Semi-supervised Framework for Clustering and Semantic Segmentation

by

Yik Lun Chow
Lakehead University

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTERS

in the Department of Computer Science

Lakehead University

Semi-supervised Framework for Clustering and Semantic Segmentation

by

Yik Lun Chow
Lakehead University

Supervisory Committee

ABSTRACT

During the past couple of decades, machine learning and deep learning methods have achieved remarkable results in many real-world applications. However, it is difficult to develop and train these artificial intelligence algorithms without a labeled dataset. Under this circumstance, it is desirable to leverage a large number of unlabeled data into the training process with fewer or even without labels. To this end, a non-supervised learning strategy (e.g., unsupervised, semi-supervised, weakly-supervised, or self-supervised) has recently been studied in different domains.

In chapter 3, a novel semi-supervised framework is proposed to solve a clustering problem fundamentally by involving only few numbers of labeled data. In this proposed framework, a non-iterative autoencoder is proposed for learning a representation of each data in an unsupervised way. The experimental results theoretically demonstrate the effectiveness of this proposed framework, where the obtained clustering accuracy for thirteen tabular and image datasets are impressive. It has also shown that the proposed autoencoder is able to capture important features of each data.

In chapter 4, the above framework is extended to a weakly-supervised semantic segmentation task for demonstrating its practical ability. Before applying the modified proposed framework to this task, computer vision methods are presented as preliminary work to generate the initial labeled data and clustering space. We achieve the current state-of-the-art performance on PASCAL VOC 2012 dataset.

This thesis shows that the proposed framework is capable not only for the traditional machine learning problem but also for the widely used real-world applications.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my principal supervisor, Dr. Yimin Yang for his invaluable advice and continuous support, and patience during my master's study. His immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. I would also like to thank to my co-supervisor, Dr. Shan Du for her constructive suggestions for my work in Chapter 4. Thanks to Dr. Thiago Oliveira and Dr. Thangarajah Akilan for their treasured comments and feedback. My appreciation also goes out to my family and friends for their encouragement and support all through my studies.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

The machine learning (ML) has revolutionized the world in past few decades, it is the technology of making intelligence machines to solve daily problems as human performs. Traditionally, the core of ML is to train machines by many pairs of labeled data and target class, which is called supervised learning. When the number of labeled data increases, classification performances of ML algorithms are naturally improved. The ML models can be well-trained to learn pattern of each data and predict its corresponding class. Furthermore, classification capacities of models also depend on the representation of data. To this end, the representation learning (or called feature learning) is proposed to study how to extract useful features from data. The neural network is one of the widely used feature extractors, it aims at encoding raw information of each data into multiple neurons. With constructing the multi-layers network, a high-dimensional input space can be projected to a low-dimensional feature space. However, an enormous number of labeled data is required to train and fine-tune the weights connecting each neuron in such complicated network.

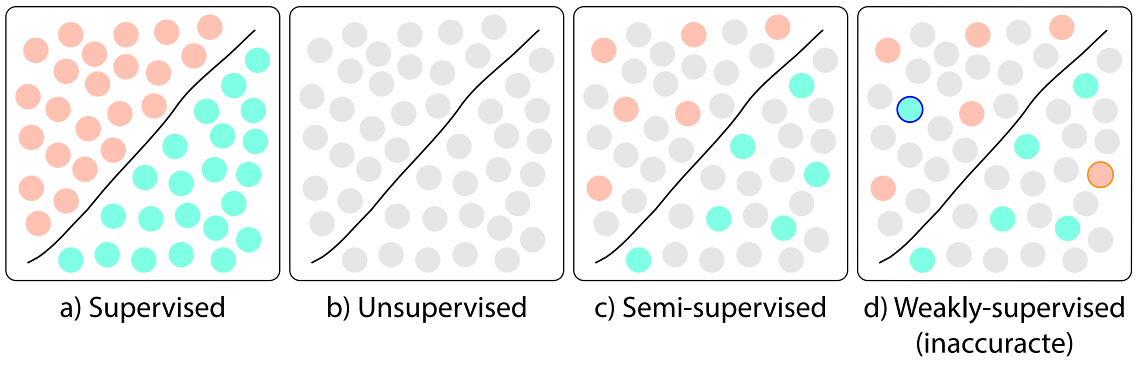


Figure 1.1: Different forms of supervision on classification

Nevertheless, it is not possible to create a dataset consisting of more than millions of labeled data because of the high annotation cost. For example, it takes 10.1 minutes to annotate each image pixel-by-pixel for instance segmentation [6]. Therefore, a common strategy to resolve the problem of lack of labeled data is to utilize unlabeled data in unsupervised, semi-supervised or weakly-supervised manners (Fig. 1.1). In general, unsupervised learning (UL) methods allow algorithms to learn internal representation or hiddens pattern among data without providing any label information. Similar to the classification tasks, the target of the clustering problem is to gather "similar" objects into one set and distribute "dissimlar" objects into different sets. As the labels are completely inaccessible, the clustering problem has been considered as the hallmark problem in ML.

However, in many real-world applications, a small number of labeled data is often attainable which can be regarded as prior knowledge to clustering tasks. To take advantage of this situation, semi-supervised learning (SSL) methods have been proposed and become a research hotspot since the early 2000s [7, 8]. The SSL falls between supervised and unsupervised learning, in which a small number of labeled data and a large number of unlabeled data can be utilized in the training process. By extending a clustering method, the semi-supervised algorithm can be easily derived. As some labeled samples are given, the semi-supervised algorithm is also known as the constrained clustering in which prior knowledge can be converted as regularization or constrained terms of an objective function [9]. Last but not least, the weakly-supervised learning (WSL) methods can be generally regarded as the special form of SSL methods, which contains three types of weak supervision [10] in the following:

1. Incomplete supervision: It gives only a small subset of labeled data, while the

rest is unlabeled, i.e., semi-supervised learning.

2. Inexact supervision: It provides a full set of labeled data, but not as exact as desired. It can be seen as the indirect supervised information. For example, the main task of semantic segmentation is to annotate every pixel of an image, i.e., pixel-level classification. Image-level annotation is the example of weak supervision, which indicates the presence of each object, but does not provide any precise spatial information about the image.

3. Inaccurate supervision: As its name implies, some data are wrongly labeled.

The weakly-supervised learning approach reduces significant human efforts for annotating samples. Therefore, it has attracted much research attention on several tasks of computer vision area, such as image classification, semantic segmentation, object detection and so on.

On the other hand, it is hard to train the complicated neural network containing multiple hidden layers when a number of labeled data is limited in reality. For this reason, the unsupervised autoencoder [11] has been proposed to learn a useful representation from a raw input space. A convention autoencoder consists of two symmetric components, i.e., an encoder and a decoder. The encoder is used to compress the data from higher dimensional input space to lower dimensional feature space, while the decoder is to map that feature space to output space which has the same dimension as input. By minimizing a discrepancy between input and output representation of the data, AE can learn its latent representation by using few hidden neurons.

1.2 Motivation

The performances of supervised Deep Convolutional Neural Network (DCNN) exceed human performances on image classification, for example, a human-level top-5 classification error rate on ImageNet dataset [12] is 5.1% [13], while the corresponding result obtained by the state-of-the-art (SOTA) of DCNN is 1.2% [14]. However, the performances of DCNN strongly rely on an extremely large number of labeled data. For instance, 1.28 and 300 million images from ImageNet and JFT-300M datasets are required respectively to train the EfficientNet-L2 network for obtaining the aforementioned performance [14]. In other words, Artificial Intelligence (AI) could only

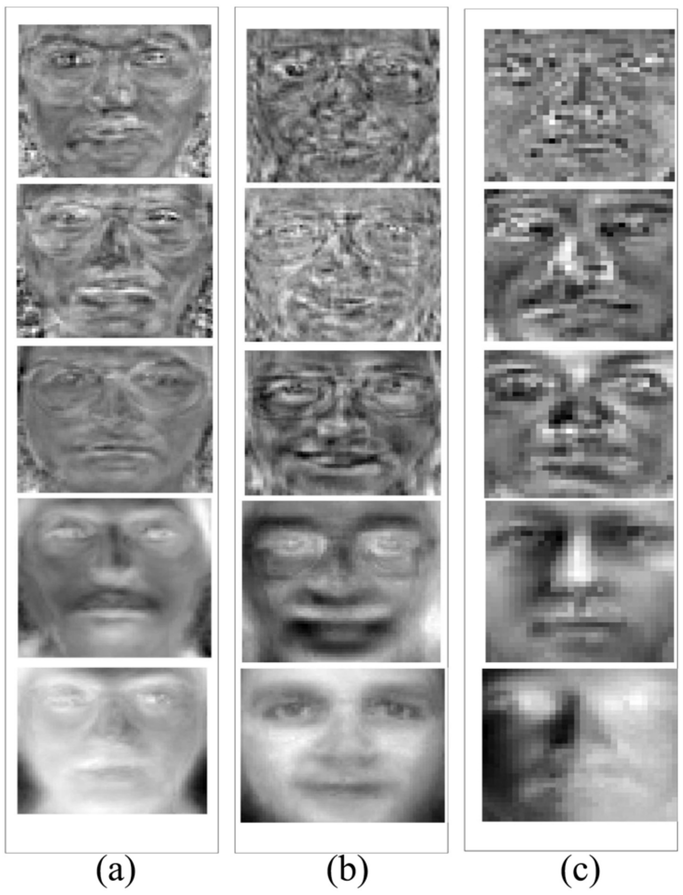


Figure 1.2: Reconstructed faces of AE with different number of hidden neurons on (a) Yale-Face, (b) Olivetti-Face and (c) Yale-Face-B datasets. From top to bottom, the used hidden neurons for each dataset are: (a): {1000, 400, 100, 10, 1}; (b): {1000, 400, 100, 10, 1} and (c): {500, 200, 50, 10, 1} (Extracted from [2])

surpass human intelligence when an extremely large number of labeled data is provided. Moreover, even a well-trained AI may fail to perform a simple task that humans can complete readily. In 2017, the 3D printed masks is made by researchers from a Vietnamese security company to directly unlock an Iphone which was encrypted by Apple's face ID. In 2018, the study [15] showed that synthesized adversarial images could consistently fool the classifier which was well-trained to classify tuples with almost 100% accuracy.

In contrast, humans can be self-trained to solve any problems easily. The newborns could even recognize their mothers immediately after birth and could visually identify their parent's faces after a few weeks. In fact, the human brains recognize faces (and even objects) by just comparing a similarity to exemplars of previously perceived faces [16]. In Ref. [2], the very insightful phenomenon has been demonstrated that the most "strong" and "common" feature of each human face can be reconstructed when a fewer number of hidden neurons is used in the hidden layer of the autoencoder. (Fig. 1.2). Inspired by this observation, the motivation naturally comes: **Could we design the autoencoder (AE) to perform clustering without involving any label information?** To be specific, if we could extract "strong" features from one sample (e.g, x_i) using this AE, we could use these features to well-reconstruct homogeneous samples, i.e., samples also belong to the same class with x_i.

Therefore, the primary focus of this thesis is to develop the unsupervised AE with the semi-supervised mechanism to resolve the semi-supervised clustering task. As mentioned previously, it is difficult to train the multi-layer network when the number of labeled data is limited. Therefore, we would develop the unsupervised AE with a extremely simple architecture as a feature extractor to learn "strong" and "common" feature. Next, we would develop the semi-supervised mechanism to incorporate this AE for guiding the clustering process, where only a few numbers of labeled data is involved. After that, the secondary focus is to apply the proposed algorithms to handle the real-word problem. We select the weakly-supervised semantic segmentation problem because it is the one of the most important task in computer vision.

1.3 Organization of Thesis

In order to alleviate the problem of lack of labeled data, the main goal of this thesis is to propose non-fully-supervised learning algorithms for clustering and semantic segmentation. In this section, an overall organization including several core problems of chapters 3 and 4 of will be presented as follows (See Fig. 1.3 also):

- Chapter 2 gives reviews of the fundamental concepts and theories to three aspects: i) unsupervised representation learning methods, such as Deep Belief Network (DBN) [11], Extreme Learning Machine Autoencoder (ELM-AE) [17] and ELM-AE with invertible functions (ELM-AE-IF) [2]; ii) semi-supervised clustering algorithm, i.e., k-means [18] and its variants [19, 20]; iii) weakly-supervised semantic segmentation methods, such as SEC [3], DSRG [4] and CIAN [5].

- Chapter 3 describes and gives detailed procedures of the proposed unsupervised AE and semi-supervised clustering mechanism. To verify performances of the

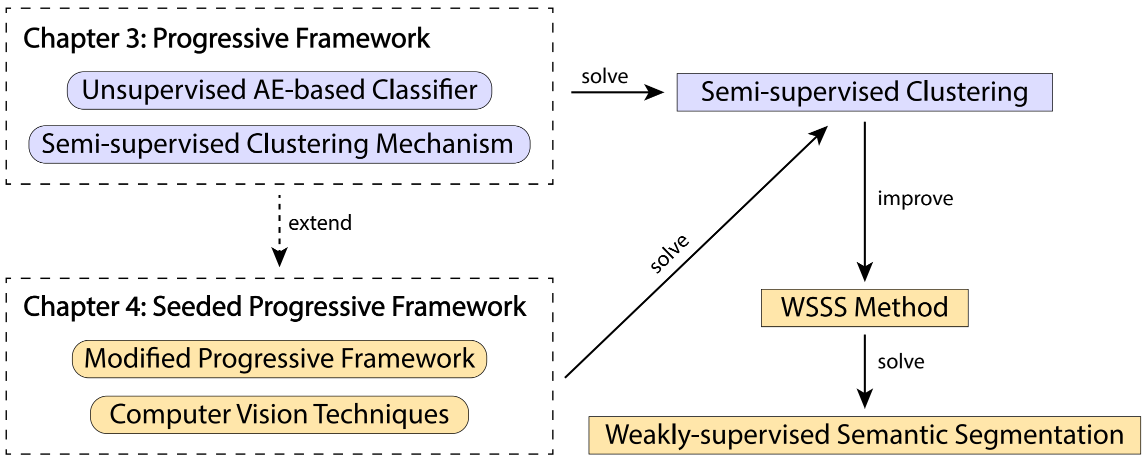

Chapter 3: Progressive Framework
Unsupervised AE-based Classifier
Semi-supervised Clustering Mechanism
solve
Semi-supervised Clustering
extend
improve
solve
Chapter 4: Seeded Progressive Framework
Modified Progressive Framework
Computer Vision Techniques
WSSS Method
solve
Weakly-supervised Semantic Segmentation

Figure 1.3: Overview of Chapter 3 and 4

proposed algorithms theoretically, extensive comparison experiments with more than 35 methods on 13 benchmark datasets are conducted. In this section, we focus on the following questions:

- Can we design a simple AE to extract "strong" but not "generalize" features of one given labeled sample x_i ?
- How can we use these extracted features to perform clustering. In other word, if two samples from the same class have the "common" feature, we can group them together because they are "similar".
- From above, what is an evaluation metric to define the term "similar"?
- Can we propose a mechanism to cluster an entire dataset progressively, rather than only two samples?
- How to evaluate the proposed clustering algorithms? And how to compare with other methods under a same condition?

- Chapter 4 applies the above methods to weakly-supervised semantic segmentation (WSSS). The original problem can be formulated as semi-supervised clustering (SSC) task, our approach can be considered as a knowledge augmentation method for boosting any WSSS methods. We show experimentally that the proposed method can improve the segmentation performance of three well-known methods. Similarly, we ask the following questions:

  - How to formulate the WSSS problem as SSC task? The main goal of WSSS is to assign labels to each pixel within an image by given only the weak supervision (in our case, image-level labels are given), which is different from SSC task.

  - From above, as weak supervised labels are the only given information, how to convert this information to labeled data for our proposed SSC method?
  - Our method can bring more labeled data, how does these labeled data improve the performance of WSSS methods? In other words, how to integrate our method with those methods?
  - As the knowledge augmentation method for boosting **any** WSSS methods, is there any conditions to select the target methods?
  - How to evaluate the performance improvements? What is the evaluation metric used for segmentation?

- Chapter 5 is our last chapter to conclude the entire work and to further provide some insights to continue this research.

Chapter 2

Literature Review

2.1 Unsupervised Representation Learning

Unsupervised representation learning (also known as feature learning) has been researched for many decades since it can capture underlying useful information of data without using any human-annotated labels. The **autoencoder (AE)** is one of the widely used representation learning models. It was traditionally developed for dimensionality reduction and representation learning. With the leading success of deep learning, many variants of AE have been proposed with abundant applications in computer vision, data mining, and natural language processing (See Table 2.1). In the rest of this section, the concepts and equations of three selected methods that are highly related to this thesis will be explained.

2.1.1 Autoencoder

Perhaps Ballard [21] was the first one to introduce the concept of AE in the late 80s. The AE, however, has become the mainstream representation learning method since Hinton and Salakhutdinov [11] proposed Deep Belief Network (DBN). It can be viewed as the multilayer AE which was constructed by a stack of Restricted Boltzmann Machines (RBM). Ref. [11] also proposed the new learning strategy to train the multilayer network layer-by-layer by solving two general situations about initial weights as follows,

- If the weights are too large, AE gets stuck into the local minima easily.
- If the weights are too small, the gradients in the early layers are getting smaller. This is also known as the vanishing gradient problem, which makes it infeasible to train multilayer AE.

Therefore, five steps were developed to train DBN:

step 1: The input layer x is regarded as the visible layer, while the hidden layer h_1 is treated as the invisible layer. These two layers are used to build the first RBM. This RBM is trained to obtain the weights and biases.

step 2: Then, the obtained weights and bias are fixed to train the second RBM constructed by the hidden layers h_1 and h_2.

step 3: Repeat the same process until all hidden layers (e.g., 3 hidden layers in Fig. 2.1) are pretrained. the DBN is constructed by stacking multiple RBMs.

step 4: Then, the DBN is unrolled to create the deep AE with multiple layers (Fig. 2.2).

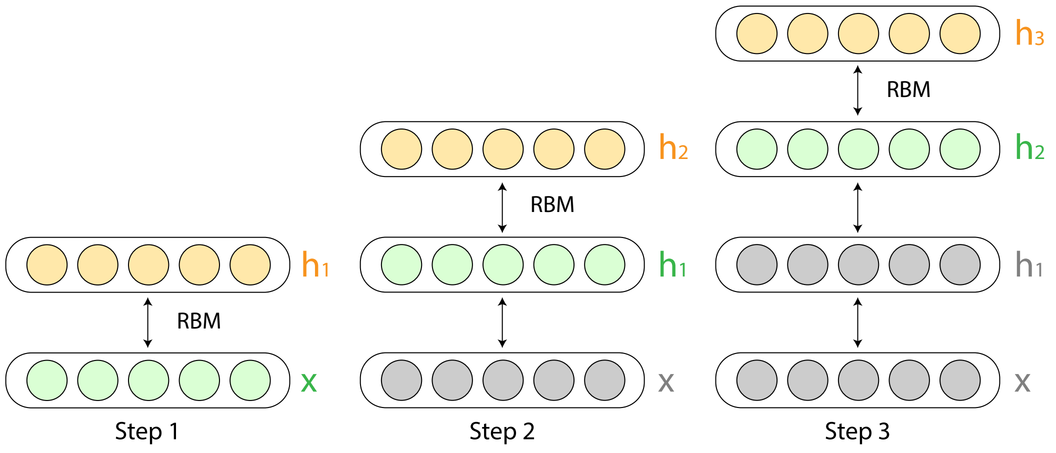


Figure 2.1: Learning layer-by-layer of a stack of RBMs for DBN (Step 1 to 3)

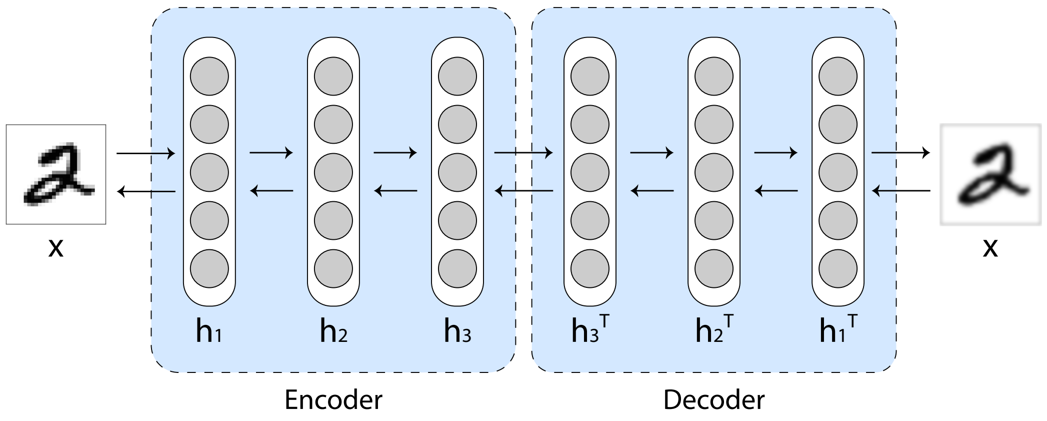


Figure 2.2: Finetuning the unrolled RBN-AE with input and output (Step 4 to 5)

step 5: For data reconstruction, the input is used as output. The weights of deep AE are finetuned through BP by given the corresponding output.

The above steps provided the learning strategy to initialize weights before BP, which makes the weights close to a good solution. These weights can then be finetuned. By extending this pioneering work, many variants of autoencoder have been proposed (Table 2.1). However, the BP-based learning algorithm is generally very slow since it requires pretraining and finetuning stages while many iterations are involved to obtain better performance. Unlike the BP approach, a Single-Layer-Feedforward Networks (SLFN) based learning algorithm can be used to build a fast-speed AE. It is also known as a Extreme Learning Machine Autoencoder (ELM-AE).

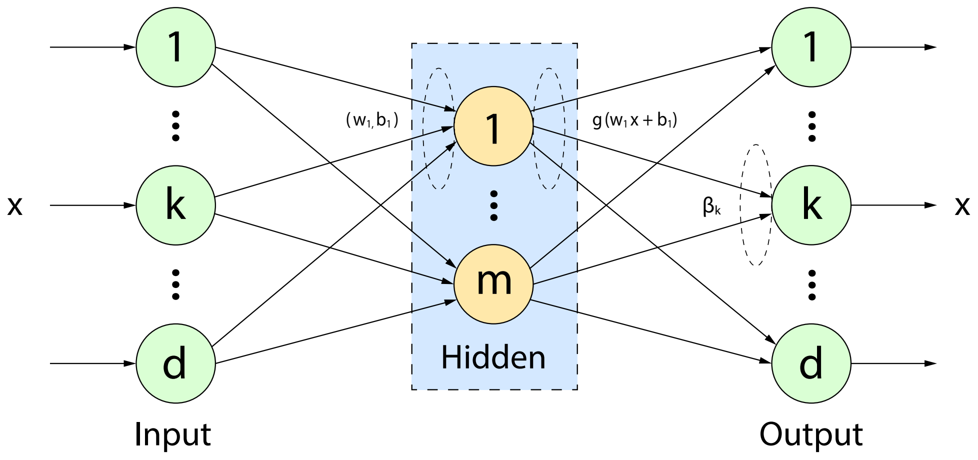


Figure 2.3: A Structure of ELM-AE

2.1.2 Non-iterative Autoencoder

In 2013, Kasun et al. [17] proposed the single hidden layer AE based on Extreme Learning Machine (ELM) [22], called ELM-AE (sometime called SLFN-AE). It can be seen as the special form of ELM, where the input is used as output, and randomly generated weights are chosen to be orthogonal. Due to the characteristic of fast learning, the ELM-AE learns significantly faster than DBN. The basic structure of ELM-AE is illustrated in Fig. 2.3. First of all, the orthogonal random weights w and biases b are initialized, then calculate the output h of the encoder:

$$w^T w = \mathrm{I},\ b^T b = 1 \tag{2.1}$$

$$\mathrm{h} = g(w \cdot \mathrm{x} + b) \tag{2.2}$$

where (w, b) is connecting between input and hidden space, g is an activation function, such as sin, sigmoid, tanh. Secondly, the output weight matrix β is calculated which is responsible for transforming a feature space to an output space. It can be calculated as follows:

$$\beta = (\frac{\mathrm{I}}{\mathrm{C}} + \mathrm{H}^T\mathrm{H})^{-1}\ \mathrm{H}^T\mathrm{X} \tag{2.3}$$

where $C \in [2^{-10}, 2^{10}]$ is a regularization term. Given N total training samples, $\mathrm{H} = [\mathrm{h}^{\mathrm{T}}(\mathrm{x}_1), ...\mathrm{h}^{\mathrm{T}}(\mathrm{x}_{\mathrm{N}})]$ is the representation of hidden space. Thirdly, the actual output matrix $\hat{\mathrm{X}}$ can be calculated by using Eq.2.4

$$\hat{\mathrm{X}} = \mathrm{H}\beta \tag{2.4}$$

As the ELM-AE is used to reconstruct an original data, the formal learning objective is to minimize the reconstruction loss E(X):

$$\min \mathrm{E}(\mathrm{X}) = \sum_{i=1}^{N} ||x_i - \hat{x_i}||^2 \tag{2.5}$$

Although the experiments showed that the ELM-AE learned useful features and achieved generalization performance than other AEs for classification, the major weakness of the ELM-AE is that all input weights are randomly generated. Hence, the generalization performance of ELM-AE could deteriorated.

Therefore, the **ELM-based autoencoder with invertible function (ELM-AE-IF)** [2] is developed. In view of the structure of AE, the ELM-AE-IF is similar to ELM-AE, which consists of encoder and decoder layers. However, the ELM-AE-IF is proposed to use to subnetwork nodes [23] rather than regular hidden nodes for constructing layers. The structure of ELM-AE-IF is depicted in Fig. 2.4. To resolve the mentioned problem of random generation of input weights, the decoder weights are calculated in ELM-AE-IF. They are used to update the encoder weights for the next iteration. Besides, the output weights β are removed for making this learning system naturally symmetric. Generally speaking, four steps are involved (See Fig. 2.4):

step 1: Initialize the input weights w_f and biases b_f by using Eq. 2.1

step 2: Given an output X and inverse of activation function g^{-1} (e.g., sigmoid or sin), the decoder weights w_n and bias b_n are calculated as follows:

$$w_n = g^{-1}(\mathrm{X}) \cdot \mathrm{H}^{\dagger} \tag{2.6}$$

$$b_n = \sqrt{mse\ (w_n \cdot \mathrm{H} - g^{-1}(\mathrm{X}))} \tag{2.7}$$

$$g^{-1}(\cdot) = \begin{cases} \arcsin(\cdot) & \text{if } g(\cdot) = sin(\cdot) \\ -\log(\frac{1}{(\cdot)} - 1) & \text{if } g(\cdot) = \frac{1}{1+e^{-1(\cdot)}} \end{cases} \tag{2.8}$$

where H is the representation of hidden space, the same as previous section. $H^{\dagger}$ is a Moore-Penrose inverse matrix. More details can be found in [2].

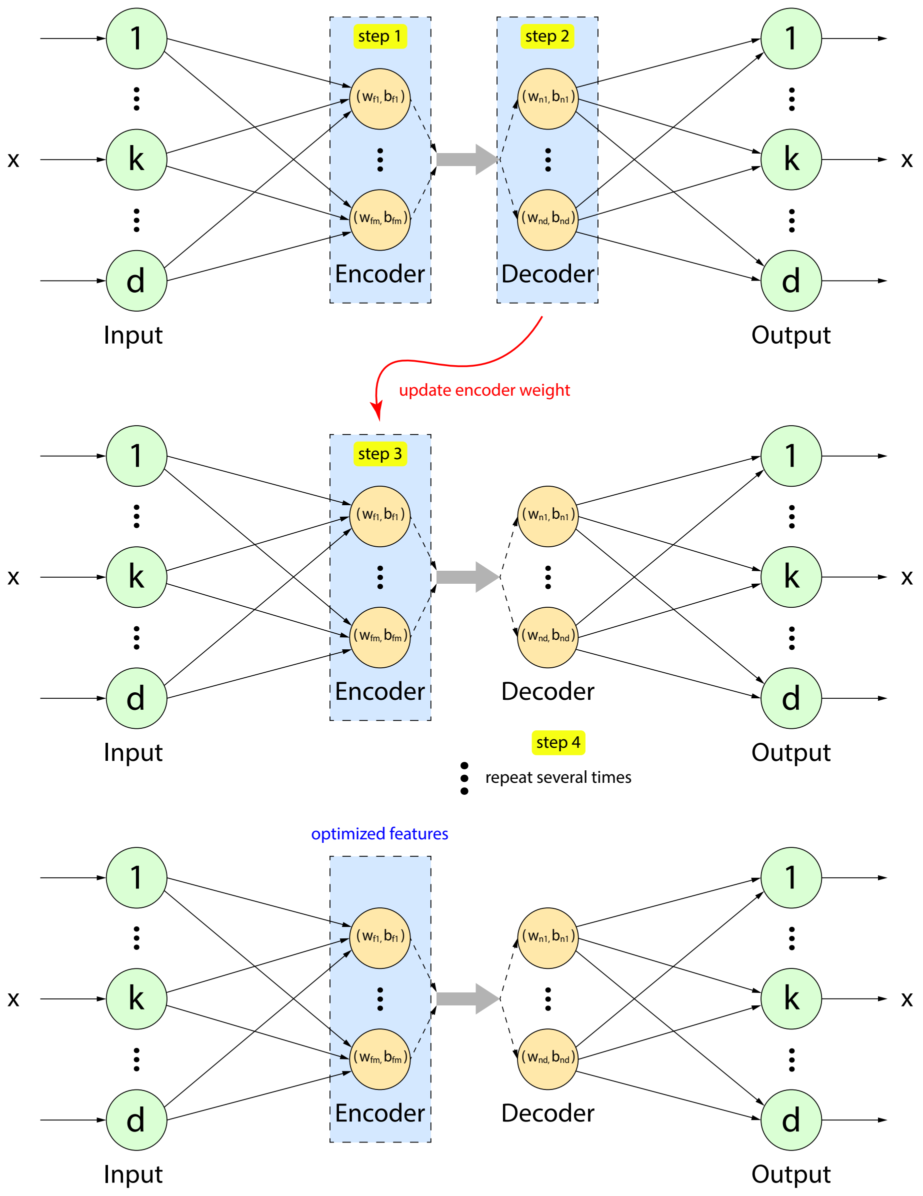


Figure 2.4: Four steps learning algorithm for ELM-AE-IF. Note that subnetwork nodes are used in encoder and decoder layers.

step 3: Update the encoder weights w_f and bias w_f by using Eq. 2.9, then recalculate the feature data H by using Eq. 2.2

$$w_f = (w_n)^T, \; b_f = b_n \tag{2.9}$$

step 4: Repeat steps 2 and 3 several times.

By updating the current encoder weight with a preceding decoder weight, ELM-AE-IF can obtain low-dimensional but useful features with much faster training speeds than ELM-AE. Similar to the relationship between ELM-AE and ML-ELM in Ref. [17], stacking a MLELM-SNs [23] on top of ELM-AE-IF can be used for classification task.

