Universiteit Leiden
ICT in Business and the
Public Sector

Ayca Sirin Kindap
s3304159

A research on the effectiveness of robust learning training on unsupervised image classification

Master's Thesis

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Supervision

Dr. Marc Hilbert and Dr. Guus J. Ramackers
October 26, 2023

Abstract

Accurately classifying images is a crucial aspect of many image processing tasks. While supervised deep network structures are dominant in machine learning problems, they have limitations in certain cases. In consequence, unsupervised image classification approaches can be employed for specific tasks. Various unsupervised image clustering methods exist with different approaches. However, concerns have been raised about the performance limitations of these approaches and the effectiveness of combining them, which requires further exploration in future work. This research aims to compare the efficacy of unsupervised image classification approaches when exposed to different datasets and, if possible, improve the accuracy of unsupervised image clustering methods by combining different approaches.

Acknowledgements

This work was performed using the compute resources from the Academic Leiden Interdisciplinary Cluster Environment (ALICE) provided by Leiden University.

Contents

Abstract **i**

Acknowledgements **iii**

Acronyms and Abbreviations **ix**

1 Introduction **1**

1.1 Research Problem 1

1.2 Research Question and Contribution of the Study 4

1.2.1 Contributions 4

1.3 Objectives and Hypotheses 5

1.4 Methodology 5

1.5 Outline 7

2 Background **8**

2.1 Deep Learning and Traditional Machine Learning in Image Classification 8

2.2 Deep Learning Categories 9

2.3 Unsupervised Image Classification (Clustering) 10

2.3.1 Discrimitive Architectures 13

2.3.2 Application Areas: An Example Business Case 16

Advantages of Self-driven Cars 16

Visual Recognition Systems in Self-driven Cars 16

Market for Self-driven Cars and Role of Computer Vision . . . 17

Technology Overview 17

Real World Examples 18

2.3.3 Clustering Components 19

Pattern Representation 20

Definition of Pattern Proximity Measure 25

Clustering . . . 25
Assessment of Clusters . . . 26
2.3.4 Approaches . . . 26
Sequential Approach . . . 27
Joint Approach . . . 28
Multi-step refinement Approach . . . 29
Add-on module to improve unsupervised clustering . . . 30
2.3.5 Image Datasets . . . 31
2.4 Learning with Noisy Labels . . . 33

3 Methods **35**
3.1 IIC . . . 35
3.1.1 Loss Function . . . 36
3.1.2 Overall Pipeline . . . 37
Sample Repeats . . . 38
Sobel Filter . . . 39
Multiple Sub-heads . . . 40
Auxiliary Overclustering . . . 40
3.1.3 Experimental Details . . . 41
Network Architecture . . . 41
Training . . . 42
Evaluation . . . 42
3.2 RUC . . . 43
3.2.1 Loss Function . . . 44
3.2.2 Overall Pipeline . . . 45
Data Sampling . . . 45
Label Smoothing . . . 47
Semi-supervised Network . . . 48
Co-training . . . 52
Co-refurbishment . . . 52
3.2.3 Experimental Details . . . 53
Network Architecture . . . 53
Training . . . 53
Evaluation . . . 54

4 Experiments **56**
4.1 Experiment 0: Preliminary Work . . . 56

4.1.1 Reproduction of IIC . . . 56
Train-test Split for IIC . . . 59
4.1.2 Reproduction of RUC . . . 60
4.2 Experiment 1: Entropy-based Clustering and Confidence Relationship 62
4.2.1 Experimental Setup . . . 63
4.3 Experiment 2: Performance Gain of RUC applied on IIC . . . 63
4.3.1 Experimental Setup . . . 63
4.4 Experiment 3: Sampling Strategy of RUC . . . 64
4.4.1 Experimental Setup . . . 65

5 Results and Discussion 66
5.1 Result 1: Confidence Analysis of IIC . . . 66
5.1.1 Confidence Distribution . . . 66
5.1.2 Calibration Skills . . . 67
Discussion: . . . 68
5.2 Result 2: Performance Analysis IIC+RUC . . . 69
5.2.1 Calibration Skills of IIC+RUC . . . 69
Confidence Distribution . . . 69
Calibration Skills . . . 70
Discussion: . . . 70
5.2.2 Accuracy Performance of IIC+RUC . . . 71
Loss and Accuracy Change . . . 71
Discussion: . . . 73
Learning Rate Effect . . . 74
Discussion: . . . 74
5.3 Result 3: Performance Analysis of Sampling Strategies . . . 75
5.3.1 Base Case: SCAN+RUC . . . 75
Discussion: . . . 77
5.3.2 Our Case: IIC+RUC . . . 77
Discussion: . . . 80

6 Conclusion and Future Work 85
6.1 About Sampling Strategy . . . 85
6.2 About Learning Rate . . . 86

A 87
A.1 Package version Comparison Table . . . 87

A.2 Loss Accuracy Curves of Sampling Strategies 87

Acronyms and Abbreviations

PCA	Principle Component Analysis
ML	Machine Learning
AI	Artificial Intelligence
DL	Deep Learning
DNN	Deep Neural Network
SSL	Self-supervised Learning
SSRL	Self-supervised Representation Learning
AE	Auto-encoder
AR	Auto-regressive
PRP	Predict Relative Position
MI	Maximize Mutual Information
k-NN	k Nearest Neighbor
NLL	Noisy-Label Learning
CNN	Convolutional Neural Network
MLP	Multilayer Perceptron
RNN	Recurrent Neural Network
SIFT	Scale Invariant Feature Transform
HOG	Histogram of Gradients
LBG	Local Binary Patterns
ECE	Expected Calibration Error
SGD	Stochastic Gradient Descent

Chapter 1

Introduction

In this section, we introduce the research problem and the associated hypotheses that have guided the development of this study. Each hypothesis is closely linked to an objective, and each of these objectives is addressed through an experiment in Chapter 4. We also cover the contributions of this research and provide an overview of the experimental steps that constitute the methodology. The chapter concludes with an outline designed for clarity and direction to the reader.

