Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

SUPClust: Active Learning at the Boundaries

Semester Thesis

Yuta Ono

yutono@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Till Aczél, Benjamin Estermann,
Prof. Dr. Roger Wattenhofer

February 26, 2024

Acknowledgements

My deepest gratitude goes to my supervisors, Till Aczél and Benjamin Estermann. Their profound knowledge and invaluable advice have always guided me in the right direction of research. None of the results presented in this thesis would have been possible without their help. In addition, their friendly personalities have helped me adapt to the life in Switzerland which is completely different from the life in Japan. I am also grateful to Prof. Dr. Wattenhofer and other members of the DISCO group for giving me the opportunity to do research in such a great environment. I will never forget the special experience of using more than 30 GPUs simultaneously. Lastly, I would like to thank my family for their support from Japan.

Abstract

In this thesis, we evaluate querying strategies for active learning based on the latent space of different representation learning models such as hierarchical VAEs and SimCLR. We also propose a novel active learning approach, SUPClust, which aims to identify the data points that are close to the decision boundaries between categories. By selecting these points, SUPClust tries to obtain more informative points for training a classifier. Our empirical results show that among these approaches, SUPClust shows a strong active learning performance in low-budget regimes. This improvement is observed even on highly imbalanced datasets.

Contents

Chapter 1

Introduction

1.1 Motivation

Progress in deep learning for image classification tasks has been following an impressive pace in recent years [1–4]. In order to achieve high classification accuracy on a target dataset, many of the methods necessitate a substantial amount of annotated data. However, in many cases, annotating data can be both time-consuming and expensive, as it often requires professionals with specific expertise, such as doctors for skin lesion images. These costs can present a challenge to the application of these successful methods. One potential solution to this problem is the use of active learning. Active learning is a framework for training machine learning models, which aims to maximize classification accuracy by selecting the most informative and valuable data points to be annotated for model training with a limited budget for annotation.

Previously, there were active learning methods for classical machine learning methods such as support vector machines (SVMs) [5]. However, recently proposed active learning methods are primarily designed for deep learning models due to the large, and sometimes unrealistic, amount of labeled data required to train them and the need to leverage them in various applications. Most of them are effective for deep learning model trainings when there is a sufficient budget for annotation, but their performance deteriorates when the budget for annotation is extremely limited. This is reffered to as the "cold start" problem in low-budget regimes, which is a critical issue for leveraging active learning in real-world settings.

To enhance the potential of active learning, we are looking for a new metric to address the cold start problem in low-budget scenarios. To this end, we consider the use of hierarchical variational autoencoders, self-supervised learning algorithms, and the idea of decision boundary inspired by SVMs. These methods are investigated in this thesis.

1.2 Organization of Thesis

This thesis is organized as follows:

- Chapter 2: The details of active learning are described in this chapter. Related ideas to the methods proposed in this thesis are also summarized
- Chapter 3: Active learning methods with a hierarchical variational autoencoder are proposed
- Chapter 4: An active learning method with self-supervised embeddings is proposed
- Chapter 5: An active learning method inspired by SVMs is proposed. The main results of this thesis are described in this chapter with some ablation studies to ensure the effectiveness of the proposed method
- Chapter 6: The thesis is concluded

Chapter 2

Related Works

2.1 Active Learning

2.1.1 Pool-based Active Learning Settings

Let $\mathcal{D}$, $\mathcal{U}$, and $\mathcal{L}$ represent the entire dataset, the unlabeled data pool, and the labeled data pool respectively. Here, the following relationships hold:

$$\mathcal{U} \cup \mathcal{L} = \mathcal{D}, \tag{2.1}$$

$$\mathcal{U} \cap \mathcal{L} = \emptyset. \tag{2.2}$$

A data point $\mathbf{x} \in \mathcal{D}$ has the dimension d, which shuold be the total number of pixels in a image or the dimension of an embedding space.

$$\mathbf{x} \in \mathrm{R}^d \tag{2.3}$$

A data point in the labeled pool has its true label $y(\mathbf{x})$. The model is trained using the labeled dataset $\mathcal{L}$ and their labels $\{y(\mathbf{x}) \mid \forall \mathbf{x} \in \mathcal{L}\}$. The optimization problem of deep active learning can be expressed as follows:

$$\underset{\mathcal{L} \in \mathcal{D}, |\mathcal{L}| = B}{\arg\min} \; \mathbb{E}_{\mathbf{x} \in \mathcal{L}} \left[l(f(\mathbf{x}),\, y(\mathbf{x})) \right], \tag{2.4}$$

where f is a deep learning model, or a classifier, B is a total budget size allocated for annotation, and $l(\cdot,\, \cdot)$ is a given loss function. In settings of active learning, a bunch of images are selected and queried for their true labels in a batch whose size is b for e itrations ($B = e \cdot b$), following a querying strategy. Ideally, the batch size should be 1 to minimize the loss, but it is often set to be more than 1 because each iteration of active learning necessitates deep learning model training and it consumes time and resources. In this thesis, e is set to 5, and two batch sizes are used:

- **Tiny budget**: $b = \#Classes$
- **Small budget**: $b = 5 \times \#Classes$

These two budget sizes can be categorized in the low-budget regimes.

2.1.2 Previous works

There are various querying strategies to achieve higher performance of active learning. These strategies can be categorized as uncertainty-based or diversity-based. Uncertainty-based approaches leverage the prediction uncertainty of the classification model during training on the labeled dataset to select informative data to be annotated. Diversity-based approaches aim to annotate a diverse range of samples spanning the complete data distribution, avoiding the selection of too similar ones. There are also hybrid methods that attempt to identify samples with high uncertainty and diversity simultaneously. Ten active learning methods are described below.

Random

The random sampling strategy randomly and uniformly selects data points from the unlabeled data pool $\mathcal{U}$. Although random sampling does not use any information about the dataset and is the simplest way to sample data, it is known to perform very well in low budget regimes compared to other designed active learning strategies. This phenomenon is often referred to as the "cold start" problem as mentioned before.

Least confidence

Least confidence [6] selects b images with the b lowest value of Q_{LC}. This means that it prioritizes the samples whose top 1 predicted probability is low.

$$Q_{LC}(\mathbf{x}; f) = P_1(\mathbf{x}; f) \tag{2.5}$$

Margin

Margin [7] selects the data points whose Q_M, or margin between the probabilities of the most probable label and second most probable label, is smaller than others. If a classifier is confident in its decision, the margin gets bigger.

$$Q_M(\mathbf{x}; f) = P_1(\mathbf{x}; f) - P_2(\mathbf{x}; f) \tag{2.6}$$

Entropy

Entropy [8] selects the data with the biggest Q_E, or entropy. If a classifier is confident, namely it has a close value to 1 for a class, the entropy is small.

$$Q_E(\mathbf{x}; f) = -\sum_i P_i(\mathbf{x}; f) \log P_i(\mathbf{x}; f) \tag{2.7}$$

BALD and DBAL

BALD [9] and *DBAL* [10] select data using the mutual information between the model prediction and the model parameters. It tries to maximize Q_{BALD}, or the sum of mutual information for $\mathbf{x}_i$ in a batch.

$$\mathbb{I}(y; \omega|\mathbf{x}, \mathcal{L}) = \mathbb{H}(y|\mathbf{x}, \mathcal{L}) - \mathbb{E}_{p(\omega|\mathcal{L})}[\mathbb{H}(y|\mathbf{x}, \omega, \mathcal{L})] \tag{2.8}$$

$$Q_{BALD}(\{\mathbf{x}_1, \ldots, \mathbf{x}_b\}, p(\omega|\mathcal{L})) = \sum_{i=1}^{b} \mathbb{I}(y_i; \omega|\mathbf{x}_i, \mathcal{L}) \tag{2.9}$$

BatchBALD

Since BALD is originally designed for acquiring individual points, similar images tend to be selected in a batch when it is applied to batch aquisition because similar images tend to have the similar mutual information. *BatchBALD* [11] extends BALD to batch aquisition problem by using the mutual information between *a joint of multiple data points* and the model parameters.

$$Q_{BatchBALD}(\{\mathbf{x}_1, \ldots, \mathbf{x}_b\}, p(\omega|\mathcal{L})) = \mathbb{I}(y_1, \ldots, y_b; \omega|\mathbf{x}_1, \ldots, \mathbf{x}_b, \mathcal{L}) \tag{2.10}$$

Coreset

Coreset [12] queries diverse samples through the selection of points that form a minimum radius cover of the remaining samples in the unlabeled data pool. In order to do this, Coreset works on the embeddings generated by the penultimate layer of the classifier.

TypiClust

TypiClust [13] focuses on typicality, see Equation (2.11), calculated in an embedding space. The embeddings of unlabeled data are generated *a priori* by SimCLR [14] which is a self-supervised learning method described in Section 2.3. In the iteration e of active learning, $b \cdot e$ clusters are built in the embedding space, and then, b samples with highest typicality are selected from different clusters. Clusters which have less labeled point and bigger size are prioritized. TypiClust is designed for low-budget regimes, and it is known to work well in those settings.

$$Typicality(\mathbf{x}) = \left(\frac{1}{K} \sum_{\mathbf{x}_n \in K-\mathrm{NN}(\mathbf{x})} \|\mathbf{x} - \mathbf{x}_n\| \right)^{-1}, \tag{2.11}$$

where $K - \mathrm{NN}(\mathbf{x})$ is a set of K nearest neighbors of $\mathbf{x}$ in an embedding space.

ProbCover

ProbCover [15] attempts to maximize the coverage with balls of a fixed size in contrast to Coreset which minimizes the ball size with covering all the data points. Probcover is designed for low-budget regimes and relies on well-structured embeddings as well as TypiClust.

2.2 Hierarchical Variational Autoencoders

2.2.1 Autoencoders

Autoencoders (AEs) [16] are neural networks that aim to compress the input information into lower dimension representation, or a latent variable. AEs usually consist of two parts: an encoder network and a decoder network.

An encoder can learn the transformation from a set of input data to efficient representations (encodings, latent variables) while a decoder can learn reproduction from the efficient representations to the original inputs. AEs are originally designed to extract compact and effective representation from inputs, but they can be used for generative tasks as well by introducing the idea of probabilistic distribution as described in Section 2.2.2.

2.2.2 Variational Autoencoders

Variational Autoencoders (VAEs) [17] introduces the idea of probabilistic distribution to the latent variable of AEs.

For VAEs, we assume discrete variable $\mathbf{x^{(i)}} \in X \ \ (i \in [N])$ is generated by some random process, involving unobserved continuous random variable $\mathbf{z}$. The random process is broken down into two steps: First, a value $z^{(i)}$ is generated from some prior distribution $p_{\theta^*}(\mathbf{z})$. Next, a value $\mathbf{x}^{(i)}$ is generated from some conditional distribution $p_{\theta^*}(\mathbf{x}|\mathbf{z})$. Unfortunately, the true parameters θ^* and the values of latent variables $\mathbf{z}^{(i)}$ are unidentifiable. See Figure 2.1 for the entire flow of VAE.

The unobserved variables $\mathbf{z}$ have an interpretation as a latent representation or *code*. In [17], the recognition model $q_\phi(\mathbf{z}|\mathbf{x})$ is referred to as a probabilistic

encoder. Given a data point $\mathbf{x}$, it produces a distribution over possible values of the code $\mathbf{z}$ from which the data point could have been generated. In a similar reason, $p_\theta(\mathbf{x}|\mathbf{z})$ is referred to as a probabilistic *decoder*, since given a code $\mathbf{z}$ it produces a distribution over the possible corresponding values of $\mathbf{x}$.

The marginal likelihood can be written as:

$$\log p_\theta(\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}) = \sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)}), \tag{2.12}$$

where

$$\log p_\theta(\mathbf{x}^{(i)}) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) \| p_\theta(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\theta, \phi, \mathbf{x}^{(i)}). \tag{2.13}$$

The first RHS term is the KL divergence of the approximate from the true posterior. Since KL divergence is non-negative, the second RHS term $\mathcal{L}(\theta, \phi, \mathbf{x}^{(i)})$ is called the *evidence lower bound* (ELBO).

$$\mathcal{L}(\theta, \phi, \mathbf{x}^{(i)}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[-\log q_\phi(\mathbf{z}|\mathbf{x}) + \log p_\theta(\mathbf{x}, \mathbf{z})] \tag{2.14}$$

ELBO can be rewritten as:

$$\mathcal{L}(\theta, \phi, \mathbf{x}^{(i)}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) \| p_\theta(\mathbf{z}|\mathbf{x}^{(i)})) + \log p_\theta(\mathbf{x}^{(i)}) \tag{2.15}$$

In the training of VAEs, we try to maximize the ELBO, or minimize inverse ELBO, for a higher likelihood. Read [17] for more details.

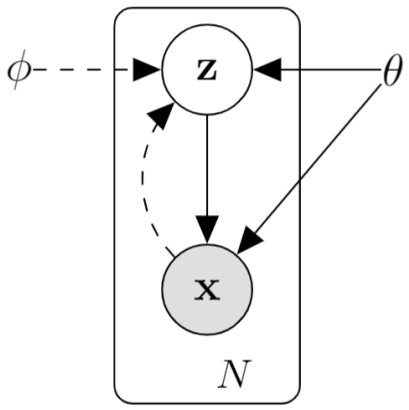

Figure 2.1: The type of directed graphical model under consideration. Solid lines denote the generative model $p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})$, dashed lines denote the variational approximation $q_\phi(\mathbf{z}|\mathbf{x})$ to the intractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$. The variational parameters ϕ are learned jointly with the generative model parameters θ. Cited from [17]

We can obtain latent variables of high quality by VAEs. In contrast to GAN [18] where the latent variables are random noises, the latent space in VAEs

is organized and can be interpreted because the distribution of the latent space follows a probabilistic distribution, which is usually Gaussian distribution.

2.2.3 Hierarchical Variational Autoencoders

Hierarchical Variational Autoencoders (HVAEs) [19–21] are variants of VAEs. They have some layers between the encoder part and the decoder part to represent multiple latent spaces as shown in Figure 2.2. These latent spaces are capable of learning hierarchical representations, such as colors, textures, and shapes of cars. In Figure 2.3, we can see that LVAE [19], an example of HVAEs, has a high capacity to learn multiple latent representations compared to VAEs. Thanks to these hierarchical representations, the quality of reproduction is greatly improved compared to the usual VAEs.