2.1.3 Summary

In this thesis, an unsupervised non-iterative autoencoder is proposed which is inspired by ELM-AE-IF. Although the structure of the proposed AE is the same as ELM-AE-IF,it focuses on another learning object such that the extracted feature can be applied on clustering task. The details of our AE can be found in Chapter 3.

2.2 Semi-supervised Clustering

Clustering is the task to partition subsets such that similar samples are gathered in the same group while dissimilar samples are arranged in different groups. Semi-supervised clustering (SSC) is the special form of clustering that can make use of a small number of labeled data. As mentioned, SSC algorithms can be easily be derived by extending the concepts of clustering, such as CDBSCAN [24] (extended from DBSCAN [25]), semi-supervised hierarchical clustering [26–28] (extended from hierarchical clustering) and LCVQE [29] (extended from CVQE [30]). Of these methods, k-means is the most well-known clustering algorithm. Hence, in this section, the concepts and related equations of k-means and its three variant of semi-supervised algorithms will be introduced. On the other hand, self-supervised learning methods can be incorporated into the semi-supervised methods to improve the performance. The concept of self-supervised learning methods will be explained in the later part of this section.

2.2.1 k-means Clustering

In 1967, MacQueen et al. [18] first defined the term “k-means”, although the standard k-means algorithm was proposed by Lloyd in 1957 [31]. Since then, a variety of k-means related surveys [32–34] has been published. Due to its simplicity, k-means is still one of the most popular unsupervised clustering methods, whose goal is to partition a dataset into M clusters based on the Euclidean distance between data:

$$d(x_i, x_m) = \sum_{j=1}^{d} (x_{ij} - x_{mj})^2 \tag{2.10}$$

where x_i, x_m are d-dimensional data. x_{ij} is the value of j-th feature of data x_i. Given a datset X = $\{x_1, ..., x_N\}, x_n \in R^{\,d}$, the M clustering problem aims at minimizing the within-cluster sum of squares (WCSS) between each data x_i and the centroid $\bar{x}_m$ of cluster C_i. The objective function can be written as:

$$\sum_{m=1}^{\mathrm{M}} n_m \sum_{C_i=m} \sum_{j=1}^{d} (x_{ij} - \bar{x}_{mj})^2 \tag{2.11}$$

where n_m is the number of data in m-th cluster. C_i represents the cluster to which data x_i is assigned. The conventional k-means clustering can be generally summarized in the following four steps

step 1: Randomly pick M data as initial centroids of clusters $\bar{x}_m$.

step 2: **Assignment:** Assign each data $x_i \in R^{\,d}$ to a cluster C_i by the following:

$$C_i = \arg\min_{m} \sum_{j=1}^{d} (x_{ij} - \bar{x}_{mj})^2 \tag{2.12}$$

step 3: **Update:** Update each centroid by calculating the mean values within clusters, for which the corresponding mean of j-th feature is computed as:

$$\bar{x}_{mj} = \frac{1}{n_m} \sum_{i \in C_i} (x_{ij}) \tag{2.13}$$

step 4: Repeat steps 2 and 3 until the converges, i.e. either the algorithm process t iterations, or the assignments no longer change.

Although the convergence of k-means is guaranteed, a large number of iterations is needed to repeat the above two steps before the convergence. As the small number of labeled data is often obtainable in real application domains, the variants of semi-supervised k-means are proposed which utilizes some background knowledge to improve the clustering performance. Here, three representative variants are showed.

2.2.2 Constrained k-means Clustering

In constrained k-means clustering [19], there are two modifications based on k-means where a small number of labeled data can be used as "seeds" to improve the clustering processing. First of all, rather than initializing k-means by randomly picking M data as centroids, the constrained k-means clustering calculates the means of M partitions $\mathrm{S_M}$ of the seed set. Secondly, the labeled data are always assigned to their pre-assigned clusters, even if they are closer to another cluster. In other words, the cluster labels of the seed data remain unchanged during the assignment step.

2.2.3 Seeded k-means Clustering

Apart from above algorithm, Basu et al. proposed another variant of k-means in the same paper [19]. The main idea is inspired by the real-life situation that some initially labeled data may be mislabeled by humans accidentally. To handle such presence of noisy supervision, the seeded k-means is recommended where the assignments of labeled data still follow Eq. 2.12. Unlike constrained k-means, the labeled data will be assigned to the nearest cluster as normal during the assignment step. As mislabeled seeds may be corrected by the algorithm, seeded k-means is quite robust against noisy seeding.

2.2.4 COP-k-means Clustering

Unlike using a small number of labeled data, the pairwise constraints can be used. As another form of prior knowledge, it indicates complex relationships among data. Two types of constraints are involved as follows:

- Must-link: It is the constraint showing that two data have to be in the same cluster, i.e., $M = \{< x_i, x_j >: x_i \in C_l, x_j \in C_h, l = h\}$
- Cannot-link: This constraint represents that two data should not lie in the same cluster, i.e., $C = \{< x_i, x_j >: x_i \in C_l, x_j \in C_h, l \neq h\}$

By using these prior knowledge to guide a k-means clustering, Wagstaff et al. [20] proposed the COP-k-means. In COP-k-means, each data x_i is assigned to the closest cluster C_i according to Eq. 2.12 such that no constraints are violated. Given must-link M, cannot-link constraints C, the violate constraints can be defined as follow:

Let consider two data points (with its cloest cluster centroid), i.e., $x_i \in C_i$ *and* $x_k \in C_k$*.* ***IF*** $< x_i, x_k > \in M$*, but* $i \neq k$*,* ***OR*** $< x_i, x_k > \in C$*, but* $i = k$*.* ***THEN****,* x_i *cannot be placed in* C_i*,* x_k *cannot either.*

2.2.5 Self-supervised Learning based Clustering

Recently many researchers in the field of semi-supervised learning propose to combine their methods with self-supervised learning [35, 36]. The concept of self-supervised learning is to train a deep learning model by pre-designed pretext tasks. The labels for pretext tasks is known as self-supervised labels (or pseudo label) which are automatically generated based on the structure of data. The model trained for these pretext tasks learns higher-level visual features [37]. The learned parameters serve as a pre-trained model and can be fine-tuned by other downstream tasks, such as semantic segmentation, object detection and human action recognition. Note that human-annotated labels are needed for downstream tasks.

Instead of solving other downstream tasks, learned parameters can be directly fine-tuned by a small amount of labeled samples. This technique of combining semi-supervised methods with self-supervised methods is called S4L technique [35]. Fig. 2.5 shows the example of the learning process of S4L. As the pretext task is to predict the rotation angle of an image, the first step of S4L is the generation of self-supervised labels (i.e., 0°, 90°, 180° and 270°). Then, the deep learning model is trained by this pretext text. Finally, the pre-trained model can be fine-tuned by a small number of labeled samples. In other words, the downstream is the regular image classification task. Inspired by the S4L technique, many semi-supervised models [36, 38, 39] have been proposed to extend their work into self-supervised methods which achieve impressive performance on ImgaeNet dataset.

2.2.6 Summary

There is a broad range of semi-supervised variants of k-means, where the prior knowledge is involved. In this thesis, a small number of labeled data is involved to guide our novel SSC mechanism. To be specific, the non-iterative AE is proposed (Section 2.1.3) which can be viewed as an unsupervised classifier to group one unlabeled data to one labeled data. As the aims of clustering is to partition an entire dataset into subsets, but not only for two data. The complete SSC mechanism is required to guide the entire process. From the prospective of SSC, the proposed algorithm is similar

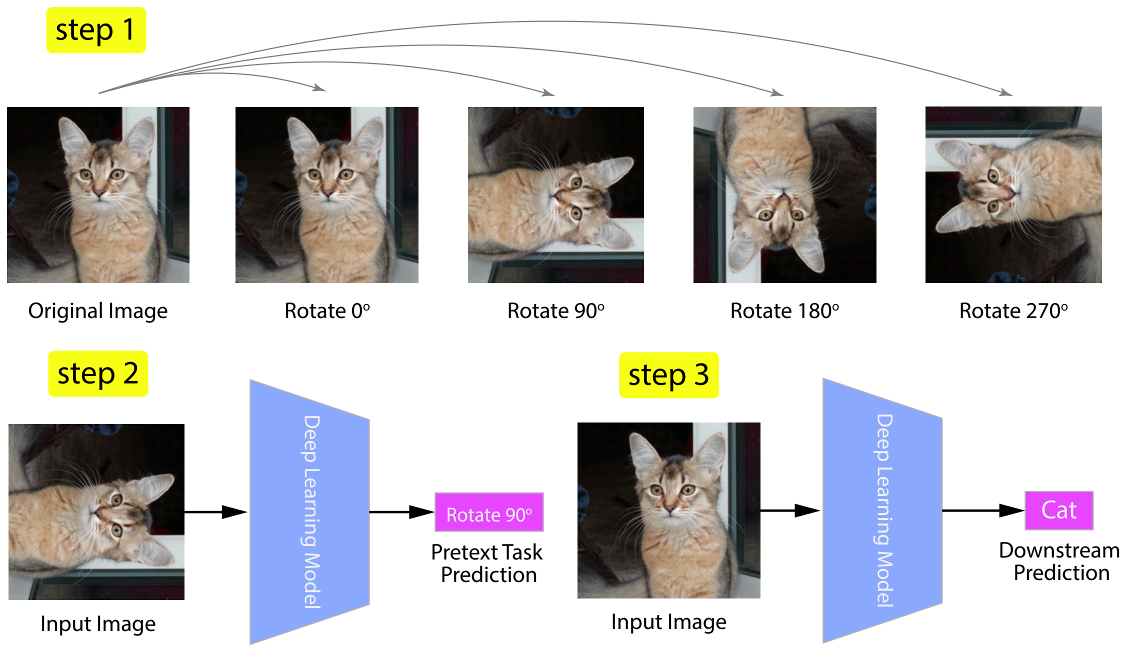


Figure 2.5: Example of the learning process of S4L where the pretext task is to predict the rotation angle and the downstream is classification.

to k-mean clustering. The details of comparison between our approach and k-means can be found in Section 3.5.

2.3 Weakly-supervised Semantic Segmentation

Image semantic segmentation is one of the essential topics in computer vision. It is also known as a pixel-level classification [40–42] because it aims to classify every pixel within an image into a discrete semantic class. It plays a important role in a broad rang of real-word applications, such as urban scene understanding [43, 44], medical image processing [45, 46], satellite image analysis [47, 48] and so on. Due to a promising success of a deep learning model, many research works have been proposed to utilize different fully supervised deep networks on image segmentation, such as fully convolutional networks (FCN [49]) and dilated convolutional models (or known as DeepLab family [50–53]). These works are considered as a milestone in fully-supervised image semantic segmentation, and have achieved significant breakthroughs for industries. However, a pixel-by-pixel annotation of each image for fully-supervised learning is time consuming and burdensome. To reduce the annotation cost, weakly-

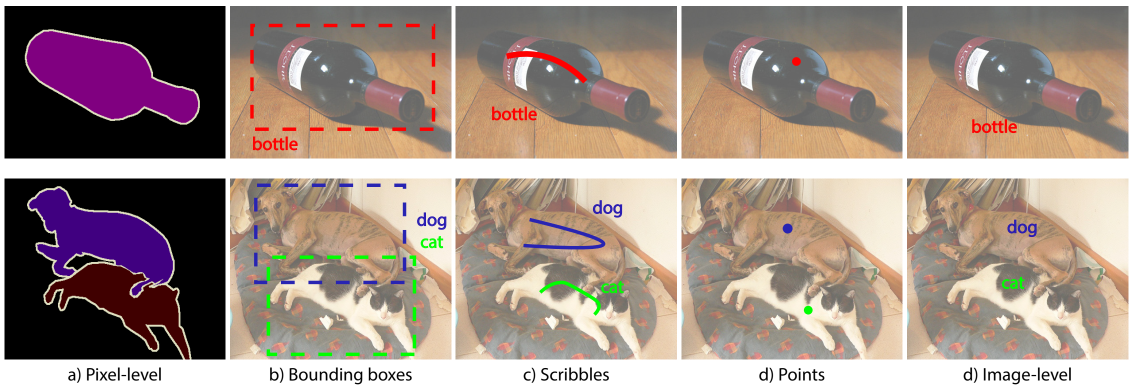


Figure 2.6: Different forms of weak supervision on image semantic segmentation

supervised approach has been proposed to use inexact supervision where only coarse-grained label information is available. There are different forms of weak supervision, such as bounding boxes [54,55], scribbles [56–58], and points [59] (Fig. 2.6). Of these forms, image-level labels are the most cheaper and quicker to obtain which indicting the presences of objects, but without providing any positional information.

To initially attain little spatial information from image-level labels, the computer vision technique is used, i.e. Class Activation Map (CAM) method [60]. The CAM image of each class is thresholded as a collection of weak localization seeds (or called cues). Although the seeds are sparse and incomplete that are far away a intact segmentation mask, the seeds highlights most discriminative regions of target object. To recover an intact segmentation mask of object, the seeds are implicitly and explicitly used as an incomplete pixel-level supervision. Table 2.2 summarizes representative methods chronologically with brief introduction and their segmentation performances in PASCAL VOC 2012, which is a benchmark in this field. Among all these approaches, there are three insightful works that explicitly uses seeds as supervision in their proposed loss function. In the rest of this section, brief introductions and related loss functions for these selected methods will be given.

2.3.1 Seed, Expand and Constrain

In the earlier year, Kolesnikov et al. [3] proposed the pioneering approach by introducing a seeding loss to encourage a segmentation network to match localization cues, while a global weighted rank pooling and a fully-connected conditional random fields (CRF) are used for expansion and constrain loss respectively. This is method is known as the SEC method. In practice, SEC is to solve the following optimization problem

$$\min_{\theta} \sum_{(X,T)\,\in\mathcal{D}} [\, L_{seed}(f(X;\theta),T) + L_{expand}(f(X;\theta),T) + L_{constrain}(f(X;\theta)) \,] \tag{2.14}$$

where X and T are an input image and a weakly-supervision (i.e., image-level labels) for training a deep network $f(X;\theta)$ using the above three terms. As seeds are generated by CAM method, a set of locations that are labeled with class c can be denoted as S_c. Thus, the seeding loss can be formulated as

$$L_{seed}(f(X;\theta),T,S_c) = -\frac{1}{\sum\limits_{c\in T}|S_c|} \sum_{c\in T} \sum_{u\in S_c} \log f_{u,c}(X) \tag{2.15}$$

As an explanation of these loss function is outside the scope of this thesis, we refer readers to the original publication [3] for more details. In a nutshell, producers of SEC can be concluded in two steps: (i) given image-level labels, seeds are generated by using CAM method; (ii) the backbone deep learning model (i.e., DeepLabv1 [50]) is then trained by the above overall loss function. (Eq.(2.14)). Although SEC obtained SOTA performances at that time, the supervision for backbone model is fixed which is not capable for providing sufficient supervisions to the network continuously. To this issues, DSRG is developed as one of the solutions to provide dynamic supervision by applying a traditional image segmentation algorithm to network.

2.3.2 Deep Seeded Region Growing

Similar to SEC, the CAM method is also employed to locate and generate the seed cues by giving only image-level labels in Deep Seeded Region Growing model (DSRG) [4]. Inspired by traditional image segmentation algorithm of seeded region growing (SRG) [61], DSRG trains a segmentation network with “dynamic supervision” in which the number of seeds is increasing progressively while training. Rather than using three different types of loss function, DSRG proposed the balanced seeding loss to consider foreground and background seeds separately as follows

$$L_{seed}(f(X;\theta),T,S_c) = -\frac{1}{\sum\limits_{c\in T}|S_c|} \sum_{c\in T} \sum_{u\in S_c} \log f_{u,c}(X) - \frac{1}{\sum\limits_{c\in \hat{T}}|S_c|} \sum_{c\in \hat{T}} \sum_{u\in S_c} \log f_{u,c}(X) \tag{2.16}$$

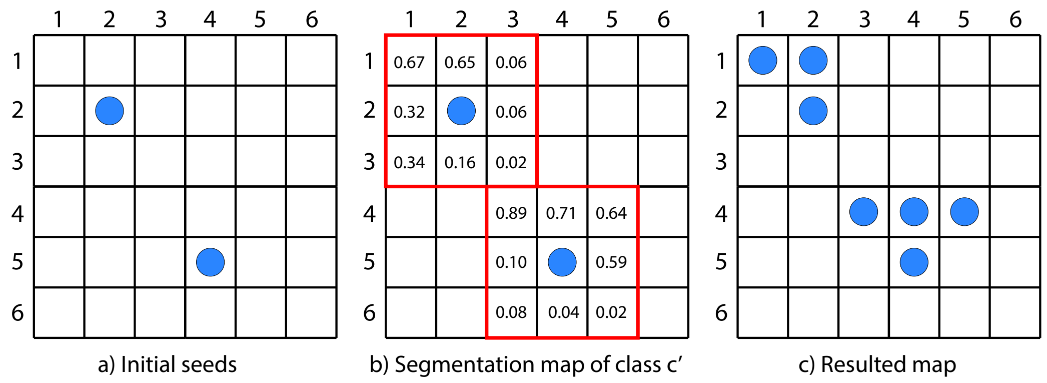


Figure 2.7: An example of performing region growing of two initial seeds by setting $\theta_{c'} = 0.60$. Note that there should be more than one probability at each position for multi-class classification, but only c' is shown for brevity .

where $f_{u,c}$ denotes the probability of class c at position u on segmentation map f of an image X. T and $\hat{T}$ are the set of classes of foreground and background respectively. Together with constrain loss proposed in [3], the overall loss function of DSRG is written as

$$L = L_{seed} + L_{constrain} \tag{2.17}$$

Apart from loss function, there are two improvements of DSRG, (i) seed generation: CAM method is used for generating foreground seeds while saliency detection technology [62] is for background; (ii) DeepLabv2 [51] is selected as the backbone model.

The mechanism of deep seeded region growing

As mentioned, SRG is a classical algorithm for image segmentation, it is known as a region-based method to examine neighboring pixels of initial seed points and determine whether nearby pixels should be added to the region. As seeds are initially generated by CAM method, SRG can be integrated into deep segmentation network and can generate more seeds in an end-to-end setting. Let consider a 8-connectivity neighborhood of the seed x_i on resulted segmentation map f, one decision rule (Fig. 2.7) is used to determine which connected non-seeds should be grouped with x_i:

> ***IF*** *the probability value* $f_{u,c'}$ *of class* c' *at position* u *is larger than a threshold value* $\theta_{c'}$ *(i.e.,* $f_{u,c'} > \theta_{c'}$ *) where class* c' *is an argmax resulted value at position* u *(i.e.,* $c' = \arg\max_c f_{u,c}$*),* ***THEN****, a label is assigned to*

that non-seed, the same as the central seed. ***OTHERWISE****, the status of the non-seeds is remained unchanged.*

2.3.3 Cross-Image Affinity Net

In contradiction to train the network by treating images independently, Fan et al. [5] first proposed to explicitly model a cross-image relationship for weakly-supervised segmentation such that supplementary information for identifying the pixels can be gained. In other words, features can be refined and amplified because the knowledge of seeds are shared across the whole dataset. To model the cross-image semantic, a cross-image affinity net (CIAN) module was developed to prompt the learning of pixel-wise relationship between two images. The idea is to select one query image q and its N reference images $\{r^{(h)}|h = 1, ...N\}$ where q and $r^{(h)}$ should have at least one common semantic class. Let x^q and x^r be the resulted representation from a CIAN model respectively, the overall loss is computed as:

$$L = L_{ce}(x^q) + L_{ce}(x^r) + L_{cp}(x^q) + L_{cp}(x^r) \tag{2.18}$$

where L_{ce} and L_{cp} are a cross-entropy loss and completion loss respectively. As a discussion about completion loss is beyond the scope of this thesis, only the cross-entropy loss is considered:

$$L_{ce}(x) = -\frac{1}{|S'_q|} \sum_{i \in S'_q} y_i^T \ \log f_c(x_i) \tag{2.19}$$

where f_c is final softmax classification layer of the segmentation network, x_i is a class probability at position i from the segmentation map x. S'_q is a set of valid pixels of an image q, i.e., a set of initial seeds generated by using CAM method. y_i is an one-hot image-level labels. In overall loss function, two terms of cross-entropy loss for the query and reference images are learned at the same time which encourages the co-learning by sharing seeding knowledge across images.

2.3.4 Summary

The mentioned WSSS methods provide the seeds knowledge to semantic models, these seeds can been viewed as a set of labeled data. By using the proposed SSC mechanism, it increases the number of labeled data (i.e., seeds) and uses these newly labeled data to improve semantic performances for the mentioned WSSS models. The entire process can be found in Chapter 4.

Table 2.1: Unsupervised autoencoder methods

Method*	**Year**	**Contribution**	**Related**
Deep Belief Network (DBN) [11]	2006	Train RBM [63] based AE w/ BP	[64–66]
Denoising Autoencoder (DAE) [67]	2008	Recover the undistorted AE's input by training AE with noisy input	[68]
Sparse Autoencoder (Sparse AE) [69]	2011	Find k largest units in hidden layer and set the rest to 0	[70]
Contractive Autoencoder (CAE) [71]	2011	Introduce the penalty term to encourage slight variations on hidden layer	[72]
Variational Autoencoder (VAE) [73]	2014	Learn global latent code by using prior distrib. and decoding distrib.	[74, 75]
Adversarial Autoencoder (AAE) [76]	2015	Extended from VAE, but use GAN for variational inference	[77, 78]
ELM Autoencoder (ELM-AE) [17]	2013	Single-Layer-Feedforward-Network based AE with fast training speed	[2, 79]

* Although similar concepts may be proposed earlier, here we cite most well-known publications.

Table 2.2: Weakly-supervised semantic segmentation methods

Method	**Year**	**Brief description**	**Type**	**VOC2012**	
				val	*test*
MIL-FCN [80]	2014	Train FCN w/ Multi-instance learning jointly	$\mathcal{U}$	N/A	25.7
EM-Adapt [55]	2015	Train DeepLabv1 w/ Expectation-Maximization + CRF	$\mathcal{U}$	38.2	39.6
SEC [3]	2016	Train DeepLabv1 w/ static seeding loss + CRF	$\mathcal{E}$	50.7	51.7
STC [81]	2017	Train DeepLabv1 progressively w/ Saliency detection + CRF	$\mathcal{U}$	49.8	51.2

HaS [82]	2017	Train CNN w/ hiding image patches + CAM	$\mathcal{I}$	N/A	N/A
TPL [83]	2017	Train first FCN + CAM for seeding mask, then second for results	$\mathcal{E}$	53.1	53.8
AE+PSL [84]	2017	Train DeepLabv1 + CAM w/ erasing gained regions + PSL	$\mathcal{I}$	55.0	55.7
GAIN [85]	2018	Train DeepLabv1 w/ trainable mask for erasing + CRF	$\mathcal{E}$	55.3	56.8
DSRG [4]	2018	Train DeepLabv2 w/ dynamic seeding loss + SRG + CRF	$\mathcal{E}$	59.0	60.4
ACoL [1]	2018	Similar to AE+PSL, but in end-to-end w/o PSL	$\mathcal{I}$	N/A	N/A
MDC [86]	2018	Train DeepLabv1 w/ multi-dilated rates + CAM	$\mathcal{I}$	60.4	60.8
AffinityNet [87]	2018	Train CNN w/ random walk for class-agnostic affinity + CAM	$\mathcal{I}$	58.4	60.5
MCOF [88]	2018	Train two DeepLabv1 w/ Saliency-guided Refinement + CRF	$\mathcal{I}$	60.3	61.2
FickleNet [89]	2019	Train DeepLabv2 w/ random dropout in network	$\mathcal{I}$	61.2	61.9
SSNet [90]	2019	Train DeepLabv2 w/ saliency detection jointly + CRF	$\mathcal{U}$	63.3	64.3
IRNet [91]	2019	Train DeepLabv2 w/ Class boundary map + Displacement field	$\mathcal{U}$	63.5	64.8
SSDD [92]	2019	Train CNN w/ knowledge & advice to remove noise	$\mathcal{U}$	64.9	65.5
CIAN [5]	2020	Train DeepLabv2 w/ co-learning across images + CRF	$\mathcal{E}$	64.3	65.3
MCIS [93]	2020	Train DeepLabv2 w/ co-attention classifiers	$\mathcal{I}$	66.2	66.9
SEAM [94]	2020	Train DeepLabv2 w/ equivariant regularization & PCM + CAM	$\mathcal{U}$	64.5	65.7
SubCat [95]	2020	Train CNN w/ clustering for parent & sub-category classifiers + CAM	$\mathcal{U}$	66.1	65.9

$\mathcal{E}$: Explicit seed learning which uses seeds directly in loss function.
$\mathcal{I}$: Implicit seed learning which uses either adversarial erasing or mining of discriminative regions.
$\mathcal{U}$: Undefined which uses CAM methods only with other techniques

Chapter 3

Progressive Framework for Semi-supervised Clustering via Representation Learning

3.0 Lead-in Section of Chapter 3 and Chapter 4

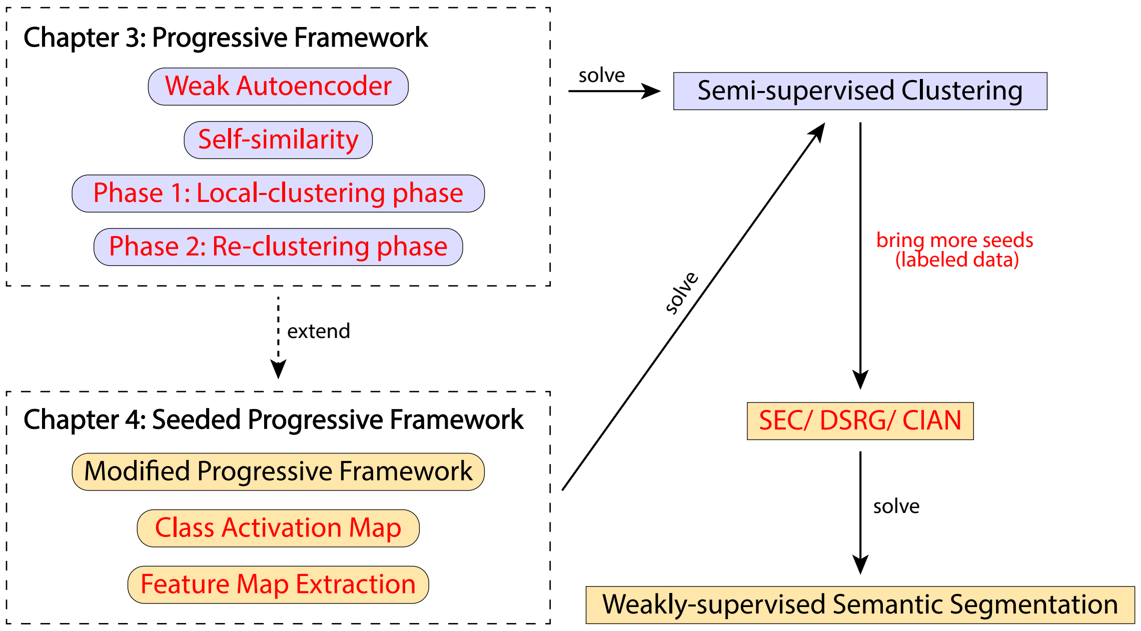

Chapter 3: Progressive Framework
Weak Autoencoder
Self-similarity
Phase 1: Local-clustering phase
Phase 2: Re-clustering phase
solve
Semi-supervised Clustering
extend
solve
bring more seeds (labeled data)
Chapter 4: Seeded Progressive Framework
Modified Progressive Framework
Class Activation Map
Feature Map Extraction
SEC/ DSRG/ CIAN
solve
Weakly-supervised Semantic Segmentation

Figure 3.1: Extended Overview of Figure 1.3. Details are highlighted in red.

In chapter 3, the semi-supervised framework will be proposed for partitioning a dataset into different subsets by using a small number of labeled data. The framework is wrapped into the cyclic structure where two phases are repeated iteratively. The first phase is local-clustering phase which aims at progressively searching unlabeled data (In this framework, it is known as member) from a small number of labeled data (namely, core). To initialize this phase, only one core is used to train a weak autoencoder (WAE) such that the strong feature of that data can be extracted. Assuming that the learned feature is shareable among the same class, it means data belonging to the same class (homogeneous data) should have similar features. Therefore, if any member can be well-reconstructed by that trained WAE, it is believed that this member can be grouped with the used core. In order to evaluate and rank the degree of reconstruction of each unlabeled data, the self-similarity is calculated by measuring the correlation coefficient between every pair of original and reconstructed data. To continue the clustering process, these two data can be used to train the new WAE again and search for another member. In fact, the WAE can learn the strong features only when the structure of autoencoder is weak such that the extracted information is minimize. This concept of minimalism in representation learning is completely different with current ML methods where the generalization performance is always the main objective.

To re-evaluate the clustering performance, the re-clustering phase is implemented where the procedure of previous phase is repeated but by interchanging the role of cores and searched members. Terminologically speaking, searched members are viewed temporarily core (i.e., t-core) while cores are regarded as temporarily member (t-mem). Similar with previous phase, only one t-core is utilized to train a WAE and search for t-mem. If t-core and searched t-mem have the same label, it is believed that this t-core is strongly similar to the t-mem which is originally labeled. Then, this t-core is accepted and regarded as a regular core in the rest of the proposed framework. On the other hand, if t-core and searched t-mem have the different label, this t-core is rejected and released as a member.

To conclude, the WAE is proposed to learn the strong feature of each data which this learned feature is useful for clustering. In order to systemically partition an entire dataset, the semi-supervised framework is implemented where two phases are repeated iteratively. As the number of labeled data is gradually increasing, this framework is called progressive framework.

In chapter 4, this framework is modified and extend to one of the important real-world applications in computer vision community, i.e., semantic segmentation. The modified framework is called seeded progressive framework. In order to reduce human efforts to prepare pixel-level annotation of each image, image-level labels which indicates the presences of objects is involved. This research area is defined as weakly-supervised semantic segmentation. Unlike previous framework, the computer vision techniques are employed for generating the initialized labeled data. Apart from this, there are several major modifications which are summarized as follows:

1. Different terminologies are used in Chapter 4. (See Table 3.1)

2. Two computer vision methods are applied to obtain the initial labeled data and clustering space (high-dimensional convolutional space).

3. To simplify the cyclic structure, we repeat the seeded progressive framework by two cycles, i.e., $\tau = 2$.

Table 3.1: Comparison of terminology between Chapter 3 and Chapter 4

	Progressive Framework	Seeded Progressive Framework
Phase 1	Local-clustering Phase	Composition Phase
Phase 2	Re-clustering Phase	Decomposition Phase
labeled data	core	seed
unlabeled data	member	non-seed
temporarily labeled data	t-core	-
temporarily unlabeled data	t-mem	-

4. In phase 2 of seeded progressive framework, we simplify two regulations proposed in Section 3.3.4.

5. The objective of previous framework is to partitionan entire dataset while now the objective is just to increase the number of labeled data. The clustered labeled data can be used to train some deep learning based segmentation models, i.e. SEC, DSRG and CIAN.