1.1 Research Problem

An emerging field of computer vision is the image and video processing. With the advances in high computational power and the availability of large-scale datasets, notable results have been achieved in the field of image classification using deep learning (DL) techniques [1]. Applications in various areas such as entertainment, healthcare, transport, and security leverage state-of-the-art techniques [2]. An example of such an application is self-driving cars, which rely on AI-based visual recognition systems for image classification [1]. The increased usage of AI-powered automation and robotics in everyday life applications has resulted in a demand for research on DL techniques to improve existing algorithms according to specific requirements.

DL is used to extract and learn features from raw data using supervised and unsupervised approaches. Supervised learning gained prominence in the deep learning field due to its high accuracy classification rates. Contemporary literature showcases state-of-the-art pre-trained supervised learning methods with various architectures; however, they are not without their limitations. While a substantial amount of labelled data is pivotal for the success of Deep Neural Networks (DNNs), the ne-

cessity for vast volumes of data for learning can hinder the performance of DNNs. The acquisition of accurate labels entails an expensive manual labelling process that demands a high level of expertise [3, 4]. Alternatively, the use of online queries and crowdsourcing has garnered attention as an alternative approach, yet it introduces a new challenge: the unreliability of labels due to the human factor [5]. On the contrary, unsupervised learning offers the ability to identify patterns and regularities within data without the need for explicit labels.

Image classification is a fundamental task in computer vision, involving the grouping of image samples into predefined categories [6]. Despite decades of research, which has led to the development of numerous algorithms and architectures, it remains a fascinating and important topic for researchers. This is due to the numerous opportunities and challenges associated with the field. The high-dimensional nature of visual data provides a rich source of information [2], and the exponential growth of available visual data offers tremendous potential for both research and business. However, the complexity of visual data also poses significant challenges, the manual labelling process of images is particularly difficult and consequently becomes an unfeasible task for extensive volumes of data. Despite the challenge in the context of image classification, the paradigm of unsupervised image classification enables machine learning models to be able to learn the inherent information in images without relying on assigned labels.

Unsupervised image classification could be used to discover patterns and structures in visual data that may not be apparent through the supervised learning methods. One area where unsupervised learning is being explored is in the field of racing, where on-board cameras inside race cars record and stream video. By converting these videos into frames of images, machine learning models can be trained to detect and classify different cars within the scene. However, the task of classifying different cars presents an obstacle, as labels associated with images of cars are not available before the training of the model. This is where unsupervised image clustering is used instead of supervised image classification which is inappropriate for the specific problem due to its limitations. While the racing industry is only one of many fields that can benefit from unsupervised image classification, there are numerous other image classification problems across various industries [7, 8] that require advanced models to gain insights from visual data.

The objective of unsupervised classification (clustering) is to group similar data into clusters without the supervision of annotation [4]. Clustering methods group similar data points based on some similarity metric or distance measure. How-

ever, traditional clustering approaches often generate degenerate or collapsed results, where one class dominates over the output labels [9, 4, 3]. To address this issue, researchers have proposed innovative clustering objectives. For instance, Ji *et al.* [4] suggest that maximizing the mutual information between the class assignments of image pairs is a novel approach for the clustering task [4]. Park *et al.* [9] categorize this type of clustering as joint approach clustering, where an end-to-end pipeline performs both representation learning and class assignment tasks concurrently [9]. Even though algorithms with an innovative objective [4] have achieved better performance compared to traditional clustering techniques, concerns regarding overconfident results have been raised in the literature.

Deep networks are prone to overfitting the noise, outliers, and other irregularities present in the data during training [10]. Faulty prediction of a sample leads to misclassification of later samples, and consequently overconfidence issue [9]. To address the overconfidence issue, Park *et al.* [9] have proposed the use of robust learning modules to improve the performance of clustering methods. Robust training corresponds to the methods that are developed to increase the robustness of the algorithms for noisy datasets. The proposed robust learning training method in the paper is based on semi-supervised learning with a flexible architecture so that it can be used as an add-on module on the pseudo-labels that are generated by any unsupervised image classification algorithm. The proposed method involves techniques from robust training categories such as robust regularization, robust loss functions, loss adjustments and sample selection [11].

Learning from noisy labels has been getting substantial attention recently due to the inevitable noise factor present in real-world datasets. With the observed results of the latest state-of-the-art approaches in Noisy-Label Learning (NLL)[12], the potential benefits of robust training to mitigate overfitting are indisputable. Despite the potential benefits, there is a lack of empirical evidence and comprehensive evaluation of the effectiveness of robust training to improve unsupervised image clustering in the context of overconfident results, as also mentioned by Park *et al.* [9]. While RUC [9] has been applied with multi-step refinement approach clustering algorithms, to the best of our knowledge, its application to joint approach algorithms which are prone to generate overconfident results has not been executed yet.

1.2 Research Question and Contribution of the Study

The research question addressed in this study is as follows: *How much of a performance improvement from an unsupervised image classification perspective could be achieved if robust training techniques are applied to joint approach clustering?*

The study is divided into 2 parts to apply the objectives for each. In the first part of the study, the overconfidence issue of joint approach clustering algorithms will be presented by the confidence distribution and the performance comparisons will be made after RUC is applied to the algorithm. The goal of this experiment is to show the presence of the overconfidence issue inherited in joint approach clustering and to observe if the proposed RUC [9] is capable of providing a better calibration as stated.

In the second part of this study, we will replace the confidence-based sample selection strategy used in RUC [9] with the Class-wise Small-loss Selection (CSS) strategy introduced by Wang et al. [13]. We will then compare the performance of these two different sample selection methods. The modification in the sampling strategy holds the potential to enhance the performance achieved by RUC. The confidence-based strategy, which categorizes samples based on high confidence scores as clean, may lead to the inclusion of a substantial number of samples with corrupted labels in the clean set when joint approach clustering algorithms exhibit overconfidence. Consequently, Class-wise Small-loss selection could achieve higher precision than confidence-based selection in forming a clean set.