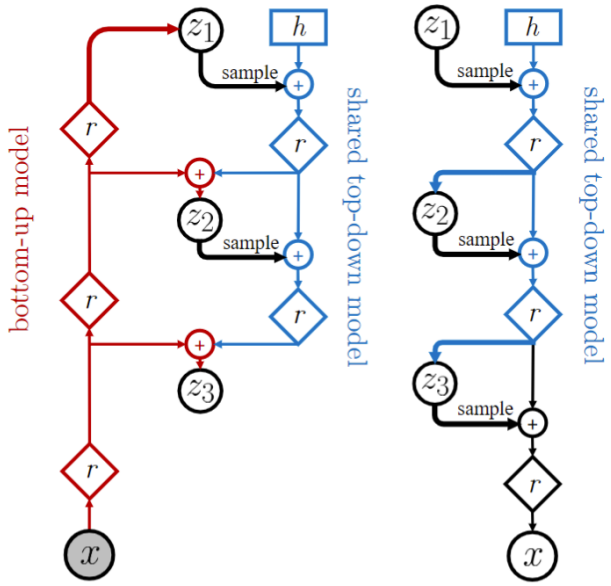

Figure 2.2: An example architecture of HVAEs. Cited from [21]

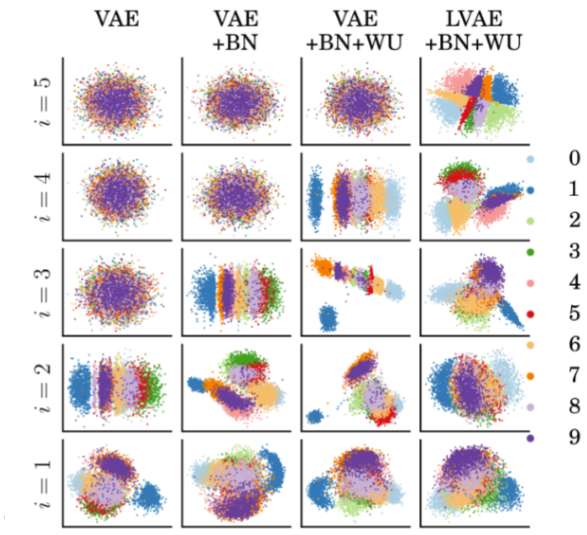

Figure 2.3: PCA plots of samples from different latent spaces of a HVAE. Cited from [19]

2.3 SimCLR

SimCLR [14] is a simple framework for contrastive learning of visual representations, in contrast to the complex frameworks such as CPC [22] and MoCo [23]. First, two different data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T},\ t' \sim \mathcal{T}$), see Figure 2.5 for some examples. These operators are then applied separately to an input image x, yielding transformed images ($\tilde{x}_i,\ \tilde{x}_j$). A base encoder network $f(\cdot)$, sometimes called a backbone model, and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After a training is completed, we discard the projection head $g(\cdot)$, and use the encoder $f(\cdot)$ and the representation h for downstream tasks. In this representation space, augmented images from the same original image attract each other, and images from different original images repel each other. Therefore, the representation space is well structured, and useful for downstream tasks.

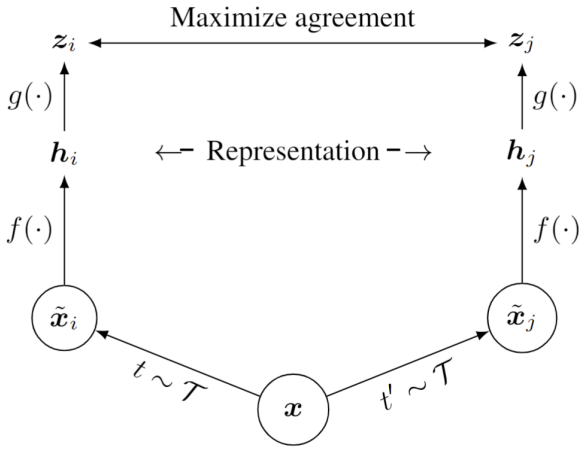

Figure 2.4: SimCLR architecture. Cited from [14]

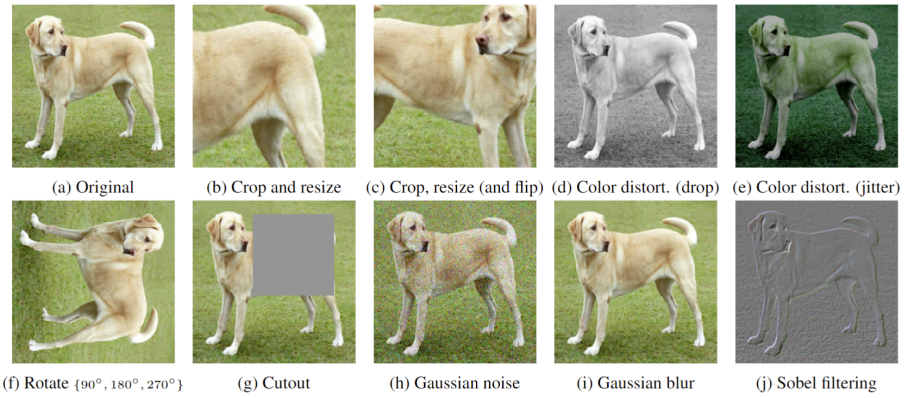

Figure 2.5: Illustrations of data augmentation operators. Cited from [14]

Since we can obtain self-supervised representation learned by SimCLR without true labels, it can be used for preprocessing of unlabeled dataset in active learning frameworks.

Chapter 3

Active Learning with Hierarchical VAEs

In this chapter, we explore the possibility of making use of hierarchical VAEs for active learning.

3.1 Background

It is shown that active learning can benefit from sampling in the latent space of a VAE [24]. Although the results on MNIST in [24] are suggestive, common VAEs are not powerful enough to extract essential information from more difficult datasets such as ImageNet [25] and CelebA [26]. To improve the ability to extract the representation and reproduce the inputs from the latent variables, hierarchical VAEs (HVAEs) have been proposed [19–21].

There are several layers in series that generate latent variables in HVAEs, and the latent variables represent embeddings of different levels of abstraction. The order of abstraction depends on the model architecture of the HVAE, but let's say that lower latent layers of HVAEs have higher abstraction and vice versa. In other words, the earlier latent layers in the forward propagation will extract broad representations, e.g., color of the object, and later ones will extract more details such as the texture of the skin. This character of latent spaces in HVAEs provides inspiration for exploiting the different levels of abstraction for diversity-based active learning.

It is crucial for the success of diversity-based active learning to find a way to ensure the diversity of samples in a queried batch. A variety of ways have been proposed to select diverse data without true labels. For example, TypiClust [13] uses clustering in an embedding space learned by SimCLR to achieve this. TypiClust works well in low-budget regimes, so we start by combining this previous work with the hierarchical representations of HVAE. We follow the protocol of TypiClust, but use the hierarchical latent space instead of the embedding space of SimCLR.

In addition, according to [13], active learning in the low budget regimes can benefit from oversampling of "typical" data points, which means that we should select easy samples to be learned by a classifier for annotation in the beginning of active learning. Unlike easy or typical samples, difficult samples tend to be atypical or outliers in the input space. Since HVAEs are able to discriminate in-distribution data and out-of-distribution data [27], we consider taking advantage of this ability to detect outliers in the input or embedding space. Figure 3.1 and Figure 3.2 show the examples of separating in-distribution data and out-of-distribution data following [27].

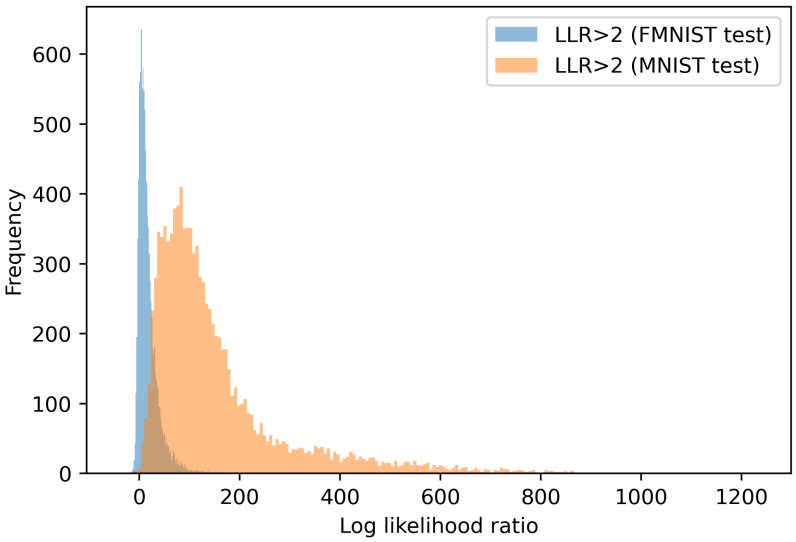

Figure 3.1: Out-of-distribution detection using FMNIST (in-ditribution) and MNIST (out-of-distribution). Out-of-distribution data tend to have larger $LLR^{>2}$.

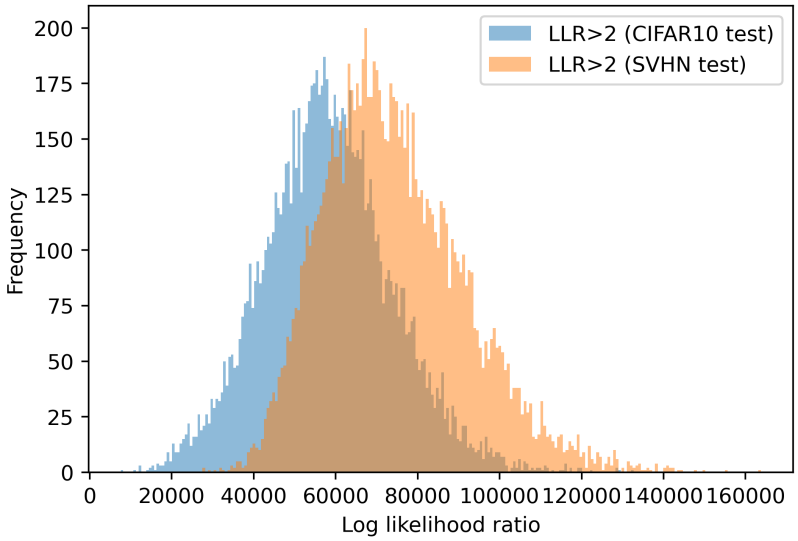

Figure 3.2: Out-of-distribution detection using CIFAR10 (in-distribution) and SVHN (out-of-distribution). Out-of-distribution data tend to have larger $LLR^{>2}$.

3.2 Method

3.2.1 TypiClust based on latent variables of original inputs

We follow the procedures proposed by TypiClust [13], but embeddings in hierarchical latent space replace the embeddings generated by SimCLR. These embeddings are used for clustering and calculating typicality. The results of the normal TypiClust and the TypiClust with hierarchical representations are compared by observing the performance of a classifier, ResNet18, trained in an active learning scheme. There are three latent spaces in the HVAE used for this experiment. These latent variables are used individually for experiments. Each latent space replaces the embedding generated by SimCLR.

3.2.2 TypiClust based on latent variables of SimCLR embedding

As shown in Section 3.3.1, latent variables generated by an HVAE from original inputs are not powerful enough to extract representation from CIFAR10. The latent spaces are not well separated by the true labels. The embeddings from SimCLR, see Figure 3.3, are so well structured that we can easily build clusters without label confusion in the embedding space without true labels.

To take advantage of this structure of SimCLR embeddings, we use these embeddings as inputs to an HVAE, and try to extract a hierarchy in the embedding space. We replace the embedding used in TypiClust with the latent variables generated by a HVAE from the embedding.

Figure 3.3: SimCLR embedding of CIFAR10 train dataset

3.2.3 TypiClust based on LLR score

Likelihood ratio score ($LLR^{>k}$) [27] is a metric to measure the level of out-of-distribution. $LLR^{>k}$ is defined as

$$LLR^{>k} = \mathcal{L}(x) - \mathcal{L}^{>k}(x), \tag{3.1}$$

where

$$\mathcal{L}^{>k} = \mathbb{E}_{p_\theta(\mathbf{z}_{\leq k}|\mathbf{z}_{>k})q_\phi(\mathbf{z}_{>k}|\mathbf{x})}\left[\log \frac{p_\theta(x|\mathbf{z})p_\theta(\mathbf{z}_{>k})}{q_\phi(\mathbf{z}_{>k}|\mathbf{x})}\right]. \tag{3.2}$$

Note that $\mathcal{L}^{>0}$ is the regular ELBO $\mathcal{L}$. High LLR indicates that the data point is an instance of out-of-distribution data, or an sample which is not included in the train dataset of the HVAE. Here, we assume that LLR indicates the degree of outlying even if the data point belongs to in-distribution data. The data points with the highest LLR or the lowest LLR are sampled in this method to sample outliers or inliers.

3.3 Result

3.3.1 TypiClust based on latent variables of original inputs

Latent spaces on the original inputs CIFAR10

Three latent variables generated by the HVAE from the CIFAR10 test dataset are shown in Figure 3.4, Figure 3.5, and Figure 3.6. Compared to the visualization of SimCLR embeddings shown in Figure 3.3, these latent spaces are clutterd in terms of true label distribution. The clustering for TypiClust is burdened by this untidiness, and it leads to a deterioration of the active learning performance as shown in the next part.