3.1 Overview

Semi-supervised and unsupervised clustering has brought significant impacts on various applications, limited prior knowledge are provided to supervise a data partitioning. Existing autoencoder (AE) based algorithms typically map high dimensional data to low dimensional space by preserving informative features. However, these features is generalized in reconstruction task but may not be appreciate in clustering. To this end, we propose a new semi-supervised clustering framework, called Progressive Framework. In the algorithm, we conversely learn discriminative features in unsupervised manner using AE with only one subnetwork node, called Weak Autoencoder (WAE). We adopt progressive approach to enlarge our labeled data through two phases recursively. Extensive comparison experiments on 13 benchmark datasets exhibit effectiveness of our framework over more than 35 state-of-the-art methods.

3.2 Introduction

Clustering analysis has been a hallmark problem during past decades in the field of artificial intelligence, it primely aims at actualizing a data compartmentalization in which similar objects are congregated and dissimilar objects are dispersed.

Without the availability of label information, it is an open question that the learning performance of any unsupervised models are in general far lower than required. Performances degrades dramatically on high-dimensional data. To this end, many researchers are mainly using the unsupervised model for feature extraction and dimension reduction to simplify the complexity of the raw data. Recent years, researchers expressed their interest into autoencoder (AE) [11], it consists of two symmetric components (i.e., encoder and decoder). It aims at minimizing the discrepancy between original and reconstructed data without any supervisory information. The central deepest hidden layer preserves highly abstract representation. Researches [96–98] has applied autoencoders as a feature extractor in clustering problem. Tian et al. [96] developed two-phases deep network to learn the non-linear embedding by massive layers of sparse AE, and further performs the k-means clustering on the code space. Literatures [97, 98] proposed the jointly optimized AE to learn deeper representation and cluster data simultaneously. Nevertheless, little of researchers use such unsupervised as a pure classifier.

On the other hand, the supervised deep convolutional neural network (DCNN) has been dominated the pattern recognition area, especially in image based objective recognition stream. With a big labeled dataset, the generalization performance with DCNN models has been significant improved during the past ten years. Therefore, researchers are more willing to use DCNN models, especially pretrained DCNN models as their classifiers.

Although labeling a large amount of data samples give the learning model a unprecedent prior knowledge which further improved the generalization performance of supervised DCNN model, labeling data samples especially a big dataset need a tremendous effort. In many real-world application, only small quantity of labeled data could be available which contributes the prior knowledge to the clustering task. Unlike supervised learning, such semi-supervised/unsupervised learning could definitely save the big workload. Therefore, plenty of semi-supervised methods have been proposed [20, 29, 99, 100].

Furthermore, by learning such huge sets of labeled training data, the network

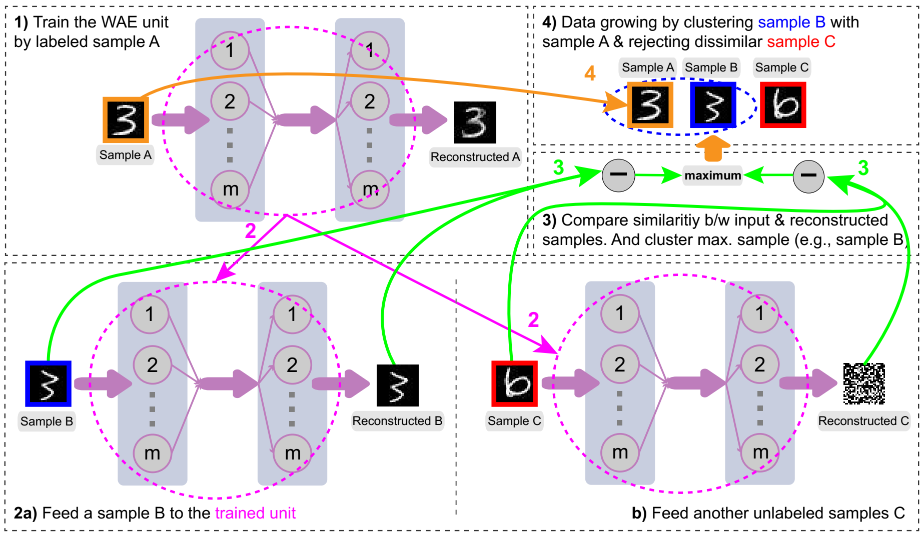

1) Train the WAE unit by labeled sample A
Sample A
Reconstructed A
1
2
m
4) Data growing by clustering sample B with sample A & rejecting dissimilar sample C
Sample A
Sample B
Sample C
4
3
maximum
3
3) Compare similaritiy b/w input & reconstructed samples. And cluster max. sample (e.g., sample B)
2
2
Sample B
Reconstructed B
Sample C
Reconstructed C
2a) Feed a sample B to the trained unit
b) Feed another unlabeled samples C

Figure 3.2: Motivation of the cluster growing with reconstructed samples; The key is to design a fast but "weak" autoencoder; Ideally samples B could be reconstructed completely since it shares the same label with sample A while sample C could be reconstructed incompletely.

simulates patterns that are consistent with the innate knowledge extracted from the given samples. But there are many things that people can do quickly that compliant artificial neural network cannot. For instance, a self-driving car can drive millions of miles but it will eventually encounter something new in which it has no experience. Similarly, a robot can learn to pick up a bottle, but if it has to pick up a cup it has to start from scratch. The way of using a large amount of labeled data samples with a giant network architecture is undeniable success in the recent transfer learning area, but the performance improvements through network architecture innovation are approaching its limitation according to the recent results on the ImageNet Large Scale Visual Recognition Challenge as mentioned in [12].

A three years old kid could easily recognize their own water bottle. If you ask them why he/she could, kids will probably tell you because it is different from others. Human repeatedly recognize items by comparing the difference between the target and the un-target items. It inspires us to have the motivation: **Could we design a classifier by progressively clustering similar samples without using label information?** Directly calculating similarity rate from the raw samples could be a most nature way, but it would be difficult to achieve competitive generalization performance. Therefore, **could we cluster the "similar" samples into one class**

based on the similarity rate of their reconstructed samples which have been generated by a "weak" auto-encoder? Our basic idea could be indicated as follow (also illustrated by Fig. 3.2):

1. We randomly select one sample A from the dataset and send it to train a "weak" autoencoder.

2. Then we randomly pick another sample B from the dataset, and then the trained "weak" autoencoder could be expected to generate very similar, if not the same, reconstructed sample B if and only if the sample A and B have the same label/-class. In other words, if the sample B and its reconstructed sample have a high similarity rate, we could cluster the sample A and B together and consider the two samples belong to the same class. (See top-right diagram of Fig. 3.2)

3. Once the clustered samples from one group increase, the more specific knowledge from the corresponding class could be enhanced as well. Then we could use one-class classifier to train the clustered samples to speed up the classification process.

In particular, main contributions of this chapter is listed below:

1. The proposed method uses an autoencoder with only one subnetwork node [101] to make sure the weakest generalization performance of the autoencoder has been achieved. Such design is opposite from the current machine learning principle that tries to build a complicate network architecture to obtain a powerful universal generalization performance. With the lightweight architecture of the autoencoder, it presumes the learned highly distinctive representation are important for discrimination. We learn such features using an autoencoder in a complete unsupervised manner.

2. Following the previous publication [102], we design a progressive approach to navigate the workflow of clustering. In Local-clustering phase, random labeled data are used to form many local clusters which included labeled and unlabeled data. In Re-clustering phase, we verify all clustered result that accept qualified data while reject others. This progressive approach gradually clusters the data by this two phases. We solve clustering problem by using extreme small amount of labeled data in a semi-supervised way. (it could be regarded as unsupervised method if only few labeled samples are used).

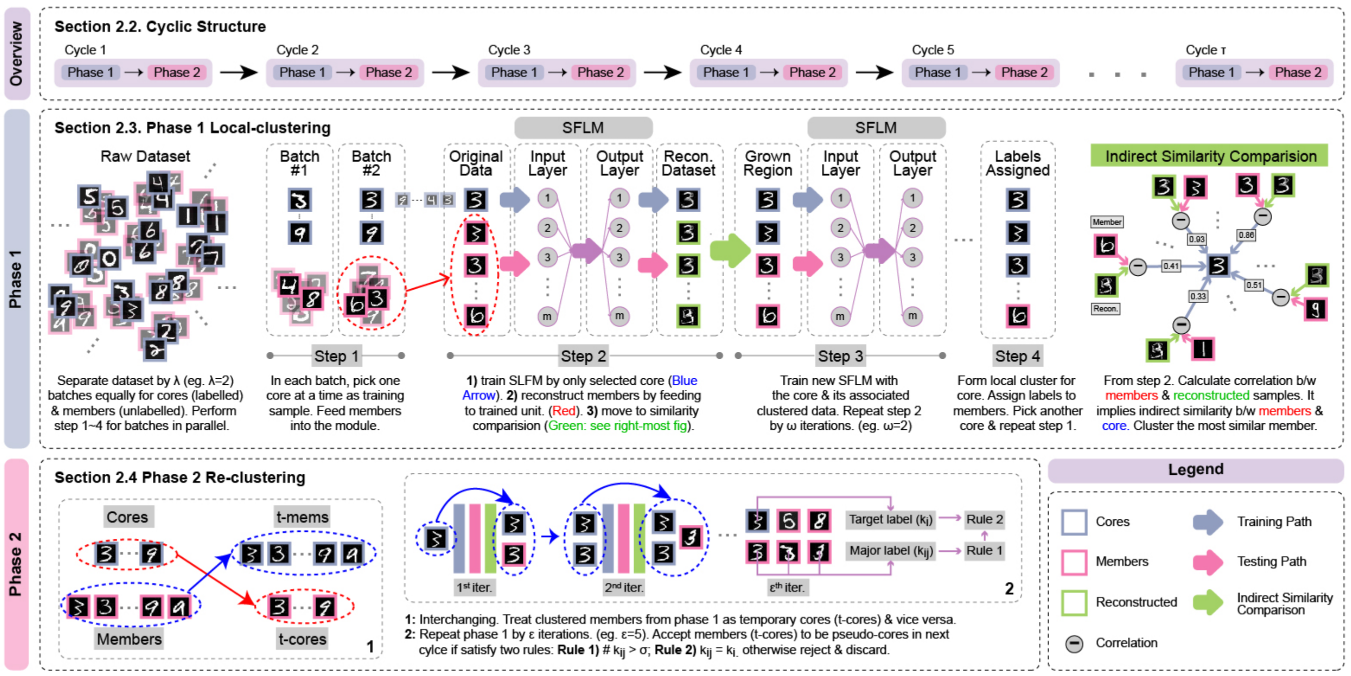

Overview
Section 2.2. Cyclic Structure
Cycle 1 Phase 1 → Phase 2
Cycle 2 Phase 1 → Phase 2
Cycle 3 Phase 1 → Phase 2
Cycle 4 Phase 1 → Phase 2
Cycle 5 Phase 1 → Phase 2
Cycle τ Phase 1 → Phase 2
Phase 1
Section 2.3. Phase 1 Local-clustering
Raw Dataset
Batch #1
Batch #2
SFLM
Original Data
Input Layer
Output Layer
Recon. Dataset
Grown Region
Labels Assigned
Indirect Similarity Comparision
Member
Recon.
Step 1
Step 2
Step 3
Step 4
Separate dataset by λ (eg. λ=2) batches equally for cores (labelled) & members (unlabelled). Perform step 1~4 for batches in parallel.
In each batch, pick one core at a time as training sample. Feed members into the module.
1) train SLFM by only selected core (Blue Arrow). 2) reconstruct members by feeding to trained unit. (Red). 3) move to similarity comparision (Green: see right-most fig).
Train new SFLM with the core & its associated clustered data. Repeat step 2 by ω iterations. (eg. ω=2)
Form local cluster for core. Assign labels to members. Pick another core & repeat step 1.
From step 2. Calculate correlation b/w members & reconstructed samples. It implies indirect similarity b/w members & core. Cluster the most similar member.
Phase 2
Section 2.4 Phase 2 Re-clustering
Cores
t-mems
Members
t-cores
1st iter.
2nd iter.
εth iter.
Target label (ki) → Rule 2
Major label (kij) → Rule 1
1: Interchanging. Treat clustered members from phase 1 as temporary cores (t-cores) & vice versa.
2: Repeat phase 1 by ε iterations. (eg. ε=5). Accept members (t-cores) to be pseudo-cores in next cylce if satisfy two rules: Rule 1) # kij > σ; Rule 2) kij = ki, otherwise reject & discard.
Legend
Cores
Members
Reconstructed
Correlation
Training Path
Testing Path
Indirect Similarity Comparison

Figure 3.3: **Overview**: framework iterates τ cycles to perform semi-supervised clustering in which phase 1 and 2 are processed alternatively in each cycle; **Phase 1**: Pick cores to perform individual clustering sequentially. The relatively confident member data are added into the coreset; **Phase 2**: Interchange cores & members as t-mems & t-cores. Repeat Phase 1 to either accept or filter the unqualified t-cores.

3. Experimental results demonstrates that our proposed framework achieves outstanding performances compared with existing semi-clustering/unsupervised methods, and even reaches the comparable performance with some supervised algorithms. It verifies the effectiveness of our novel representation learning strategy in a semi-supervised/unsupervised task.

The remainder of this Chapter is structured as follows. Chapters 3.3 - 3.4 outline the progressive framework with detailed procedures. Chapter 3.6 evaluates the experimental performances. Chapter 3.7 investigates the sensitivity of involved parameters. Chapter 3.8 briefly concludes the Chapter.

3.3 Progressive Framework

As detailed schematic diagram depicted in Fig. 3.3, the proposed framework implements two phases alternatively by numerous repetitions, which congregates members (unlabeled) around cores (labeled) progressively (top sub-diagram of Fig. 3.3). In phase 1, the "weak" autoencoder (WAE) is involved to learn "weak" but specific feature in an unsupervised manner (middle). Step 1-4 demonstrates the symbolic workflow of how one core sample constitutes the local cluster with its detected members. In phase 2, the clustering result of phase 1 are verified (bottom). The accepted clustered data will be regarded as pseudo-cores while rejected samples will be released as members.The cyclic searching-checking progressively enlarges the number of cores. Procedure of two phases are summarized as follows:

Phase 1: Local-clustering phase

step 1: **Batch Separation**: Ahead of clustering, the dataset X $\in \mathcal{R}^{n \times d}$ is divided into λ batches, i.e., X = [X$_1$,X$_2$,...,X$_\lambda$]. Clustering will be executed simultaneously.

step 2: **Individual Clustering**: Train the WAE by a selected i-th core (x_i), then feed the learned WAE unit by an unlabeled data. Derive self-similarities by measuring correlations between ordinal and reconstructed samples. The most similar result is clustered.

step 3: **Clustering Progression**: Repeat step 2 by ω iterations but a new WAE is trained with clustered member(s) and its core.

step 4: **Sequential Process**: Assign labels to members clustered by the core within a local cluster. Then pick another core, and repeat step 2-3 for remaining data in sequential order.

Phase 2: Re-clustering phase

step 1: **Data Interchanging**: Gather all members from batches, switch members to temporary cores (t-cores) and vice versa.

step 2: **Results Verifying**: Repeating all steps of phase 1 by including ε iterations in step 3. t-cores (i.e., members from phase 1) are either accepted or eliminated by examining two rules.

In this section, we present the big picture of the proposed framework phase by phase. From top to down, Section 3.3.1 formulates the clustering problem and notation used mathematically. Next, Section 3.3.2 states the wrapped cyclic structure. For each cycle, two phases are involved. Therefore, Section 3.3.3 and 3.3.4 discusses the operation of Local-clustering and Re-clustering phase. Note that the underlying ideology of WAE 3.4 will be interpreted in Section 3.4.

Table 3.2: Notations of Progressive Framework

Notations	Decription
General	
n, d	number of instances & features
λ, τ	number of batches & cycles
$\mathcal{L}, \mathcal{U}$	labelled & unabelled Set
X = [X$_1$, X$_2$, ... X$_\lambda$]	batches (subsets) in phase 1
Phase 1 (P1)	
$l_{i,b}^{t,1}$	i-th core in b-th batch of t-th cycle in P1
$p_{i,j,b}^{t,1}$	j-th found member from i-th core in b-th batch of t-th cycle in P1
ω	number of required member in searching region
Phase 2 (P2)	
$l_i^{t,2} = p_i^{t,1}$	i-th t-core of b-th cycle in P2
$p_{i,j}^{t,2} = l_i^{t,1}$	j-th found t-mem from i-th t-core of t-th cycle in P2
k_i, k_{ij}	target label of i-th t-core & its associated major label from t-mems
ε	number of required t-mem in checking region
σ	qualified rate for checking
WAE	
(a_f, b_f)	input weight and bias in encoder
β	output weight in decoder
g	invertible Activation function
$g(aX + b)$	output of encoder (i.e input of decoder)
m	number of hidden neurons

3.3.1 Problem Formulation

Let consider the X be a $n \times d$ matrix, each row represents an instance of data with d-dimensional features. Suppose it is composed of a set of l cores and a collection of $n - l$ members, i.e.,

$$\{\mathcal{L},\mathcal{U}\} \subseteq \mathrm{X} = \left\{\{(x_i, y_i)\}_{i=1}^{l}, \{(x_j)\}_{j=1}^{n-l}\right\} \subseteq \mathrm{X} \tag{3.1}$$

where x_i and x_j denote i-th cores and j-th members respectively. In set $\mathcal{L}$, $y_i \in \{1, 2, ..., \mathrm{k}\}$ represents the associated label of i-th instance, where k is a predefined number of clusters k. In our proposed framework, the number of k is directly pre-set

Algorithm 1 The Proposed Algorithm of Progressive Framework

```
    Initialization: Given X, $\mathcal{L}$, $\lambda$, $\omega$, $\varepsilon$, $\sigma$, $\tau$ & k

 1: for $t \leftarrow 1$ to $\tau$ do
 2:     if $t < \tau$ then
 3:         Phase 1: Local-clustering
 4:         Step 1 Batch separation: X= [X$_1$, X$_2$, ... X$_\lambda$]
 5:         Initialize $p^{t,1} = [p_1^{t,1}, p_2^{t,1}, ..., p_\lambda^{t,1}] = \emptyset$
 6:         parfor $j \leftarrow 1$ to $\lambda$ do
 7:             Step 2 to 4
 8:         end parfor
 9:         Obtain the clustered set $p^{t,1}$
10:         Phase 2: Re-clustering
11:         Step 1: Data Interchange
12:         $p \leftarrow$ # found members of $p^{t,1}$
13:         for $i \leftarrow 1$ to $p$ do
14:             Step 2: Results Verifying
15:         end for
16:         Obtain the re-clustered set $p^{t,2}$
17:         Update $\mathcal{L} \leftarrow \{\mathcal{L} \cap p^{t,2}\}$
18:     else
19:         Repeats phase 1 to 2 without eliminating results
20:     end if
21: end for
```

the same as the number of class of the dataset. Within the cycle, we employ the progressive clustering in which l local clusters are formed by l cores associated with ω members as mentioned in Step 3 of phase 1. Therefore, the cycle objective function is generally formulated as follows:

$$\min_{\theta} E(\theta) = \sum_{i=1}^{l} \sum_{j=1}^{\omega} D(x_i, x_{ij}; \theta) \tag{3.2}$$

where x_{ij} are members detected by x_i labeled patterns, $D(\cdot\ ;\theta)$ indicates pairwise dissimilarities in which θ are model parameters in WAE. We will further interpret and reformulate the above function in the following subsection. To derive our proposed framework systemically, complex notations with the format $A_{c,d}^{a,b}$ are defined in Table 3.2. The superscript a and b represent time-related status (e.g.,, the cycle or phase.), and the subscript c and d indicate another dynamic information (e.g., index of batch, core and member data).

3.3.2 Cyclic Structure

The progressive framework F are wrapped into the cyclic structure such that it iterates Local-clustering and Re-clustering phase by τ cycle(s) where τ >1 as additional ending cycle is needed to clustering all remaining data. As mentioned, it rejects and discards clustered members in regular phase 2. However, all detected members will not be eliminated in ending cycle. Remaining data are used to form a local cluster with ε cores, we instead assign label to a member by majority voting among cores.

3.3.3 Phase 1: Local-clustering Phase

In this phase, we utilize the batch separation to shorter complexity and computational time. We use the Weak Autoencoder (WAE) indicated in Section 3.4 as a unit to determine similarities between given core and remaining members. We propose the progressive approach to cluster unlabeled data (members) with labeled data (cores) sequentially. Hence, it leads us to describe how our algorithm works with the batch separation. Following by presenting the operation, where the notations and figures are involved comprehensively.

Step 1: Batch Separations

Unlike other semi-supervised clustering methods where distances between every pair of connected data are required to measure. The proposed framework estimates only local linkages between two data. Precisely, the idea is to progressively cluster small sets of data. An entire dataset could be pre-separated equally into subsets, i.e., $\mathrm{X} = [\mathrm{X}_1, \mathrm{X}_2, ..., \mathrm{X}_\lambda]$, where λ represents the pre-defined number of batch. In each batch, we process individual clustering for each cores sequentially. Batch process can be executed simultaneously that highly reduces the computational time. In fact, the value of λ depends on the size and dimensionality of dataset for which the related analysis will be conducted in Section 3.7.2.

Step 2: Individual Clustering with Self-similarity

Different from typical algorithms calculates the similarities between labeled and unlabeled samples, we adopt the **self-similarity** to estimates between original and

Algorithm 2 Locally Clustering Phase

```
    **Initialization:** Given $\mathbb{X}$, $\mathcal{L}$, b, $\omega$, K
    **Output:** Clustered set $p^1$
 1: Let Set $\mathcal{P} = \mathbb{X} \in \mathcal{R}^{n \times d}$
 2: Conduct batch separation on $\mathcal{P}$, s.t. $\mathcal{P}$= [$\mathcal{P}_1$, ... ,$\mathcal{P}_b$]
 3: Initialize $p^1 = [p^1_1, p^1_2, ..., p^1_b] = \emptyset$

 4: parfor $j \leftarrow 1$ to $b$ do
 5:     $l_j \leftarrow$ # cores in $\mathcal{P}_j$
 6:     for $i \leftarrow 1$ to $l_j$ do
 7:         Obtain $\mathcal{P}^i_j$ by removing $i$-th core
 8:         for $m \leftarrow 1$ to $\omega$ do
 9:             $ni \leftarrow \#\mathcal{P}^i_j$
10:             Train WAE by core $l^1_{i,j}$
11:             Encode and decode the $\mathcal{P}^i_j$
12:             Compute correlation matrix $\Phi_i$
13:             Find $q = \text{argmax}_{q \in [1,ni]} \Phi_i(q)$
14:             Find member data $p^1_{m,i,j} = \mathcal{P}^i_j(q)$
15:             Enlarge the core set, $l^1_{i,j} \leftarrow \{l^1_{i,j}, p^1_{m,i,j}\}$
16:             Update $\mathcal{P}^i_j \leftarrow \{\mathcal{P}^i_j \cup \mathcal{P}^i_j(q)\}$
17:         end for
18:         Consider the member set $\mathcal{Q} = \{p^1_{m,i,j}\}^{\omega}_{m=1}$
19:         Obtain set $\mathcal{Q}' \leftarrow \mathcal{Q} \cap \mathcal{L}_j$
20:         if $\mathcal{Q}' \notin 0$ then
21:             $\mathcal{Q} \leftarrow \{\mathcal{Q} \cup \mathcal{Q}'\}$
22:         end if
23:         Update $\mathcal{P}^i_j \leftarrow \{\mathcal{P}^i_j \cup \mathcal{Q}\}$
24:         Update $p^1_j \leftarrow \{p^1_j \cap \mathcal{Q}\}$
25:     end for
26: end parfor
```

reconstructed members. By mentioning that, the above objective function can be formally rewrote as:

$$\min_{\theta} E(\theta) = \sum_{i=1}^{l} \sum_{j=1}^{\omega} D(x_{ij}, f(x_{ij}, x_i, \theta)) \tag{3.3}$$

where $f(x_{ij}, x_i, \theta)$ signifies reconstructed members via the WAE learned by x_i. The cycle objective function is oppositely to maximize the similarity $S(\cdot)$, so it is equivalent to redefine Eq. 3.3 as:

$$\min_{\theta} E(\theta) \Leftrightarrow \max_{\theta} E'(\theta) = \sum_{i=1}^{l} \sum_{j=1}^{\omega} S(x_{ij}, \hat{x_{ij}}) \tag{3.4}$$
$$\text{s.t.} \; \forall \; ij, \; \hat{x_{ij}} = f(x_{ij}, x_i, \theta)$$

Assume we have an input x_i and a hidden code z_i, the prior distribution can be denoted as $p(x_i)$ while the encoding and decoding distribution can be represented as $q(z_i|x_i)$ and $p(x_i|z_i)$ respectively. By using WAE, z_i retains specialized features provided by x_i which are most characteristic to present the hidden pattern. From our prospective, these features are highly shareable within the class. That is supposing the input x_i is used to train the WAE, then an unlabeled x_{ij} is feed-forward into that WAE. After encoding, the squashed hidden vector is projected back by applying the trained decoding mapping $p(x_i|z_i)$. x_{ij} can only be well-reconstructed when the used decoding function approximates to expected, i.e., a desired decoding distribution $p(x_{ij}|z_{ij}) \approx p(x_i|z_i)$. Therefore, it implies the self-similarity rate between x_i and x_{ij}:

$$S(x_{ij}, \hat{x}_{ij}) \implies S(x_{ij}, x_i) \tag{3.5}$$

Specifically, a statistic similarity metric, correlation, is involved to find out which member is closest to the particular core, and it is calculated by Eq. 3.6:

$$\phi_{ij} = corr(x_{ij}, \hat{x}_{ij}) = \frac{cov(x_{ij}, \hat{x}_{ij})}{\sigma_{x_{ij}} \cdot \sigma_{\hat{x}_{ij}}} \tag{3.6}$$

where $cov(\cdot)$ defines covariance while $\sigma_{x_{ij}}$ and $\sigma_{\hat{x}_{ij}}$ mean standard deviation of two given variables. In practice, members are all propagated into trained WAE, for which it yields the correlation matrix $\Phi_i = \{(\phi_{im})\}_{m=1}^{n-l}$. However, we select only individual x_{ij} as a clustered result each time (clustered by x_i), for which it has the largest correlation in reconstruction. This process is called individual clustering.

Step 3-4: Sequential Clustering

The proposed procedure is highly linked with our previous research works [102], the early developed structure of cluster growing algorithm, PLM, was to approximate any hybrid system (i.e., linear/non-linear) by using multi-SLFNs which are trained by n parent clusters (Eq. 3.7), such that each parent cluster can obtain its corresponding

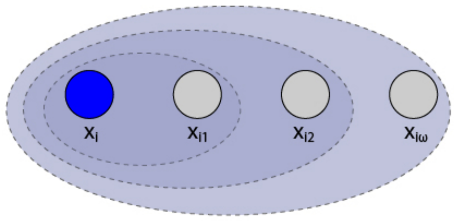


Figure 3.4: Progressive Search: Blue labeled data x_i will search the "nearest" unlabeled data x_{i1} to form the local cluster; They train WAE and search other, and so on.

offspring clusters (Eq. 3.8).

$$\phi^0 = [\phi^0_1, \phi^0_2, ...\phi^0_n] \tag{3.7}$$

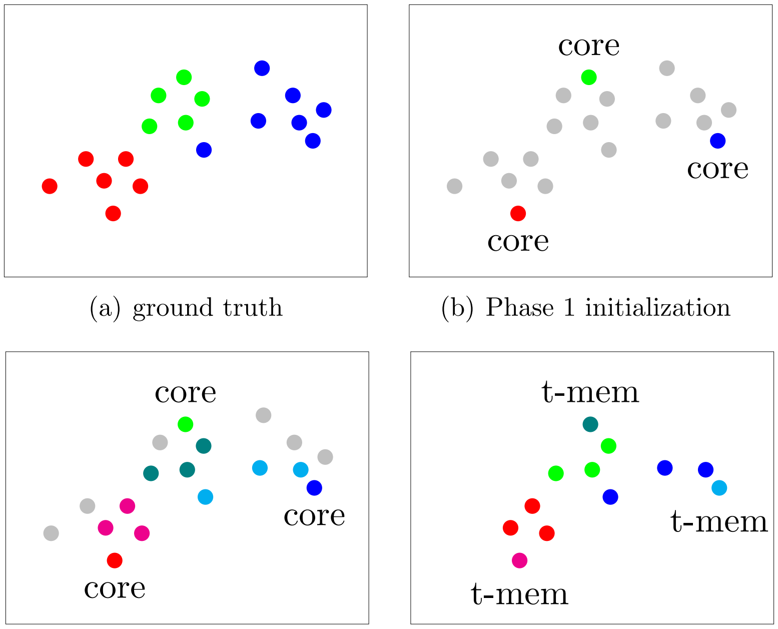
$$\phi^1 = [\phi^1_1, \phi^1_2, ...\phi^1_n] \tag{3.8}$$

PLM algorithm repeats the $(s1 - 1)$ times. It proved that the clustering can be completed on all unlabeled data for setting positive finite value $s1 : s1 \to \infty$. However, the main problem of PLM is higher computational complexity. Therefore, we propose the following workflow to reduce the computational time.

As mentioned in Section 3.3.3, our first modification is to add the batch separation to reduce the computational cost where the related experiments will be conducted in Section 3.7.2. For brevity, we assume λ=1 in this section. Similar to PLM, we cluster members by core x_i into a local cluster:

$$\begin{aligned} p_i &= \{\{\{x_i, x_{i1}\}, x_{i2}\}, ...\} = \{\{p_{i1}, x_{i2}\}, ...\} \\ &= \{x_i, x_{ij}\}^{\omega}_{j=1} \end{aligned} \tag{3.9}$$

Few top-rank members are normally reconstructed completely from correlation matrix Φ_i, it implies they share the same label with its training sample x_i. However, we conservatively cluster only one member x_{ij} with highest self-similarity in each timestep. It is because our goal is to train "weak" autoencoder with "weak" performance, the enhanced trained knowledge will "generalize" the autoencoder (See Section 3.4). Fig. 3.4 shows how we form the local cluset p_i progressively, we train first WAE by x_i to cluster x_{i1}. Then, we train another WAE by p_{i1} (with adding x_{i1}) and cluster x_{i2}. As "mis-cluster" members possibly destroy the algorithm. Therefore, we only cluster one sample at a time. And we call it "progressive" search.



(a) ground truth (b) Phase 1 initialization

(c) Phase 1 result (d) Phase 2 initialization

Figure 3.5: Procedure of Phase 1 (ω=3.): Three primary colors (red, blue & green) with similar colors (magenta, cyan & teal) & gray-color represent cores, associated members & unseen data. e.g., cyan-data are members clustered from blue-data.