1.2.1 Contributions

The unique contributions of this study are as follows:

1. Application of RUC [9] on joint approach clustering to improve its unsupervised image classification performance
2. Modification of sampling strategy on RUC [9] with a strategy proposed in ProMix [13]

The reasons why the contributions mentioned above are unique:

- To the best of our knowledge, the robust training approach in contribution 1 has not been applied to a joint approach clustering and corresponding performance comparisons are not conducted.

- By tailoring a technique from a different algorithm (ProMix) into an existing method (RUC), novel insights and improvements that might not have been explored within the original scope of RUC could be achieved.

1.3 Objectives and Hypotheses

Each objective has a corresponding hypothesis and a deliverable in Chapter 4 associated with it. In total, there are three objectives and three hypotheses listed below.
Objectives:

1. Identify the relationship between entropy-based joint approach clustering and confidence of labels
2. Identify the performance gain[1] achievable when robust learning training approach proposed in RUC[9] is applied to the joint approach clustering algorithm, namely IIC
3. Search sampling strategies in the literature, apply and determine the sampling strategy which has the potential to form a rich set [2] of clean samples

Hypotheses:

Hypothesis 1 (H1)**.** *Entropy-based joint approach clustering generates overconfident results.*

Hypothesis 2 (H2)**.** *RUC [9] improves the accuracy and calibration performance of the joint approach clustering it is applied to.*

Hypothesis 3 (H3)**.** *Confidence-based sampling strategy outputs a less decent clean set* [3] *when applied to noisy labels generated by an overconfident joint approach clustering.*

1.4 Methodology

This study follows a quantitative research methodology. The type of research is experimental research which focuses on setting up appropriate experiments following scientific code to establish the cause-effect relationship among the group of variables

[1] Accuracy, calibration
[2] by comparing Precision and Recall metrics
[3] compared to metric-based and Class-wise Small-loss Selection (CSS)

that concern the study. The steps designed according to the aim of the study incorporate the following steps:

1. Dataset Selection: Choose a dataset among benchmark datasets that is suitable for unsupervised image classification and the limitations surrounding the study.

2. Joint Approach Clustering Algorithm Selection: Select and implement a joint approach clustering algorithm.

3. Reproduction of Results: Train the algorithms on the chosen dataset. Verify the reliability of the reproduction by assessing the outcomes of the chosen algorithm. Observe the performance of each algorithm and use it as the reference value for later experiments.

4. Confidence and Calibration Analysis: Analyse the confidence distribution of the joint approach clustering algorithm. Observe the overconfidence issue and its impact on performance.

5. RUC Application and Accuracy Comparison: Apply the Robust training add-on module (RUC) approach to the joint approach clustering algorithm to address the overconfidence issue. Compare the performance of the algorithms before and after applying RUC. Measure the improvement achieved in unsupervised image classification accuracy.

6. Calibration Comparison: In addition to the accuracy performance, investigate the effect of RUC on the model calibration to the joint approach clustering.

7. Sampling Strategy Selection: Conduct research about sampling strategies in the literature, and choose at least one that may increase the performance of RUC. Replace the sample selection strategy of RUC with the chosen strategy. Train the modified RUC algorithm on the dataset and evaluate its performance.

8. Sampling Strategy Comparison: Compare the performance of the sampling strategies from the original RUC to the proposed strategy when applied to predictions of the joint approach clustering algorithm. Measure the accuracy difference that is achieved after the modification to determine the best sampling strategy for the joint approach algorithm.

1.5 Outline

In Chapter 1, an introduction to the research topic where the research questions, objectives and hypotheses are outlined. Crucial concepts and distinctions between machine learning and deep learning; and the categorisation of deep learning are discussed in Chapter 2. Findings from the prior research about unsupervised image classification were reviewed also in this chapter including components of clustering, different approaches in the field, related algorithms, architectures and datasets. Another essential concept of NLL is summarised. The chosen unsupervised classification algorithms are scrutinized by discussing the details of the loss function, pipeline and experimental setup in Chapter 3. Chapter 4 gives detailed information about the implementation of the experiments corresponding to each hypothesis in Chapter 1 and the results obtained from each experiment. The results are analysed in Chapter 5 to interpret the findings gathered throughout the study. We evaluated whether our hypotheses were supported by the findings. Lastly, the study is summarised and a synthesis of the key findings is included in Chapter 6 with an addition of possible future work that could further extend the current research.

Chapter 2

Background

In this chapter, we start with an examination of the fundamental distinctions between deep learning architectures and their traditional counterparts. Within the realm of deep learning, a categorisation of learning approaches is included. These initial chapters serve a dual purpose: firstly, to shed light on the prevailing trends in the dynamic landscape of artificial intelligence, and secondly, to pinpoint the specific domain to which this research contributes, namely unsupervised image classification. Following the two sections, we explored the related theories, essential concepts, and noteworthy findings from previous research in-depth for unsupervised image classification. Lastly, the chapter is concluded with a summary of noisy-label learning.

2.1 Deep Learning and Traditional Machine Learning in Image Classification

Deep Learning, a sub-field of Machine Learning (ML), has the principle of automated learning. The main distinction of this superior technology compared to traditional ML algorithms is the multi-layer neural network architecture, which enables advanced learning capabilities [14]. Each layer of the network consists of multiple neurons (also referred to as processing units), and together (almost) all contribute to the learning process by establishing non-linear mappings between input and output [15]. Deep neural networks (DNN) can imitate human analytical processes, which yield successful results on many real-world tasks such as classification and regression. Deep learning has the potential to extract abstract representations from data, capturing informative hidden features to be utilized in complex analysis and

decision-making. The main challenge of constructing a DNN for a given task lies in effectively training the model with the given input data which includes selecting a suitable model by taking into account the nature of the data and the learning capabilities of deep learning techniques [14].

While the input data for a DL model can take a wide variety of forms, such as text, images, videos, and graphs, this study focuses on digital image data. Digital image data is typically represented by 2D matrices consisting of numerical values that correspond to pixel intensities for each of the three colour channels (red, green, and blue). The combination of channels forms an RGB image that depicts an object or a scene of interest.