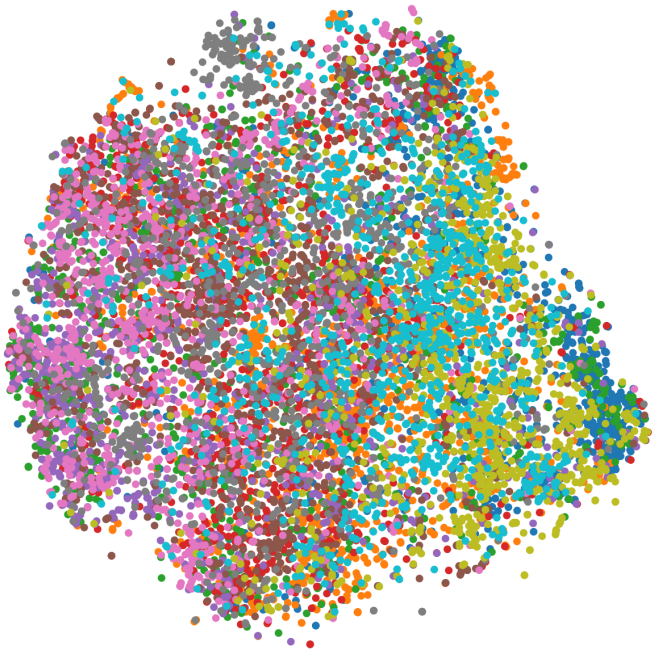

Figure 3.4: t-SNE visualization of the latent variables z_0 (the lowest layer of latent spaces). Each color represents a true label.

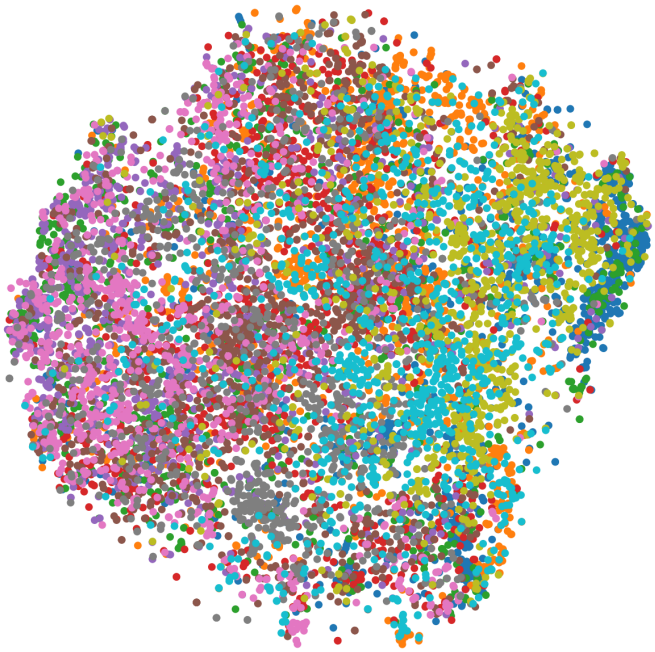

Figure 3.5: t-SNE visualization of the latent variables z_1 (the middle layer of latent spaces). Each color represents a true label.

Figure 3.6: t-SNE visualization of the latent variables z_2 (the highest layer of latent spaces). Each color represents a true label.

Training Results of Active Learning

The results of active learning with the tiny budget are shown in Figure. 3.7 and Figure 3.8[1]. "typiclust-rp (zi)" or "zi" ($i = 0,\ 1,\ 2$) is a result of TypiClust using the latent variable z_i generated by the reparameterization trick ($z_i(x) = \mu_i(x) + \sigma_i(x)\varepsilon,\ \varepsilon \sim \mathcal{N}(0, 1)$), and "typiclust-rp ($zi$ mean)" or "zi (mean)" is a result using the latent variable $z_i(x) = \mu_i(x)$, which is the mean value of the estimated distribution.

[1] I understand that means and standard errors with different random seeds should be shown for fair comparison, but it's difficult to repeat this experiment because the latent spaces of the HVAE are already lost due to the tragic catastrophe in the computer cluster

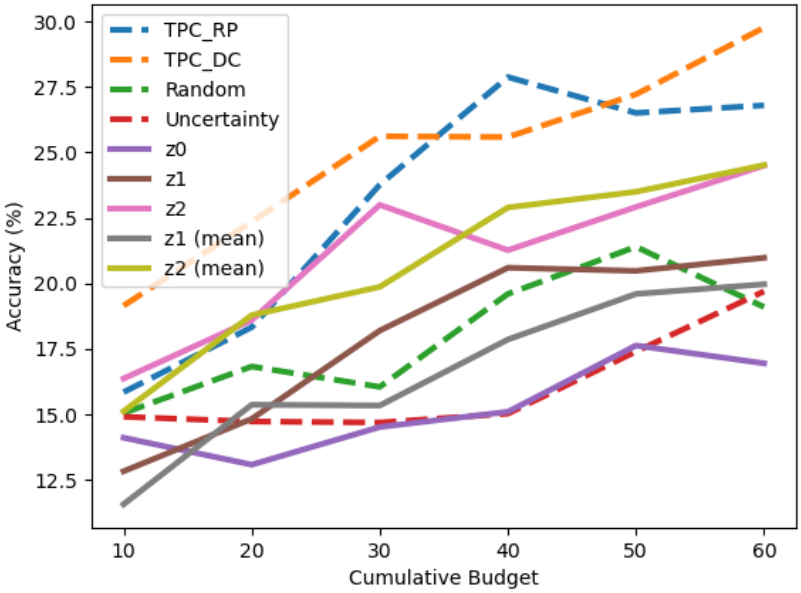

Figure 3.7: Result of active learning with tiny budget.

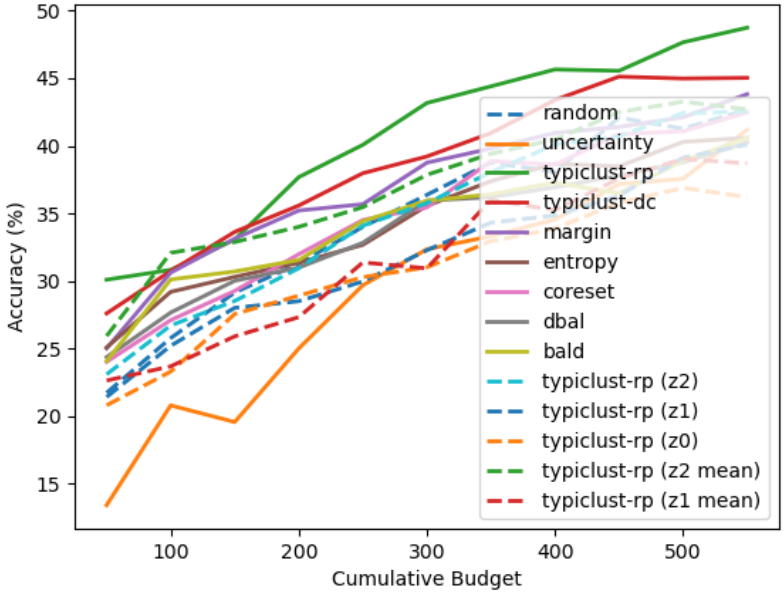

Figure 3.8: Result of active learning with small budget.

3.3.2 TypiClust based on latent variables of SimCLR embedding

The results of TypiClust-RP using the latent variables obtained by an HVAE from SimCLR embedding are shown in Figure 3.9 and Figure 3.10. The notation "typiclust-rp (e-zi)" ($i = 0,\ 1,\ 2$) represents the result of TypiClust-RP with the latent variable z_i of the HVAE generated from the SimCLR embedding inputs.

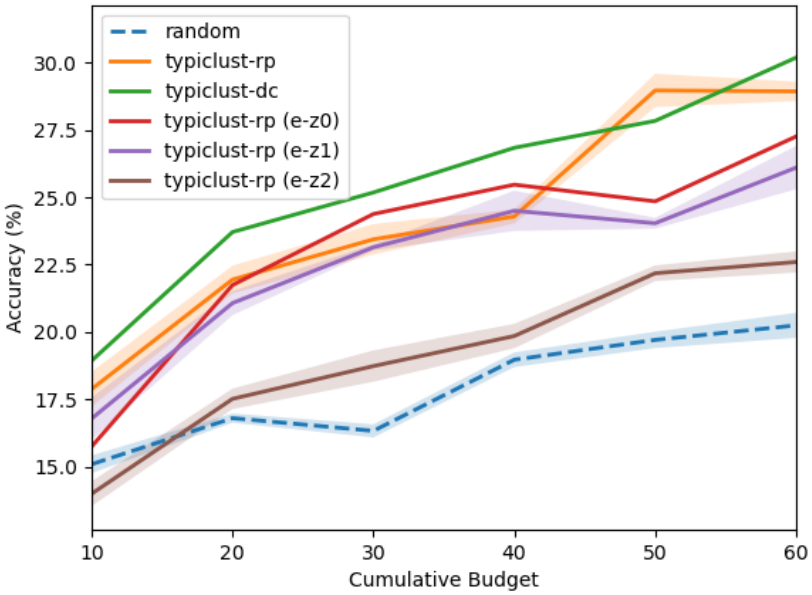

Figure 3.9: Result of active learning with tiny budget.

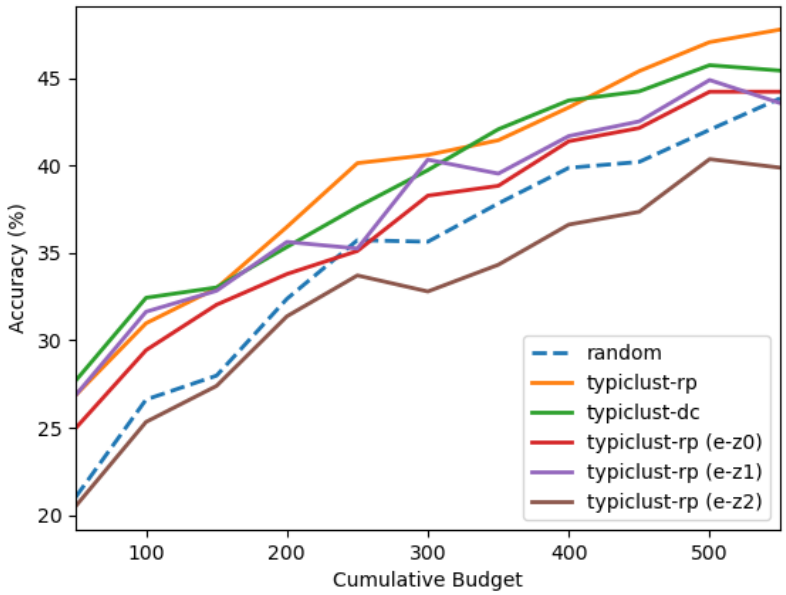

Figure 3.10: Result of active learning with small budget.

3.3.3 TypiClust based on LLR score

"ood-hr (clst:z2, smp: LLR>2)" builds clusters in the top latent space, and selects a data point with the lowest $LLR^{>2}$ from each cluster. "ood-hr (clst:z2, smp: LLR>2-inv)" also builds clusters in the topmost latent space, but it selects a data point with the *highest* $LLR^{>2}$ from each cluster.

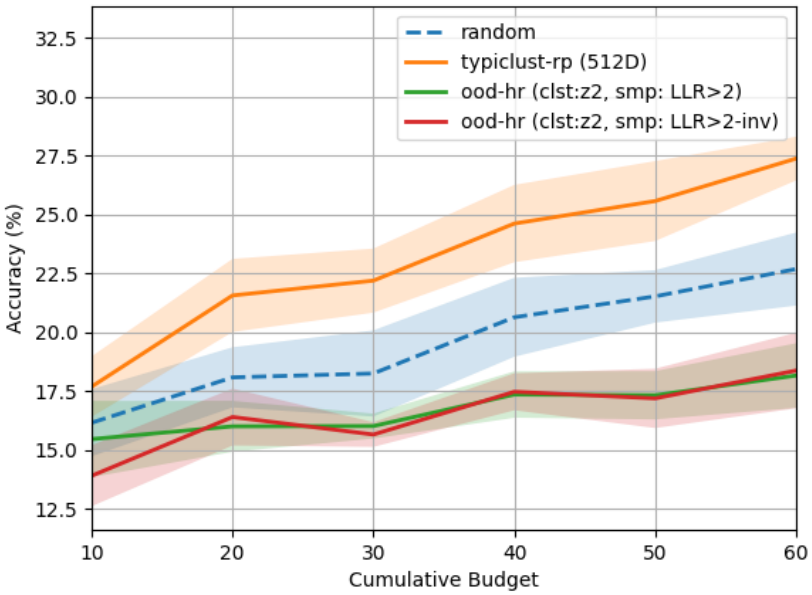

Figure 3.11: Result of active learning with tiny budget.

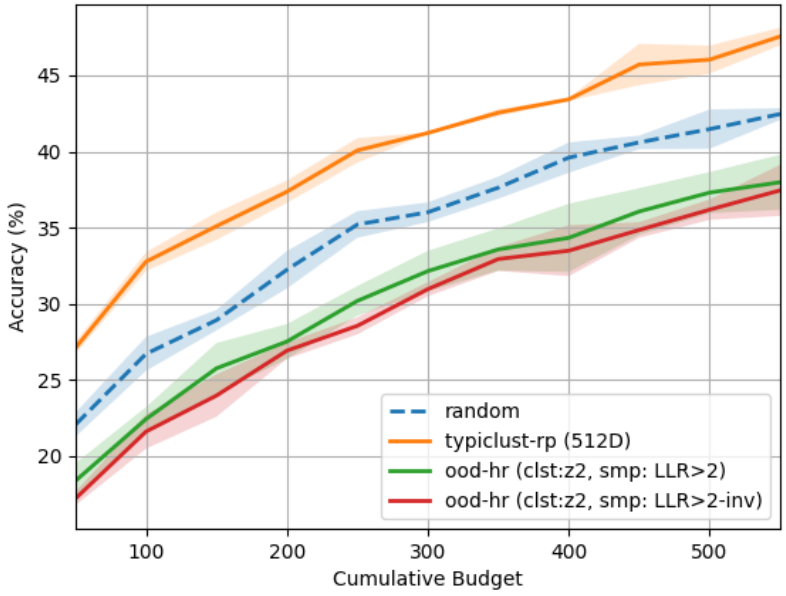

Figure 3.12: Result of active learning with small budget.

3.4 Discussion

3.4.1 TypiClust based on latent variables of original inputs

Both Figure 3.7 and Figure 3.8 show that TypiClust performs better in higher latent space regardless of the way how the representations are generated, random re-parameterization or mean sampling. These results suggest that active learning with TypiClust can benefit from representations at lower levels of abstraction, or from fine-grained characters in images. These results are consistent with our intuition that details of images are more important to measure the informativeness of the images.