After that, p_i will be removed from the searching pool. We pick up the other core x_{i+1} and form p_{i+1}. l local clusters will be form as follows:

$$p = [p_i, p_{i+1}, ..., p_{i+l-1}] \tag{3.10}$$

Fig. 3.5 demonstrates procedures of phase 1. We have 18 samples from 3 clusters, the real labels are shown in (a). Given only 1 core from each cluster (b), local clusters are formed after phase 1 with ω members (c). It initializes the phase 2 by considering only cores and members (d). In each batch, we cluster data in sequential. However, multi-batches can be implemented at the same time. It reduce the higher complexity issues from PLM.

3.3.4 Phase 2: Re-clustering Phase

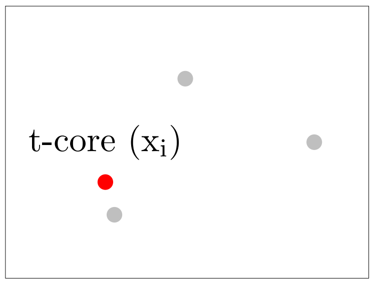
To re-evaluate the clustered result from phase 1, the progressive framework implements the Re-clustering phase. In this phase, we execute individual clustering as phase 1 does by interchanging cores and members. Consider the core x_i labeled as

Algorithm 3 Re-clustering Phase

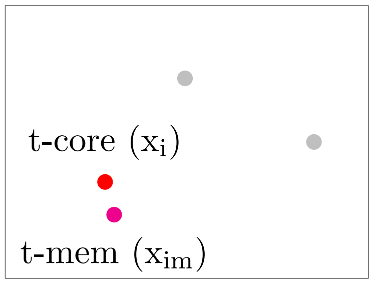
```
Initialization: Given $p^1$, $\varepsilon$ & $\sigma$
Output: Re-clustered set $p^2$

1: $p \leftarrow$ # found members of $p^1$
2: for $i \leftarrow 1$ to $p$ do
3:     Consider t-core $l_i^2 = p^1(i)$
4:     for $m \leftarrow 1$ to $\varepsilon$ do
5:         Repeat procedures of Phase 1 by using $l_i^2$
6:     end for
7:     Consider the t-mem set $\mathcal{Q} = \{p_{m,i}^2\}_{m=1}^{\varepsilon}$
8:     Assume the label $km_i$ is a majority label of $\mathcal{Q}$
9:     Count $y \leftarrow$ # t-mem assigned $km_i$
10:    $N \leftarrow$ # total t-mem in $\mathcal{Q}$
11:    if ( $y$ / $N$) $\geqslant \sigma$ then
12:        Let $kc_i$ is the label assigned to t-core $l_i^2$
13:        if ( $kc_i \neq km_i$ ) then
14:            Update $p^1 \leftarrow \{p^1 \cup p^1(i)\}$
15:        end if
16:    end if
17: end for
18: Assign $p^2 \leftarrow p^1$
```

k_i clusters the member x_{ij} (also assigned as k_i), we treat x_{ij} as temporary core, i.e., t-core and x_i as t-mem. Similar with ω, we use ε in this phase to represent the number of t-mems are required to cluster. Suppose many t-mems belong to same class k_{ij}, we call it "majority label". We then calculate the majority rate *m-rate* by number of k_{ij}/ε.

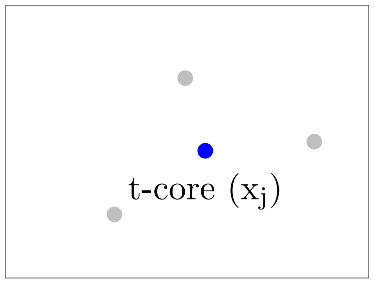
After that, we apply two regulations to verify whether the t-core x_{ij} are accepted: i) majority rule: *m-rate* $> \sigma$ which is qualified rate; ii) equivalence rule: $k_i = k_{ij}$. The t-cores are accepted and regarded as pseudo cores in next cycle if they satisfy those rules. The unqualified clustered results will be rejected and release as member. Follow Fig. 3.5, we present schematic success and failure example. For a & b in Fig. 3.6, the t-core (red) successfully clusters the correct t-mem (pink). However, the t-core (blue) cluster the wrong t-mem (teal) in c & d.



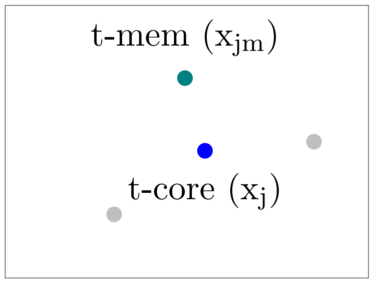
(a) Success: Starting



(b) Success: Result



(c) Failure: Starting



(d) Failure: Result

Figure 3.6: Two Scenarios in Phase 2 (ε=1 & σ=1): Following Fig. 3.5, we present success & failure cases. For success, i-th t-core finds ε t-mem(s) *m-rate* $> \sigma$. For failure, t-core x_j finds t-mem(s) but with wrong label.

3.4 Minimalism in Representation Learning

Informax approach has been a long held belief in field of representation learning, for which its underlying ideology is to retain exhaustive information through the parameterized mapping from input x to a latent code z. With the goal of generalization capability, most AEs are dedicated to model generative non-linear function, for which unseen instances are capable to be compressed into and decompressed from the latent space.

As depicted previously, the key of the proposed method is to design an AE with **weak performance (see Fig. 3.2) but fast training speed** because such AE will be used for thousands to ten thousands of times in the entire clustering process. Therefore, we have used a two layer network with an non-iterative learning strategy which we have proposed in [101]. We only use one subnetwork node consisted of m hidden nodes in order to achieve a fast but limited generalization performance. Although it drops a universal approximation capability of feature learning, such simplicity approach is workable on clustering task. Two data can be straightly dispersed when they are dissimilar, it implies that an discriminative features is necessitated to depict the similarities. In contradiction of infomax, we then advocate the naive representation learning strategy, **infomin approach**, which can be regard as information minimalism.

Weak Autoencoder (WAE) is a SLFN-AE which can be decoupled into encoder and decoder. The input X is projected to a m-dimensional random feature space through the encoder. Then, it is mapped to an output space via the decoder. Similar to [2], the invertible function is used to calculate the decoder weights, but we use only two steps in our learning system:

step 1: Initialize the orthogonal input weight w_f and bias b_f:

$$w_f^T w = \mathrm{I},\ b_f^T b_f = 1 \tag{3.11}$$

step 2: Given an output X which is equivalent with its input, and inverse of activation function g^{-1} (e.g.,, sigmoid or sin), the decoder weight w_n and bias b_n are calculated as follows:

$$w_n = g^{-1}(\mathrm{X}) \cdot \mathrm{H}^{\dagger} \tag{3.12}$$

$$b_n = \sqrt{mse\ (w_n \cdot \mathrm{H} - g^{-1}(\mathrm{X}))} \tag{3.13}$$

$$g^{-1}(\cdot) = \begin{cases} \arcsin(\cdot) & \text{if } g(\cdot) = sin(\cdot) \\ -\log(\frac{1}{(\cdot)} - 1) & \text{if } g(\cdot) = \frac{1}{1+e^{-1(\cdot)}} \end{cases} \tag{3.14}$$

where H is a representation on hidden space, the same as previous section, $\mathrm{H}^{\dagger}$ is called a Moore Penrose Pseudo-inverse matrix and can be calculated as follows.

$$\begin{cases} \mathrm{H}^{\dagger} = \mathrm{H}^{\mathrm{T}} \cdot (\frac{\mathrm{C}}{\mathrm{I}} + \mathrm{H}\mathrm{H}^{\mathrm{T}})^{-1}, & \mathrm{H}^T\mathrm{H} \text{ is singular} \\ \mathrm{H}^{\dagger} = (\frac{\mathrm{C}}{\mathrm{I}} + \mathrm{H}\mathrm{H}^{\mathrm{T}})^{-1} \cdot \mathrm{H}^{\mathrm{T}}, & \mathrm{H}^T\mathrm{H} \text{ is non-singular} \end{cases} \tag{3.15}$$

Note that the positive parameter $\mathrm{C} \in [2^{-10}, 2^{10}]$ is used as a regularization term to improve the stability.

There are two major modifications made on top of the autoencoder [2] to achieve information minimization:

1. For model structure, we select single hidden layer with one subnetwork node to guarantee not only the weakest reconstruction performance but also the fastest processing speed, i.e., the single hidden unit is merely used as an input weight.

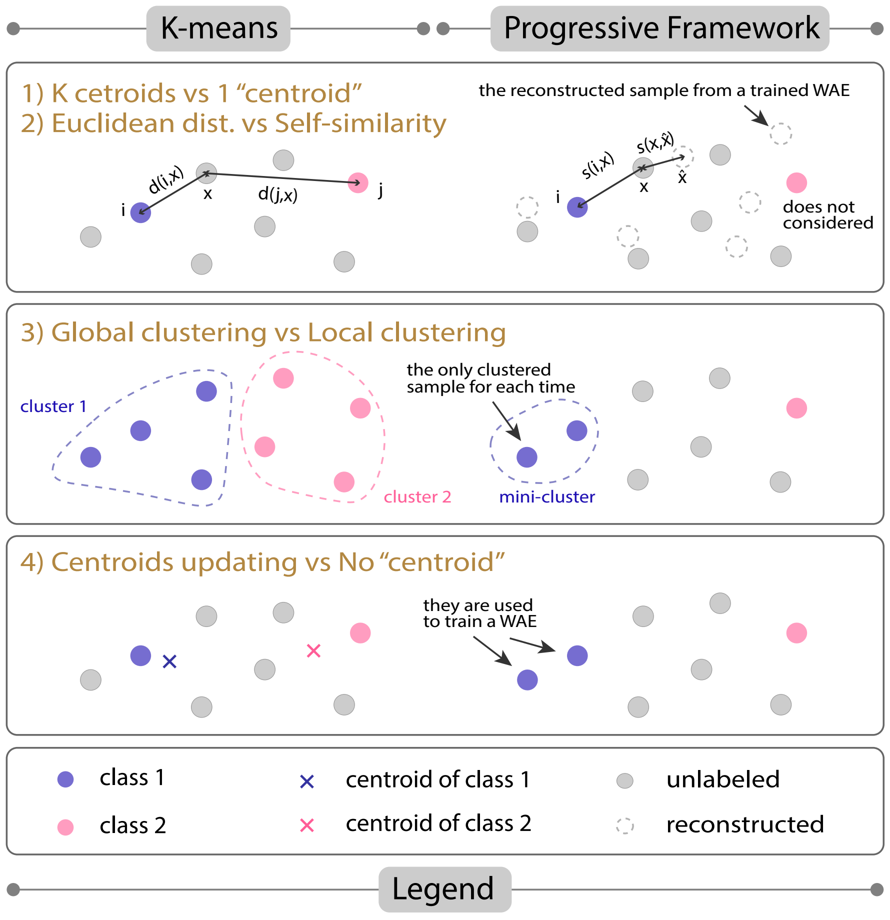

K-means
Progressive Framework
1) K cetroids vs 1 "centroid"
2) Euclidean dist. vs Self-similarity
the reconstructed sample from a trained WAE
d(i,x)
d(j,x)
i
x
j
s(i,x)
s(x,x̂)
x̂
does not considered
3) Global clustering vs Local clustering
cluster 1
cluster 2
the only clustered sample for each time
mini-cluster
4) Centroids updating vs No "centroid"
they are used to train a WAE
class 1
class 2
centroid of class 1
centroid of class 2
unlabeled
reconstructed
Legend

Figure 3.7: Comparison with k-means clustering

2. For learning strategy, we adopt the non-iterative learning process takes only two steps and is quite simple. In other words, we would not update the input weight and bias (w_f, b_f) by preceding (w_n, b_n) iteratively contrary to [2]. It "learns" only special features while we intentionally weaken the generalized reconstruction ability of such AE. Therefore, we name it as the weak autoencoder (WAE).

In fact, the WAE could be replaced by other data reconstruction methods such as low-rank representation [99]. With the low-rank representation, the actual performance of the proposed framework could be slightly improved. However, as the WAE unit need to be used for thousands to ten thousands of times in the entire process, in this chapter, we have to use the proposed WAE unit, rather than the low-rank representation method to speed up the entire clustering process.

3.5 Comparison with k-means clustering:

In order to help readers to under the proposed clustering algorithm, we compare the progressive framework with the classical clustering method, k-means clustering. Five major differences are summarized in the following (See Fig. 3.7 also):

1. The proposed method considers single "centroid" at a time, i.e., k=1 in k-means since the only one labeled sample is used to train the WAE while all unlabeled samples are involved.

2. Our method clusters one sample for each time, i.e., the nearest sample to the training set. Thus, it has a local clustering ability. Instead, the k-means assigns labels to all unlabeled samples. Hence, it performs the global clustering.

3. We calculate the self-similarity rather than the conventional Euclidean distance. Therefore, there is no statistical assumption of sample distribution, e.g., normal distribution.

4. Our method updates the "centroid" by considering only the certain clustered samples, i.e., add clustered samples into the training set and train another WAE. In contrast, k-means updates the centroids by summarizing all clustered results for each class. In other words, there is no statistical centroid in the proposed algorithm.

5. The update is made in the composition phase (similar to the assign step in k-means) but not in the decomposition phase (similar to the update step).

3.6 Experiment Verification

In this section, we investigate our proposed progressive framework by comparing with the existing methods. Our 13 tested datasets will be splitted into two groups and compared with other methods separately. In Section 3.6.1, we discuss the experimental setting by specifying the details of datasets, environments and evaluation metrics. In Section 3.6.2, we conduct an experiment using the 5 present semi-supervised methods on tabular datasets. In Section 3.6.3, the clustering performance of our proposed method on image datasets will be evaluated.

Table 3.3: Dataset Descriptions

Dataset	Dimension	Sample	Class	Remark
Tabular Dataset				
Hill-Valley	100	1212	2	UCI
Mushroom	112	8124	2	UCI
IJCNN1#	22	10000	2	LIBSVM
Satimage	36	4435	6	LIBSVM
Protein#	357	10500	3	LIBSVM
German	4	3025	2	LIBSVM
Image Dataset				
USPS	256	9298	10	Handwriting
USPS1K	256	1000	10	Handwriting
MNIST1K	784	1000	10	Handwriting
COIL20	1024	1440	20	Object
CIFAR10*	256	60000	10	Object
Caltech101*	256	9144	102	Object
Caltech256*	256	30607	256	Object
Scene15*	256	4485	15	Scene
Yale-Face	4096	165	15	Human Face
Olivertti-Faces	4096	400	40	Human Face

* Convolutional Features: Extracted from Random-VGG16
# Original sample size: IJCNN1=141691; Protein=17766

3.6.1 Experimental Setting

Dataset Specification. Our tested datasets typically divided into two groups, i) 6 tabular datasets which are obtained from the UCI Repository and LIBSVM and ii) 7 well-known image datasets. Table 3.3 shows dataset descriptions of both categories. Image benchmarks can be specifically separated into three sub-categories (handwritten, scene and object). Note that the sign * represents that we are using Convolutional Features (DF) from those dataset rather than raw data, the features are extracted from DCNN Model (VGG16) with random weights. (See Section 3.6.3)

Environment. Experiments are all carried out in MATLAB 2019b on the Linux Ubuntu 18.04 platform with 12 cores of Intel Core i9-7920X @ 2.9 GHz and 128 GB memory.

Evaluation Metrics. To measure the clustering performance, we employ Rand Index (RI) and classical clustering accuracy (ACC). RI is the statistical measurement score scaled between 0 and 1 for quantifying the similarity between actual and desired label assignments. We compute RI using the below equation:

$$RI = \frac{a+b}{{}_nC_2} = \frac{2 \cdot (a+b)}{n \cdot (n-2)} \tag{3.16}$$

where $(a+b)$ represents the number of agreement between cluster y_i and class y_j, and $({}_nC_2)$ denotes a total unordered pair of all elements in these sets. The RI ranges from 0 to 1 in which higher RI means the better clustering performance.

3.6.2 Tabular Datasets

In this subsection, we focus on 6 small datasets, Hill-Valley, Mushroom, IJCNN1, Satimage and German. We evaluate our proposed method by computing the clustering ACC and RI on these datasets using progressive framework along with five selected semi-supervised clustering methods (COP-K-means [20], LCVQE [29], CSP [103], CEVCLUS [104] & SSELM [100])

For those methods which marks use of pairwise constraints rather than label information, constraints transformation is needed such that positive label information is converted to pairwise constraints. For experimental parameters, we basically set the number of cluster (i.e., k) to the real number of class. To control the performance of our model, we empirically fix our sensitive parameters in this experiment. The setting is as follows, τ=3, ω=4, ε=9 and σ=0.8. All definitions of these variables can be reviewed in Table 3.2. In each scenario, the data are average performances of 3 runs with different sets of random labeling initialization. It is noted that parameter λ depends on sizes of datasets, we use parentheses to enclose the exact used value next to the dataset name, e.g., we divide Satimage into 5 subsets. Additionally, the clustering is completed by using our WAE as the optional clustering algorithm.

Table 3.4 demonstrates the performance (ACC & RI) of various semi-supervised clustering methods on 6 datasets by providing different portions of labeled data, we primarily select 5, 10, 20, 30 and 40% as the percentage of the labeled data. It is obvious that the progressive framework nearly outperforms 5 existing methods expect only Satimage in which SSELM leads the performance constantly by increasing ration

Table 3.4: Generalization Performances (Clustering Accuracy) on 6 Datasets

Datasets(λ)	Ratios	Ours	COP [20]	LCVQE [29]	CSP [103]	CEVCLUS [104]	SSELM [100]
	0.05	**0.8490**	0.5228	0.5184	0.5305	0.5429	0.5206
	0.10	**0.9860**	0.5545	0.5118	0.5212	0.5718	0.5462
Hill_Valley(1)	0.20	**0.9942**	0.5982	0.5091	0.5569	0.6130	0.5965
	0.30	**0.9942**	0.6328	0.5113	0.6119	0.6097	0.6469
	0.40	**0.9917**	0.6900	0.5061	0.5864	0.6394	0.6972
	0.05	**0.9728**	0.7953	0.7052	0.5181	0.8966	0.5060
	0.10	**0.9777**	0.7925	0.8024	0.5181	_	0.5300
Mushroom(5)	0.20	**0.9934**	0.9142	0.9143	0.5181	_	0.5854
	0.30	**0.9986**	0.9246	0.9230	0.5181	_	0.6373
	0.40	**0.9991**	0.9344	0.9346	_	_	0.6893
	0.05	**0.7845**	0.5249	0.5772	0.5003	0.5259	0.5230
	0.10	**0.8182**	0.5505	0.6567	0.5505	_	0.5230
IJCNN1(15)	0.20	**0.8266**	0.6848	0.7230	_	_	0.5998
	0.30	**0.8642**	0.7240	0.5846	_	_	0.6500
	0.40	**0.8882**	0.8876	0.5323	_	_	0.7000
	0.05	0.7578	0.6784	0.6749	0.2521	0.7186	**0.8428**
	0.10	0.7910	0.6684	0.6973	0.4767	0.7012	**0.8631**
Satimage(5)	0.20	0.8232	0.6646	0.7772	0.5071	_	**0.8764**
	0.30	0.8573	0.7294	0.7431	0.2832	_	**0.8797**
	0.40	0.8852	0.7453	0.7154	0.7069	_	**0.8962**
	0.05	0.4655	0.3664	0.3727	0.3335	_	**0.5257**
	0.10	0.4974	0.42806	0.3605	0.3335	_	**0.5601**
Protein(15)	0.20	0.5539	0.5186	0.3879	_	_	**0.5437**
	0.30	0.6162	0.6460	0.3584	_	_	**0.6357**
	0.40	0.6709	0.7496	0.3860	_	_	**0.7179**
	0.05	**0.9061**	0.8000	0.7996	0.8381	0.8516	0.6281
	0.10	**0.9269**	0.8114	0.8096	0.9083	0.8698	0.5950
German(15)	0.20	**0.9458**	0.8450	0.8378	0.9126	_	0.5289
	0.30	**0.9481**	0.8648	0.8677	0.9166	_	0.5372
	0.40	**0.9560**	0.8893	0.8731	0.9341	_	0.6033

_: Overtime issues which takes over 6 hours for computing

of labeled data. We conclude the following important observation:

1. In term of stability, the clustering ACC is improved by growing number of labels. Takes LCVQE as a counter-example, the performance **decrease** from 51.18 to 50.91% supporting by 10 and 20% labels respectively on Hill-Valley. However, our accuracy keeps improving when the portion of labeled data increases which means we could rely on such method in the real application.

Table 3.4: (continued) Generalization Performances (Rand Index) on 6 Datasets

Datasets(λ)	Ratios	Ours	COP [20]	LCVQE [29]	CSP [103]	CEVCLUS [104]	SSELM [100]
Hill_Valley(1)	0.05	**0.7434**	0.5008	0.5006	0.5017	0.5033	0.5004
	0.10	**0.9723**	0.5058	0.5000	0.5005	0.5099	0.5039
	0.20	**0.9885**	0.5191	0.4998	0.5101	0.5252	0.5182
	0.30	**0.9885**	0.5349	0.4999	0.5319	0.7508	0.5428
	0.40	**0.9836**	0.5719	0.4997	0.5307	0.5385	0.5774
Mushroom(5)	0.05	**0.9471**	0.6853	0.6299	0.5006	0.8146	0.5001
	0.10	**0.9564**	0.7199	0.6933	0.5006	_	0.5022
	0.20	**0.9868**	0.8430	0.8433	0.5006	_	0.5146
	0.30	**0.9973**	0.8606	0.8578	0.5006	_	0.5377
	0.40	**0.9983**	0.8775	0.8778	_	_	0.5716
IJCNN1(15)	0.05	**0.6618**	0.5012	0.5327	0.5000	0.5013	0.5012
	0.10	**0.7025**	0.5051	0.5719	0.5087	_	0.5012
	0.20	**0.7133**	0.5971	0.6029	_	_	0.5202
	0.30	**0.7653**	0.6236	0.5182	_	_	0.5453
	0.40	**0.8014**	0.8008	0.5021	_	_	0.5803
Satimage(5)	0.05	0.8570	0.8598	0.8600	0.2193	0.8545	**0.9122**
	0.10	0.8800	0.8523	0.8630	0.5666	0.8560	**0.9218**
	0.20	0.8996	0.8569	0.8805	0.6152	_	**0.9274**
	0.30	0.9167	0.8672	0.8684	0.2769	_	**0.9277**
	0.40	0.9324	0.8756	0.8596	0.8386	_	**0.9365**
Protein(15)	0.05	0.4795	0.3968	0.45777	0.3334	_	**0.5932**
	0.10	0.5173	0.5183	0.4988	0.3334	_	**0.6074**
	0.20	0.5705	0.5732	0.5264	_	_	**0.5999**
	0.30	0.6778	0.6462	0.5234	_	_	**0.6778**
	0.40	0.7111	0.7294	0.5355	_	_	**0.7037**
German(15)	0.05	**0.8298**	0.6799	0.6794	0.7541	0.7471	0.5327
	0.10	**0.8645**	0.6938	0.6916	0.8335	0.7734	0.5179
	0.20	**0.8974**	0.7379	0.7281	0.8405	_	0.5015
	0.30	**0.9016**	0.7661	0.7703	0.8471	_	0.5026
	0.40	**0.9159**	0.8030	0.7799	0.9341	_	0.5212

_: Overtime issues which takes over 6 hours for computing

2. Our framework is completely advantageous from the large dataset. We use the label information constraint as a supervisory in clustering task, such larger dataset constributes more labeled data. However another pairwise constraint models may consider it as a limitation, the result label assignment is being rejected due to the violation. Therefore, they can only freely enjoy the supervision when the amount of labeled data reaches certain levels. For instance, the

performances of COP-K-Means jumps hugely from 55% (using 10%) to 89% (using 40%) on IJCNN1. However, our method keeps regular improves without inner-jumping.

3. It is significant that our model can effectively cluster the dataset by using the extremely small amount of labeled data. By using 5% labeled data on Hill-Valley, performances are between 51.8 to 53.05% for all existing methods. However, the progressive framework reaches incredible 84.9% by providing the same portion of supervisory. Despite using more lablled data (40%), other methods only reaches maximum performance up to 69.72% (SSELM). For extreme small supervision, our framework could be regarded approximates to unsupervised model.

4. Although SSELM sightly outperforms ours on Satimage and Protein, the progressive framework obtains higher-level of generalized performance on different tabular datasets. For Hill-Valley and Mushromm, ours performance are 30+ % higher than SSELM constantly using different rations of labeled data. Also, we use fix the experimental setting in order to perform baseline performance. Improved Performances are shown Table 3.11.

3.6.3 Image Datasets

In this subsection, we concentrate on 7 well-known image datasets, they are USPS, COIL20, MNIST, CIFAR10, Scene15 (S15), Caltech101 (C101) and Caltech256 (C256). We carry out 4 sub-experiments in the following subsections depending on both the supervisory level of comparing methods and complexity of datasets. Note that clustering accuracy is the only metrics to access our performance over other approaches.

Classical Semi-Supervised Methods

In this section, less complex datasets will be tested. Apart from COIL20, we also utilize MNIST1K and USPS1K which are formed by randomly picking 1K samples from MNIST and USPS. We select total 4 conventional methods from Spectral Clustering family (CSP [103], IRSC [105] and SCSRR [106]) and factorization category (SNMF [107]). Significantly, our proposed framework is dominant in term of clustering accuracy over another 4 methods by using same percentage of labeled data. From Table 3.5, we chose 5, 7.5 and 10% of labeled data as 3 baselines.

Table 3.5: Comparison on USPS1K, COIL20, MNIST1K

Methods	**USPS1K**			**COIL20**			**MNIST1K**		
	5%	7.5%	10%	5%	7.5%	10%	5%	7.5%	10%
CSP [103]	0.825	0.845	0.857	0.777	0.813	0.815	0.758	0.774	0.790
IRSC [105]	0.765	0.827	0.843	0.772	0.779	0.789	0.664	0.681	0.747
SCSSR [106]	0.823	0.861	0.881	0.806	0.816	0.860	0.700	0.740	0.761
SNMF [107]	0.635	0.641	0.638	0.628	0.632	0.624	0.486	0.490	0.491
Ours (NFPF)	**0.836**	**0.883**	**0.940**	**0.824**	**0.855**	**0.875**	**0.759**	**0.840**	**0.871**

Firstly, the performance of our framework is beneficial with increasing amount of labeled data stably while another methods only take minor advantages of it. For example, the accuracy goes up from 83.6 till 94.0% on USPS1K using our method, it increases over 11%, while the growth of others are far away from ours. Secondly, we reach the comparable accuracy by using half of labeled data than existing methods. For COIL20, we obtain 82.4% using only 5% labeled data, it almost outperforms all other methods even compared with 10% of supervisory. To briefly conclude, it primary shows that our methods is capable in clustering task on simple image datasets.

Deep Semi-Supervised Embedded Clustering Methods

Deep semi-supervised embedded clustering (DSSEC) is one of the remarkable approach for such large dataset with partial supervision, plenty of researches are conducts by considering huge CIFAR10 datasets. Few DSSEC methods will be utilized in this section, they are SDEC [108], ClusterNet [109] and SCDMLGE [110].

Current DSSECs consider latent features instead of raw features, we will therefore utilize the DCNN structure to obtain the convolutional features as a pre-processing **without involving any pretrained weights**. A CNN architecture is replicated and modified by connecting the 256-dimensional dense layer before the classification. In this experiment, the architecture of VGG16 is used, we call it **Random-VGG** model.

Fig. 3.8 demonstrates the performance of our progressive framework is superior than 3 DSSEC models. It achieves 60+% accuracy with using only 1% labeled information, while other DSSECs reach at most 40% of accuracy (SCDMLGE with 10% labeled data). It significantly shows the ability our framework in handling the large dataset (CIFAR10). Most importantly, it proves that the advocated belief which uses

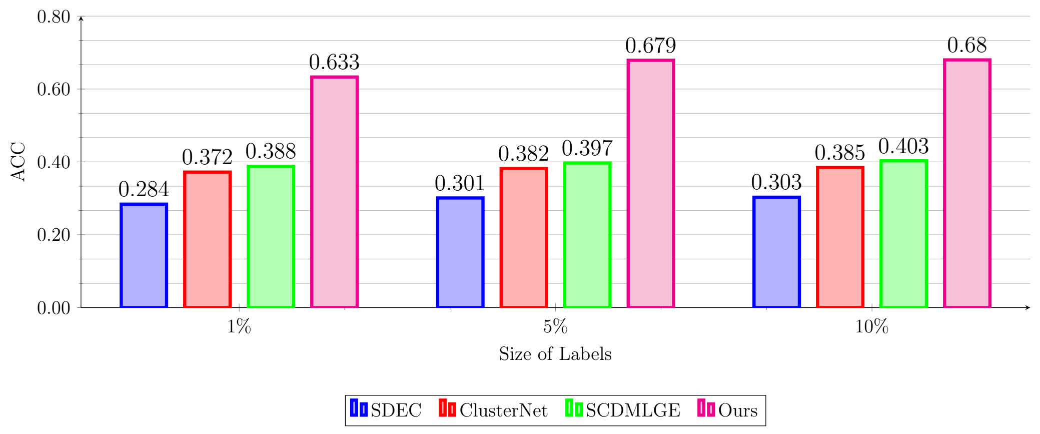


Figure 3.8: Comparison on CIFAR10 using DSSEC

Table 3.6: Comparison on Caltech101, Caltech256 and Scene15 using DCNN with semi-supervised manner

	Caltech101			**Caltech256**			**Scene15**		
Img / Method	1	5	9	1	6	12	3	15	30
AlexNet (Pretrained)	0.069	0.271	0.560	0.016	0.111	0.271	0.328	0.548	0.728
VGG16 (Pretrained)	0.255	0.647	0.742	0.035	0.264	0.365	0.476	0.717	0.775
ResNet50 (Pretrained)	0.443	0.689	0.810	0.227	0.632	**0.727**	0.431	0.541	0.733
VGG16 (Random Init.)	**0.663**	**0.713**	**0.818**	**0.370**	**0.654**	0.672	**0.800**	**0.861**	**0.879**

critical feature can be the potential approach of feature extraction in clustering task.

Semi-Supervised DCNN Methods

In this subsection, we compare our progressive framework with DCNN models in semi-supervised manner on more complex image datasets: Scene15 (S15), Caltech101 (C101) and Caltech256 (C256). Due to their outstanding results on the ILSVRC recognition competition, AlexNet, VGG16 and ResNet50 are selected as our competitors where they are all pretrained by the ImageNet dataset. Similar with previous experiment, the 256-dim convolutional features will be extracted using the Random-VGG. We utilize the extreme small amount of labeled data to perform clustering using DCNN such that these 1%, 5%, 10% labeled data can be considered as “training” dataset in DCNN model, while the rest forms the “testing” dataset.