2.2 Deep Learning Categories

From the perspective of learning tasks, deep neural network-based techniques are categorized into supervised, unsupervised, and semi-supervised and reinforcement by Sarker [14]. A slightly different approach for the categorization is shared by Alzubaidi *et al.* [16]. The learning techniques are classified as supervised, semi-supervised (also referred to as partially supervised) and unsupervised. Deep reinforcement learning is not considered a separate category but rather a sub-category for semi-supervised learning, while supervised and unsupervised applications exist. Due to the more compact structure and simplicity of the latter taxonomy, for the remainder of the study, deep learning schemas will be grouped into the major three categories - namely supervised, unsupervised and semi-supervised.

The availability of training labels plays a pivotal role in shaping the divergence of deep learning approaches. When labels are present for the training data, a supervised learning approach is typically employed. In this paradigm, the machine's objective is to discern the relationships between data samples and their corresponding labels so that the knowledge acquired can be utilized to classify the unseen data samples which can be also referred to as test data. A significant drawback of supervised methods lies in their substantial reliance on a high volume of labelled data. Especially for computer vision tasks where large-scale datasets are essential for training multi-layered deep neural networks [17]. Each data sample in the dataset should be accompanied by a human-annotated label. These labels serve as references for optimizing a large number of parameters at each layer, achieved by minimizing the loss function, which is composed of the machine's class prediction and the actual label of the data sample.

The requirement of labels in supervised learning creates a deployment bottleneck, to make use of fastly-growing data without the process of manual labelling, an unsupervised learning schema is developed. In unsupervised learning, the labels are not provided so the machine learns the relationship between the features of data samples without the supervision of the class assignment information. By following this learning approach, one may take away the burden of labelling. Self-supervised learning which started to get a significant amount of attention in the field of deep learning, particularly in the applications of feature representation learning is considered under the category of unsupervised learning by [18, 19]. Further detail about Self-supervised Learning (SSL) is available under the Section 2.3.3.

To mitigate the heavy reliance on labels while still making use of a comparatively limited amount of labelled data, one can adopt a semi-supervised approach. Semi-supervised learning represents a learning paradigm that avails the advantages of both supervised and unsupervised learning methods. Unlabeled data can play a pivotal role in enhancing the machine's learning capabilities within this framework, wherein machine performance trained in a semi-supervised paradigm outperforms that of a machine trained solely on a labelled dataset [20]. To realize this, unlabeled data must provide additional insights about the data distribution that cannot be derived from the labelled data.

The supervised learning models are task-dependent where the model learns based on the labelled data. Unlike supervised learning, the lack of supervision from labels renders the unsupervised learning data-driven. The business case influencing this study (see Section 2.3.2) is constrained by the absence of labels, consequently the content of the succeeding sections focuses on the unsupervised learning paradigm to provide a consistent style. However, supervised learning will be mentioned in Sections 2.3 and 2.3.3 to address the possible problems and limitations when compared to unsupervised learning and its sub-field SSL. Furthermore, semi-supervised learning will be introduced again in Section 2.4 within the context of robust learning.

2.3 Unsupervised Image Classification (Clustering)

According to the taxonomy by Sarker [14], deep learning techniques are divided into 3 categories: discriminative, generative, and hybrid learning. The nature of the DL techniques differs based on their learning capabilities when exposed to different environments. The aforementioned categorisation is based on the distinct learning capabilities exhibited by various Deep Neural Network (DNN) architectures.

Discriminative DNNs are particularly noteworthy for their ability to produce representative features and excel in distinguishing input data among different classes, especially in label-present scenarios. While discriminative DNNs can be applied to supervised, unsupervised, and semi-supervised learning settings, the most prevalent form of deep learning for a classification task is supervised learning. Its popularity can be attributed to its proven success which overshadows most of the unsupervised learning algorithms. Nevertheless, the long-term perspective, as suggested by LeCun *et. al.* [15], anticipates a shift towards greater popularity for unsupervised and semi-supervised learning.

One of the primary reasons for this shift is the limitations of supervised learning as mentioned before. Although supervised neural networks have demonstrated undeniable success, the outgrowing data and the labour-intensive annotation process, particularly in domains requiring specialized expertise or involving confidential information, pose significant challenges. Another reason is related to the learning principles followed by the network structures. Network structures aim to replicate the learning patterns observed in humans and animals, which are predominantly unsupervised. In this context, it is increasingly likely that future developments in deep learning, especially in computer vision tasks, will align more closely and adapt their nature with the natural learning processes they seek to emulate. As a result, unsupervised learning has begun to garner considerable attention within the machine learning community. Given the inherent nature and learning capabilities of discriminative DNNs, supervised or unsupervised, they are well-suited for classification tasks and continue to be a prevalent choice for such applications.

The network structures that belong to the discriminative learning category learn a mapping function between the training data which for example can consist of images of different animals and the associated class of each image. During training, the model adjusts the weights in each layer by applying the backpropagation principle to minimize the loss function [15]. After the training the model becomes capable of classifying unseen data into the predetermined classes, the model could be imagined as a black box which has a discriminative function inside, the input is the image data and the output is the class assignments.

By exploiting the multi-layer structure of DNNs, it is possible to learn very complicated functions and even too exactly which might raise the issue of overfitting. DNNs are prone to overfitting the noise, outliers, and other irregularities present in the data which affects their performance during prediction for the unseen data. In case the model is overfitted to the training data, since it will not be able to

generalize, the classification performance will be subsequently low. The challenge could be alleviated by applying regularization where the objective is to restrict the subset of hypothesis space for the model [10].

By leveraging the multi-layer structure DNNs, it becomes possible to learn highly complex functions, even too precisely that this can give rise to the issue of overfitting. DNNs are susceptible to overfitting noise, outliers, and other irregularities within the data, which can negatively impact their predictive performance on unseen data. When a model is overfitted to the training data and lacks the ability to generalize, its classification performance tends to be notably reduced. This challenge can be addressed through the application of regularization techniques, with the objective of constraining the hypothesis space of the model [10]. The hypothesis space represents the range of all potential mappings, including the optimal mapping, that the machine strives to discover during training.