Although higher latent spaces are more suitable for active learning, they cannot outperform the original TypiClust-RP and TypiClust-DC. Since TypiClust heavily relies on the representation extracted from the inputs in terms of clustering and typicality, this poor performance is likely due to the lower quality of the latent space. They are less suitable for clustering than the embedding space from SimCLR, as the latent spaces are not clearly partitioned by true labels as shown in Figure 3.6 compared to the SimCLR embedding space shown in Figure 3.3. We also observed that the quality of the latent spaces could not be improved even with deeper HVAEs such as Biva [20]. Therefore, we assume that the latent spaces generated by HVAEs from the original images do not have a structure suitable for clustering.

3.4.2 TypiClust based on latent variables of SimCLR embedding

The active learning results of TypiClust using the latent variables from the SimCLR embedding have the opposite tendency to the results of TypiClust using the latent variables from the original images. It is shown in Figure 3.9 that e-z0 has the highest accuracy and e-z2 has the lowest accuracy among the results using latent variables from SimCLR embeddings.

This inversion is thought to be due to the collapse in higher latent space. Since the inputs, the SimCLR embeddings, are already well structured, an HVAE does not need to learn hierarchical representations with multiple layers. It can simply learn the representation with the first layer for the latent variable. This leads to the collapse in the higher latent layer, and will degrade the quality of the higher latent space. Because of this collapse, clustering and typicality calculation in higher latent spaces are no longer meaningful.

More importantly, the performance is greatly improved by using the latent variables from the SimCLR embeddings, but it is still not better than the original TypiClust.

3.4.3 TypiClust based on LLR score

The accuracies with TypiClust using the LLR score are lower than the those of the original TypiClust. Moreover, both of them are worse than the accuracies by random sampling, and interestingly, selecting the highest LLR and the lowest LLR does not make a big difference. From the results, we can conclude that the LLR score of the in-distribution dataset does not provide good information for active learning. Originally, LLR score is designed to distinguish between in-distribution data and out-of-distribution data. Therefore, it is not contradictory that LLR does not work well for detecting outliers within the in-distribution dataset.

Difference by the space for clustering

As an ablation study, we perform active learning experiments with clustering in the SimCLR embedding space and sampling by the LLR score. The results are shown in Figure 3.13 and Figure 3.14. "ood-rp (clst:ND, smp:LLR>2)" and "ood-rp (clst:ND, smp:LLR>2-inv)" use the LLR score in the same way as "ood-rp (clst:zi, smp:LLR>2)" and "ood-rp (clst:zi, smp:LLR>2-inv)" do, but clusters are assigned in the SimCLR embedding space whose dimension is N.

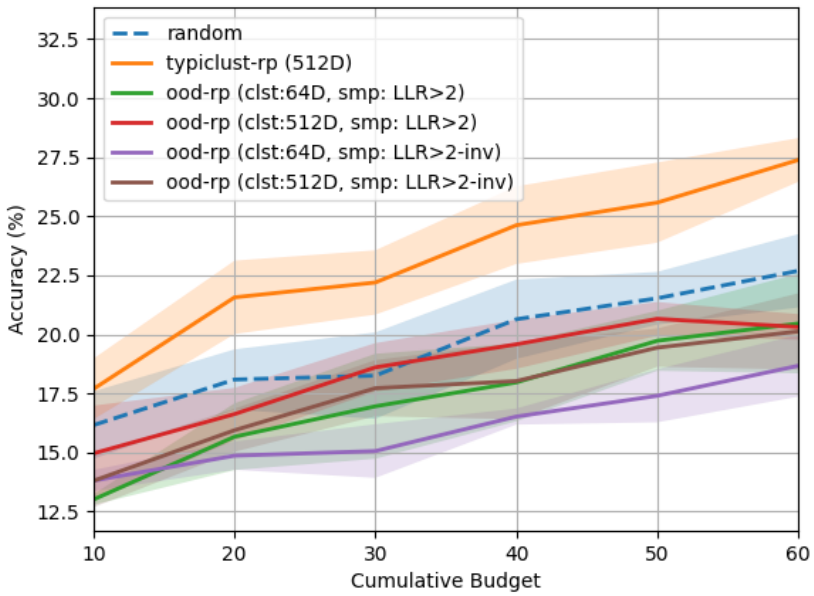

Figure 3.13: Result of active learning with tiny budget.

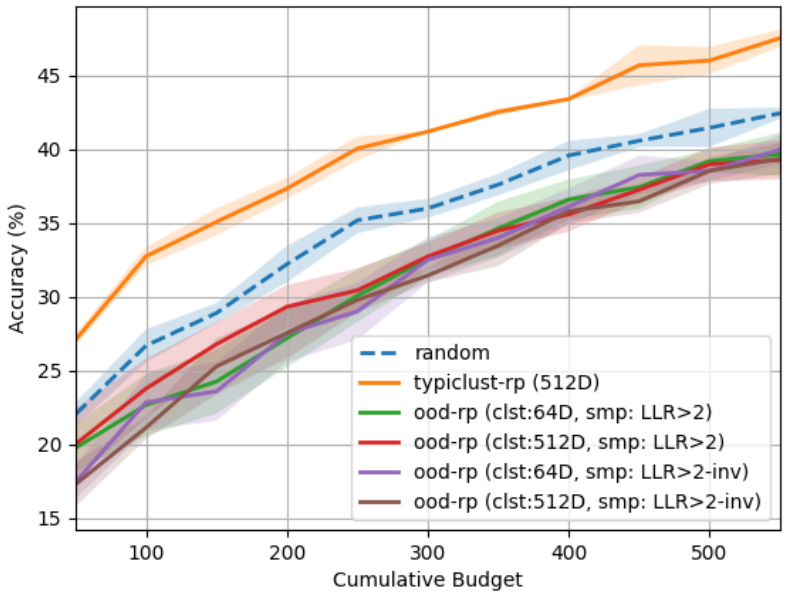

Figure 3.14: Result of active learning with small budget.

Clustering in the SimCLR embedding space improves the performance. Therefore, it is suggested that cluttered representation space, or latent space by HVAEs would affect the selection of diverse samples in a batch because of the collapsed clusters. In addition, LLR is not a good metric to measure the informativeness of data for active learning because the accuracy is still worse than the random sampling even if the SimCLR embeddings are used for clustering.

Chapter 4

Active Learning with different SimCLR embeddings

4.1 Background

TypiClust [13] uses the same SimCLR embedding by a single backbone model for both clustering and typicality. Although the default dimension of the embedding space is 512, it can be set to a different dimension. It is also possible to use other embeddings for clustering and typicality. In this chapter, we explore the best combination of dimensions for successful active learning.

4.2 Method

Several SimCLR embeddings of different dimensions ($D = 8,\ 16,\ 32,\ 64,\ 128,\ 256,\ 512$) are prepared first. Each combination of D_{clst} and D_{typ} is examined on CIFAR10 with the tiny budget, where D_{clst} is the embedding dimension for clustering and D_{typ} is for typicality.

4.3 Result

4.3.1 Result with Fixed D_{clst}

The results with fixed D_{clst} and baselines (typiclust-rp (512D) and random) are shown from Figure 4.1 to Figure 4.7. The notation "typ-rp (clst:MD, typ: ND)" means $D_{clst} = M$ and $D_{typ} = N$.

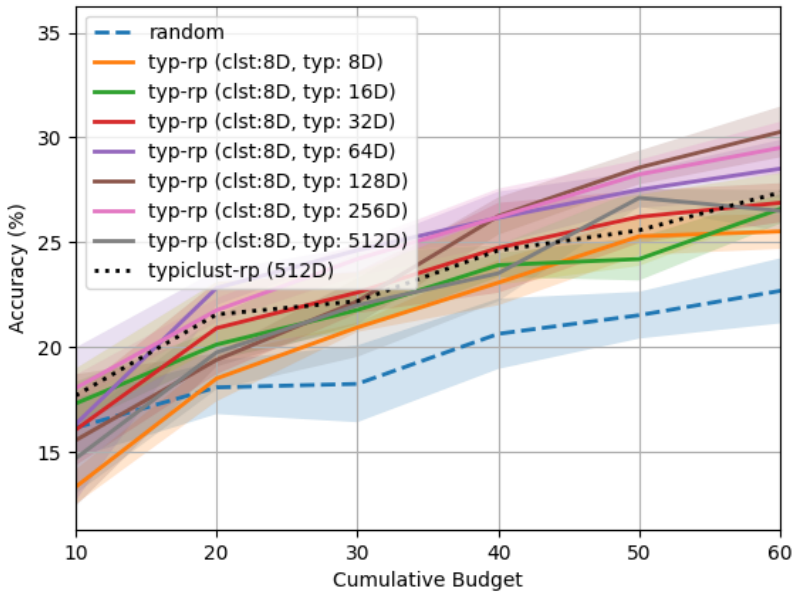

Figure 4.1: Active Learning Results (Fixed: $D_{clst} = 8$)

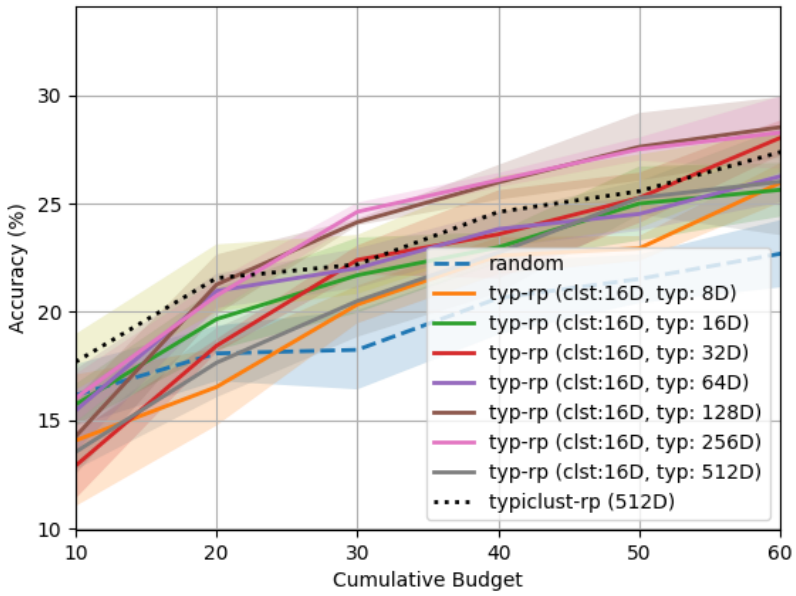

Figure 4.2: Active Learning Results (Fixed: $D_{clst} = 16$)

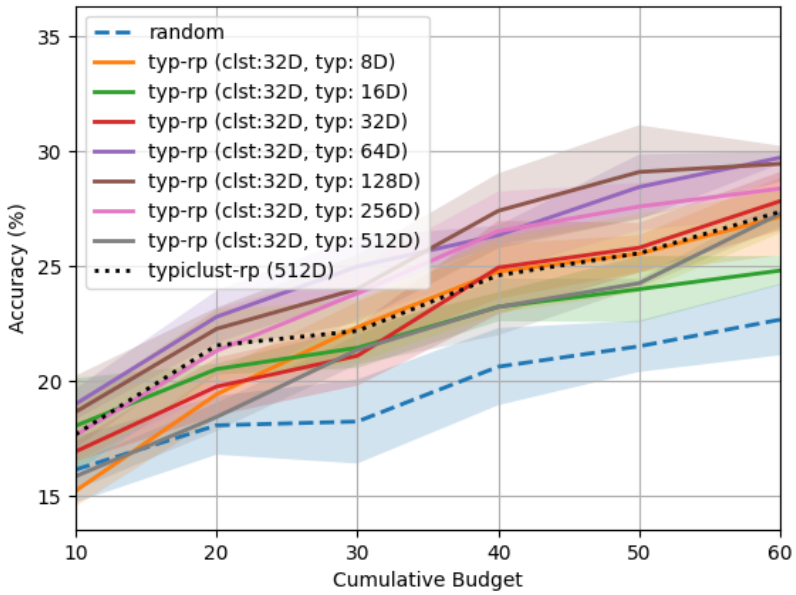

Figure 4.3: Active Learning Results (Fixed: $D_{clst} = 32$)

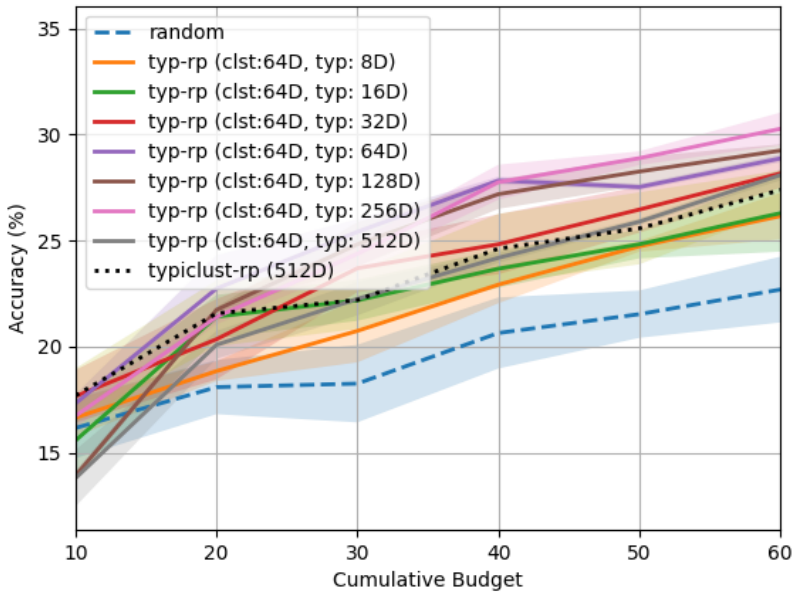

Figure 4.4: Active Learning Results (Fixed: $D_{clst} = 64$)