Our proposed methods outperforms AlexNet and VGG16 models on all tested datasets. From Table 3.6, the proposed framework reaches 80.0% on S15 by using 1% labeled data, while AlexNet and VGG16 achieve 72.8% and 77.5% respectively by using 10 times labeled data than us. For the ResNet50, our model obtains comparable accuracies by using 10% labeled data, for example, the progressive framework achieves 81.8% on C101, and ResNet50 achieves 81.0%. However, the accuracies of our proposed method are completely higher than 3 DCNNs by using extreme tiny amount of labeled data, for example, it reaches 66.3% on C101 which is nearly 1.5 time higher than ResNet50 and 10 times than AlexNet. It is essential to highlight that those models also take the huge advantage from using the pretrained weight (ImgaeNet) as indirect supervisions to extract the key features on clustering (or classification) task, but our model use 256 dimensional features by training from scratch which can be considered as the fully unsupervised features extractor. Therefore, it shows the capability to handle the highly sophisticated datasets when only few labeled information is given.

Fully Supervised DCNN Methods

In order to test the supervised fashion of our methods, 3 scene-centric datasets (S15, C101 and C256) are selected with relative higher amount of labeled data following the widely accepted standard. We randomly draw 30 and 100 images from each classes to form the “training” dataset on S15 and C256. For C101, we pick 15 and 30 images from each classes. As same as other methods, all the remaining images will be the “testing” images.

From Table 3.7, we evaluate our framework with 20 state-of-the-art methods [23, 111–129] and 3 self-tested methods, these methods can be categorized into 6 major types, i.e., Traditional, Sparse Coding Based, Dictionaries Based, Feature Coding, Ensemble Network and DCNN-based. We obtain almost the highest accuracy on C101 and C256, and the comparable performance on S15. It demonstrates the capability of our method with using natural specific feature.

Our proposed framework achieve the highest performance on both C101 (86.4%) and C256 (70.1%). From Table 3.7, it is obvious that our method consistently leads the performance on two complex datasets when comparing with other methods except feature coding method. Hierarchical Matching [123] as a feature coding methods

Table 3.7: Comparision on Caltech101, Caltech256 and Scene15 using DCNN with full-supervised manner

	Caltech101		**Caltech256**	**Scene15**
Img/Class / Method	15	30	30	100
Traditional Methods				
NBNN [111]	0.650	0.704	0.268	-
Kernel Codebooks [112]	-	0.642	-	0.767
O2C Kernel [113]	-	0.655	-	0.888
LC-KSVD [114]	0.677	0.736	0.357	<u>0.929</u>
Sparse Coding Based Methods				
Linear SPM [115]	0.670	0.732	0.277	0.803
LLC [116]	0.654	0.734	0.412	0.892
SRC [117]	0.649	0.707	-	0.918
Dictionaries Based Methods				
K-SVD [118]	0.652	0.732	-	0.867
D-KSVD [119]	0.651	0.730	-	0.891
DLSI [120]	0.609	0.703	-	0.925
COPAR [121]	0.623	0.718	-	**0.955**
Features Coding Methods				
Local Pyramidal [122]	0.715	0.801	0.447	-
Hierarchical Matching [123]	-	<u>0.825</u>	<u>0.480</u>	-
Visual Word Ambiguity [124]	-	0.641	0.272	0.767
Feature Pooling [125]	-	0.718	-	0.806
SPF [126]	0.564	0.646	-	0.814
Soft Assignment [127]	-	0.765	-	0.822
Ensemble Network Methods				
Multi-layer ELM [23]	0.717	0.776	0.382	0.824
Mid Level Feature Coding [128]	-	0.757	-	0.843
DHVF Coding [129]	<u>0.721</u>	0.797	0.417	0.854
DCNN				
AlexNet Training from scratch	-	0.603	0.242	0.472
VGG16 Training from scratch	-	0.344	0.095	0.448
ResNet50 Training from scratch	-	0.454	0.147	0.421
Progressive Framework with Random-VGG features	**0.730**	**0.864**	**0.701**	0.910

achieves relative close performance (82.5%) to ours on C101. However, our approach (70.1%) extremely outperforms all others when handling the 257 classes (C256) while the second best is only 48.0% [123]. Although our performance slightly behinds others on S15, it still obtains the acceptable accuracy. The reasons is that the task complexity of S15 is relatively uncomplicated and straightforward than other two datasets. State-off-the-art methods, like SRC [117] and LC-KSVD [114], can simply scores 91.8% to 92.9%, it proves that the feature learning task is probably direct. Therefore, our proposed method may not be the best solution in such situation. In short, our methods reaches the reasonable accuracy on S15, while we are completely the best solution on C101 and C256. It further proves the strong capability and effectiveness of using our approach that learn specific features rather that informative features.

3.7 Analysis of Progressive Framework

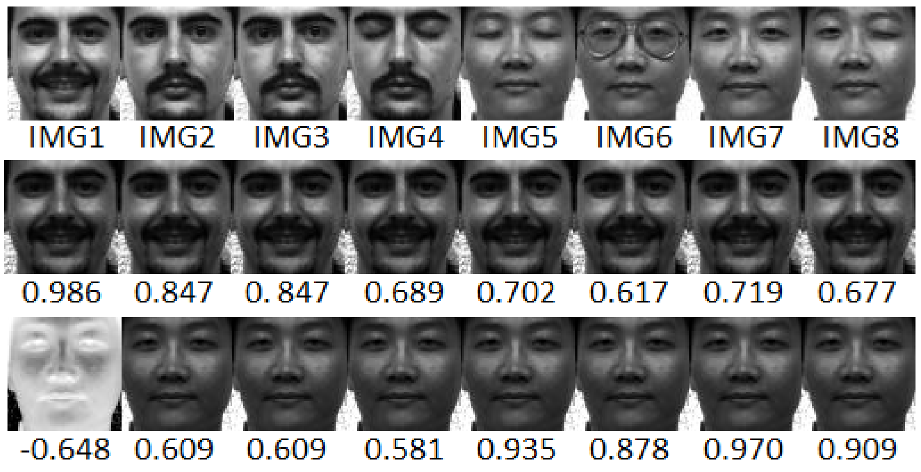
The purpose of this section is to analysis the proposed progressive framework. First of all, 4 image datasets (i.e., MNIST1K, COIL20, Yale-Face, Olivetti-face) are involved to show the reconstruction performance of WAE. Secondly, 4 tubular datasets (i.e., Hill-Valley, German, Satimage and USPS) are used to test the sensitivity of involved hyperparameter of our proposed framework. Table 3.8 generally defines the categories of above datasets depends on data dimensions and sizes.

Table 3.8: Dataset Properties

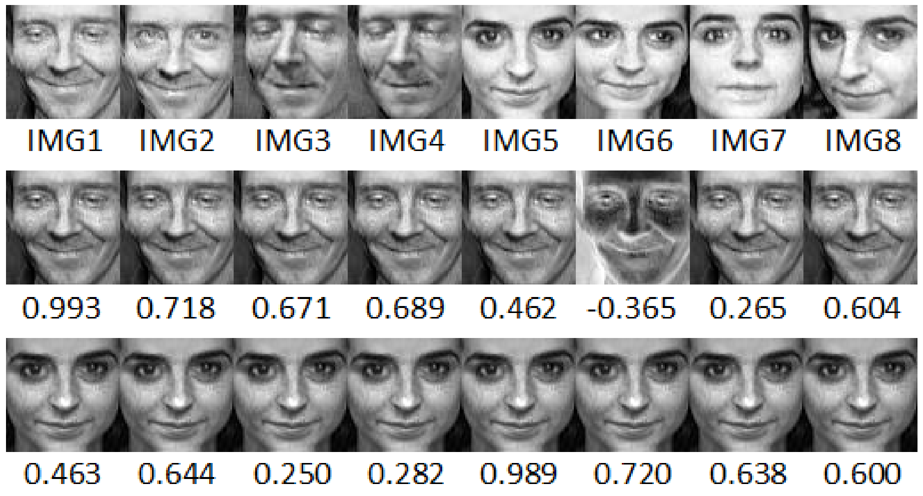
Dataset	**Dimension Category**	**Size Category**
Hill-Valley	High: 100	Small: 1212
German	Low: 4	Large: 3025
Satimage	Low: 36	Large: 4435
USPS	High: 256	Large: 9298

3.7.1 Reconstruction Performance of WAE

In order to show the reconstruction performace of WAE, we conducted two experiments by using 4 image datasets in this section. We first train the WAE by only one randomly selected training image, then reconstruct all images. Fig. 3.9 shows original images (first row), reconstructed images generated by two WAE which are trained by IMG1 (second row) and IMG5 (third row). It can be observed that the training image can yield the highest self-similarity. The image that is similar with the training



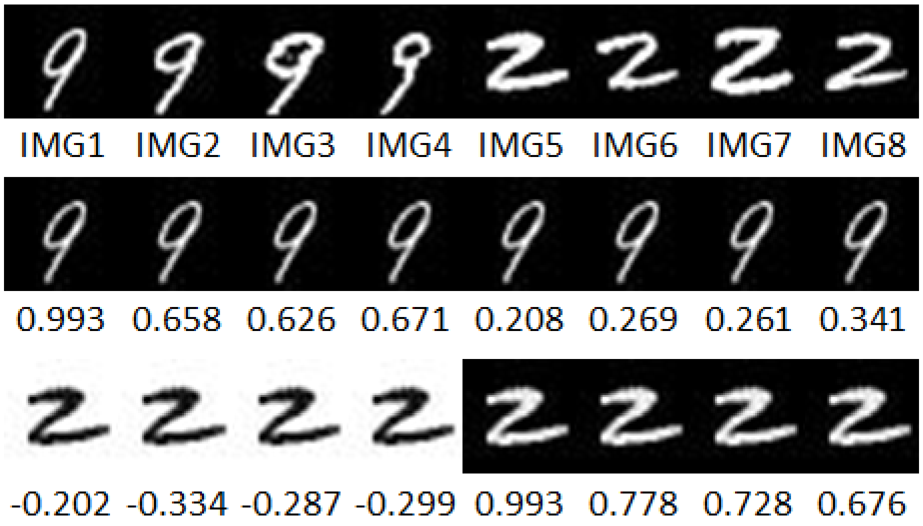
(a) Yale-Face



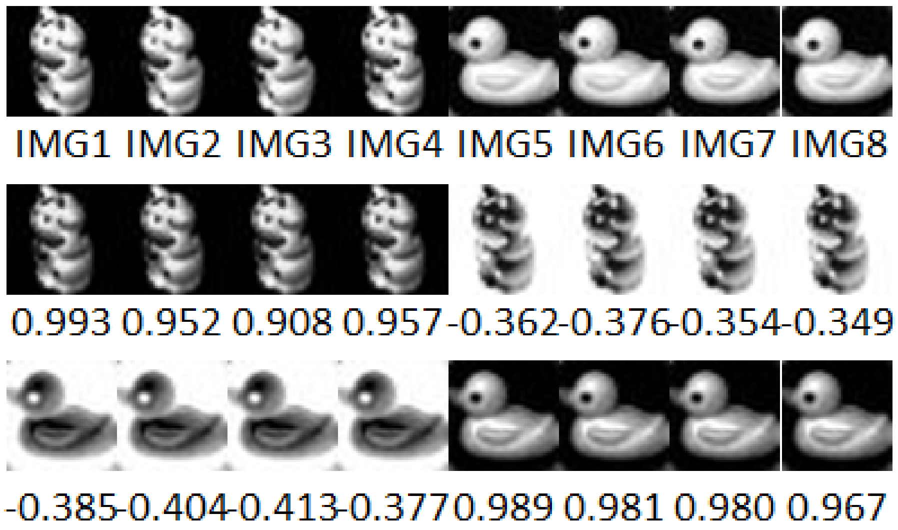
(b) Olivetti-Faces

Figure 3.9: Reconstruction performances of WAE trained by two separated samples. Corresponding self-similarity are shown under each image. Note that IMG1 to IMG4 are from one class while IMG5 to IMG8 are from another class.

image can produce the second best self-similarity. For example, IMG1 and IMG2 of Fig. 3.9(b) are similar with each other since they have opened eyes. Therefore, the trained WAE can well-reconstruct both IMG1 and IMG2. On the other hand, the images which are dissimilar with IMG1 cannot be well-reconstructed by that trained WAE. For example, the IMG6 are extremely dissimilar with IMG1 such that the the self-similarity is -0.365. Therefore, the reconstruction performance of WAE is not generalized to all images but only specific to few images.



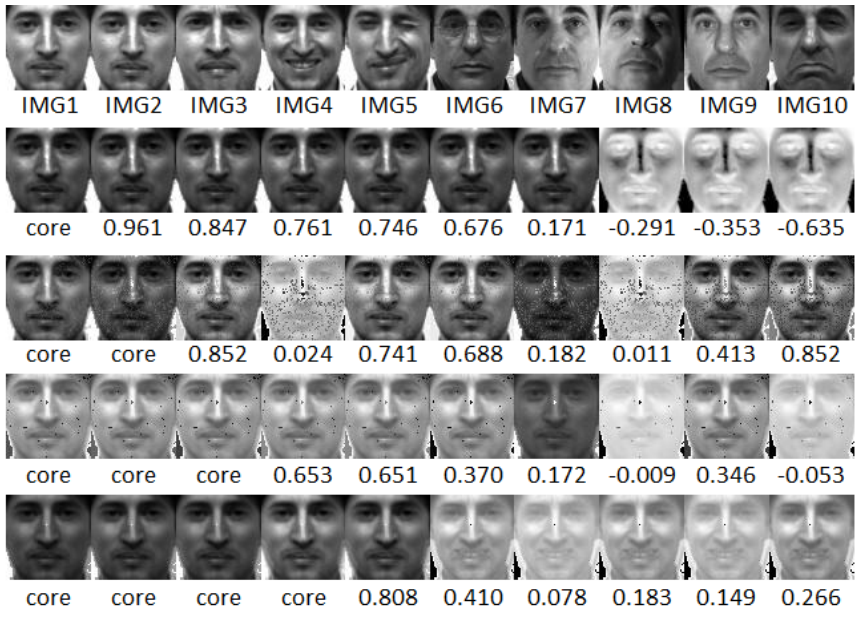
(a) MNIST



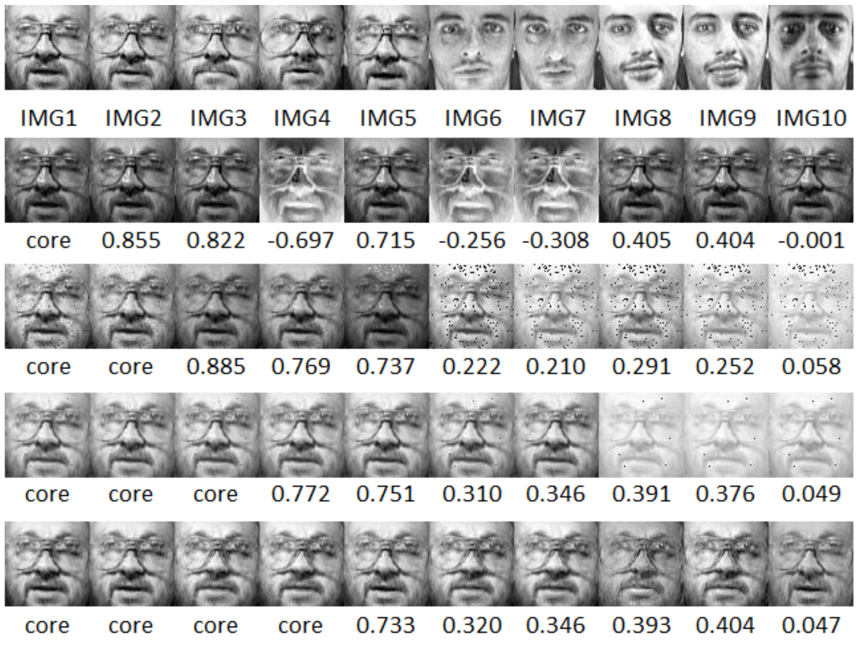
(b) COIL20

Figure 3.9: (continued) Reconstruction performances of WAE trained by two separated samples

After showing the performance of WAE trained by two separated samples, We now visualize the reconstruction performance of WAE trained by continuous samples. We first train the WAE by using only one training image (i.e., IMG1 in Fig. 3.10), then we reconstruct all images (second row). The reconstructed image that has the highest self-similarity (i.e., IMG2) is grouped with previous training image. Then, we train the WAE by using these two images and reconstruct all images again (third row). This

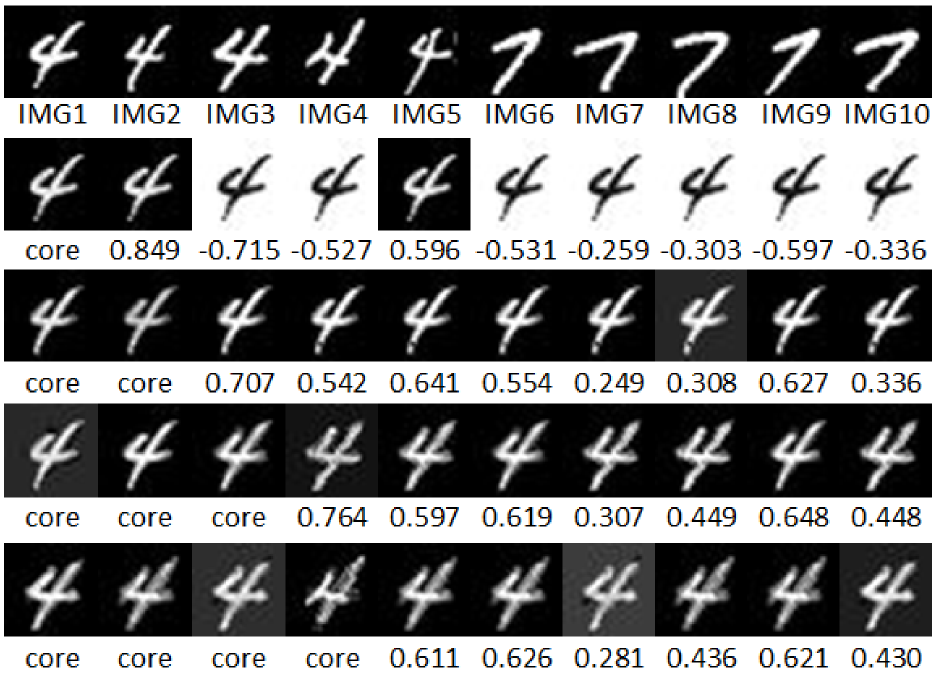

IMG1
IMG2
IMG3
IMG4
IMG5
IMG6
IMG7
IMG8
IMG9
IMG10
core 0.961 0.847 0.761 0.746 0.676 0.171 -0.291 -0.353 -0.635
core core 0.852 0.024 0.741 0.688 0.182 0.011 0.413 0.852
core core core 0.653 0.651 0.370 0.172 -0.009 0.346 -0.053
core core core core 0.808 0.410 0.078 0.183 0.149 0.266

(c) Yale-Face

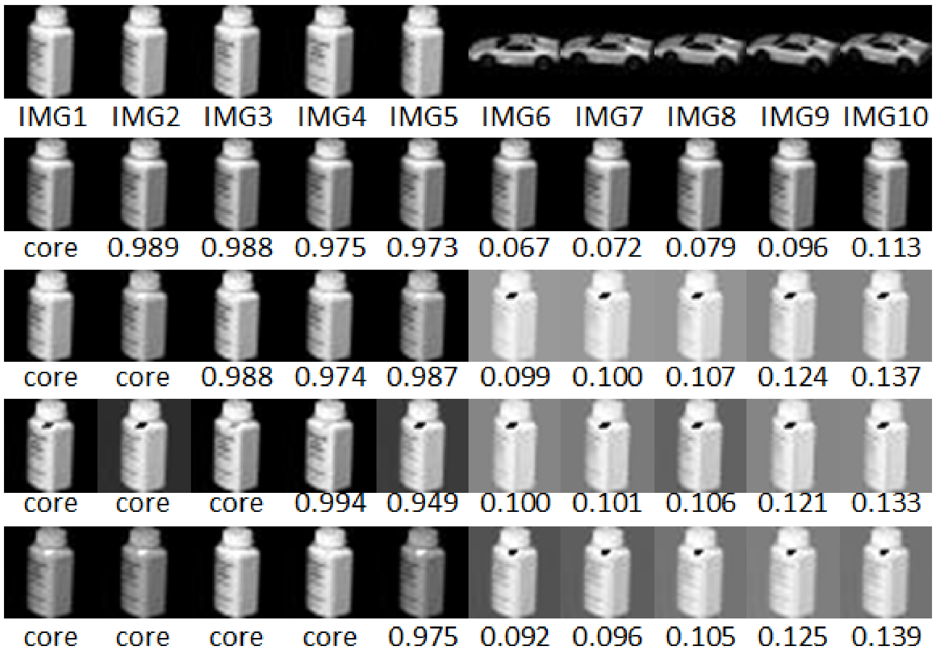

IMG1
IMG2
IMG3
IMG4
IMG5
IMG6
IMG7
IMG8
IMG9
IMG10
core 0.855 0.822 -0.697 0.715 -0.256 -0.308 0.405 0.404 -0.001
core core 0.885 0.769 0.737 0.222 0.210 0.291 0.252 0.058
core core core 0.772 0.751 0.310 0.346 0.391 0.376 0.049
core core core core 0.733 0.320 0.346 0.393 0.404 0.047

(d) Olivetti-Faces

Figure 3.10: Reconstruction performances of WAE trained by continuously samples. Images that are marked as “core” represent the training data for each WAE in each iteration. Note that IMG1 to IMG5 are from one class while IMG6 to IMG10 are from another class.

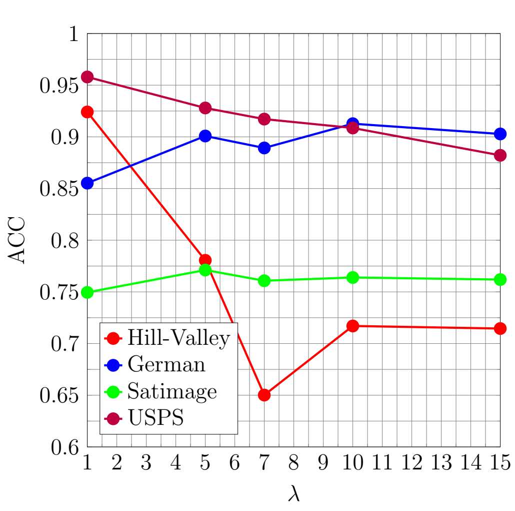

IMG1 IMG2 IMG3 IMG4 IMG5 IMG6 IMG7 IMG8 IMG9 IMG10
core 0.849 -0.715 -0.527 0.596 -0.531 -0.259 -0.303 -0.597 -0.336
core core 0.707 0.542 0.641 0.554 0.249 0.308 0.627 0.336
core core core 0.764 0.597 0.619 0.307 0.449 0.648 0.448
core core core core 0.611 0.626 0.281 0.436 0.621 0.430

(a) MNIST

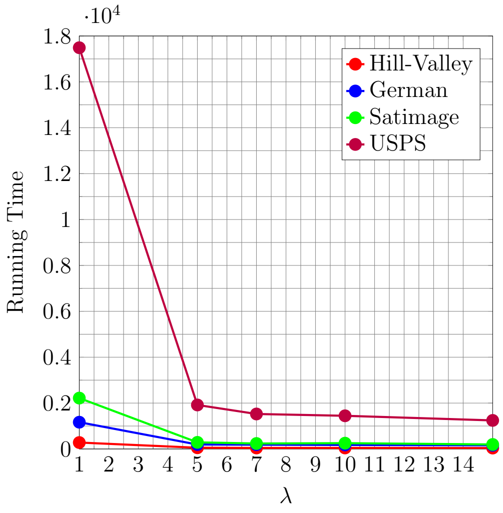

IMG1 IMG2 IMG3 IMG4 IMG5 IMG6 IMG7 IMG8 IMG9 IMG10
core 0.989 0.988 0.975 0.973 0.067 0.072 0.079 0.096 0.113
core core 0.988 0.974 0.987 0.099 0.100 0.107 0.124 0.137
core core core 0.994 0.949 0.100 0.101 0.106 0.121 0.133
core core core core 0.975 0.092 0.096 0.105 0.125 0.139

(b) COIL20

Figure 3.10: (continued) Reconstruction performances of WAE trained by continuously samples



(c) Clustering Accuracy



(d) Running Time

Figure 3.11: Batch Numbers Analysis

process is repeated for ω times (3.7.2). The reconstruction performance is generalizing when the number of labeled images increases. For example, the average self-similaity of second row of Fig. 3.11(a) is -0.405 while that of final row is 0.479. It reals that the learned feature is no longer weak and specific when more training samples is used. To ensure the weak reconstruction performance, we set hyperparameter ω to control the number of iterations for adding the labeled samples.

3.7.2 Sensitivity of Parameter ω & λ

Ahead of jointly testing λ and ω, we first conduct an independent experiment on parameter λ. Experiment is conducted by using different values of λ with fixed ω=4 in order to primitively conclude the rules for adjusting λ. We conclude two observations from Fig. 3.11. Firstly, λ partly depends on data size. For the larger dataset, the proposed framework still performs reasonable performance. For instance, USPS and German keeps achieving 90+% clustering accuracy when λ=10. It obtains stable performance on low-dimensional dataset, Satimage by achieving around 76% with various λ. Secondly, batch separation reduces computational cost while still offers the acceptable result. It decreases only around 5% from 95.79% (λ=1) to 90.86% (λ=10), but computational time reduces by 12 times. Therefore, the principle of adjusting of λ is summarized below:

Rule 1. *Larger λ can be selected on large and low-dimensional datasets, but it certainly deteriorates the clustering performance on small datasets (no matter the dimension):*

We then further study impacts of using various combinations of ω and λ. ω is represents how many members are required to be searched by cores in phase 1. (See Section 3.3.3). Firstly, ω is implicitly regulated by used label rations (i.e., $r \in [0, 1]$). Since one core will search ω data, it yield the following:

Rule 2. *Given that label rations r, the ω is regulated as*

$$(\omega + 1) \cdot r \leq 1$$

From Section 3.3.3, the core set is incremented sequentially. Our mechanism works perfectly when data from the same class are clustered gradually. Thus, small ω could maintain stable performance conservatively. In other words, the high amount of ω will destruct the entire belief of the proposed framework.

Rule 3. *ω is controlled to be limited number in order to maintain the conservative mechanism by narrowing down the search area.*

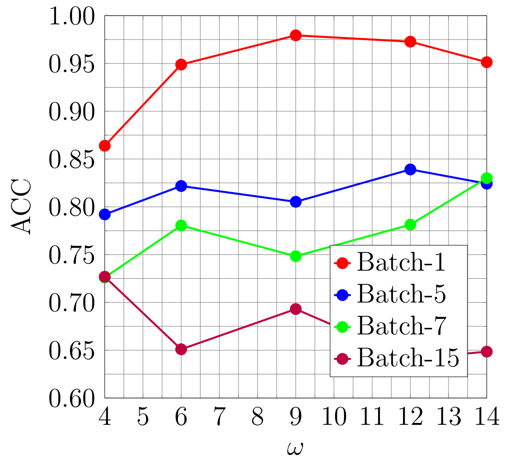
According to our study, λ and ω can generally be chosen from [1, 5, 7, 10, 15] and [4, 6, 9, 12, 14] respectively. Other parameters are kept as same as early experiments. 5% labeled data is used. Two observations can be stated from Fig. 3.12:

1. For higher λ, the performance is degraded when ω increases gradually. For example, the accuracy decreases with increasing ω on Satimage when λ=10. Similar trends can be observed on USPS λ=10 and 15.

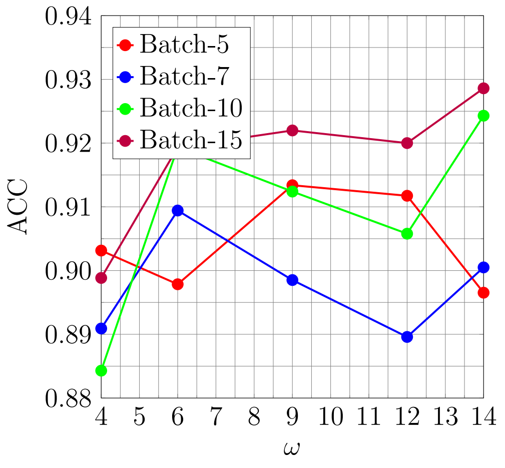
2. For smaller λ, ω can be selected openly. Interestingly, the performance is improved by applying increasing number of ω on Hill-Valley when λ=1. Also, it impressively achieves the highest accuracy, 92.86% by using $\omega = 14$ on German when λ=15.

Rule 4. *For higher λ, ω is strictly be a limited number. For lower λ, ω can be selected openly not smaller value is suggested.*

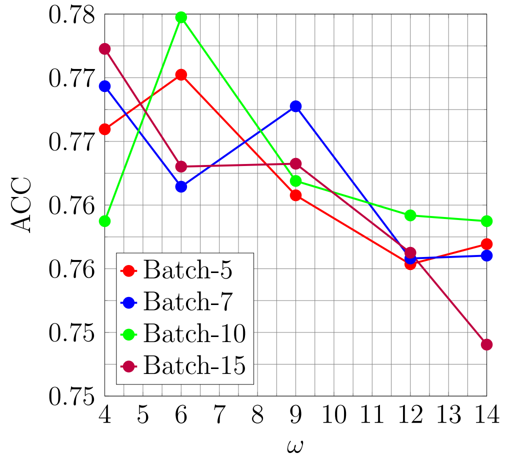
Although the selection of λ and ω is not generalized, the above 4 rules offer a clean guidance to adjust optimal combination pair. Therefore, the computational time can be hugely reduced on particularly handling larger datasets while the performance is kept.