Regularisation methods can be categorized as either implicit or explicit. Explicit regularisation includes techniques such as data augmentation, weight decay (L2 regularisation), and dropout. While implicit regularisation involves approaches like early stopping and batch normalization.

Data augmentation involves the alteration of images through various transformations, selected based on the specific task at hand. In Section 2.3.3, examples of different data augmentations and their purposes are discussed in detail when examining examples of SSL.

Weight decay, also referred to as L2 regularisation, imposes constraints on the model's weights by adding a regularisation term to the loss function that penalizes large weight values. The regularisation term is the square of the model parameters and its effect is controlled by a hyperparameter. This constraint is introduced to encourage the model to keep the loss function with the added square of the parameters which effectively curbs the growth of parameter values.

Dropout is a technique that randomly deactivates nodes within each layer of the network during training. The dropout process, as the name suggests, involves randomly selecting nodes and dropping them out, which helps prevent the model from becoming overly reliant on specific nodes and promotes better generalization.

Batch normalisation follows the principle of normalizing input data before feeding it into a deep learning model which is a standard practice to ease the convergence process. The normalisation is applied after each activation function of a hidden layer before it enters the next one. With the normalisation, the loss landscape achieves a more consistent shape among its parameters and a more uniform gradient across the

parameters. Batch normalization layers include learnable parameters for mean and standard deviation, allowing the model to adapt and optimize its performance during training. Batch normalisation serves to alleviate the model's reliance on specific initialisation for convergence and makes it less susceptible to the challenges posed by higher learning rates [21].

These regularisation techniques play an essential role in enhancing the robustness and generalization capability of deep learning models while counteracting the possibility of overfitting. The discriminative architectures involve regularisation techniques in their body to harness their benefits. It's essential to note that while regularisation methods are valuable tools, they are not the sole solution. The architecture of deep learning models also holds immense significance, and it is often the synergy between regularisation techniques and architectural choices that yields optimal results. In the next section, various discriminative architectures built for classification tasks and ResNet [22], a seminal state-of-the-art architectural design, which is utilised as a backbone network for unsupervised image classification models are mentioned.

2.3.1 Discrimitive Architectures

The architectures belonging to this category consist of Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and their variants. MLP is the groundwork of any DNN architecture, the fully connected network has the elements of input, output, and at least one hidden layer. RNN is an artificial neural network which is substantially different in design from the other categories. RNNs process sequential data by utilizing feedback connections by maintaining internal states that allow them to capture temporal dependencies and patterns in sequential data. The architecture which is focused on in this study is CNNs due to its nature in alignment with the business case and the problem targetted. Other discriminative architectures or categories of generative and hybrid architectures are not included.

CNN architectures are composed of convolution and pooling layers. CNNs possess essential attributes that make them conducive to effective training and high generalisability, distinguishing them from fully connected network architectures. These attributes include local connections, weight sharing, pooling, and the incorporation of multiple layers [15]. By leveraging these elements, CNNs iteratively adjust the weights of each layer to minimize loss while concurrently reducing dimensionality.

The architectural design of CNNs renders them well-suited for applications deal-

ing with 2D data, such as visual recognition, medical image analysis, image segmentation, and natural language processing [14]. Notably, various CNN variants, including VGG, AlexNet, Xception, Inception, ResNet, DenseNet, and MobileNets, serve as network backbones in diverse applications, chosen based on the alignment between structural characteristics and specific requirements. For example, the ResNet and DenseNet with more than 100 layers compared to AlexNet and VGG with 8 and 16 layers respectively make them suitable for different fields of applications. The design of DenseNets prioritizes efficiency in parameter usage and the reusability of features, while MobileNets are purpose-built for mobile applications, emphasizing computational efficiency and real-time performance [1].

ResNet, short for Residual Network, is a deep learning architecture designed to tackle challenges that arise when networks become increasingly deep. While deep neural networks excel at learning high-level features as their layer depth increases, this advantage comes with certain challenges. The relation between performance and depth is eminent for the applications of deep networks especially in the field of visual recognition tasks.

As deeper structures are proposed in the community, several obstacles based on the increasing dept have emerged. One such challenge is the vanishing gradient problem, where the gradient of the loss function which affects the parameters in the beginning layers approaches zero. This condition challenges increasing the deepness of the model. To address this, techniques like normalized initialization and intermediate normalization layers were introduced, but as the convergence problem was eliminated, the accuracy of DNNs started to saturate [22].

The issue which is referred to as the degradation problem, arises from the varying difficulty of learning different functions within a deep network. This problem decreases the performance of the DNNs even after the vanishing gradient problem is eliminated. It's crucial to note that the degradation problem is distinct from overfitting. A significant experiment detailed in the paper by He et al. [22] highlights this difference. The paper [22] discloses the difference by setting up an experiment including two NNs, both theoretically representing the same function. If we refer to a DNN as a black box with an input and output, the black box computes a specific function on the input which results in the output. The independent variable of the experiment is the number of layers of the network. n constructing the deeper network, the parameters of the original network are reused while adding identity mappings to deepen it. Consequently, two models with distinct network architectures are obtained: a shallow one and a deep one, while keeping the network's

complexity constant. During training, it is observed that the deeper counterpart has higher training error. The counter-intuitive results validated the fact that the ease of learning differs from function to function and identity mapping can impede the machine's learning process.

The authors [22] address the problem by introducing residual learning, based on the hypothesis that optimizing a residual mapping is easier than optimizing the original function. The key functionality of the residual network is the inclusion of direct connections between the current and previous layers. The information flow is provided by shortcut connections which function as identity mapping by bypassing layers and forwarding the learning function of the previous layers' inputs to the subsequent layers.