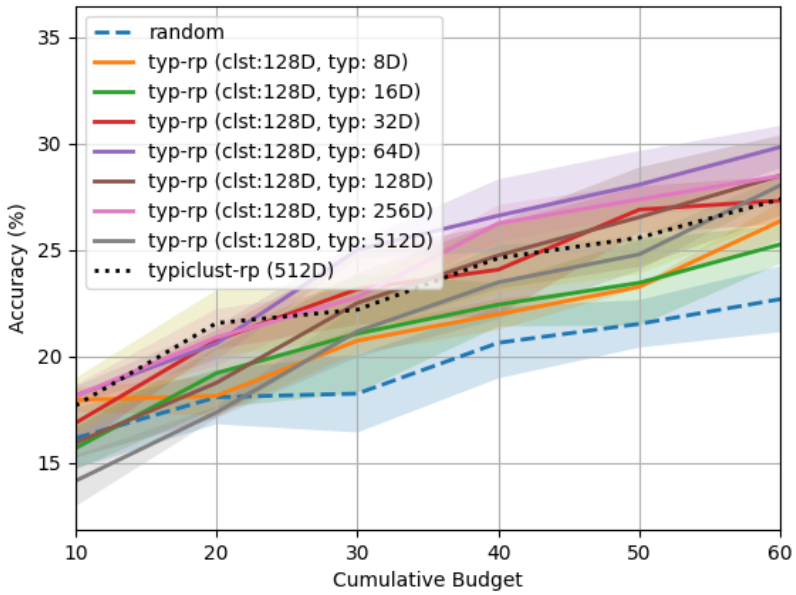

Figure 4.5: Active Learning Results (Fixed: $D_{clst} = 128$)

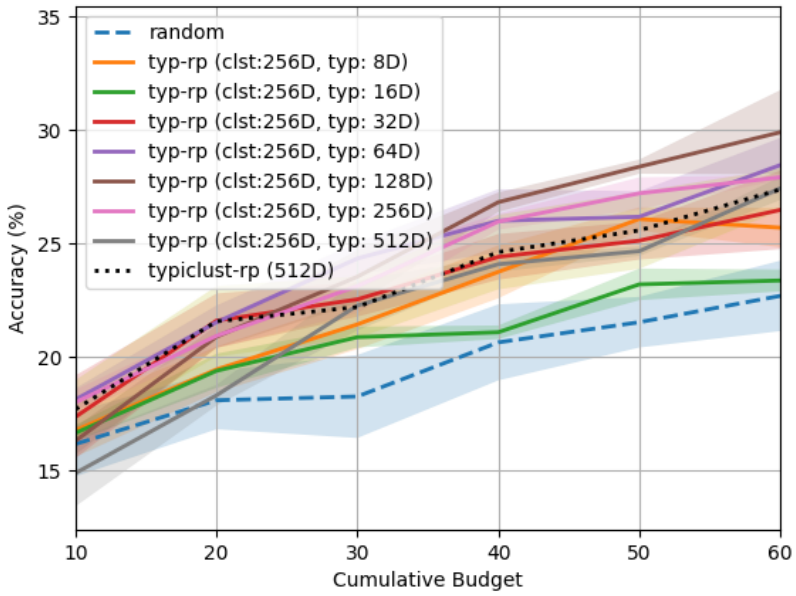

Figure 4.6: Active Learning Results (Fixed: $D_{clst} = 256$)

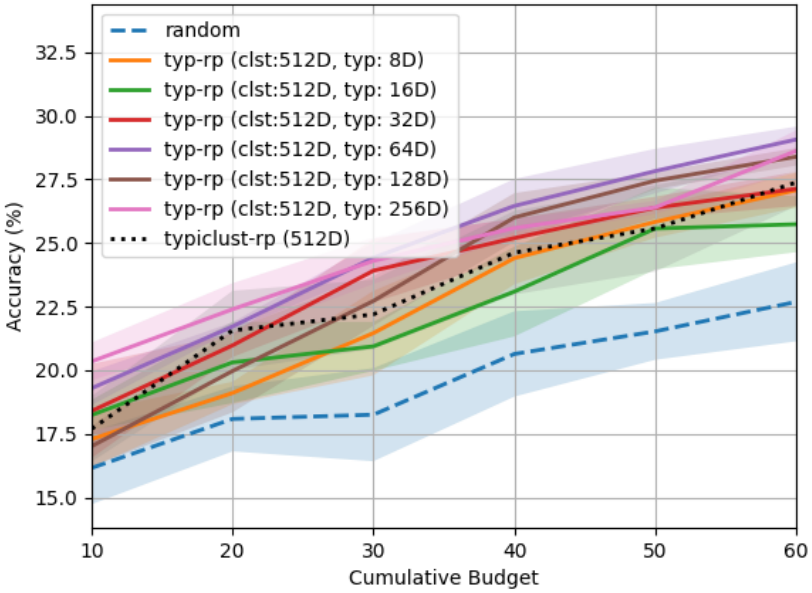

Figure 4.7: Active Learning Results (Fixed: $D_{clst} = 512$)

4.3.2 Result with Fixed D_{typ}

The results with fixed D_{typ} and baselines (typiclust-rp (512D) and random) are shown from Figure 4.8 to Figure 4.14. The notation "typ-rp (clst:MD, typ: ND)" means $D_{clst} = M$ and $D_{typ} = N$.

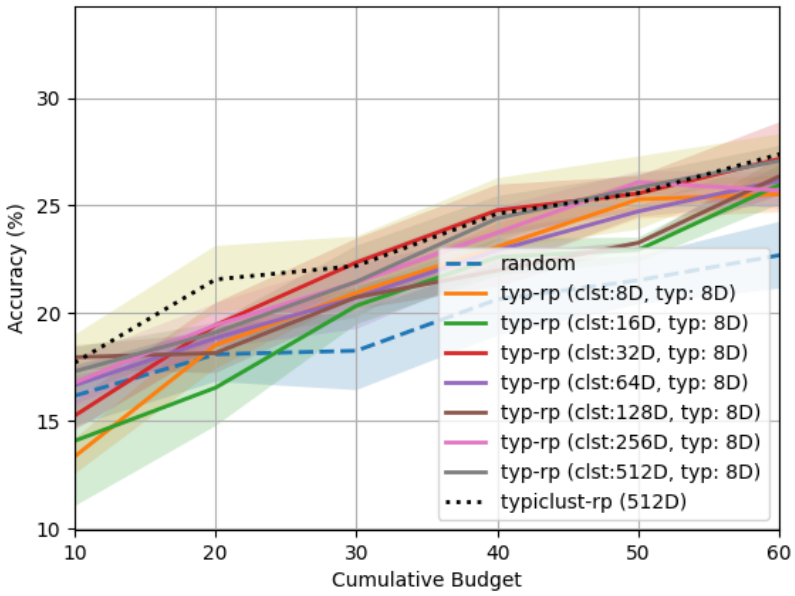

Figure 4.8: Active Learning Results (Fixed: $D_{typ} = 8$)

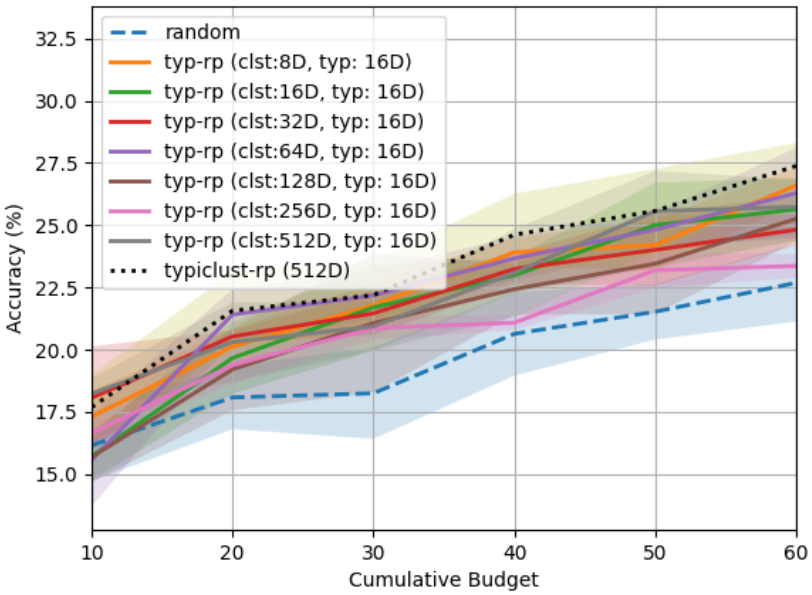

Figure 4.9: Active Learning Results (Fixed: $D_{typ} = 16$)

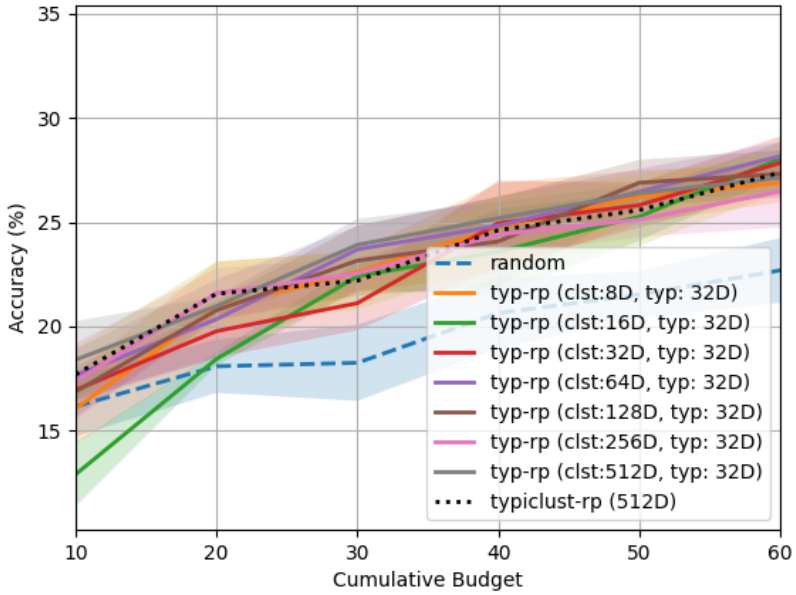

Figure 4.10: Active Learning Results (Fixed: $D_{typ} = 32$)

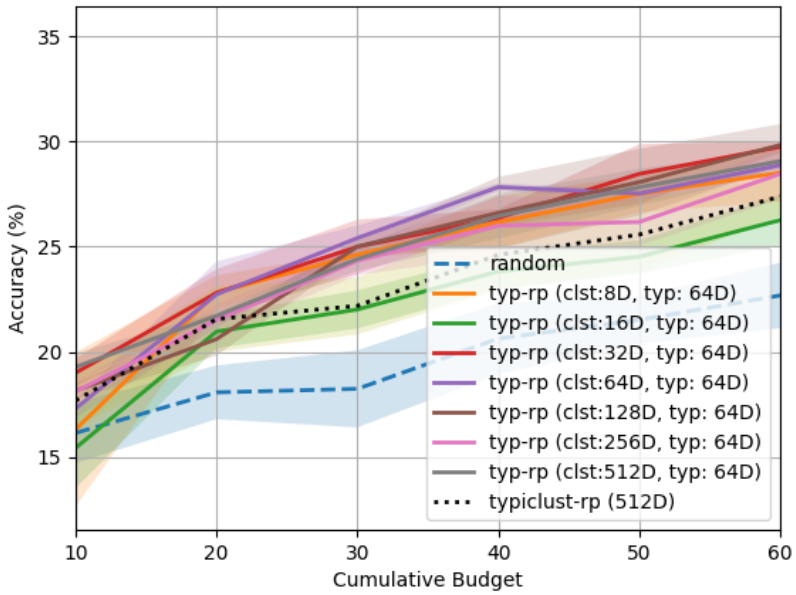

Figure 4.11: Active Learning Results (Fixed: $D_{typ} = 64$)

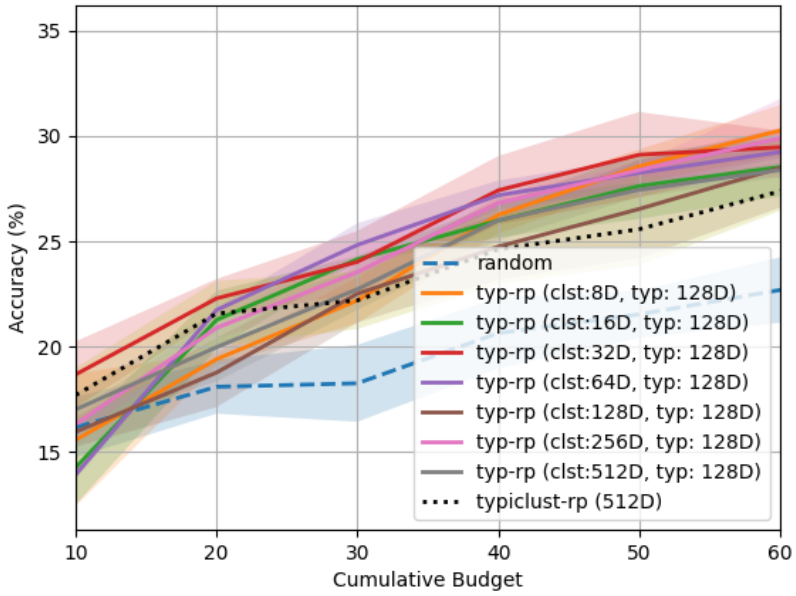

Figure 4.12: Active Learning Results (Fixed: $D_{typ} = 128$)

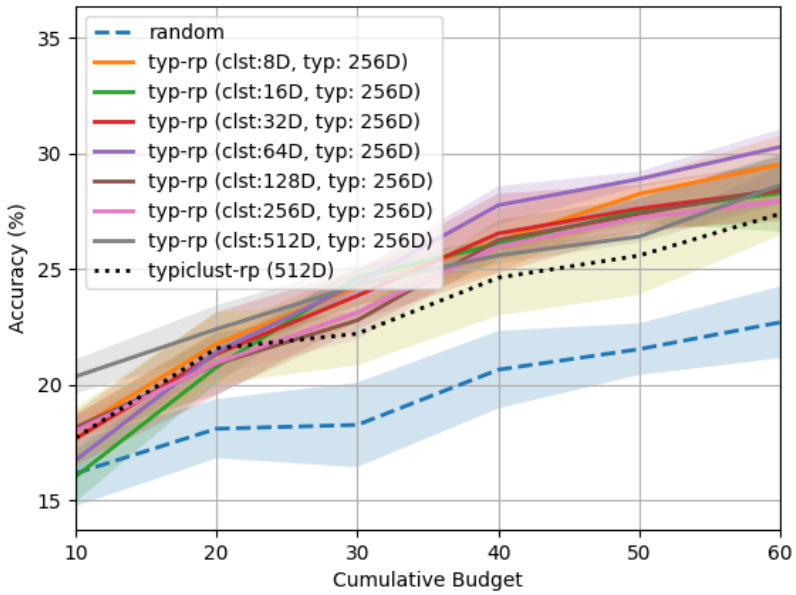

Figure 4.13: Active Learning Results (Fixed: $D_{typ} = 256$)