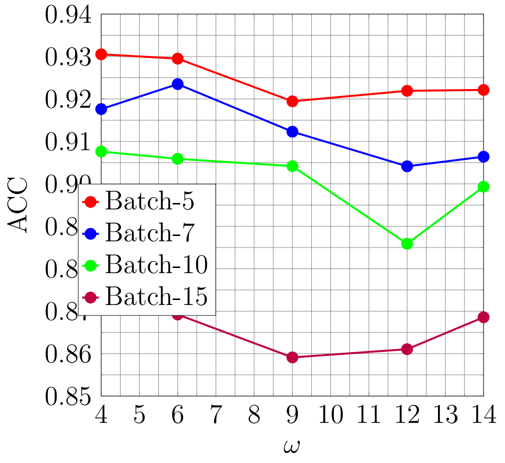
(a) Hill-Valley



(b) German



(c) Satimage

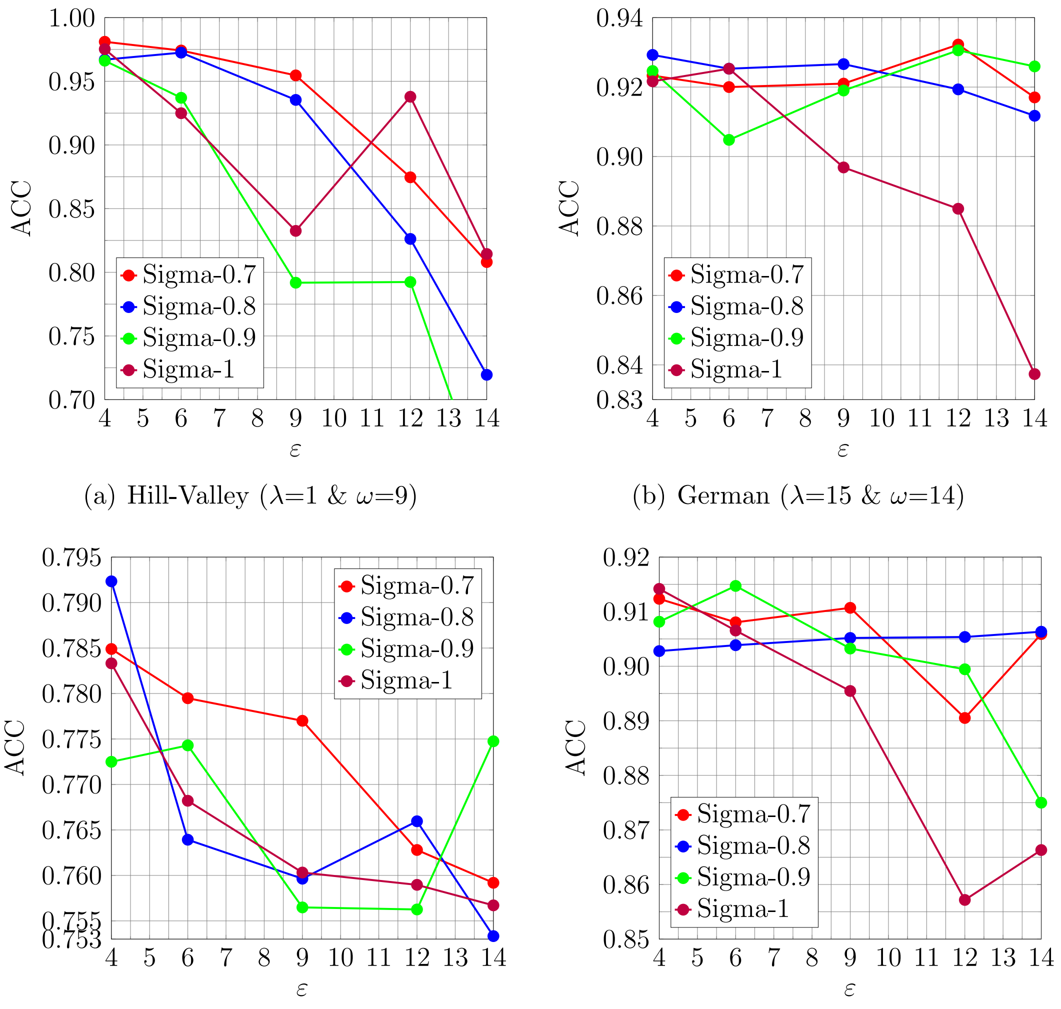


(d) USPS

Figure 3.12: Impact of of ω and λ (τ=3, ϵ=9, σ=0.8 with 5% labelled data)

3.7.3 Restrictiveness of Parameter ε & σ

We evaluate the impact on different combinations of ε & σ. Similar to ω, ε indicates how many t-mem are required to be searched by t-core in phase 2 (See Section 3.3.4). ε could be regarded as an indicator showing how far the searching space is while σ could be treated as a cut-off factor. The higher the value of ε and σ, the stricter the algorithm which ideally leads lower mis-clustering rates. Conversely, over-conservative will eliminates all found data. Thus, we investigate how the restrictiveness affect the final performances. 4 datasets are continually utilized while we select the optimal pairs of λ and ω from Section 3.7.2. For example, we use λ=10 with ω=9 on USPS.



(a) Hill-Valley (λ=1 & ω=9) (b) German (λ=15 & ω=14)

(c) Satimage (λ=10 & ω=6) (d) USPS (λ=10 & ω=4)

Figure 3.13: Impact of ε and σ (τ=3 with 5% labelled data used)

We conclude experimental result below:

1. Horizontal Comparison: In Fig. 3.13, curves generally goes down by increasing ε on every datasets apparently. Precisely, ε=4 gives highest accuracy consistently in Table 3.9

2. Vertical Observation: Four curves of σ on every dataset are in the consist order, that is, the accuracy of σ=0.7>0.8>0.9>1. We can see that setting σ=0.7 is not a bad choice. The greater value may over-regulates the framework, it eliminates found data such that labeled (or p-labeled) data are insufficient to cluster remaining data.

Table 3.9: Average Performance of σ to varied ε

ε	**Hill-Valley**	**German**	**Satimage**	**USPS**
4	**0.972**	**0.925**	**0.783**	**0.909**
6	0.952	0.919	0.771	0.908
9	0.879	0.916	0.763	0.904
12	0.858	0.917	0.761	0.888
14	0.741	0.898	0.761	0.888

Table 3.10: Average Performance of ε to varied σ

σ	**Hill-Valley**	**German**	**Satimage**	**USPS**
0.7	**0.918**	**0.923**	**0.773**	**0.905**
0.8	0.884	0.922	0.767	0.905
0.9	0.822	0.921	0.767	0.900
1.0	0.897	0.893	0.766	0.888

3. Exceptional Cases: However, we observe that higher ε sometimes may offers better result from Fig. 3.13. It reaches the highest accuracy (90.63%) on USPS when ε=14 and σ=0.8. That is to say, smaller ε limits the searching space. Few new unlabeled data could be clustered in each cycle, resulting in inadequate labeled samples in final phase.

In summary, the average accuracy performance of using smaller value of ε and σ is stably superior than using larger. However, ε can be set higher on large datasets.

3.8 Discussion and Conclusion

In Table 3.11, we present our best performances over 8 datasets comparing with 3 supervised methods: ELM [22], SVM [130] and SNN [101]. We randomly select 1 sample per class, 1 and 5% as labeled data for our progressive framework while 70% data are picked as training data for supervised methods. By using only extreme small amount of labeled data, our method even obtains higher performance than supervised methods (eg. Hill-Valley & USPS). More importantly, our framework achieves acceptable results by using only 1 sample per class which could be regarded as “unsupervised learning”. To conclude, we propose a new clustering framework

Table 3.11: Best Performances on 8 Datasets

	1 / class	**1%**	**5%**		**70%**	
	Ours (NFPF)			ELM	SVM	SNN
Hill-Valley	0.8358	0.8457	0.9810	0.6640	0.5130	0.9091
Mushroom	0.9175	0.9839	0.9941	0.9993	0.9998	0.9977
IJCNN	0.7037	0.8036	0.8763	0.9848	0.9453	0.9095
Satimage	0.7639	0.7619	0.7982	0.8855	0.8670	0.8465
Protein	0.4170	0.5213	0.5474	0.6647	0.6325	0.6685
German	0.7841	0.8456	0.9322	0.9558	0.9404	0.9535
USPS	0.8079	0.9126	0.9305	0.9229	0.9495	0.8618
COIL20	0.6986	0.7055	0.8236	0.9602	0.9827	0.5133

that learns discriminative features of high-dimensional data using WAE. We employ the WAE to perform clustering by comparing the reconstructed and original data. As a consequence, we gradually enlarge the labeled dataset by adding the confident unlabeled data via two phases. Experimental result shows the progressive framework is capable to solve the clustering problem.

Chapter 4

Seeded Progressive Framework for Weakly-supervised Semantic Segmentation

4.1 Overview

Weakly-supervised semantic segmentation with only image-level labels is an essential application since it reduces considerable human efforts to fully annotate images. Most of the recent state-of-the-art methods consider discriminative regions or seeds which are generated by using the Classification Activation Maps (CAM) method. As seeds could be considered as segmentation knowledge, increasing the number of seeds will naturally improve the performance of segmentation methods. Therefore, we formulate the weakly-supervised segmentation challenge as a seed exploration task using clustering learning. In this chapter, we propose a seeded progressive framework including autoencoders to explore numerous and accurate seeds to further improve the performance of segmentation methods. Our explored seeds can replace the initial seeds, and thus they can be integrated into current weakly-supervised semantic segmentation methods for performance improvement. We show experimentally that the explored seeds lead to better model training. As a result, we obtain performance improvement from corresponding counterparts to reach 68.1% and 69.2% mIoU on PASCAL VOC 2012 validation and test dataset respectively, which are the current state-of-the-art performances.

4.2 Introduction

Image semantic segmentation is a task of pixel-level classification for a given image and has attracted much attention in the field of computer vision. With the development of Deep Convolutional Neural Network (DCNN), the performance of semantic segmentation has been considerably improved in a fully supervised manner [49, 51, 131–136]. Nevertheless, these DCNN models require a large-scale dataset consisting of large amount of accurate pixel-level annotations, which are laborious and burdensome to obtain. To this end, the weakly-supervised semantic segmentation (WSSS) is proposed which intuitively makes use of different forms of weak supervision for training the segmentation model, such as bounding boxes [54, 55], scribbles [56–58], and points [59]. Among these different forms of weak supervision, the image-level labels is the most appealing and readily obtainable.

Image-level labels only indicate the presence of objects, but it does not provide precise spatial information. To primarily explore few sparse location cues, the Classification Activation Maps (CAM) [60] method can be seen as a cornerstone in this particular area of the weakly-supervised segmentation using image-level labels [1,3–5,82,84,85,87,137–139]. In general, related methods can be roughly divided into two streams, namely implicit and explicit seed learning methods. The former usually treats seeds as a whole, which is called the discriminative regions (i.e., a collection of seeds). With an adversarial erasing strategy, it iteratively hides and erases the regions that have been found. Existing works [82, 84, 85] commonly use a single classification network to explore the sub-discriminative regions by image-level erasing mechanism. For example, HaS [82] randomly hid a region from the given training image, as it encouraged a network to search for other discriminative regions via CAM method. Besides, research works [1, 137, 138] generally use two classifiers. One of them will be designed to identify the most discriminative region, while the other is proposed to find complementary parts. These methods attempt to explore and expand the discriminative region on a feature map using a network.

In contrast, the seeds is regarded as a single unit which will be utilized in a model loss function in the explicit seed learning. With the involved stimulated pixels being termed as “seeds” above, recent methods have been proposed to utilize these seeds in the loss functions of deep convolutional neural networks for improving the pixel-level classification performance. Kolesnikov et al [3] regard seeds as a static supervision to propose their seeding loss to train a segmentation network. They named this process as seed, expand and constrain (SEC) [3]. Ref. [4] proposes a strategy of deep seeded region growing (DSRG) by dynamic seed supervision. On top of [3], it integrated a classical Seeded Region Growing (SRG) into the deep segmentation network that generated accurate and complete pixel-level labels given original seeds. With the help of co-segmentation, CIAN [5] defines the model loss function considering cross image learning. It shared supplementary information among related images to jointly train the seeding loss or completion loss as in [5]. Similarly, seeds are explicitly used in [87, 139]. These methods gradually push the state-of-the-art performance on semantic segmentation in recent years with limited prior knowledge.

However, seeds are always sparse and incomplete, which are far away from the intact segmentation mask. In other words, if more prior knowledge (accurate seeds) are provided at the initial stage, the performance of these methods could be improved. Therefore, a motivation naturally comes: Can we design a clustering algorithm to explore more accurate seeds to enhance the initial conditions for performance improvement of these semantic segmentation methods? In detail, could we cluster the unlabeled "similar" pixel-level samples into one class based on the similarity rate of their reconstructed samples that have been generated by a "weak" autoencoder?

Autoencoder has been an important research topic for a long time. Consisting of encoder and decoder, the conventional autoencoder [11,65,140] projects raw samples to latent feature space using the encoder. Then, the sample is decompressed and reconstructed to original representation space via the decoder. As an unsupervised model, the autoencoder learns the low dimensional data representation with hidden neurons, and aims at minimizing the reconstruction discrepancy. For the reconstruction capability's improvement, more layers have been added to the deep autoencoder to achieve the maximum representation learning. However, here we would like to compare the similarity rate between the input sample and its reconstructed samples. We try to use an autoencoder with minimum representation learning capability only extract common shared features among samples. Rather than achieving the generalization reconstruction performance, our proposed autoencoder can be trained to only reconstruct the input data itself.

In this chapter, we formulate the weakly-supervised semantic segmentation task as a semi-supervised seed clustering problem. We propose to first collect the most confident initial seeds using the CAM method. In semi-supervised learning perspective, such initial seeds can then be chosen as labeled samples and the remaining as unlabeled samples. The aim of the proposed approach is to increase the number of labeled samples, the resulted seeds is called explored seeds. These seeds includes both initial and newly clustered seeds, and are used to train the segmentation models in the same way as the initial seeds do to improve the segmentation performance.

As illustrated in the schematic diagram (Fig. 4.1), the proposed method firstly obtain the initial seeds on deep convolutional space. Then, the single layer feedforward network (SLFN) based autoencoder (AE) is utilized to group one unlabeled sample with one labeled sample. To perform clustering, the seeded progressive framework is proposed to guide the clustering mechanism while mentioned AEs are involved. These labeled samples can be used to replace the initial seeds to improve the performance of the weakly-supervised semantic networks (i.e., [3–5]).

We conduct thorough experiments on PASCAL VOC 2012 benchmark [141] to demonstrate the explored seeds can boost the segmentation performance from existing weakly-supervised semantic segmentation methods using the original seeds only. In summary, we make three main contributions in the following:

- Feasible for any weakly-supervised segmentation methods. The seeded progressive framework is proposed to explore additional accurate seeds from the initial seeds without any additional knowledge. In other words, our approach is used as the knowledge augmentation method for boosting any weakly-supervised semantic segmentation methods, without any additional knowledge involved in the learning process.

- Simplest autoencoder. We propose to use the simplest autoencoder to group one unlabeled sample with one labeled sample, called weak autoencoder (WAE). Unlike the current machine learning principle, it is designed to ensure the weakest but not universal generalization performance. With numerous but weak autoencoders for clustering learning, we provide another way of thinking to the semi-supervised/unsupervised learning problems.

- High performance. The explored seeds are adoptable for current weakly-supervised semantic segmentation networks, i.e., SEC [3], DSRG [4] and CIAN [5]. We obtain around 1.5% to 3.9% performance improvement of mIoU on PASCAL VOC 2012 validation and test dataset. The mIoU of our framework are 68.1% and 69.2% respectively, which are the state-of-the-art performance. [1]

The remainder of this chapter is structured as follows. Preliminaries are briefly introduced in Section 4.3. In Sections 4.4, detailed procedures of our proposed framework are described. Section 4.5 provides experimental performances and analysis of our algorithm. Section 4.6 presents our conclusion and future work.

[1]The source code is available for reviewers and will be made public after publication http://www.yiminyang.com/weakly_supervised.html

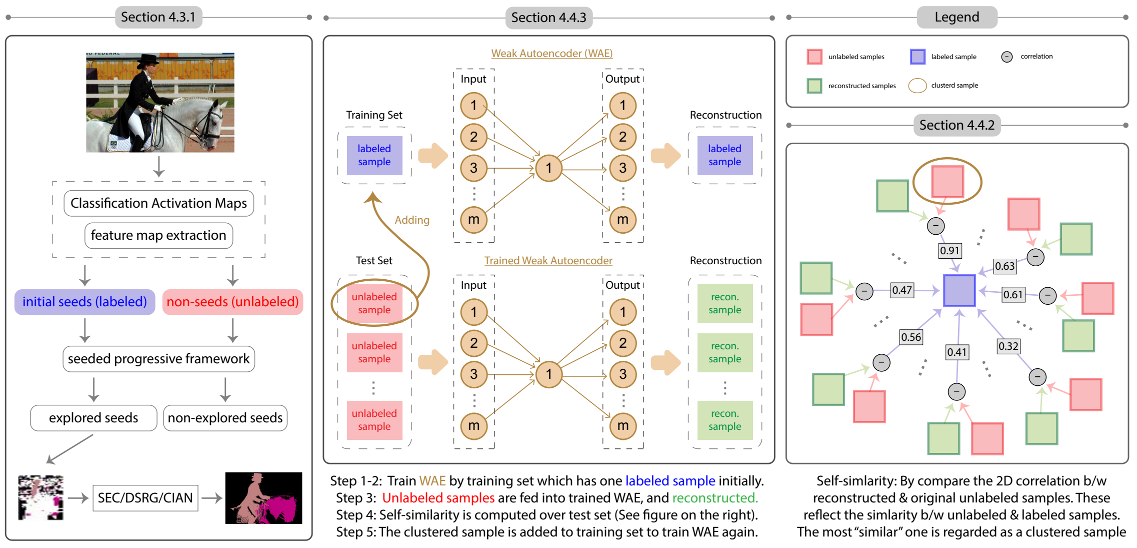

Section 4.3.1
Classification Activation Maps
feature map extraction
initial seeds (labeled)
non-seeds (unlabeled)
seeded progressive framework
explored seeds
non-explored seeds
SEC/DSRG/CIAN
Section 4.4.3
Weak Autoencoder (WAE)
Training Set
labeled sample
Input
Output
1
2
3
m
Reconstruction
labeled sample
Adding
Test Set
unlabeled sample
Trained Weak Autoencoder
recon. sample
Step 1-2: Train WAE by training set which has one labeled sample initially.
Step 3: Unlabeled samples are fed into trained WAE, and reconstructed.
Step 4: Self-similarity is computed over test set (See figure on the right).
Step 5: The clustered sample is added to training set to train WAE again.
Legend
unlabeled samples
labeled sample
correlation
reconstructed samples
clusterd sample
Section 4.4.2
0.91
0.63
0.47
0.61
0.56
0.41
0.32
Self-simlarity: By compare the 2D correlation b/w reconstructed & original unlabeled samples. These reflect the simlarity b/w unlabeled & labeled samples. The most "similar" one is regarded as a clustered sample

Figure 4.1: Overview of the chapter: The initial seeds is obtained by CAM and feature map extraction (left). By using seeded progressive framework, the explored seeds is generated and can be used to trained segmentation models ([3–5]). Here, we show the process of grouping one unlabeled sample with the labeled sample (Section 4.4.3).

4.3 Problem Analysis and Preliminaries

Given an arbitrary image-level label (tag) for a certain training image, we discover discriminative regions through the CAM method [60]. In spite of the insufficient number of seeds, we can use them as labeled samples for the semi-supervised seed clustering (SSSC) task. Hence, in this chapter, we formulate the weakly-supervised semantic segmentation as a SSSC problem. As mentioned previously, we aim at increasing the number of labeled samples. The explored seeds can be used to replace the initial seeds in the related methods (e.g., [3–5]) to improve the segmentation performances. As the generated explored seeds only requires knowledge of the initial seeds, no additional strong supervision are needed in the chapter. In other words, with the provided initial seeds of an image, the explored seeds are obtained under a purely unsupervised learning condition. Therefore, the proposed method could be feasible to strengthen the knowledge of initial seeds used in any semantic segmentation methods. We briefly describe how to obtain the initial seeds through CAM, and then illustrate the clustering preparation, specifically on how to construct searching space by convolutional feature map (Section 4.3.1). Since SLFN based autoencoder will be implemented for the SSSC task, we discuss related concepts and definitions mathematically in Section 4.3.2.

4.3.1 Seed as labeled samples, convolutional feature map as clustering space

To obtain seed samples, we follow [3–5] to use the CAM method. In particular, a global average pooling (GAP) layer is added right after the final convolutional layer from a CNN model, named as *Conv7* (e.g., VGG16 [142]). Through backpropagation, such model is iteratively trained in an end-to-end manner. The classification layer is added on top of *Conv7* to obtain 2-D heatmap, where a k-classes probability vector can be found in every single position. After that, thresholding is employed on the heatmap to obtain the 20% most significant confident units as the initial seeds. We refer readers to [3–5] for details. Now, we need to establish a clustering space. Given a raw feature representation (RGB channel of a colored image), we encounter two main difficulties, i) high inner-class variance on background object and, ii) weak feature representation for classification, e.g., two pure white pixels (255, 255, 255) may come from class "person" (sweater) or "dog" (doghair) respectively. Since the convolution model extracts semantic and spatial information, we employ the DeepLab model [50] to extract convolutional features. In practice, after the model training, the classification layer is replaced by three fully-connected layers, named *fc7, fc8* and *fc9*. The clustering space used in the proposed method is formed by extracting the feature map on the *fc7* layer.

4.3.2 Single layer feedforward network based autoencoder

As aforementioned, let X be an $n \times d$ matrix, where each row represents an instance of data with d-dimensional convolutional features. In a general semi-supervised learning, $\mathrm{X}^{\mathrm{L}} = \{(x_i, y_i)\}_{i=1}^{l}$ consists of l labeled data, $\mathrm{X}^{\mathrm{U}} = \{(x_i)\}_{i=1}^{n-l}$ be an unlabeled data. The set of class labels is represented as $c = \{y_1, y_2, \cdots, y_k\}$ for k classes. As a promising representation learning method, autoencoder is often used for semi-supervised clustering. As a small number of labeled samples can be regarded as an additional regularized term, they are utilized to train autoencoder jointly on the representation learning and clustering. Specifically, the generic cost function can be denoted by the following simplified equation:

$$minJ = J_{recon} + \alpha J_{semi} \tag{4.1}$$

where J_{recon} is a common reconstruction term, whereas α is a hyperparameter for controlling the weights of semi-supervised clustering loss J_{semi}. In particular, J_{recon} can be computed as

$$J_{recon} = \frac{1}{2}\sum_{i=1}^{n} \| \hat{x_i} - g(x_i, w, b) \cdot \beta \|^2 \tag{4.2}$$

where $\| \cdot \|$ is a L^2 norm, x_i is reconstructed as $\hat{x_i}$ by single-layer-feedforward network, (w, b) are encoder parameters connecting a input to a hidden layer. β is an output weight, and $g(\cdot)$ is an activation function, such as sigmoid, sine, tanh and so on. Besides, the cross-entropy loss is often applied on J_{semi}:

$$J_{semi} = \frac{1}{l}\sum_{i=1}^{l}\sum_{j=1}^{k} y_{ij} \cdot log(\hat{y_{ij}}) \tag{4.3}$$

where $\hat{y_{ij}}$ is a class probability on the j-th label of data x_i and y_{ij} is a corresponding desired output. Given l label samples and k classes, $\hat{y_{ij}}$ can be either calculated on the bottleneck or the output layer of autoencoder.

4.3.3 Summary of the seeded progressive framework

In this chapter, the proposed seeded progressive framework is used to resolve a semi-supervised clustering problem. Given the initial seeds and clustering space, our framework aims at increasing the number of seeds, in other words, more labeled samples are explored. The entire process can be summarized as follows:

1. The seeded progressive framework includes two main phases: composition phase and decomposition phase.

2. In the composition phase, numerous mini-clusters are formed by numerous labeled samples where one labeled sample and several unlabeled samples are comprised inside the mini-cluster. The weak autoencoder (WAE) is proposed and used for clustering one unlabeled sample to one labeled sample.

3. In the decomposition phase, all mini-clusters are decomposed. The aim of this phase is to refine the clustered samples. Therefore the most confident unlabeled samples are accepted and regarded as labeled samples, while the rest are rejected and treated as unlabeled samples.

4. Labels are assigned to accepted samples (the same as the corresponding labeled samples from the same mini-cluster). Then, the above two phases are repeated by using all labeled samples to initialize the composition phase.

5. Consequentially, the number of labeled samples is increased via the above seeded progressive framework.

To better understand the proposed framework, the bottom-up approach is employed in this chapter. In Section 4.4.1, we will introduce the structure of the weak autoencoder. The process of clustering one unlabeled sample with one labeled sample will be explained in Section 4.4.2. In Section 4.4.3, the procedure of forming the mini-clusters will be described. Then, we will illustrate the composition and decomposition phases in Section 4.4.4. Finally, we will conclude the entire framework in Section 4.4.5.

4.4 Seeded Progressive Framework

The information maximization approach has been applied in a representation learning of autoencoder, the underlying ideology is to retain exhaustive information from raw to latent feature spaces. To achieve this goal, most autoencoders are designed with the complicate network architecture. In other words, more layers are added to the autoencoder to obtain the powerful universal generalization performance for reconstruction. In contrast, the information minimization approach is proposed to extract only the "strong" and "common" feature, for which it can be applied on clustering.

4.4.1 Information minimization on weak autoencoder

We utilize a single hidden layer feedforward network based autoencoder, where the invertible function is applied in a decoder. The refer readers to Chapter 3.4 to more details about the structure of the weak autoencoder.

4.4.2 Self-similarity learning

The reconstructed samples obtained from the WAE are severely corrupted since there is only one hidden neuron on a single hidden layer. Therefore, only the "strong" and "common" feature could be learned by such extremely simple network structure. The WAE is firstly trained by one labeled sample $\{(x_i, y_i)\}$, and then we feed an unlabeled sample x_j to such trained WAE. Assuming that $\hat{x_j}$ is the reconstructed sample of x_j, and y_j is the class label for the sample x_j, we set the following rule:

1. If $\hat{x_j}$ is similar to x_j, we believe x_j and x_i belong to the same category ($y_j = y_i$).
2. If $\hat{x_j}$ is dissimilar to x_j, we believe x_j and x_i belong to different categories ($y_j \neq y_i$).

Given that the WAE is trained only by x_i, it retains the strong common feature that is shared among samples in the same class. Thus, the WAE is capable of reconstructing homogeneous samples (i.e., samples that belong to the same class with x_i) while incapable of rebuilding heterogeneous samples. By measuring the similarity between $\hat{x_j}$ and x_j, it reflects the reconstruction ability of WAE. Thereby, it implies the similarity between test sample x_j and training sample x_i (See Eq.(4.4)). The 2-D correlation coefficient is used as the statistical similarity measure, which is simply termed as self-similarity (See Eq.(4.5)).

$$S(x_j, \hat{x}_j) \implies S(x_j, x_i) \tag{4.4}$$

$$S(x_j, \hat{x}_j) = \frac{(x_j - \bar{x_j})(\hat{x_j} - \bar{\hat{x_j}})}{\sqrt{(x_j - \bar{x_j})^2(\hat{x_j} - \bar{\hat{x_j}})^2}} \tag{4.5}$$

4.4.3 Composition of the mini-cluster

The most similar unlabeled sample x_k which has the highest self-similarity score, is grouped with the labeled sample x_i. Hence, a first mini-cluster is formed. We train the WAE again by a collection of labeled samples (x_i, x_k), and search another homogeneous sample by repeating the same procedure. Therefore, with i iterations, the WAE clusters i samples which belongs to the same class. As we use non-iterative WAE with only one neuron, the training speed could be extremely fast, usually less than 0.1 millisecond. However, when the number of training samples increase significantly, the reconstructed ability of the WAE is aggregately strengthen since more features will be learned by the WAE. Therefore, it is hard to make sure all the samples of the mini-cluster belong to the same category. To maintain the weakest performance of the WAE, we denote $\omega \in \mathcal{R}^+$ as a maximum number of samples of the mini-cluster, i.e., the capacity of mini-cluster. The process of the composition of the mini-cluster is summarized by following steps (See Fig. 4.1 also):

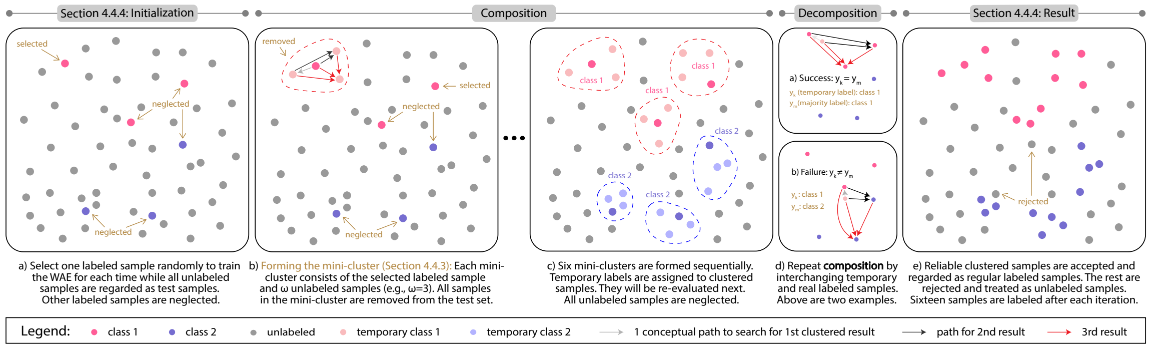

Section 4.4.4: Initialization
Composition
Decomposition
Section 4.4.4: Result
selected
neglected
neglected
removed
selected
neglected
neglected
class 1
class 1
class 1
class 2
class 2
class 2
a) Success: y_k = y_m
y_k (temporary label): class 1
y_m (majority label): class 1
b) Failure: y_k ≠ y_m
y_k: class 1
y_m: class 2
rejected
a) Select one labeled sample randomly to train the WAE for each time while all unlabeled samples are regarded as test samples. Other labeled samples are neglected.
b) Forming the mini-cluster (Section 4.4.3): Each mini-cluster consists of the selected labeled sample and ω unlabeled samples (e.g., ω=3). All samples in the mini-cluster are removed from the test set.
c) Six mini-clusters are formed sequentially. Temporary labels are assigned to clustered samples. They will be re-evaluated next. All unlabeled samples are neglected.
d) Repeat composition by interchanging temporary and real labeled samples. Above are two examples.
e) Reliable clustered samples are accepted and regarded as regular labeled samples. The rest are rejected and treated as unlabeled samples. Sixteen samples are labeled after each iteration.
Legend:
class 1
class 2
unlabeled
temporary class 1
temporary class 2
1 conceptual path to search for 1st clustered result
path for 2nd result
3rd result

Figure 4.2: A simplified example of the composition and decomposition phase shows six labeled samples from two classes are used to form six mini-clusters from (a) to (c). Clustered samples are re-evaluated while two cases are illustrated in (d). The acceptable samples from (e) will be used in the rest of the proposed framework.

step 1: The training set $\{x_i\}$ is initialized by a single training sample (x_i, y_i), while the test set $\{x_j\}$ is composed of all unlabeled samples $\{x_j\}_{j=1}^{n-l}$. Note that we assume there are l labeled samples and $(n-l)$ unlabeled samples.

step 2: The WAE (Section 4.4.1) is trained by the training set.

step 3: All unlabeled samples from the test set are fed to the trained WAE, and reconstructed as $\{\hat{x_j}\}_{j=1}^{n-l}$ by a decoder of WAE.

step 4: The self-similarity is calculated and ranked (Section 4.4.2). The top-ranked sample x_k is selected as the clustered sample.

step 5: The label y_i of the initial labeled sample is assigned to x_k. The clustered sample is eliminated from $\{x_j\}_{j=1}^{n-l}$ and added to the training set, i.e., $\{x_i, x_k\}$.

step 6: If the number of samples of mini-cluster reaches the upper limit ω, the process is terminated. Otherwise, we repeat steps 2 to 5.

4.4.4 Composition and decomposition phase

Since we directly add the clustered sample into the training set to train WAE, our proposed algorithm highly relies on the accurate clustering performance for every round. It is necessary to refine the clustering results, and hence a two phases framework is proposed. By repeating the composition (Section 4.4.4) and decomposition (Section 4.4.4) phases, more labeled samples are discovered.