The performance analysis of the ResNet is conducted on several datasets such as ImageNet, COCO and CIFAR (for further detail on the datasets, please refer to Section 2.3.5). It has been shown in the paper [22] that the residual networks are easier to optimise and due to their reformulated structure, it is possible to add more layers which results in accuracy gain compared to the plain networks with the same depth. The fundamental learning block enabling deep residual learning is illustrated in Figure 2.1.

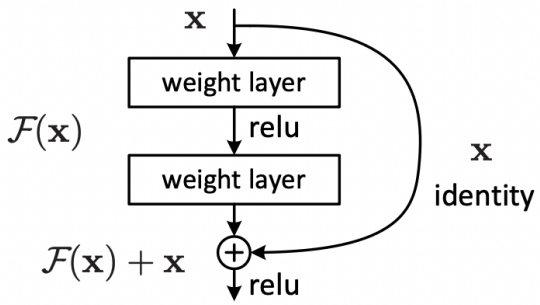

Figure 2.1: Residual learning block: Shortcut connections are formulated as $\mathcal{F}(\mathbf{x})+\mathbf{x}$ where $\mathbf{x}$ and $\mathcal{F}(\mathbf{x})$ are the input and output of current layer, respectively. $\mathcal{F}(.)$ represents the function which corresponds to the combination of weighted layers and ReLU activation. The figure is reproduced based on the paper [22].

With the specified architectural modifications, ResNets address both the vanishing gradient problem and the degradation problem. These issues typically hinder neural networks from achieving greater depth without overfitting the input data. This becomes especially critical in image classification tasks, where the complexity of the input data necessitates deep neural networks. Furthermore, ResNet enhances

training efficiency and facilitates faster convergence, which proves to be highly advantageous for computer vision tasks that demand deep networks and a substantial number of training epochs. The aforementioned reasons make ResNets a compelling candidate for the backbone network architecture in unsupervised image classification models including the following business case.

2.3.2 Application Areas: An Example Business Case

Different tasks require specific architectural designs with some attributes to be compatible with the task they are used for. For example, in the task of pedestrian and vehicle detection in the context of self-driving cars, high accuracy and short running time are both indispensable features whereas in the context of security systems, accuracy is the most critical feature of the model to detect criminals [1].

Advantages of Self-driven Cars

Some of the advantages according to Gupta *et. al.* [5] are as follows:

- Enhanced efficiency in fuel consumption, scheduling and routing, resulting in lower travel costs
- Improved safety for the users by strict considerations of speed limits, blind spots, issues related to distracted driving and rule violations
- Increased mobility for the elderly and disabled
- Reduction of car ownership, increase in shared access while continuing to provide a personalised experience for the user

Visual Recognition Systems in Self-driven Cars

Detection of pedestrians and vehicles is a crucial task in the realisation process of self-driving cars. Without achieving high accuracy performance in visual recognition systems and obtaining reliability to the abilities of a self-driving car from several perspectives, becoming competent in both perception and cognition, it is not possible for them to be trusted to come to action in daily commutes. The objectives from different perspectives according to Chen *et. al.* [1] are as follows:

Objectives:

1. Very high accuracy

2. High processing time

3. Good memory consumption

Market for Self-driven Cars and Role of Computer Vision

According to Statistica, the forecasted Size of the global autonomous vehicle market based on the values of 2021 and 2022 are as follows: An article by Statistica indicates

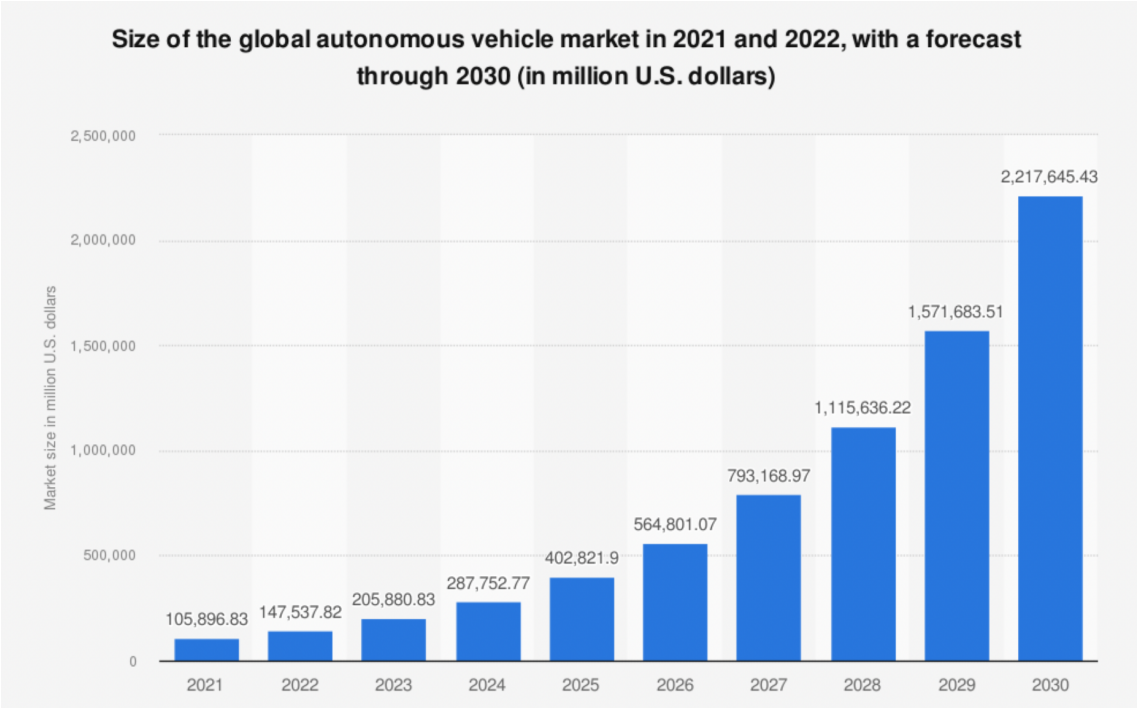

Figure 2.2: Size of the global autonomous vehicle market in 2021 and 2022, with a forecast through 2030 [23]

that by the year 2030, the market for fully automated cars will achieve 13.7 billion. It is also mentioned that the growth is positively correlated with the integration speed of the customers to the product and the production line efficiency.