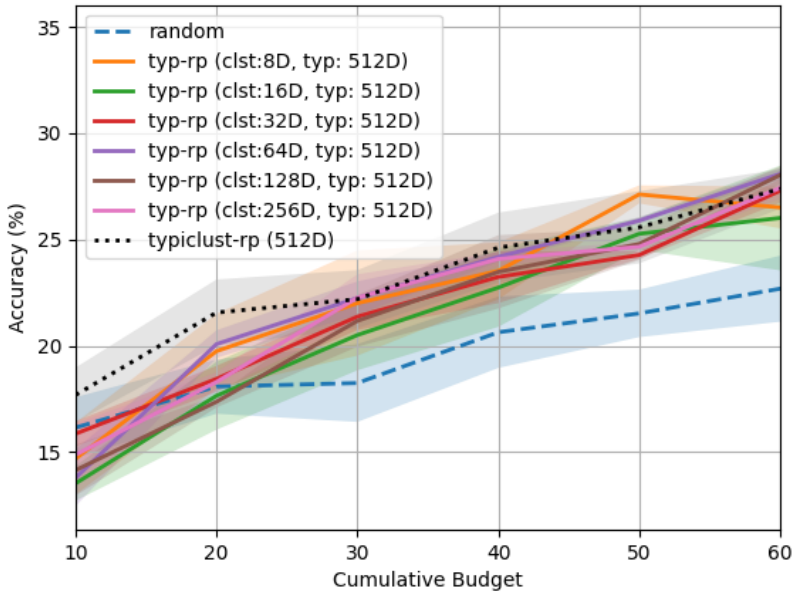

Figure 4.14: Active Learning Results (Fixed: $D_{typ} = 512$)

4.4 Discussion

There are always some combinations of D_{clst} and D_{typ} that outperform the original typiclust (shown as typiclust-rp (512D)) in every figure with fixed D_{clst}. On the other hand, there are not always such combinations with fixed D_{typ}. From these results it can be concluded that D_{typ} affect the performance of active learning more than D_{clst}. Good D_{typ} values lead to higher accuracy of the classifier, but it cannot outperform the original TypiClust regardless of D_{clst} if a bad D_{typ} is chosen. In this setting, a good D_{typ} is one of $(64, 128, 256)$. When D_{typ} is set to the good value, every combination of D_{typ} and D_{clst} outperforms the original TypiClust-RP except the case of $(D_{clst},\ D_{typ}) = (16, 64)$. This suggests that the embedding space for clustering does not have a much effect on the performance of TypiClust.

The figures from Figure 4.15 to Figure 4.21 show that the embedding spaces of different dimensions have similar structures without critical collapses.

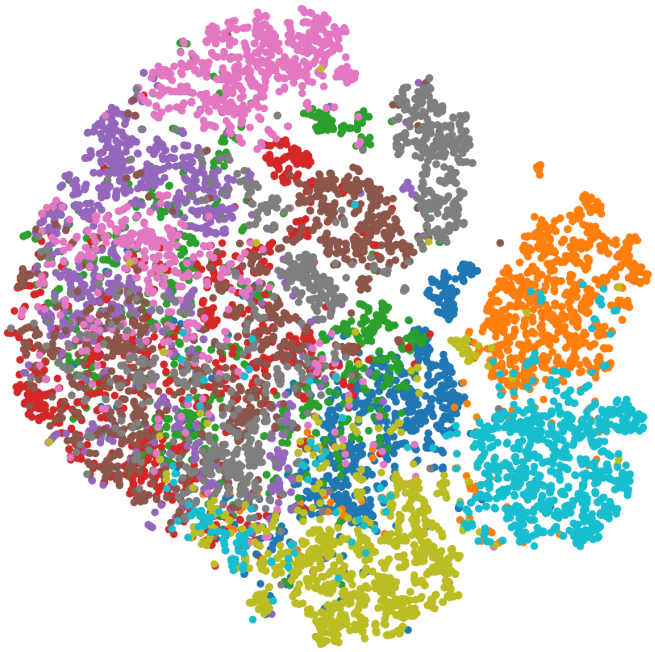

Figure 4.15: t-SNE plot of CIFAR10 test dataset ($D = 8$)

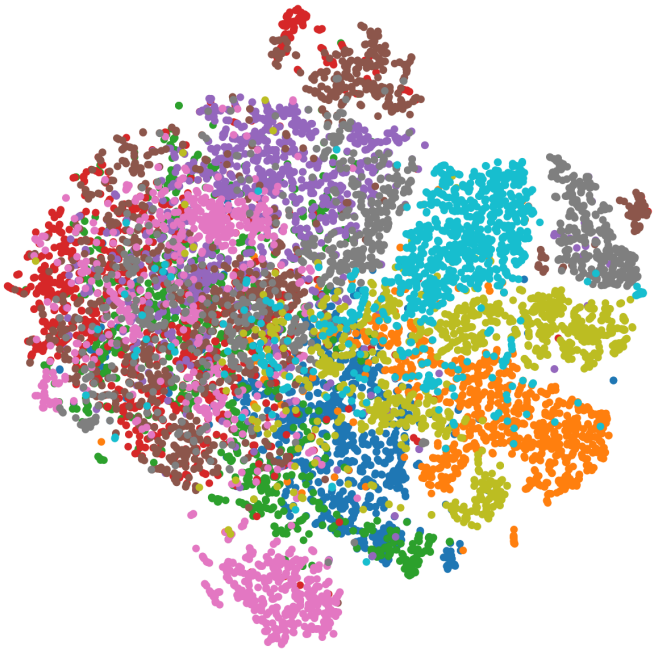

Figure 4.16: t-SNE plot of CIFAR10 test dataset ($D = 16$)

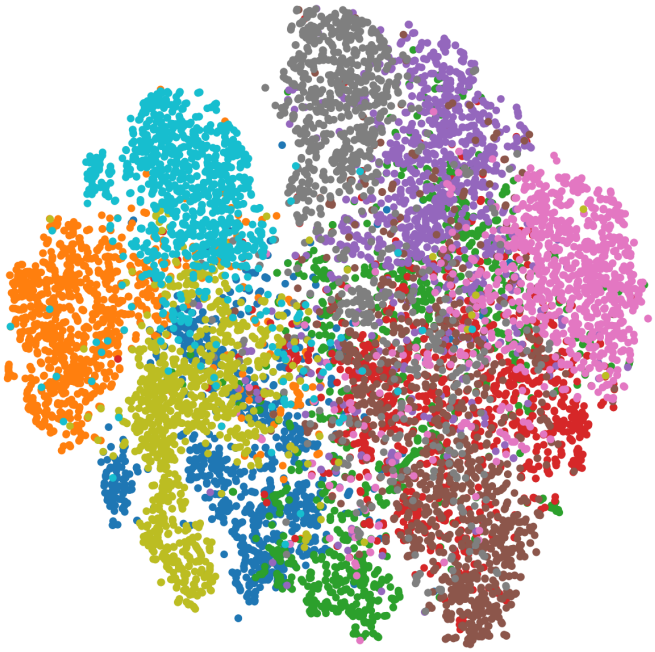

Figure 4.17: t-SNE plot of CIFAR10 test dataset ($D = 32$)

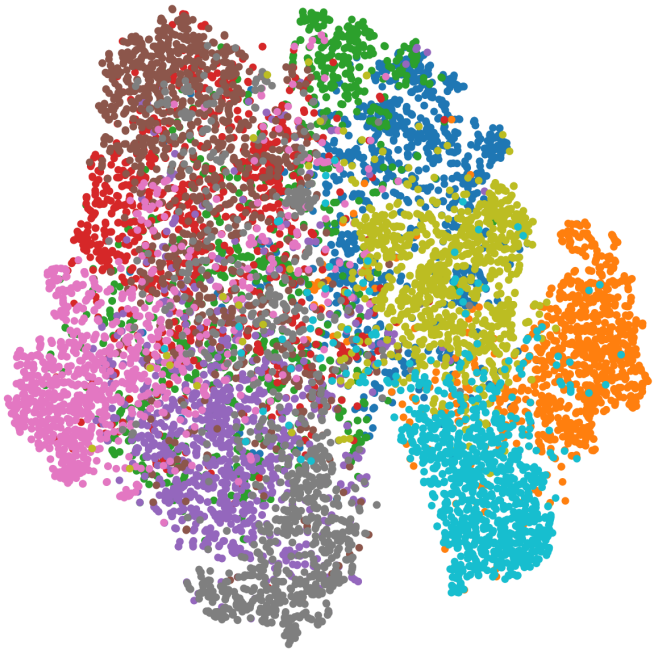

Figure 4.18: t-SNE plot of CIFAR10 test dataset ($D = 64$)

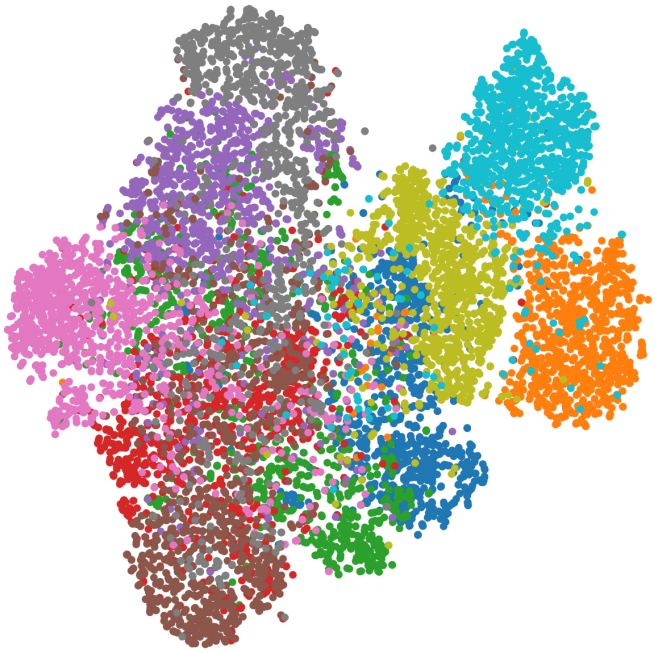

Figure 4.19: t-SNE plot of CIFAR10 test dataset ($D = 128$)

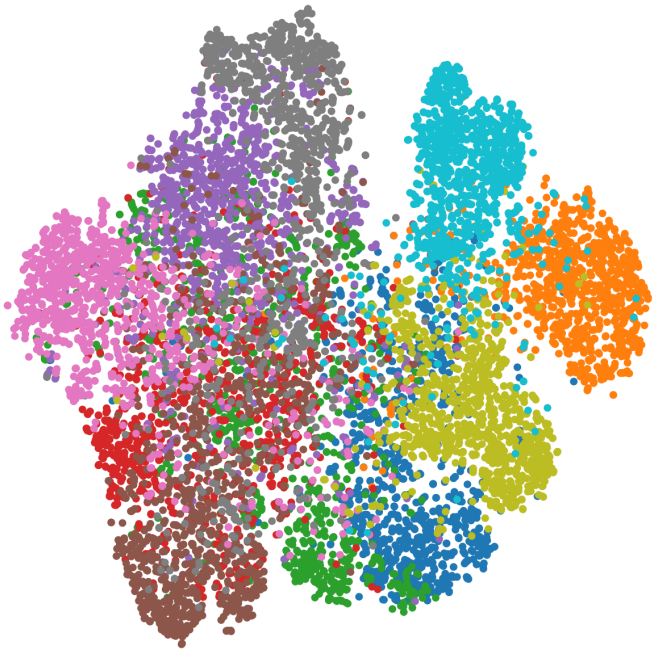

Figure 4.20: t-SNE plot of CIFAR10 test dataset ($D = 256$)

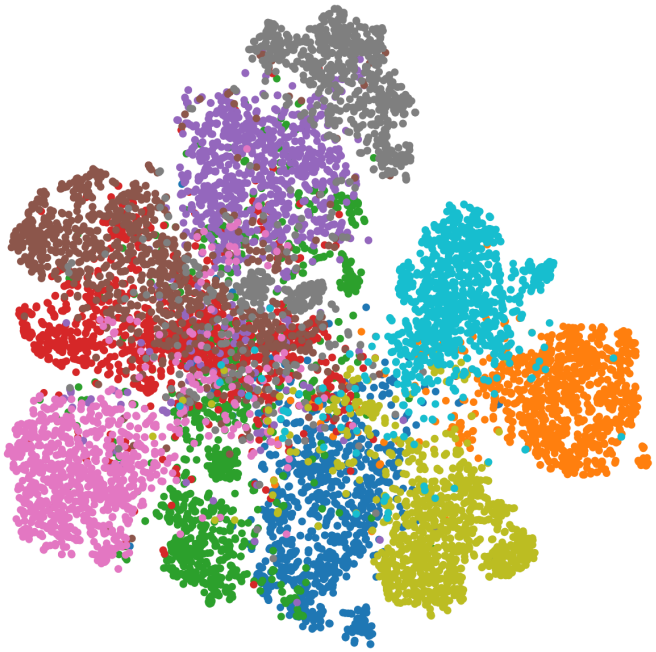

Figure 4.21: t-SNE plot of CIFAR10 test dataset ($D = 512$)

Chapter 5

SUPClust: Active Learning at the Boundaries

5.1 Background

How can a model correctly classify points of different classes? Classical support vector machines search for a hyperplane that separates two classes with the largest possible margin (See Figure 5.1). The points that lie on this decision boundary are called *support vectors*. In other words, these support vectors define the boundary of all samples of a class and are critical for a model to know in order to correctly separate the classes. We hypothesize that points close to the decision boundary are similarly relevant for neural network-based models.