Composition phase for explored seeds

In this phase, l mini-clusters are formed by l labeled samples, $\{(x_1, y_1), ..., (x_l, y_l),\} \in \mathrm{X}^{\mathrm{L}}$. Our method forms mini-clusters sequentially, in which every mini-set consists of ω samples. The details can be found in Section 4.4.3. In the t-th composition, we assign temporary labels $y_i \in c$ to all clustered samples from the mini-cluster $x_i^t = \{(x_i, x_{i1}, x_{i2}, ..., x_{i\omega})\}$. Sequentially, we form multiple mini-clusters until all labeled samples are processed.

Decomposition phase for refinedly explored seeds

In order to improve the accuracy of the proposed clustering mechanism, we introduce the decomposition phase to re-evaluate the samples. It basically repeats the composition phase but by interchanging the role of labeled samples and clustered samples, called temporarily-labeled samples. Assume $\{(x_{ij}, y_i)_{i=1,j=1}^{l,\omega} \in \mathrm{X}^{\mathrm{t}}\}$ be the t-th temporarily-labeled sample set. We take x_k from X^{t} as an example to explain the decomposition phase as follows:

step 1: The training set $\{x_k\}$ is initialized by the single temporary-labeled sample (x_k, y_k), while the test set $\{x_j\}$ is composed of all labeled samples $\{x_j\}_{j=1}^{l}$. Note that there are l labeled samples.

step 2: A single mini-cluster is formed after repeating the process of composition phase, i.e., $\{(x_k, x_{k1}, ..., x_{k\epsilon}), (y_k, y_{k1}, ...y_{k\epsilon})\}$. Since the clustered samples are primarily labeled, there are ϵ labels, (ϵ can be different from ω)

step 3: All samples from the above mini-cluster carry labels (both temporary and real labels). Assuming that y_k is a temporary label for x_k, we denote y_m as the majority label among clustered (labeled) samples via majority voting. We then consider the following two conditions to decide whether to keep the temporary-labeled samples:

(a) If $y_k = y_m$, we believe x_k is strongly similar to the labeled sample from one class since it finds the correct label reversely. We assign y_k to x_k as a regular label and include it into the labeled sample set.

(b) If $y_k \neq y_m$, we believe x_k is not strongly similar to the labeled sample from one class since it finds the incorrect label reversely. We release x_k to the unlabeled sample set.

step 4: We repeat steps 1 to 3 until all samples are verified.

In short, the reliable temporarily-labeled samples are accepted, while the rest are rejected. Similar to composition phase, the capacity of the mini-cluster can be denoted as ϵ, which can be different from ω. Note that the accepted samples are regarded as regular labeled samples in the rest of the proposed framework.

Method improvement using random sampling

As the above two phases involves thousands times of self-similarity calculation, the proposed approach requires a lengthy running time for completing the composition and decomposition phases. Therefore, the random sampling is proposed to make our algorithm much faster. As depicted in Fig. 4.2 (b), the proposed method considers only local relationship between one labeled sample and all unlabeled samples for each time. When deciding whether the unlabeled is closest to the selected labeled sample, the remaining labeled sample is neglected. In other words, the proposed clustering mechanism has the local clustering ability, different from other clustering methods (See Section 4.4.5). Thus, the random sampling can be employed prior to the composition phase to randomly separate the entire dataset X into λ subsets.

$$\mathrm{X} = [\,\mathrm{X}_1,\, \mathrm{X}_2,\, ...,\, \mathrm{X}_\lambda\,]$$

$$\mathrm{X}_k^{\mathrm{L}} = \{(x_i, y_i)\}_{i=1}^{lk},\ \mathrm{X}_k^{\mathrm{U}} = \{(x_i)\}_{i=1}^{nk-lk},\ k = 1, 2, ..., \lambda$$

where n is a number of samples of entire dataset, nk and lk are the number of total and labeled samples of k-th subsets respectively. $\mathrm{X}_k^{\mathrm{L}}$ and $\mathrm{X}_k^{\mathrm{U}}$ are the labeled and unlabeled sample sets from the k-th subset. We equally separate the dataset, in which lk is the number of labeled samples in the k-th subset. We apply sampling without replacement. Therefore the composition phase can be processed with parallel computing. As the number of unlabeled samples is reduced for the k-th batch, we reduce the calculation of self-similarity from $(n - l)$ to approximately $(nk - lk)$ at a time. Hence, it speeds up our algorithm. Note that as the random sampling probably degrades clustering performance, the fair test on selection of λ is conducted in Section 4.5.7.

4.4.5 The complete workflow of seeded progressive framework for weakly-supervised segmentation

Complete workflow: Prior to the weakly-supervised semantic segmentation (WSSS) challenge, the seeded progressive framework is proposed to search for the accurate explored seeds. In practice, the initial seeds is generated by CAM method while the clustering space is established by extracting the convolutional feature map from fc7 layer of DeepLab model. In the composition phase of seeded progressive framework, unlabeled samples are grouped to labeled samples in corresponding mini-clusters. In the decomposition phase, the clustered samples are re-evaluated such that the reliable samples are acceptable and regarded as the regular labeled samples while the rest are rejected. The proposed framework is wrapped into a cyclic structure such that it iterates the above phases by τ cycles. To make sure only the most reliable seeds are obtained, we suggest to set $\tau = 2$. Eventually, the explored seeds will fully replace the initial seeds, being used to train any WSSS methods (e.g., [3–5]), for boosting the segmentation performance.

The proposed method will be applied to one of the semantic segmentation dataset, i.e., PASCAL VOC 2012. We will briefly introduce the experimental setup in 4.5.1. In Sections 4.5.2, 4.5.3 and 4.5.4, the hypothesis validation, the baseline performance and the comparison experiment on clustering space will be presented respectively. We will compare our method with current state-of-the-art methods in Section 4.5.5. The qualitative results can be found in Section 4.5.6. Eventually, we will provide the step-by-step analysis on the proposed framework in Section 4.5.7.

4.5 Experiment on weakly-supervised segmentation

4.5.1 Experimental setup

Dataset: For conducting fair comparisons, we evaluate our proposed method on the PASCAL VOC 2012 (VOC-2012) segmentation benchmark dataset [141, 143]. Including the background, it contains 21 semantic classes. Following the common practice [50], we augment the training dataset by adding additional images which are provided from [144]. The resulting augmented training set consists of 10,582 images while validation and test dataset are formed by 1,449 and 1,456 images re-

Table 4.1: Network settings of segmentation models

Hyper-parameters	**SEC** [1]	**DSRG** [3]	**CIAN** [4]
Selected backbone	VGG16	VGG16	ResNet101
Optimizer	SGD	SGD	SGD
Momentum	0.9	0.9	0.9
Weight decay	5e-4	5e-4	5e-4
Batch size	32	64	16
Learning rate	1e-3	1e-3	1e-3
Learning rate drop factor	0.1	0.1	Poly: 0.9
Learning rate drop period	2000	2000	

spectively. As per the weakly-supervised method, only image-level-labels are adopted for supervision. We use standard mean intersection over union (mIoU) to assess the segmentation performance.

Reproducibility: For semi-supervised clustering, experiments are all carried out in MATLAB 2019b on the Linux Ubuntu 18.04 platform with 12 cores of Intel Core i9-7920X @ 2.9 GHz and 128 GB memory. The codes are available for reviewers at http://www.yiminyang.com/weakly_supervised.html. For weakly-supervised segmentation, we use published Python codes for CIAN[2] and reproduced codes for SEC[3] and DSRG[4].

4.5.2 Hypothesis validation

In this experiment, we validate the core hypothesis, explored seeds generated by the proposed method could bring more prior knowledge to the segmentation model. To this end, we randomly select N_r unseen real-seeds from each ground-truth image, and add into the corresponding initial seeds as simulation of the explored seeds. Qualitatively, we add N_f fake-seeds to test how does the segmentation performance of the weakly supervised model is affected by incorrect labels. The SEC is used as the segmentation model in this experiment. All used network settings are summarized in Table 4.1 which are recommended from [3].

Table 4.2 shows segmentation results on the validation dataset of VOC-2012, it reaches to the expectation that additional real-seeds bring positive impacts on seg-

[2]https://github.com/js-fan/CIANl
[3]https://github.com/halbielee/SEC_pytorchl
[4]https://github.com/xtudbxk/DSRG-tensorflow

Table 4.2: Hypothesis validation on VOC-2012 validation dataset using SEC [1] with varying number of simulated real-seeds (N_r) and fake-seeds (N_f).

	N_r				
N_f	0	100	300	500	full
0	49.44	53.74	56.43	57.35	58.37
100	40.33	46.20	51.51	53.86	–
300	32.16	37.17	43.91	–	–
500	27.11	31.37	–	–	–

mentation while fake-seeds severely hurt the performances. It is remarked that the replicated SEC result with initial seeds (49.44%) somewhat inferior to its originally reported (50.70%). We suspect there is minor differences between Caffe and PyTorch implementation. Furthermore, we denote "full" to simulate all real-seeds are used to train the model, i.e. fully-supervised training. The obtained result is 58.37%, which can be regarded as the theoretical "upper-bound" [42] of the selected model.

4.5.3 Baseline performance

Preparation of initial seeds and clustering space: The proposed method relies on the initial seeds. To the best of our knowledge, only [3, 4] research on initial seed generation while their seeds can be represented as $\mathcal{S}_{sec}$ and $\mathcal{S}_{dsrg}$ respectively. Regarding the essential clustering space, we utilize the well-known semantic model, DeepLab [50], since it provides larger receptive fields during the convolution process. We initialize the model by ImageNet, and directly extract feature map $\mathcal{F} \in \mathcal{R}^{41\times41\times4096}$ from the *fc7* convolutional layer as the baseline clustering space.

Setting: Since there are different hardware settings, we run related source codes to provide our replicated segmentation performances of three models in order to conduct fair comparisons. For references, we also show the benchmark results which are obtained directly from the published chapters. These results can be found in the first two rows of Table 4.3. For semi-supervised clustering, three hyperparameters are involved including both the capacities of the mini-cluster in two phases (ω, ϵ) and the number of subsets (λ). In this entire section, we use $(3, 2, 20)$ for $(\omega, \epsilon, \lambda)$ unless otherwise specified.

Table 4.3: Baseline performances on VOC-2012 validation dataset. The first row is cited from published papers while second row is our reproduction. The best results are in bold while the second bests are underlined

	SEC [3]	DSRG [4]	CIAN [5]
Benchmark from published papers	50.70	57.60	64.30
Our replicated performances	49.44	57.11	63.71
(a) Baseline Performances			
Init. seed $\mathcal{S}_{sec}$ on clustering space $\mathcal{F}$	51.81	53.98	66.34
Init. seed $\mathcal{S}_{dsrg}$ on clustering space $\mathcal{F}$	<u>53.60</u>	58.63	64.21
(b) Feature Map Comparison			
Init. seed $\mathcal{S}_{sec}$ on clustering space $\mathcal{F}_{sec}$	52.19	57.80	<u>66.79</u>
Init. seed $\mathcal{S}_{sec}$ on clustering space $\mathcal{F}_{dsrg}$	53.51	<u>59.10</u>	**68.14**
Init. seed $\mathcal{S}_{dsrg}$ on clustering space $\mathcal{F}_{sec}$	53.56	59.00	63.51
Init. seed $\mathcal{S}_{dsrg}$ on clustering space $\mathcal{F}_{dsrg}$	**54.51**	**59.19**	65.58

Results: The $\mathcal{S}_{sec}$ and $\mathcal{F}$ are selected as the initial seeds and clustering space as setups for the proposed method. The explored seeds generated by the proposed method is denoted as $\mathcal{S}^{'}_{sec}$ (similarly $\mathcal{S}^{'}_{dsrg}$ for DSRG). Then, we substitute $\mathcal{S}^{'}_{sec}$ to the original seeds and train the segmentation model, while fixing all networking setting. Table 4.3 (a) shows that 1.11% of performance gain is obtained when using our $\mathcal{S}^{'}_{sec}$ instead of $\mathcal{S}_{sec}$. On the other hand, we repeat above procedure by using $\mathcal{S}^{'}_{dsrg}$ on the DSRG and CIAN since they both employ $\mathcal{S}_{dsrg}$ rather that $\mathcal{S}_{sec}$. By directly referring to replicated results, the result demonstrates our methods explores numerous and accurate seeds which achieve better performances compared with corresponding competitors in term of mIoU.

As the number of seeds of $\mathcal{S}_{dsrg}$ is larger than that of $\mathcal{S}_{sec}$, the $\mathcal{S}^{'}_{dsrg}$ contributes more informative supervision to the segmentation model such that higher mIoU results are achieved on both SEC and DSRG. Nevertheless, the counterexample is observed on CIAN in which $\mathcal{S}^{'}_{sec}$ provides the better result. We suspect that more false-positive seeds are discovered in $\mathcal{S}^{'}_{dsrg}$. Due to the cross-image learning, the false-positive seeds offers an imprecise attention information which severely deteriorates the co-segmentation. In general, the performances on CIAN achieved by the proposed seeded progressive framework are still higher than that in the original chapter [5]. Therefore, our semi-supervised clustering method explores sufficient and accurate seeds that contribute to the improved segmentation performance of [3–5]. Note that the results demonstrated in Table 4.3 (a) can be regarded as the baseline performance.

4.5.4 Comparison with clustering spaces from feature maps

As the proposed method also depends on the initial clustering space, the experiments to test how our method is affected by different clustering spaces is conducted. We extract feature maps from the last convolutional layer of the trained SEC and DSRG models respectively, namely $\mathcal{F}_{sec}$ and $\mathcal{F}_{dsrg}$. To compare fairly, the $\mathcal{S}_{sec}$ is selected as the initial seeds. The results from upper part of Table 4.3 (b) generally indicates that better convolutional spaces further enhance segmentation results. For instance, we obtain 5.12% improvement on DSRG from 53.98% ($\mathcal{F}_{dsrg}$) to 59.10% ($\mathcal{F}$).

In order to jointly evaluate the effectiveness on the initial seeds and clustering spaces, we repeat the above experiment but by using $\mathcal{S}_{dsrg}$ (lower part of Table 4.3 (b)). Similar to previous section, $\mathcal{S}_{dsrg}$ with trained clustering spaces (both $\mathcal{F}_{sec}$ and $\mathcal{F}_{dsrg}$) mostly provide better segmentation results. However, the counterexample is still observed on the CIAN. For example, by using clustering space $\mathcal{F}_{sec}$, 66.79% is obtained when $\mathcal{S}_{sec}$ is used while 63.51% is collected when $\mathcal{S}_{dsrg}$ is applied.

4.5.5 Comparison with state-of-the-art methods

Regardless of different initial seeds and clustering spaces, the proposed method surpasses three original method in terms of mIoU [3–5]. Besides, we also compare our method with previous relate works. With different annotation settings and backbone model architectures, we emphasize that these comparison results can only be referenced and should not be straightly compared. Tables 4.4 and 4.5 summarize state-of-the-art results using VGG16 and ResNet-based backbone models respectively. We provide our best performances with the corresponding initial seeds and clustering space listed in parentheses. All the segmentation results on both validation and test dataset are uploaded to the official PASCAL VOC evaluation server.[5]

[5]http://www.yiminyang.com/weakly_supervised.html

Table 4.4: Comparison of weakly-supervised semantic segmentation methods on VOC-2012 validation and test dataset. The supervision (Sup.) includes: image-level ($\mathcal{I}$), implicitly use pixel-level ($\mathcal{P}$), bounding box ($\mathcal{B}$) or additional data ($\mathcal{A}$). The methods listed here use VGG16 as backbone model. Values in bold are for SEC while the results for DSRG are underlined

Method (VGG16)	**Sup**	**Val.**	**Test**
With additional supervision			
MIL-seg [145]	$\mathcal{I} + \mathcal{P}$	42.0	40.6
CrawlSeg [146]	$\mathcal{I} + \mathcal{A}$	58.1	58.7
GuidedSeg [139]	$\mathcal{I} + \mathcal{S}$	55.7	56.7
STC [81]	$\mathcal{I} + \mathcal{A}$	49.8	51.2
EM-Adapt [55]	$\mathcal{I}$	38.2	39.6
AE_PSL [84]	$\mathcal{I}$	55.0	55.7
GAIN [85]	$\mathcal{I}$	55.3	56.8
MCOF [88]	$\mathcal{I}$	56.2	57.6
AFFNet [87]	$\mathcal{I}$	58.4	60.5
SeeNet [138]	$\mathcal{I}$	61.1	60.7
FickleNet [89]	$\mathcal{I}$	61.2	61.9
LSISU [147]	$\mathcal{I}$	61.2	62.5
MCIS [93]	$\mathcal{I}$	63.5	63.6
ACFN [148]	$\mathcal{I}$	63.6	64.2
SEC [3]	$\mathcal{I}$	<u>50.7</u>	<u>51.7</u>
DSRG [4]	$\mathcal{I}$	**57.6 (59.0*)**	**60.4**
Ours SEC ($S_{dsrg} + F_{dsrg}$)	$\mathcal{I}$	<u>54.5</u>	<u>55.0</u>
Ours DSRG ($S_{dsrg} + F_{dsrg}$)	$\mathcal{I}$	**59.2**	**60.0**

* Applying retraining technique

VGG16-based models

Apart from SEC, DSRG and CIAN, several well-known methods using the adversarial erasing approach are listed in Table 4.4, such as GAIN [85], SeeNet [138], AFFNet [139] and so on. Compared directly with SEC, the proposed approach achieves mIoU scores of 54.5% and 55.0% on VOC-2012 validation and test dataset respectively. For DSRG, 59.2% are obtained on validation dataset by using the explored seeds $\mathcal{S}'_{dsrg}$. These investigation results reveal that the explored seeds provides significantly

Table 4.5: Comparison of weakly-supervised semantic segmentation methods on VOC-2012 validation and test dataset. The methods listed here use ResNet-based model. Values in bold are for CIAN.

Method (ResNet)	**Backbone**	**Val.**	**Test**
MCOF [88]	ResNet101	60.3	61.2
AFFNet [87]	ResNet38	61.7	63.7
DSRG [4]	ResNet101	61.4	63.2
SeeNet [138]	ResNet101	63.1	62.8
FickleNet [89]	ResNet101	64.9	65.3
CSENet [149]	ResNet101	62.3	63.4
WSIF [150]	ResNet101	63.8	64.7
SEAM [94]	ResNet38	64.5	65.7
SubCat [95]	ResNet101	66.1	65.9
MCIS [93]	ResNet101	66.2	66.9
ACFN [148]	ResNet101	66.0	66.6
LSISU (DRFI) [147]	ResNet101	62.5	62.7
LSISU (DeepSaliency) [147]	ResNet101	68.4	68.9
CIAN [5]	ResNet50	**62.4**	**63.8**
CIAN [5]	ResNet101	**64.3**	**65.3**
Ours CIAN ($S_{sec} + F_{dsrg}$)	ResNet101	**68.1**	**69.2**

better supervision. Additionally, a predicted segmentation masks can be used as the pseudo-labels to train the network for another round, i.e., retraining strategy. As listed in the parentheses of DSRG, the retraining can improve the segmentation performance since the pseudo-labels can provide finer details to further fine-tune the model. Nevertheless, the explored seeds generated by the seeded progressive framework is already at a fine level. Therefore, without applying retraining, the proposed method still surpasses these two variation of DSRGs.

Compared with most recent models (SeeNet [138], FickleNet [89], MCIS [93] and ACFN [148]), they achieve superior performance to ours. However, high computational resources are required for these models. MCIS uses the co-segmentation learning approach which demands multi-GPUs to train the network iteratively. Even though the Map Expansion Technique is proposed, FickleNet still requires intensive GPUs memory usage when training the newtwork. In contrast, our framework is simple yet efficient since we adopt the non-iterative learning strategy for our proposed weak autoencoder. Hence, it requires extremely little computing resources.

Table 4.6: Performance comparison of mIoU on VOC-2012 validation dataset, compared to related counterparts (our reproduction of SEC, DSRG and CIAN). The best results are in bold while the second bests are underlined for each semantic objects. (classes 1 to 11)

Method	bkg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow
SEC [3]	81.0	51.0	24.8	55.4	24.0	43.9	69.0	57.9	73.8	21.1	55.7
DSRG [4]	85.7	68.7	31.0	60.9	35.4	64.6	70.5	66.4	79.2	24.9	65.5
CIAN [5]	88.1	79.0	**33.6**	74.6	**55.0**	**70.9**	84.2	72.4	79.0	23.9	74.8
Ours SEC	85.7	65.9	27.3	68.1	34.8	49.6	70.6	66.3	77.0	21.2	54.2
Ours DSRG	88.0	77.7	32.6	75.4	49.8	55.8	73.3	68.6	80.6	22.9	58.4
Ours CIAN	**90.2**	**81.7**	33.1	**83.8**	48.9	**70.9**	**86.3**	**78.1**	**90.9**	**33.2**	**84.3**

Table 4.6: (continued) Performance comparison of mIoU on VOC-2012 validation dataset. (classes 12 to 21)

Method	table	dog	horse	mbk	person	plant	sheep	sofa	train	tv	mean
SEC [3]	29.7	65.5	54.1	62.1	55.2	37.0	57.9	33.0	44.8	41.5	49.4
DSRG [4]	25.5	76.3	67.5	66.2	65.2	36.3	70.2	35.1	54.1	49.9	57.1
CIAN [5]	**41.1**	75.7	78.9	**75.0**	64.3	44.1	78.2	29.2	61.7	54.2	63.7
Ours SEC	31.0	68.3	58.1	61.3	67.2	37.7	65.1	30.9	52.9	51.3	54.5
Ours DSRG	23.3	73.0	65.6	64.5	68.9	39.5	69.0	33.1	**64.1**	58.7	59.2
Ours CIAN	33.2	**83.3**	**79.7**	74.9	**77.6**	**49.5**	**85.2**	**37.1**	64.0	**65.2**	**68.1**

Furthermore, it is surprising that our method surpasses some early weakly-supervised semantic methods with additional supervision. By using VGG16 as backbone model fairly, STC [81] and CrawlSeg [146] (upper part of Table 4.4) use additional images while only 10K images are involved in our method.

ResNet-based models

In Table 4.5, all methods use ResNet-based backbone model. Integrating with our seeds, we improve CIAN performances and achieve mIoU of 68.1% and 69.2% from original 64.3% and 65.3% on validation and test dataset of VOC-2012 respectively. Note that saliency methods are employed in LSISU to obtain an additional knowledge to train the segmentation model. As the proposed method builds on top of CIAN, we take huge advantages from their cross-image affinity module and currently achieve state-of-the-art performance at the time of writing this chapter by using only image-level supervision.

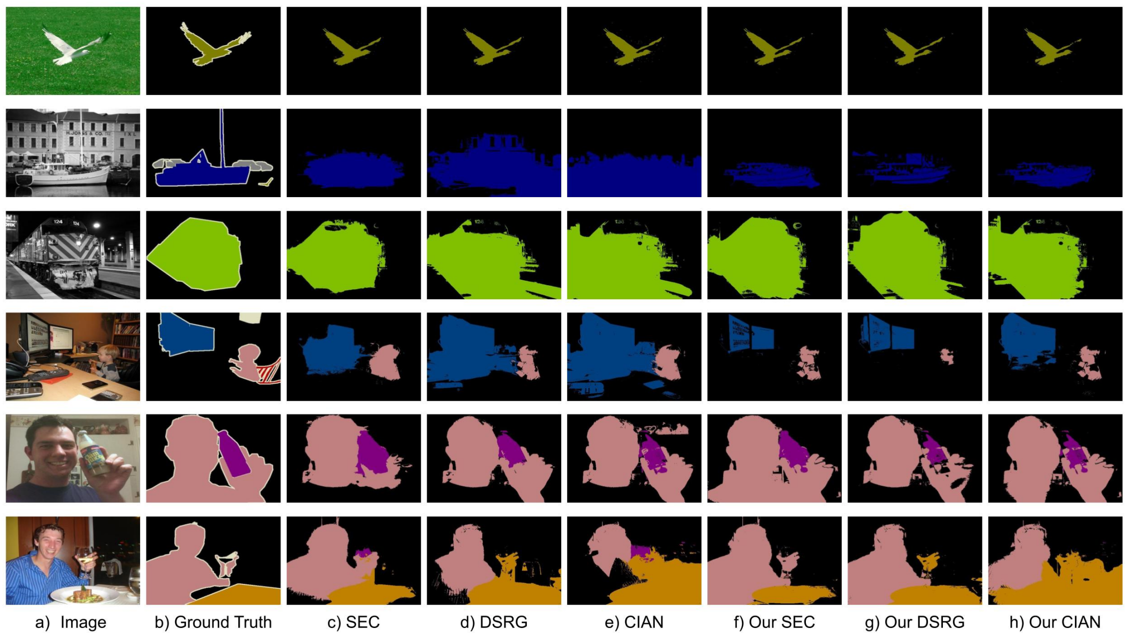


Figure 4.3: Visualization examples on VOC-2012 validation dataset.

4.5.6 Qualitative results

Fig. 4.3 and Table 4.6 present numerical and visual qualitative results of the final segmentation mask. By comparing our methods (f to h) to corresponding counterparts (c to e), it shows that our methods can recover fine details of the boundary, and discover more accurate mask in a larger area of a target object. The first row demonstrates the simplest example, which can be segmented among different methods. The next three rows exhibit that our methods can produce more precise result by inhibiting false positive prediction since the explored seeds offer stronger location information than that of the deep segmentation model. On the other hand, larger area can be explored in the last two rows. Since our methods give more accurate initialize seeds, it allows the segmentation network to consider a wider range of target objects.

4.5.7 Step-by-step analysis on semi-supervised clustering

From semi-supervised learning standpoint, we analyse how the proposed seeded progressive framework is affected using different hyperparameters thoughtfully. To this end, we conduct fair tests on a small-scale VOC-2012 dataset, namely Mini-VOC-2012 which consists of 500 randomly selected images from VOC-2012. Since we target at increasing the number of seed samples, there are two evaluation metrics in this experiments. Given an initial seed $\mathcal{S}_{sec}$ and feature map $\mathcal{F}$ for each image on

Table 4.7: Comparison of mAcc, mCr, and processing time on Mini-VOC-2012 without and with Random Sampling (RS). Values in bold are corresponding results of the select λ.

λ	**mAcc(%)**	**mCr(%)**	**time(s)**
0 (Without RS)	91.36	66.40	229.4688
5	89.72	65.51	28.9852
10	89.37	65.68	19.7908
15	89.00	65.83	17.9757
20	**88.78**	**65.85**	**17.3673**

Mini-VOC-2012, the mean accuracy (mAcc) is calculated to evaluate the quality of the explored seeds. Similarly, the mean cover rate (mCr) is computed for assessing how many seeds are explored by the proposed approach including both the initial and newly clustered seeds. These metrics are utilized to analyse three hyperparameters as follows:

1. Number of subsets λ: We apply random sampling to separate the the dataset into λ subsets (Section 4.4.4).

2. The capacity of mini-cluster in the composition phase ω: We start with single labeled samples, and form the mini-cluster which contains ω samples (Section 4.4.4).

3. The capacity of mini-cluster in the decomposition phase ϵ: The same as above. (Section 4.4.4).

Trade-off between efficiency and effectiveness using random sampling

As discussed in Section 4.4.4, we speed up the proposed approach by using a random sampling. The larger the number of subsets (λ), the shorter the processing time. However, this technique probably degrades the clustering performance because the dataset is pre-separated into subsets. Therefore, the comparative test on λ is required while ω and ϵ are fixed as 3 and 2. Table 4.7 demonstrates the mAcc, mCr and the time obtained by the proposed method using various λ. When larger λ is used, the total clustering time is reduced, i.e., approximately 13 times faster when comparing λ=20 with λ=0. Although the mAcc are slightly decreasing constantly, the mCr

Table 4.8: Comparison of mAcc and mCr on Mini-VOC-2012 using different combination of ω_1 and ϵ_1. Values in bold are corresponding results of the select (ω_1, ϵ_1).

	mAcc(%)			**mCr(%)**		
ω_1	$\epsilon_1 = 1$	$\epsilon_1 = 2$	$\epsilon_1 = 3$	$\epsilon_1 = 1$	$\epsilon_1 = 2$	$\epsilon_1 = 3$
1	90.65	90.94	90.28	51.94	50.53	50.77
2	89.31	89.90	88.61	61.24	58.84	59.37
3	88.10	**88.78**	87.26	69.08	**65.89**	66.72
4	87.27	88.02	86.30	74.60	71.31	72.13
5	86.67	87.59	85.55	77.97	74.85	75.73

performances are stably maintained in the range of 65.5% to 65.8%. Hence, the random sampling improves the efficiency of semi-supervised clustering, and maintains the effectiveness at the same time. In the remaining experiments, we use λ=20 unless otherwise specified.

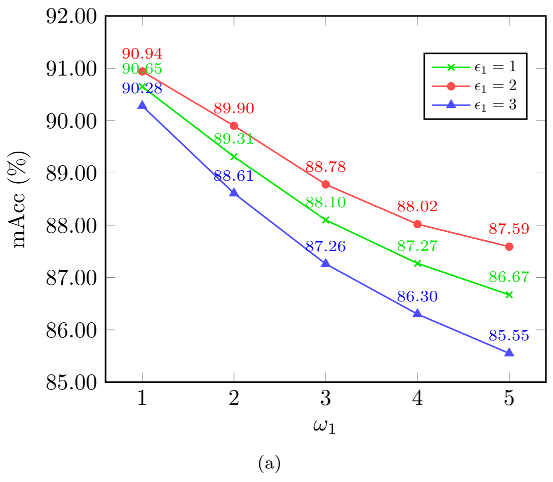
Comparison on capacity of the mini-cluster

Due to the characteristics of the proposed weak autoencoder, we need to set a maximum number of samples of the mini-cluster (i.e., ω and ϵ) to maintain a weakest performance of WAE. In other words, we need to control the number of iterations for adding clustered samples to the mini-cluster. As the labeled samples is increased progressively, we can design a dynamic capacity of the mini-cluster involved three cycles (Section 4.4.5 of the proposed framework. That is, we can use (ω_1, ϵ_1) for the first cycle while (ω_r, ϵ_r) for the rest. In general, we can select a smaller number for (ω_1, ϵ_1) to ensure only the reliable samples is explored by few initial labeled samples in an early stage. As the number of labeled samples increases, a larger number for (ω_r, ϵ_r) can be applied.