Technology Overview

In the field of self-driving cars, there are two fundamental systems which play a central role: perception and decision-making. The core objective of an autonomous vehicle is to navigate optimally to its destination, considering factors such as minimizing travel time, energy efficiency, user safety, comfort, and adherence to traffic regulations. Perception is of paramount importance, as it empowers a self-driving

car to autonomously assess its state relative to its surroundings. Tasks encompassed within the perception objective of self-driving cars, as outlined by Badue et al. [24], include localization, offline obstacle mapping, road mapping, moving obstacle tracking (MOT), and traffic signalization detection and recognition. To enable perception and decision-making in a machine, a substantial volume of data is essential, especially in the case of self-driving cars, which struggle with the complexity of an immense number of parameters. Given that perception in self-driving cars is predominantly visual-based [25], effective data utilization is imperative.

Proposed solutions revolve around subsystems that constitute autonomous car perception, with a focus on sensor fusion. Advanced camera technologies provide valuable data by offering 360-degree high-resolution visual information about surrounding objects and depth. However, these technologies exhibit a significant reliance on expensive hardware, impacting overall costs. Research suggests combining cameras, RADAR, and LiDAR with deep learning to enhance machine interpretation. While LiDAR technology offers high precision, financial costs give rise to research on alternative cost-effective solutions from the computer vision community [25, 26]. Although camera technology demonstrates significant technological excellence and the importance of LiDAR technology cannot be denied in data collection, the role of deep learning in data analyses becomes evident. Deep learning solutions are commonplace for various tasks, with their performance often surpassing that of model-based counterparts, particularly in tasks such as MOT, traffic sign and signal detection, and pavement recognition. However, the scarcity of labelled data remains a significant challenge. Addressing this data availability issue could be alleviated by shifting focus toward unsupervised deep learning methods and expanding research in this domain. This emphasises the significance of unsupervised deep learning for image processing and its potential for business opportunities in the autonomous vehicle sector.

Real World Examples

Research on self-driving cars and the essential technological developments required for their realization has received substantial attention from both the academic and commercial sectors. According to Badue *et. al.* [24], some of the important research platforms are Navlab's mobile platform, the University of Pavia's and the University of Parma's car, ARGO. An important milestone in the history of self-driving cars was marked by the Defense Advanced Research Projects Agency (DARPA), which organized several competitions. These competitions attracted the participation of

prestigious universities such as Carnegie Mellon, Stanford, and Virginia Tech. Subsequently, numerous other competitions emerged, further fueling advancements in the field. The industry has also displayed keen interest through substantial investments.

Torc, Google, Tesla, and Didi are among the many players striving to establish a presence in this industry which is currently in its developing stages by producing one of the most advanced intelligent technologies that hold the promise of shaping the future [24].

When the trends in the market and the growing interest from both educational institutions and industrial companies are observed, it is evident that the research to design and implement technologies that can contribute to autonomous vehicles will increase significantly. Within this realm of research, unsupervised image classification for object detection tasks stands out as a cost-efficient alternative compared to advanced cameras, and it proves to be robust, particularly in the case of labelled data scarcity. Due to its natural advantage on image data, CNN architectures are the common architectures used for object detection tasks which can be utilised for both supervised and unsupervised models. Some of the proposed designed networks among this category are R-CNN, R-FCN, SSD, and YOLO, each offers advantages and disadvantages suited to different tasks with varying requirements that are specified in more detail in [1]. In the next section, the essential unsupervised clustering components employed within CNN architectures are introduced.

2.3.3 Clustering Components

Data clustering is the process of arrangement of the data samples into groups (clusters) based on a similarity metric such that the similar samples share membership within the same cluster [27]. Clustering is the same task as unsupervised classification, where, in the absence of labels, classification is accomplished by forming clusters, each corresponding to a class. This sets it apart as a distinct task from supervised classification, where the model is trained using both data samples and their associated labels and performs labelling for the unseen data samples after training is completed.

The clustering task garners attention from various fields due to its relevance in numerous industries. Its vital potential to be used in data analysis is widely acknowledged [27]. Consequently, extensive research efforts have been devoted to this complex problem, the research from different disciplines involves discrete assumptions and applications.

Clustering algorithms find applications in a variety of domains such as image segmentation, information retrieval, data mining, etc. Further knowledge on the application of clustering is available in literature [27], while this study is only focused on clustering from the perspective of an unsupervised image classification task. In the following section, we focus on the main components of clustering to be able to inspect and analyse the unsupervised image classification methods that will be focused on in Chapter 3. According to Jain *et. al.* [27] clustering task incorporates the steps of representation learning on the feature vector(sample), application of an appropriate similarity measure between the data samples, grouping the data samples accordingly and evaluation of the output.

Pattern Representation

The representation learning step can encompass feature selection or feature extraction. The former process limits the number of features in the feature set while the latter performs transformations to the prior feature set and obtains a new feature set. Feature selection is a highly used pre-processing method and the goal is to filter out the reluctant or irrelevant features and left with comparatively informative features [28]. With Big Data, and digital cameras, Internet data is growing faster. In the field of image analysis, large-scale high-dimensional image datasets are present in various real-life contexts. Each of them suffers from the profound challenge of requiring excessive computational power to be analyzed. The high dimensionality of images despite the rich content exacerbates processing and analysing the data which uncovers the need for feature selection and/or feature extraction to reduce the dimensionality of data while representing it in a more suitable way for the learner that is employed for the specific problem. There are certain positive and negative attributes associated with each of the techniques; the merit of feature extraction is since each feature generated corresponds to a condensed version of multiple features, they are more competent in discrimination compared to the original features. A disadvantageous impact of feature transformation is the decline in the interpretability of the model. Given the paramount importance of model accuracy, feature extraction typically takes precedence in image analysis problems, and will be focused on in this study while applications of feature selection methods used in image analysis could be found in [28].