In this chapter, we propose a novel active learning method *SUPClust* that attempts to identify these points so that they can be annotated. Since the labels of the points are not known a priori, we rely on self-supervised representation learning combined with clustering to decompose the high-dimensional input space. For each cluster, we then identify the points that are close to a neighboring cluster, thereby selecting potential support vector points. By selecting points from all clusters, we ensure a broad coverage of the input space. In practice, data distributions often contain outliers and the decision boundary between different classes is not always clear. For this reason, we further constrain our points to be somewhat typical according to a typicality metric introduced by [13].

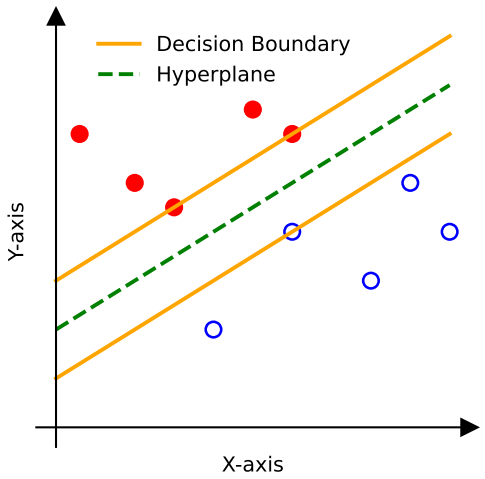

Figure 5.1: Decision boundary of an SVM classifier.

5.2 Method

SVM classifiers are defined by a few key points located on the decision boundary between the categories. Our querying strategy selects instances close to the decision boundary because they also provide a strong signal to the learning process of neural network-based models. Traditional active learning methods have approached this problem by using model uncertainty as a cue for samples at the decision boundary. However, these methods suffer from the cold start problem, where in low budget scenarios, the model uncertainty is unable to identify hard instances. In this work, we introduce a novel method to find such samples by exploiting pre-trained representations. We can see in Figure 5.2, using CIFAR-10 as an example, that similar categories are clustered together in the representation space.

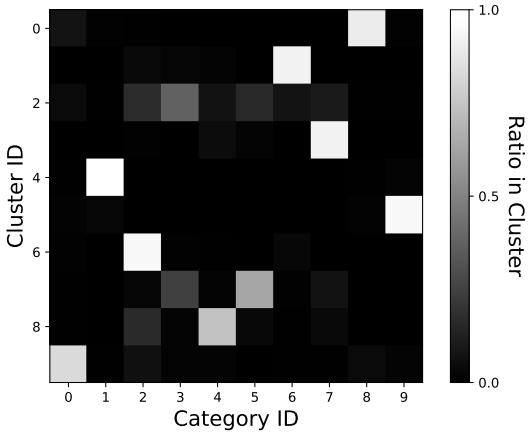

Figure 5.2: Distribution of classes within each cluster on SimCLR embeddings for CIFAR10. Cluster boundaries align with category boundaries

Since category boundaries align with cluster boundaries, we use clustering to identify samples of interest. To quantify the proximity to the decision boundary, we compute, for each sample, the weighted average distance to all other cluster centers. The weights are the same for all samples within a cluster and depend on the distance of the cluster center to all other cluster centers. Clusters located at the "edge" of the data distribution select an instance that is close to the nearest cluster. Conversely, clusters in the "middle" of the distribution will not select instances that are close to only one of the clusters. To normalize the weights to 1, we use the softmax function with the negative L2 distance as the logits and the temperature parameter T. For a point in cluster i, the weight for the cluster j is given by

$$w_i^j = \frac{\exp\left(-\frac{\|\mathbf{c}_i - \mathbf{c}_j\|}{T}\right)}{\sum_{k \in C \setminus \{i\}} \exp\left(-\frac{\|\mathbf{c}_i - \mathbf{c}_k\|}{T}\right)}, \tag{5.1}$$

where $\mathbf{c}_i$, $\mathbf{c}_j$ and $\mathbf{c}_k$ are the centers of cluster i, j and k respectively, and C is the set of all clusters. For cluster i, we select the sample $\mathbf{x}$, that has the minimum distance, or maximum SUP to the decision boundary computed by Equation (5.2).

$$SUP(\mathbf{x}) = \left(\sum_{j \in C \setminus \{i\}} w_i^j \|\mathbf{x} - \mathbf{c}_j\| \right)^{-1} \tag{5.2}$$

The entire procedure of SUPClust is shown in Algorithm 1.

Algorithm 1 SUPClust

Require: $B = N \cdot b$ (B is an annotation budget for active learning)

Pretrain a backbone by SimCLR
Extract self-supervised embedding
for $e = 1, \ldots, N$ **do**
  Build $e \cdot b$ clusters $\mathcal{C}_1, \ldots, \mathcal{C}_{e \cdot b}$ in the embedding space
  Choose b clusters $\mathcal{C}'_1, \ldots, \mathcal{C}'_b$ with less labeled data points and bigger cluster size
  for $i = 1, \ldots, b$ **do**
    Calculate typicalities of data points in cluster $\mathcal{C}'_i$
    Filter data points using typicality (only top 10% data points pass)
    Calculate $SUP(\mathbf{x})$ for data points $\mathbf{x}$ in cluster $\mathcal{C}'_i$
    Select the data point with highest SUP to be annotated
  end for
  Query true labels of newly selected points
  Train a classifier on the labeled dataset $\mathcal{L}$
end for

5.3 Result

5.3.1 Experimental setup

All strategies are evaluated on image classification tasks using CIFAR-10, CIFAR-100 [1], CIFAR-10-LT [28], and ISIC-2019 [29], following the benchmarking suite proposed by [30]. CIFAR-10 and CIFAR-100 consist of 60k natural images of size 32x32 with 10 and 100 classes. The datasets are divided into two subsets: 50k images are allocated for training purposes, while the remaining 10k images are reserved for testing. CIFAR-10-LT is a class-imbalanced subset of CIFAR-10. We apply an imbalance ratio of 50, which means a 50-fold difference in the number of images between the most and least frequent class. ISIC-2019 consists of 25331 skin cancer images with 8 imbalanced classes. To standardize the image dimensions, all images are resized to 224x224 pixels. Given the imbalanced nature of the dataset, the balanced accuracy (Mean Recall) is used as the evaluation metric.

$$\text{Recall of class } c = \frac{TP_c}{TP_c + FN_c} \tag{5.3}$$

$$\text{Mean Recall} = \frac{1}{C} \sum_{c=1}^{C} \frac{TP_c}{TP_c + FN_c} \tag{5.4}$$

In line with TypiClust, we adopt tiny and small budget sizes, with query step sizes of 1 and 5 times the number of classes, respectively. Note that TypiClust, ProbCover, and our querying strategy are designed for the low-budget regimes, while others are suitable for high-budget regimes.

We evaluate active learning strategies in the following two frameworks.

1. Fully supervised (FSL): training a deep neural network, ResNet18 [31], exclusively on the labeled set which is acquired by active queries.
2. Fully supervised with self-supervised embedding (SSL): training a linear classifier on the labeled embeddings obtained by active queries.

These self-supervised embeddings for the classifier are obtained from a pre-trained SimCLR [14]. Within these frameworks, we compare SUPClust to nine baseline strategies: Random, Margin, Least confidence, Entropy, BALD, Coreset, DBAL, TypiClust, and ProbCover. For the clustering and sampling with TypiClust and SUPClust, we use SimCLR 512-dimensional representations, namely the ResNet18 backbone for CIFAR-10, CIFAR-10-LT50 and ISIC-2019, and the ResNet34 for CIFAR-100.

5.3.2 Ablation Study

To assess the importance of individual components within SUPClust, we perform ablation experiments for each component. The results are shown in Figure 5.3. When we omit our SUP-based acquisition metric (SUPClust w/o SUP) and instead randomly selecting a sample from the top 10% typical samples within each cluster, the performance drops significantly, falling below that of TypiClust. Similarly, relying solely on SUP without considering typicality for sample selection (SUPClust w/o typicality) fails to achieve the performance levels observed with other querying strategies. For comparison, we also show the result of TypiClust (typiclust-rp), which always selects the most typical sample of a cluster. Our results showcase that all components of SUPClust are necessary and contribute to its performance.

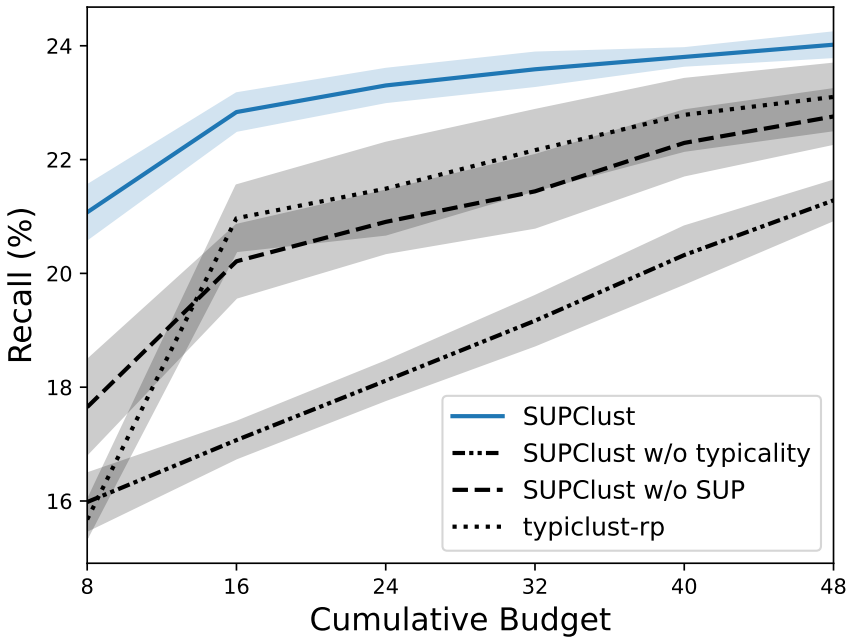

Figure 5.3: Ablation study on ISIC-2019 with tiny-budget (SSL)

5.3.3 Sampled Points

In order to describe the differences between TypiClust and SUPClust, 100 labeled points that were selected by each strategy are shown in Figure 5.3.3 and Figure 5.3.3. Active learning is performed for 10 episodes, selecting 10 data points at each episode on CIFAR10.

For clusters at the "edge" of the data distribution, SUPClust tends to select samples that are closer to other clusters in the embedding space. For clusters in the "center" of the data distribution, SUPClust tends to select samples that are near the center of its cluster. Conversely, TypiClust selects data points which are typical regardless of its location. This is the reason why we introduced softmax function into the metric. Although TypiClust does not take into account the relationship of data points to other clusters, SUPClust uses it to select more informative data for active learning.

Figure 5.4: Sampled points by TypiClust (marked as "+") in 2-dimensional t-SNE plot by CIFAR10 embedding. Colors represent corresponding true labels

Figure 5.5: Sampled points by SUPClust (marked as "+") in 2-dimensional t-SNE plot by CIFAR10 embedding. Colors represent corresponding true labels

5.3.4 Cluster Boundary vs Category Boundary

SUPClust relies on clusters formed in an embedding space for diversity in a batch. Therefore, clusters need to be well structured. In addition, since SUPClust aims to exploit the points on the decision boundary, the boundary of clusters should be aligned with the category boundary.

The relationships between category (true label) and cluster ID with different number of clusters are shown in Figure 5.2, 5.6, and 5.7. Most of clusters consist of one or two primary category data, aligning the cluster boudary with the category boundary.

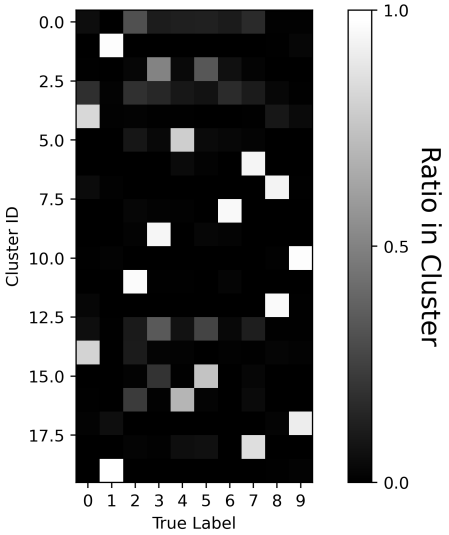

Figure 5.6: Distribution of classes within 20 clusters on SimCLR embeddings for CIFAR10

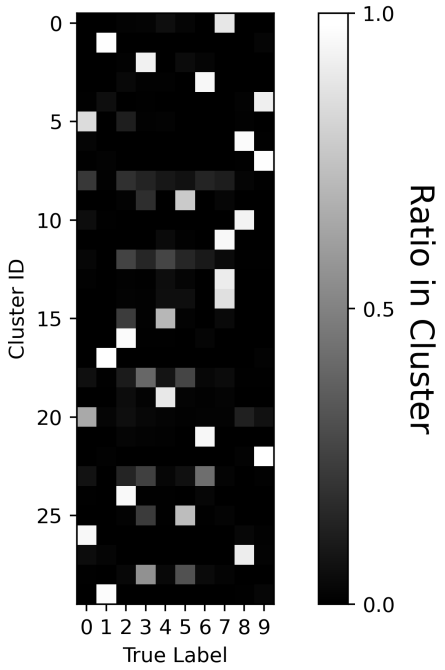

Figure 5.7: Distribution of classes within 30 clusters on SimCLR embeddings for CIFAR10

5.3.5 Relation between Typicality and SUP

Typicality and SUP are not correlated, see Figure 5.8 and 5.9, so using both metrics for sample selection can improve performance.

