL	BWCL
CIFAR10	14.915%	0.9258	0.9258	**0.9262**	0.9234	0.9254	0.9250
CIFAR20	16.67%	0.8825	**0.8820**	0.8738	0.8787	0.8764	0.8775
CIFAR40	21.09%	0.8159	**0.8130**	0.8071	0.8085	0.8115	0.8129
CIFAR100	31.48%	0.7444	0.7398	0.7279	0.7353	0.7404	**0.7406**

Table A.1: Logistic Regression test set accuracy by averaging 12 runs

Datasets	Retention Rate	10%	20%	30%	40%
CIFAR10	14.915%	**0.9253**	0.9249	0.9246	0.9251
CIFAR20	16.67%	0.8783	**0.8789**	0.8758	0.8744
CIFAR40	21.09%	0.8125	0.8122	**0.8134**	0.8132
CIFAR100	31.48%	0.7401	0.7407	**0.7410**	**0.7410**

Table A.2: Logistic Regression BWCL test set accuracy by averaging 12 runs

A.3 CNN Original History

Datasets	Retention Rate	Whole	POP	EGDIS	CL	WCL	BWCL
CIFAR10	14.915%	0.9522	0.8265	0.8111	0.8257	**0.8414**	0.8260
CIFAR20	16.67%	0.8660	0.5190	0.5320	**0.6145**	0.6005	0.5600
CIFAR40	21.09%	0.8233	0.5268	0.5260	**0.6238**	0.5875	0.5816
CIFAR100	31.48%	0.7842	0.5900	0.5783	0.6487	**0.6527**	0.6406

Table A.3: CNN test set accuracy

Datasets	Retention Rate	10%	20%	30%	40%
CIFAR10	14.915%	0.8318	**0.8405**	0.8189	0.8128
CIFAR20	16.67%	0.5735	**0.5790**	0.5565	0.5310
CIFAR40	21.09%	**0.5953**	0.5913	0.5800	0.5600
CIFAR100	31.48%	**0.6505**	0.6363	0.6305	0.6450

Table A.4: BWCL test set accuracy