Comparison on ω_1 and ϵ_1 To jointly conduct the test on (ω_1, ϵ_1), the grid search is employed by using $\omega_1 = \{1, 2, 3, 4, 5\}$ and $\epsilon_1 = \{1, 2, 3\}$ while (ω_r, ϵ_r) is fixed as $(2, 2)$. As the incorrect seeds servery hurts the segmentation, mAcc is prioritized over mCr. Fig. 4.4 shows the corresponding performances of mAcc and mCr on Mini-VOC-2012. By using ϵ_1=2, the best mAcc constantly is obtained regardless the value of ω_1 (Fig. 4.4 (a)). To consider mCr at the same time, the combination of $(3, 2)$ for

Table 4.9: Comparison of mAcc and mCr on Mini-VOC-2012 using different combination of ω_r and ϵ_r. Values in bold are corresponding results of the select (ω_r ϵ_r).

	mAcc(%)			**mCr(%)**		
ω_r	$\epsilon_r = 1$	$\epsilon_r = 2$	$\epsilon_r = 3$	$\epsilon_r = 1$	$\epsilon_r = 2$	$\epsilon_r = 3$
1	89.25	89.67	88.68	58.20	56.76	57.76
2	88.34	**88.74**	87.28	68.45	**66.04**	67.61
3	87.81	88.28	86.62	75.48	72.36	74.41



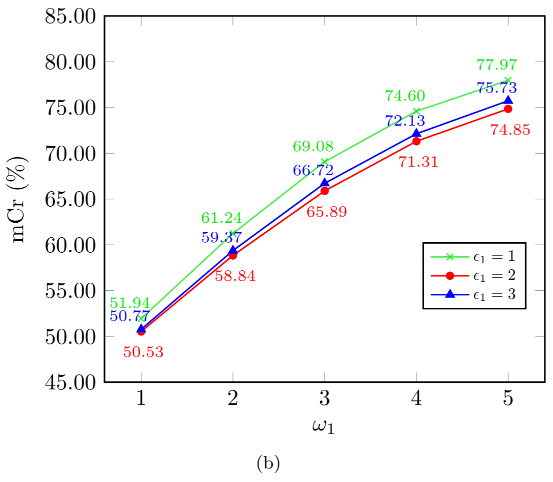


Figure 4.4: Comparison of mAcc and mCr on Mini-VOC-2012 dataset using different combination of ω_1 and ϵ_1.

(ω_1, ϵ_1) is recommended since it strikes a better balance between mAcc (88.78%) and mCr (65.89%).

Comparison on ω_r and ϵ_r By using the best hyperparameters for (ω_1, ϵ_1), we conduct the final comparative test for (ω_r, ϵ_r). Similarly, the grid search is applied such that $\omega_r = \{1, 2, 3\}$ and $\epsilon_r = \{1, 2, 3\}$. It is discovered that ϵ_r=2 contributes the best mAcclong with various ω_r. To reach reasonable compromise between two mentioned metrics, $(2, 2)$ for (ω_r, ϵ_r) is preferable to use such that 88.34% and 66.04% are reached for mAcc and mCr respectively.

Summary In general, mAcc is consistently decreased when ω is increasing with the fixed ϵ. That is to say, the reconstruction performance of WAE is generalized when

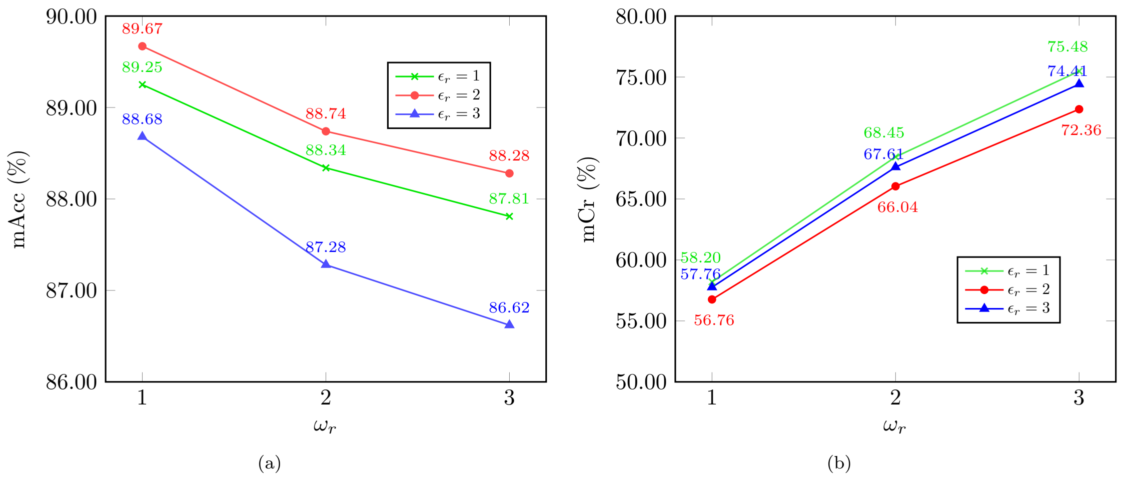


Figure 4.5: Comparison of mAcc and mCr on Mini-VOC-2012 dataset using different combination of ω_r and ϵ_r.

more labeled samples are used to train such WAE. The learned features of WAE is no longer "strong" and "common" among the same class. Therefore, the clustering process may not work well when it is repeated too many times, e.g., more than 5 times. In addition, the proposed seeded progressive framework is not sensitive to dynamic setting for (ω, ϵ). Therefore, for simplicity, it is suggested to use uniform setting, i.e., $(\omega_r, \epsilon_r) = (\omega_1, \epsilon_1)$.

4.6 Conclusions

In this chapter, we formulate the weakly-supervised semantic segmentation task using only image-level labels as a semi-supervised seed clustering problem where initial seeds generated by CAM method can be regarded as labeled samples while the rest are unlabeled samples. In such perspective, the non iterative weak autoencoder, WAE is proposed to group one unlabeled sample with one labeled sample. We propose the seeded progressive framework to search for numerous and accurate seeds to further improve the segmentation performance of [3–5]. We demonstrate the effectiveness of the seeded progressive framework on PASCAL VOC 2012 dataset, while thorough analyses have been conducted to suggest the proper choices of involved hyperparameters.

Chapter 5

Conclusion & Future Work

Most of the machine learning or deep learning algorithms require a very large number of labeled data as prior knowledge. However, it is difficult and expensive to obtain a large number of labeled data. By contrast, the unlabeled data is usually accessible and can be easily to collect. Therefore, **the primary research focus is to develop machine learning algorithms for solving the semi-supervised clustering task.**

As shown in previous research [2], "strong" and "common" features of each data could be extracted by using deeper dimensional compression. Inspired by this observation, the weak autoencoder (WAE) is proposed as an unsupervised representation learning method to learn the "common" feature of each data. Assuming that the learned feature is shareable among the same class, it means data belonging to the same class should have similar features. Therefore, if any unlabeled data can be well-reconstructed by the WAE that is trained by only one labeled data, those unlabeled data should be similar with the labeled data. The most well-reconstructed unlabeled data can be grouped with that labeled data. In order to evaluate and rank the degree of reconstruction of each unlabeled data, the self-similarity is calculated by measuring the correlation coefficient between every pair of original and reconstructed data.

The progressive framework is proposed to continue the clustering process. This framework is wrapped into the cyclic structure where two phases are repeated iteratively. The aim of first phase is to search for the unlabeled data starting from a small number of labeled data. The target of second phase is to re-evaluate the clustered results such the most confident data are accepted and regarded as labeled samples while the rest are rejected and treated as unlabeled samples. By repeating the first and second phase alternatively, the number of labeled data increases.

The experimental results on thirteen datasets show the effectiveness of the progressive framework. In fact, these works are important to researchers in many domains of computer science as these works fundamentally provided another way of approach to extract useful feature using a weak autoencoder instead of a complicated network.

After verifying the effectiveness of the above algorithms on thirteen tubular and image datasets, **the second focus of this thesis is to extend these algorithms to the application in real-world scenario, i.e., semantic segmentation.** Two preliminary concerns have to be handled when formulating such task as the semi-supervised clustering problem: how to obtain the initial labeled data and how to construct clustering space.

The solution for first concern is to generate initial seeds using computer vision-based method. Given only the image-level labels to the deep learning model, the Class Activation Map method is used to back propagate the activated weights to the final convolutional heat map such that some units will be heated up. These units are usually the significant regions of image which indicates the most representative region of each object. By thresholding the top 20% units of this heatmap, the initially labeled data are generated. Unlike using the raw data representation to construct the clustering space, the mentioned 2-D heatmap is extracted from a deep learning model. In fact, each data in the heatmap is a 4096-dimensional feature vector rather than just RGB data.

After initialization, the original semantic segmentation problem is transferred to a semi-supervised clustering problem which can be solved by the modified progressive framework. Terminologically, labeled data can be viewed as seeds while unlabeled data can be treated as non-seeds. As seed is the important concept of this application, the modified framework is termed as "Seeded Progressive Framework". This framework can be used as a knowledge augmentation method to increase the number of labeled data. In other words, more seeds can be generated by using this framework. Then, the initially and newly clustered seeds can be used to train few current weakly-supervised segmentation methods. As these seeds are better than initial seeds, they can be used for boosting the segmentation performances.

By experiments, around 1.5% to 3.9% mIoU of performance improvements are obtained, and achieve the current state-of-the-art performance 69.2% on PASCAL VOC

2012 test dataset. To conclude, this work suggests a way to improve performances of any weakly-supervised segmentation models by using our proposed framework. In fact, other existing semi-supervised methods can be also used to increase the number of seeds and capable for enhancing the segmentation performances.

This thesis handles different tasks and problems in the field of ML, and it can be summarized as follows:

- **Weak Autoencoder (WAE).** It is an unsupervised classifier which is used to learn strong and common feature of each data. As this autoencoder is repeatedly used for clustering task, the non-iterative autoencoder is used such that there are only one encoder layer and one decoder layer.

- **Self-similarity.** Most semi-supervised clustering methods calculate the similarity among data by measuring distances between labeled data and unlabeled in the same data space. In our framework, the self-similarity is used, i.e., a correlation coefficient of every pair of original and reconstructed data. This similarity reflects the reconstruction performance of WAE, and further reflects the similarity between labeled data and unlabeled data.

- **Progressive Framework.** The progressive framework is proposed to perform the clustering on an entire dataset in a systematic way. This framework is wrapped into the cyclic structure where two phases are involved. The first phase is called local-clustering phase while the second phase is called re-clustering phase.

- **Seeded Progressive Framework.** To apply this work on weakly-supervised semantic segmentation, the seeded progressive framework is proposed where the initial labeled data and clustering space are generated by using the computer vision techniques. Apart from above, there are five modifications to simplify the previous framework.

- **Using any semi-supervised clustering method to improve segmentation performances.** As our work can be viewed as a knowledge augmentation method to increase a number of labeled data, any other related methods can replacement ours framework.

This thesis proposes many novel methods for different tasks in ML and CV including representation learning, clustering and segmentation. There are room for improvement in respect of model architecture, similarity computing and clustering framework. Therefore, this thesis can be concluded by suggesting several possible research directions in the future:

- **Utilizing self-supervised learning methods for visual feature extraction.** In our experiment for image data, deep convolutional features are extracted from the DCNN model without involving any pretrained weights. Instead, deep self-supervised neural network can be utilized to extract visual feature of images. Self supervised learning method is to train the neural network with supervisory signals that are generated from the data itself (self-supervision) by leveraging its structure [37]. There are no labeled data involved in the leaning process. In our future work, any trained self-supervised models can be applied as visual feature extractor for any image datasets.

- **Extending to fully-unsupervised clustering task.** During this thesis, a small number of labeled data was utilized to guide the clustering process. As the weak autoencoder can be viewed as a local classifier to group one data to another, it is not naturally necessary to use any label information. Therefore, an existing unsupervised clustering method can be combined with our weak autoencoder. Besides, this modification can make our framework more robust, structural and systematic.

- **Integrating the proposed framework into weakly-supervised segmentation models in an end-to-end way.** Currently, the proposed framework can only be used as a pre-processing method before training segmentation models. In the future, the framework can be simplified as a module, and integrate it into segmentation models for shorting a total processing time.

Bibliography

[1] X. Zhang, Y. Wei, J. Feng, Y. Yang, and T. S. Huang, "Adversarial complementary learning for weakly supervised object localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1325–1334, 2018.

[2] Y. Yang, Q. J. Wu, and Y. Wang, "Autoencoder with invertible functions for dimension reduction and image reconstruction," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 7, pp. 1065–1079, 2016.

[3] A. Kolesnikov and C. H. Lampert, "Seed, expand and constrain: Three principles for weakly-supervised image segmentation," in *European conference on computer vision*, pp. 695–711, Springer, 2016.

[4] Z. Huang, X. Wang, J. Wang, W. Liu, and J. Wang, "Weakly-supervised semantic segmentation network with deep seeded region growing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7014–7023, 2018.

[5] J. Fan, Z. Zhang, T. Tan, C. Song, and J. Xiao, "Cian: Cross-image affinity net for weakly supervised semantic segmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 10762–10769, 2020.

[6] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[7] O. Chapelle, B. Schölkopf, and A. Zien, "Introduction to semi-supervised learning," 2006.

[8] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.

[9] X. Yang, Z. Song, I. King, and Z. Xu, "A survey on deep semi-supervised learning," *arXiv preprint arXiv:2103.00550*, 2021.

[10] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National science review*, vol. 5, no. 1, pp. 44–53, 2018.

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

[12] Y. Yang, Q. M. J. Wu, X. Feng, and T. Akilan, "Recomputation of the dense layers for performance improvement of dcnn," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 11, pp. 2912–2925, 2020.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[14] H. Pham, Z. Dai, Q. Xie, and Q. V. Le, "Meta pseudo labels," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11557–11568, 2021.

[15] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *International conference on machine learning*, pp. 284–293, PMLR, 2018.

[16] D. A. Ross, M. Deroche, and T. J. Palmeri, "Not just the norm: Exemplar-based models also predict face aftereffects," *Psychonomic bulletin & review*, vol. 21, no. 1, pp. 47–70, 2014.

[17] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, "Representational learning with elms for big data," 2013.

[18] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967.

[19] S. Basu, A. Banerjee, and R. Mooney, "Semi-supervised clustering by seeding," in *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*, Citeseer, 2002.

[20] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, *et al.*, "Constrained k-means clustering with background knowledge," in *Icml*, vol. 1, pp. 577–584, 2001.

[21] D. H. Ballard, "Modular learning in neural networks.," in *AAAI*, vol. 647, pp. 279–284, 1987.

[22] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[23] Y. Yang and Q. J. Wu, "Multilayer extreme learning machine with subnetwork nodes for representation learning," *IEEE transactions on cybernetics*, vol. 46, no. 11, pp. 2570–2583, 2015.

[24] C. Ruiz, M. Spiliopoulou, and E. Menasalvas, "C-dbscan: Density-based clustering with constraints," in *International workshop on rough sets, fuzzy sets, data mining, and granular-soft computing*, pp. 216–223, Springer, 2007.

[25] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *kdd*, vol. 96, pp. 226–231, 1996.

[26] L. Zheng and T. Li, "Semi-supervised hierarchical clustering," in *2011 IEEE 11th International Conference on Data Mining*, pp. 982–991, IEEE, 2011.

[27] S. Miyamoto and A. Terami, "Semi-supervised agglomerative hierarchical clustering algorithms with pairwise constraints," in *International Conference on Fuzzy Systems*, pp. 1–6, IEEE, 2010.

[28] I. Davidson and S. S. Ravi, "Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results," *Data mining and knowledge discovery*, vol. 18, no. 2, pp. 257–282, 2009.

[29] D. Pelleg and D. Baras, "K-means with large and noisy constraint sets," in *European Conference on Machine Learning*, pp. 674–682, Springer, 2007.

[30] I. Davidson and S. Ravi, "Clustering with constraints: Feasibility issues and the k-means algorithm," in *Proceedings of the 2005 SIAM international conference on data mining*, pp. 138–149, SIAM, 2005.

[31] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[32] D. T. Pham, S. S. Dimov, and C. D. Nguyen, "Selection of k in k-means clustering," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 219, no. 1, pp. 103–119, 2005.

[33] E. Bair, "Semi-supervised clustering methods," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 5, no. 5, pp. 349–361, 2013.

[34] X. Mai, J. Cheng, and S. Wang, "Research on semi supervised k-means clustering algorithm in data mining," *Cluster Computing*, vol. 22, no. 2, pp. 3513–3520, 2019.

[35] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, "S4l: Self-supervised semi-supervised learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1476–1485, 2019.

[36] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.

[37] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[38] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *arXiv preprint arXiv:2001.07685*, 2020.

[39] O. Henaff, "Data-efficient image recognition with contrastive predictive coding," in *International Conference on Machine Learning*, pp. 4182–4192, PMLR, 2020.

[40] M. Zhang, Y. Zhou, J. Zhao, Y. Man, B. Liu, and R. Yao, "A survey of semi- and weakly supervised semantic segmentation of images," *Artificial Intelligence Review*, pp. 1–30, 2019.

[41] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[42] L. Chan, M. S. Hosseini, and K. N. Plataniotis, "A comprehensive analysis of weakly-supervised semantic segmentation in different image domains," *International Journal of Computer Vision*, vol. 129, no. 2, pp. 361–384, 2021.

[43] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The apolloscape open dataset for autonomous driving and its application," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 10, pp. 2702–2719, 2019.

[44] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv preprint arXiv:1805.04687*, vol. 2, no. 5, p. 6, 2018.

[45] G. Aresta, T. Araújo, S. Kwok, S. S. Chennamsetty, M. Safwan, V. Alex, B. Marami, M. Prastawa, M. Chan, M. Donovan, *et al.*, "Bach: Grand challenge on breast cancer histology images," *Medical image analysis*, vol. 56, pp. 122–139, 2019.

[46] K. Sirinukunwattana, J. P. Pluim, H. Chen, X. Qi, P.-A. Heng, Y. B. Guo, L. Y. Wang, B. J. Matuszewski, E. Bruni, U. Sanchez, *et al.*, "Gland segmentation in colon histology images: The glas challenge contest," *Medical image analysis*, vol. 35, pp. 489–502, 2017.

[47] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, and R. Raskar, "Deepglobe 2018: A challenge to parse the earth through satellite images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 172–181, 2018.

[48] C. Tian, C. Li, and J. Shi, "Dense fusion classmate network for land cover classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 192–196, 2018.

[49] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[50] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[51] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[52] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[53] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818, 2018.

[54] J. Dai, K. He, and J. Sun, "Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, pp. 1635–1643, 2015.

[55] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille, "Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation," in *Proceedings of the IEEE international conference on computer vision*, pp. 1742–1750, 2015.

[56] D. Lin, J. Dai, J. Jia, K. He, and J. Sun, "Scribblesup: Scribble-supervised convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3159–3167, 2016.

[57] P. Vernaza and M. Chandraker, "Learning random-walk label propagation for weakly-supervised semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7158–7166, 2017.

[58] W. Lu, D. Gong, K. Fu, X. Sun, W. Diao, and L. Liu, "Boundarymix: Generating pseudo-training images for improving segmentation with scribble annotations," *Pattern Recognition*, vol. 117, p. 107924, 2021.

[59] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei, "What's the point: Semantic segmentation with point supervision," in *European conference on computer vision*, pp. 549–565, Springer, 2016.

[60] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, 2016.

[61] R. Adams and L. Bischof, "Seeded region growing," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 6, pp. 641–647, 1994.

[62] H. Jiang, J. Wang, Z. Yuan, Y. Wu, N. Zheng, and S. Li, "Salient object detection: A discriminative regional feature integration approach," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2083–2090, 2013.

[63] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive science*, vol. 9, no. 1, pp. 147–169, 1985.

[64] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[65] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, pp. 153–160, 2007.

[66] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in *Artificial intelligence and statistics*, pp. 448–455, PMLR, 2009.

[67] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

[68] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.," *Journal of machine learning research*, vol. 11, no. 12, 2010.

[69] A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[70] A. Makhzani and B. Frey, "K-sparse autoencoders," *arXiv preprint arXiv:1312.5663*, 2013.

[71] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Icml*, 2011.

[72] S. Rifai, G. Mesnil, P. Vincent, X. Muller, Y. Bengio, Y. Dauphin, and X. Glorot, "Higher order contractive auto-encoder," in *Joint European conference on machine learning and knowledge discovery in databases*, pp. 645–660, Springer, 2011.

[73] D. P. Kingma and M. Welling, "Auto-encoding variational bayes in 2nd international conference on learning representations," in *ICLR 2014-Conference Track Proceedings*, 2014.

[74] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[75] C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt, "Advances in variational inference," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 2008–2026, 2018.

[76] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[77] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein auto-encoders," *arXiv preprint arXiv:1711.01558*, 2017.

[78] L. Mescheder, S. Nowozin, and A. Geiger, "Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks," in *International Conference on Machine Learning*, pp. 2391–2400, PMLR, 2017.

[79] K. Sun, J. Zhang, C. Zhang, and J. Hu, "Generalized extreme learning machine autoencoder and a new deep neural network," *Neurocomputing*, vol. 230, pp. 374–381, 2017.

[80] D. Pathak, E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional multi-class multiple instance learning," *arXiv preprint arXiv:1412.7144*, 2014.

[81] Y. Wei, X. Liang, Y. Chen, X. Shen, M.-M. Cheng, J. Feng, Y. Zhao, and S. Yan, "Stc: A simple to complex framework for weakly-supervised semantic segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 11, pp. 2314–2320, 2016.

[82] K. K. Singh and Y. J. Lee, "Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization," in *2017 IEEE international conference on computer vision (ICCV)*, pp. 3544–3553, IEEE, 2017.

[83] D. Kim, D. Cho, D. Yoo, and I. So Kweon, "Two-phase learning for weakly supervised object localization," in *Proceedings of the IEEE international conference on computer vision*, pp. 3534–3543, 2017.

[84] Y. Wei, J. Feng, X. Liang, M.-M. Cheng, Y. Zhao, and S. Yan, "Object region mining with adversarial erasing: A simple classification to semantic segmentation approach," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1568–1576, 2017.

[85] K. Li, Z. Wu, K.-C. Peng, J. Ernst, and Y. Fu, "Tell me where to look: Guided attention inference network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9215–9223, 2018.

[86] Y. Wei, H. Xiao, H. Shi, Z. Jie, J. Feng, and T. S. Huang, "Revisiting dilated convolution: A simple approach for weakly-and semi-supervised semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7268–7277, 2018.

[87] S. J. Oh, R. Benenson, A. Khoreva, Z. Akata, M. Fritz, and B. Schiele, "Exploiting saliency for object segmentation from image level labels," in *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 5038–5047, IEEE, 2017.

[88] X. Wang, S. You, X. Li, and H. Ma, "Weakly-supervised semantic segmentation by iteratively mining common object features," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1354–1362, 2018.

[89] J. Lee, E. Kim, S. Lee, J. Lee, and S. Yoon, "Ficklenet: Weakly and semi-supervised semantic image segmentation using stochastic inference," in *Proceed-*

ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5267–5276, 2019.

[90] Y. Zeng, Y. Zhuge, H. Lu, and L. Zhang, "Joint learning of saliency detection and weakly supervised semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7223–7233, 2019.

[91] J. Ahn, S. Cho, and S. Kwak, "Weakly supervised learning of instance segmentation with inter-pixel relations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2209–2218, 2019.

[92] W. Shimoda and K. Yanai, "Self-supervised difference detection for weakly-supervised semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5208–5217, 2019.

[93] G. Sun, W. Wang, J. Dai, and L. Van Gool, "Mining cross-image semantics for weakly supervised semantic segmentation," in *European Conference on Computer Vision*, pp. 347–365, Springer, 2020.

[94] Y. Wang, J. Zhang, M. Kan, S. Shan, and X. Chen, "Self-supervised equivariant attention mechanism for weakly supervised semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12275–12284, 2020.

[95] Y.-T. Chang, Q. Wang, W.-C. Hung, R. Piramuthu, Y.-H. Tsai, and M.-H. Yang, "Weakly-supervised semantic segmentation via sub-category exploration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8991–9000, 2020.

[96] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[97] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, pp. 478–487, 2016.

[98] J. Chang, G. Meng, L. Wang, S. Xiang, and C. Pan, "Deep self-evolution clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 809–823, 2020.

[99] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 171–184, 2013.

[100] G. Huang, S. Song, J. N. D. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2405–2417, 2014.

[101] Y. Yang and Q. M. J. Wu, "Extreme learning machine with subnetwork hidden nodes for regression and classification," *IEEE Transactions on Cybernetics*, vol. 46, pp. 2885–2898, Dec 2016.

[102] Y. Yang, Y. Wang, Q. M. Jonathan Wu, X. Lin, and M. Liu, "Progressive learning machine: A new approach for general hybrid system approximation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 9, pp. 1855–1874, 2015.

[103] X. Wang, B. Qian, and I. Davidson, "On constrained spectral clustering and its applications," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 1–30, 2014.

[104] V. Antoine, B. Quost, M.-H. Masson, and T. Denoeux, "Cevclus: evidential clustering with instance-level constraints for relational data," *Soft Computing*, vol. 18, no. 7, pp. 1321–1335, 2014.

[105] K. Kamvar, S. Sepandar, K. Klein, D. Dan, M. Manning, and C. Christopher, "Spectral learning," in *International Joint Conference of Artificial Intelligence*, Stanford InfoLab, 2003.

[106] Y. Jia, S. Kwong, and J. Hou, "Semi-supervised spectral clustering with structured sparsity regularization," *IEEE Signal Processing Letters*, vol. 25, no. 3, pp. 403–407, 2018.

[107] L. Yang, X. Cao, D. Jin, X. Wang, and D. Meng, "A unified semi-supervised community detection framework using latent space graph regularization," *IEEE Transactions on Cybernetics*, vol. 45, no. 11, pp. 2585–2598, 2015.

[108] Y. Ren, K. Hu, X. Dai, L. Pan, S. C. Hoi, and Z. Xu, "Semi-supervised deep embedded clustering," *Neurocomputing*, vol. 325, pp. 121–130, 2019.

[109] A. Shukla, G. S. Cheema, and S. Anand, "Semi-supervised clustering with neural networks," *arXiv preprint arXiv:1806.01547*, 2018.

[110] X. Li, H. Yin, K. Zhou, and X. Zhou, "Semi-supervised clustering with deep metric learning and graph embedding," *World Wide Web*, pp. 1–18, 2019.

[111] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[112] J. C. Van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. Smeulders, "Kernel codebooks for scene categorization," in *European conference on computer vision*, pp. 696–709, Springer, 2008.

[113] L. Zhang, X. Zhen, and L. Shao, "Learning object-to-class kernels for scene classification," *IEEE Transactions on Image Processing*, vol. 23, no. 8, pp. 3241–3253, 2014.

[114] Z. Jiang, Z. Lin, and L. S. Davis, "Label consistent k-svd: Learning a discriminative dictionary for recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2651–2664, 2013.

[115] Jianchao Yang, Kai Yu, Yihong Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1794–1801, 2009.

[116] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3360–3367, 2010.

[117] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[118] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[119] Q. Zhang and B. Li, "Discriminative k-svd for dictionary learning in face recognition," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2691–2698, 2010.

[120] I. Ramirez, P. Sprechmann, and G. Sapiro, "Classification and clustering via dictionary learning with structured incoherence and shared features," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3501–3508, 2010.

[121] D. Wang and S. Kong, "A classification-oriented dictionary learning model: Explicitly learning the particularity and commonality across categories," *Pattern Recognition*, vol. 47, no. 2, pp. 885–898, 2014.

[122] L. Seidenari, G. Serra, A. D. Bagdanov, and A. Del Bimbo, "Local pyramidal descriptors for image recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 5, pp. 1033–1040, 2014.

[123] L. Bo, X. Ren, and D. Fox, "Multipath sparse coding using hierarchical matching pursuit," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 660–667, 2013.

[124] J. C. van Gemert, C. J. Veenman, A. W. M. Smeulders, and J. Geusebroek, "Visual word ambiguity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1271–1283, 2010.

[125] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 111–118, 2010.

[126] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 2169–2178, 2006.

[127] Lingqiao Liu, Lei Wang, and Xinwang Liu, "In defense of soft-assignment coding," in *2011 International Conference on Computer Vision*, pp. 2486–2493, 2011.

[128] Y. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2559–2566, 2010.

[129] H. Goh, N. Thome, M. Cord, and J. Lim, "Learning deep hierarchical visual feature coding," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 12, pp. 2212–2225, 2014.

[130] C. Cortes and V. Vapnik, "Support vector machine," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[131] S. Borjigin and P. K. Sahoo, "Color image segmentation based on multi-level tsallis–havrda–charvát entropy and 2d histogram using pso algorithms," *Pattern Recognition*, vol. 92, pp. 107–118, 2019.

[132] J. Fu, J. Liu, Y. Li, Y. Bao, W. Yan, Z. Fang, and H. Lu, "Contextual deconvolution network for semantic segmentation," *Pattern Recognition*, vol. 101, p. 107152, 2020.

[133] A. López-Cifuentes, M. Escudero-Viñolo, J. Bescós, and Á. García-Martín, "Semantic-aware scene recognition," *Pattern Recognition*, vol. 102, p. 107256, 2020.

[134] Z. Wang, R. Song, P. Duan, and X. Li, "Efnet: Enhancement-fusion network for semantic segmentation," *Pattern Recognition*, p. 108023, 2021.

[135] Y. Zhang, X. Sun, J. Dong, C. Chen, and Q. Lv, "Gpnet: Gated pyramid network for semantic segmentation," *Pattern Recognition*, vol. 115, p. 107940, 2021.

[136] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Pattern Recognition*, vol. 90, pp. 119–133, 2019.

[137] A. Chaudhry, P. K. Dokania, and P. H. Torr, "Discovering class-specific pixels for weakly-supervised semantic segmentation," *arXiv preprint arXiv:1707.05821*, 2017.

[138] Q. Hou, P.-T. Jiang, Y. Wei, and M.-M. Cheng, "Self-erasing network for integral object attention," *arXiv preprint arXiv:1810.09821*, 2018.

[139] J. Ahn and S. Kwak, "Learning pixel-level semantic affinity with image-level supervision for weakly supervised semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4981–4990, 2018.

[140] G. Huang, L. Kasun, H. Zhou, and C. Vong, "Representational learning with extreme learning machine for big data," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 31–34, 2013.

[141] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[142] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[143] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[144] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *2011 International Conference on Computer Vision*, pp. 991–998, IEEE, 2011.

[145] P. O. Pinheiro and R. Collobert, "From image-level to pixel-level labeling with convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1713–1721, 2015.

[146] S. Hong, D. Yeo, S. Kwak, H. Lee, and B. Han, "Weakly supervised semantic segmentation using web-crawled videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7322–7330, 2017.

[147] W. Luo, M. Yang, and W. Zheng, "Weakly-supervised semantic segmentation with saliency and incremental supervision updating," *Pattern Recognition*, vol. 115, p. 107858, 2021.

[148] L. Xu, H. Xue, M. Bennamoun, F. Boussaid, and F. Sohel, "Atrous convolutional feature network for weakly supervised semantic segmentation," *Neurocomputing*, vol. 421, pp. 115–126, 2021.

[149] J. Liu, C. Yu, B. Yang, C. Gao, and N. Sang, "Csenet: Cascade semantic erasing network for weakly-supervised semantic segmentation," *Neurocomputing*, 2020.

[150] Y. Li, Y. Liu, G. Liu, and M. Guo, "Weakly supervised semantic segmentation by iterative superpixel-crf refinement with initial clues guiding," *Neurocomputing*, vol. 391, pp. 25–41, 2020.