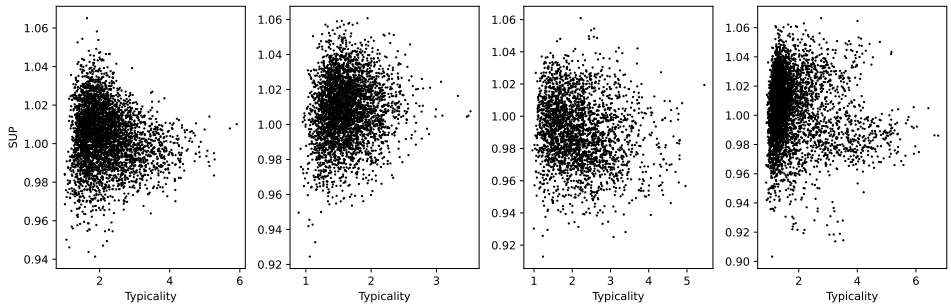

Figure 5.8: Relation between typcality and SUP on CIFAR10 on 4 randomly selected clusters at episode 2 in the tiny budget regime, with temperature 1. Typicality and SUP has no strong correlation, using both metrics to select instances can improve the querying strategy

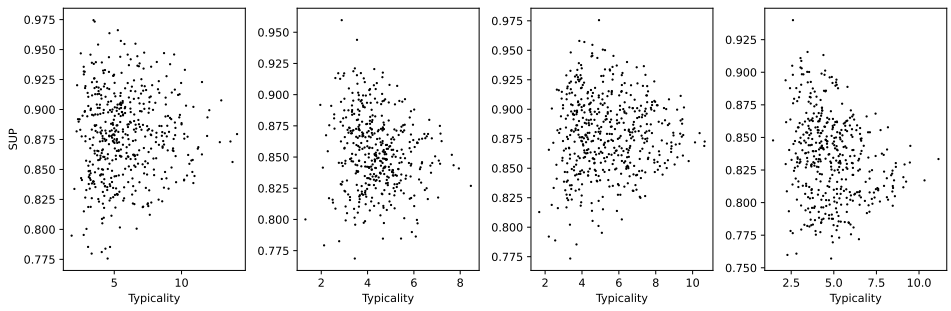

Figure 5.9: Relation between typcality and SUP on ISIC2019 on 4 randomly selected clusters at episode 5 in the small budget regime, with temperature 1

5.3.6 Main Results

The results in tiny-budget and small-budget regimes are shown in Figure 5.10. The results with the SSL settings are shown as solid lines, and the results with the FSL settings are shown as dotted lines.

In the low-budget regime, diversity-based methods such as TypiClust, Coreset and ProbCover generally outperform their uncertainty-based counterparts. This is to be expected, since uncertainty-based methods bring stronger benefits only in higher budget regimes because they use a classifier trained on a labeled dataset to measure "uncertainty". Building on the self-supervised pre-trained embeddings improves performance on all datasets. The performance of Coreset on CIFAR10-LT50 in the SSL setting is surprising. The embeddings of the pre-trained backbone allow Coreset to select very informative samples. Unfortunately, when training in the FSL setting or on any other dataset, the performance of Coreset is reduced compared to other algorithms.

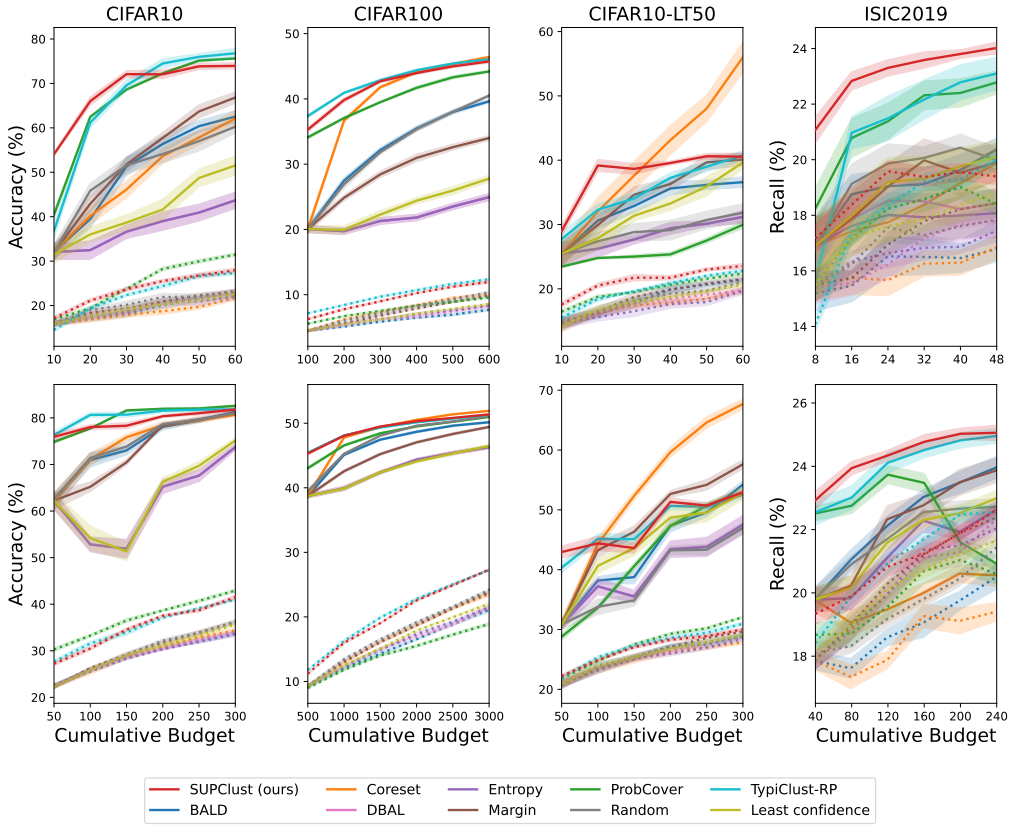

Figure 5.10: Main results in tiny-budget (top) and small-budget (bottom) regimes. Solid lines represent results with the SSL settings, and dotted lines represent results with the FSL setting. The mean and standard error with 10 different random seeds are shown. Our method (SUPClust) shows robust performance compared to other baselines, across all datasets and both budget regimes here

5.4 Discussion

SUPClust introduces a novel metric SUP, which is a non-label-based means of quantifying the distance of each sample to the decision boundary, for active learning, and the performance is on par with or better than previous methods. Selecting instances close to the decision boundaries between categories for annotation provides the classifier with strong signal to be trained well in low-budget regimes. SUPClust outperforms baseline methods especially on imbalanced datasets, CIFAR10-LT50 and ISIC2019. According to [32], data points around the decision boundaries tend to be balanced even if the overall distribution is imbalanced.

This may be the reason why SUPClust has an advantage over other methods on imbalanced datasets.

In our research, SimCLR is used to obtain embeddings following TypiClust [13]. There are other self-supervised learning methods which can be used for feature extraction such as SwAV [33] and DINO [34, 35]. The possiblity of other self-supervised pretext tasks for SUPClust remains to be explored.

Chapter 6

Conclusion

In this thesis, we delved into active learning to mitigate the cold start problem of active learning in low-budget regimes. In Chapter 2, active learning strategies with HVAEs was proposed, only to find the representation spaces by HVAEs are not suitable for active learning and out-of-distribution detection are not able to find outliers in in-distribution dataset. In Chapter 3, we tried out smaller dimensions of SimCLR embeddings for TypiClust. Some combinations of D_{clst} and D_{typ} worked better than the original TypiClust, offering the possibility to compress the embedding dimension. Although a smaller SimCLR embedding dimension leads to faster iteration of active learning, different embedding dimensions require multiple model trainings with SimCLR, which consumes more computational resources. Moreover, the faster selection does not have a significant impact on the total active learning time because the biggest bottleneck of active learning in practice is the time required for annotation by human. In Chapter 4, we developed a novel querying strategy for active learning, SUPClust, which is designed to take advantage of data points close to the decision boundary. SUPClust was investigated on four different datasets including imbalanced datasets, and the empirical study showed that SUPClust performs well compared to existing methods. Moreover, SUPClust showed powerful performance especially on imbalanced datasets. From these results, we conclude that the data points on the decision boundaries are useful for active learning in low-budget regimes.

Future work could include theoretical analysis of SUPClust. There exists a body of previous research on active learning strategies that sample instance on the decision boundaries of SVMs. However, these methods are not designed for deep active learning. Going through these nevertheless would be helpful to analyze our strategy from a theoretical point of view. Another future direction is to consider the sampling strategy for imbalanced datasets. Although our approach works on imbalanced datasets, it is inevitable to oversample data points with the majority labels. By using the label information in the labeled dataset to balance the label distribution of the labeled dataset, we may be able to further improve our querying strategy.

Bibliography

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, pp. 84–90, Dec. 2012.

[2] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, 2015, pp. 448–456.

[3] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," Oct. 2020. arXiv: 2010.11929 [cs.CV].

[4] S. Srivastava and G. Sharma, "OmniVec: Learning robust representations with cross modal sharing," *ArXiv*, vol. abs/2311.05709, Nov. 2023.

[5] M.-F. Balcan, A. Broder, and T. Zhang, "Margin based active learning," in *Learning Theory*, Springer Berlin Heidelberg, 2007, pp. 35–50.

[6] D. D. Lewis, "A sequential algorithm for training text classifiers: Corrigendum and additional data," *SIGIR Forum*, vol. 29, no. 2, pp. 13–19, Sep. 1995.

[7] A. I. Schein and L. H. Ungar, "Active learning for logistic regression: An evaluation," *Mach. Learn.*, vol. 68, no. 3, pp. 235–265, Oct. 2007.

[8] A. J. Joshi, F. Porikli, and N. Papanikolopoulos, "Multi-class active learning for image classification," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2009, pp. 2372–2379.

[9] N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel, "Bayesian active learning for classification and preference learning," Dec. 2011. arXiv: 1112.5745 [stat.ML].

[10] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Mar. 2017, pp. 1183–1192.

[11] A. Kirsch, J. Van Amersfoort, and Y. Gal, "Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[12] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A Core-Set approach," Aug. 2017. arXiv: 1708.00489 [stat.ML].

[13] G. Hacohen, A. Dekel, and D. Weinshall, "Active learning on a budget: Opposite strategies suit high and low budgets," in *Proceedings of the 39th International Conference on Machine Learning*, vol. 162, PMLR, Jul. 2022, pp. 8175–8195.

[14] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning*, H. D. Iii and A. Singh, Eds., ser. Proceedings of Machine Learning Research, vol. 119, PMLR, 2020, pp. 1597–1607.

[15] O. Yehuda, A. Dekel, G. Hacohen, and D. Weinshall, "Active learning through a covering lens," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., May 2022, pp. 22 354–22 367.

[16] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," en, *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[17] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[18] I. Goodfellow *et al.*, "Generative adversarial nets," *Adv. Neural Inf. Process. Syst.*, vol. 27, 2014.

[19] C. K. Sønderby, T. Raiko, and L. Maaløe, "Ladder variational autoencoders," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2016.

[20] L. Maaløe, M. Fraccaro, V. Liévin, and O. Winther, "Biva: A very deep hierarchy of latent variables for generative modeling," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[21] A. Vahdat and J. Kautz, "NVAE: A deep hierarchical variational autoencoder," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 19 667–19 679, 2020.

[22] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," Jul. 2018. arXiv: 1807.03748 [cs.LG].

[23] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick, "Momentum contrast for unsupervised visual representation learning," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 9726–9735, Nov. 2019.

[24] F. Pourkamali-Anaraki and M. B. Wakin, "The effectiveness of variational autoencoders for active learning," Nov. 2019. arXiv: 1911.07716 [cs.LG].

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2009, pp. 248–255.

[26] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," *ICCV*, pp. 3730–3738, Nov. 2014.

[27] J. D. Havtorn, J. Frellsen, S. Hauberg, and L. Maaløe, "Hierarchical VAEs know what they don't know," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 2021, pp. 4117–4128.

[28] K. Cao, C. Wei, A. Gaidon, N. Arechiga, and T. Ma, "Learning imbalanced datasets with label-distribution-aware margin loss," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[29] M. A. Kassem, K. M. Hosny, and M. M. Fouad, "Skin lesions classification into eight classes for ISIC 2019 using deep convolutional neural network and transfer learning," *IEEE Access*, vol. 8, pp. 114 822–114 832, 2020.

[30] C. T. Lüth, T. J. Bungert, L. Klein, and P. F. Jaeger, "Toward realistic evaluation of deep active learning algorithms in image classification," Jan. 2023. arXiv: 2301.10625 [cs.CV].

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 770–778, Dec. 2015.

[32] S. Ertekin, J. Huang, and C. L. Giles, "Active learning for class imbalance problem," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '07, Amsterdam, The Netherlands: Association for Computing Machinery, Jul. 2007, pp. 823–824.

[33] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 9912–9924, 2020.

[34] M. Caron *et al.*, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, openaccess.thecvf.com, 2021, pp. 9650–9660.

[35] M. Oquab *et al.*, "DINOv2: Learning robust visual features without supervision," Apr. 2023. arXiv: 2304.07193 [cs.CV].

Appendix A

Hyperparameters

Hyperparameters used for training a classifier in the active learning scheme are shown in Table A.1.

Table A.1: Hyperparameters for Active Learning

Property	CIFAR10/100/10-LT	ISIC2019
Epochs	200	200
Batch Size	$\min(100, \lvert\mathcal{L}\rvert)$	$\min(50, \lvert\mathcal{L}\rvert)$
Optimizer	SGD	SGD
Learning Rate	0.025	0.0125
LR Scheduling	Cosine Annealing	Cosine Annealing
LR Dampening	0.0	0.0
LR T_{max}	200	200
Momentum	0.9	0.9
Nesterov	True	True
Gamma	0.1	0.1
Weight Decay	0.0003	0.0003

Hyperparameters for SimCLR training are shown in Table A. For CIFAR100, we used a pretrained backbone shared on Hugging Face[1].

Table A.2: Hyperparameters for SimCLR

Property	CIFAR10/10-LT	CIFAR100	ISIC2019
Backbone	ResNet18	ResNet34	ResNet18
Epoch	500	*	500
Batch Size	512	*	32
Optimizer	SGD	*	SGD
Learning Rate	0.4	*	0.025
LR Decay Rate	0.1	*	0.1
LR Scheduling	Cosine Annealing	*	Cosine Annealing
LR Dampening	0	*	0
LR T_{max}	500	*	500
Momentum	0.9	*	0.9
Nesterov	False	*	False

Transforms for SimCLR training and active learning follow the ones in TypiClust[2].

[1] https://huggingface.co/edadaltocg/resnet34_simclr_cifar100
[2] https://github.com/avihu111/TypiClust