The field of low-dimensionality representation and associated methods has witnessed extensive research. Principal Component Analysis (PCA) stands out as one of the popular approaches, which reduces dimensions through linear transformations

applied to the features. This transformation yields new features called principal components, with the first component capturing the highest variance in the data. Machine learning problems frequently employ feature extraction methods like PCA. The choice of transformations applied to the data significantly influences the representation and, consequently, the model's performance. Hence, careful engineering and expertise in the field are essential when constructing a mapping from the initial input space to a low-dimensional embedding space [15].

Particularly in scenarios involving high-dimensional data such as images, traditional machine learning classifiers without sophisticated features struggle to detect invariant objects in augmented images. To address this challenge, handcrafted (or hand-engineered) features like Scale Invariant Feature Transform (SIFT), Shape Context, Haar-like features, Histogram of Gradients (HOG), Local Binary Patterns (LBG), and others were utilized as a solution to the selectivity-invariance dilemma [15] before the success of CNNs in learning representations. The selectivity-invariance dilemma is related to the balance that feature extractors must hold between retaining relevant features for later use and discarding irrelevant features during the discrimination process. In terms of image representation learning, the capability to classify distinct objects, such as birds and airplanes, hinges on feature learning being sensitive to even minute distinguishing patterns critical for object detection. Simultaneously, the influence of features like background, object position, or lighting must be minimized. Deep learning simplifies this intricate and impractical feature extraction process. Unlike traditional machine learning, feature extraction in deep learning models is not a separate process. As mentioned earlier, deep learning's layered structure progressively enhances representation by increasing complexity in each layer. To illustrate the learning process, a deep learning model, when inputted with an image, begins by learning edges in the initial layer. In the second layer, it identifies motifs as combinations of the detected edges, and in the third layer, it recognizes objects formed by these motifs from the previous layer [15]. This learning scheme makes it possible for DL models to eliminate the hand-engineered feature extraction. In the following section, we delve into self-supervised representation learning, a prevalent technique in pattern representation for unsupervised deep learning models.

To learn the visual representation of the images, Self-Supervised Representation Learning (SSL)-based techniques are alternatives to supervised representation learning. Self-supervised Representation Learning (SSRL) learns relevant features without the limitation of labels. This characteristic of SSRL establishes occasional

superiority over conventional representation learning approaches that heavily rely on label supervision. As it is mentioned before, SSL considered to fall under the category of unsupervised learning type since it does not require human-annotated data. Instead, SSL relies on the machine's ability to assign labels to data samples based on their intrinsic attributes [19]. These labels, often referred to as pseudo-labels, are generated through the process of predicting them from observable aspects of the input, encapsulating the goal of the self-supervised notion [29].

In certain domains of deep learning, such as medicine, data annotation is often scarce due to the high resource costs involved, including both time and money. As a result, supervised learning is constrained by the limited labelled data, if any exists. This scarcity of labelled data can lead to overfitting issues, as the model may struggle to generalize when trained on a small dataset. Furthermore, supervised feature representation is susceptible to spurious correlations and adversarial attacks, which can compromise the model's overall robustness and generalizability [29]. In situations where labelled data is inadequate for effective supervised representation learning, Self-Supervised Representation Learning (SSRL) exhibits a significant advantage. SSRL has the capacity to leverage unlabeled data effectively and establish its own form of supervision. This characteristic makes it a valuable approach for scenarios where labelled data is limited or lacking over the traditional supervised approach.

SSRL approach relies on the idea of a suitable pretext task to learn the representations from the unlabelled data, the pretext tasks correspond to the preliminary learning of the machine where the model updates its parameters accordingly to the objective function of the pretext task. The knowledge that is gained by the pretext task could be transferred to the downstream task which is the actual goal of the application. In computer vision, examples of downstream tasks include classification, segmentation, and detection. Typically, downstream tasks are supervised. However, when labelled data is limited for the target task, SSRL can utilize unlabelled data for self-supervision, uncovering inherent relationships between features and occurrences while simultaneously recovering missing information [29]. Even though the most common scenario for downstream tasks is supervised learning, unsupervised downstream tasks such as clustering (also referred to as unsupervised classification) [3] make use of SSRL. This paper does not delve into the examples and benefits of SSRL in supervised learning, as its focus is primarily on cases where the downstream task is also unsupervised, serving as the pretext task and resulting in a complete unsupervised process in representation learning and classification.

In the literature, a wide variety of pretext tasks exist for SSRL. Examples include image inpainting [30], image colourization [31], image jigsaw puzzles [32], and clustering [33]. Considering the abundant types of pretext tasks, one may question which one to choose. The choice of pretext task has a significant effect on the learned features which means that it should be selected carefully by contemplating both the nature of the data and the downstream task hence the topic is an active track of research [34]. By choosing a certain type of pretext, the objective of the pretext influences the parameters therefore the initialization for the downstream task. For example, if we apply data augmentation of scaling to the input image while solving the pretext task of discriminating the same image with different scaling values, the machine learns to stay invariant to scaling transformations while discriminative to others on the data such as colour transformations. Selecting the right pretext task allows SSRL to serve as an effective pre-trained feature extractor, while in other cases, the learned features may be ineffective or detrimental.

Categorizing SSRL approaches varies in the literature. Ericsson et al. [18] focused on discriminative models while excluding generative models from classification. They categorize pretext tasks into transformation prediction, masked prediction, instance discrimination, clustering, and contrastive instance clustering. Jing and Ting [19] propose a categorization that includes generation-based, context-based, and free semantic label-based categories for pretext tasks, specifically for image data representation learning. However, this taxonomy does not include a category for contrastive learning-based models (which both IIC and RUC belong to) and is not comprehensive enough for this study's focus. Instead, we adopt the categorization by Liu *et. al.* [29] to classify the pretext tasks that are employed in representation learning. The taxonomy is constructed based on the district objectives of the SSRL and comprises generative, contrastive, and generative-contrastive (adversarial) categories. The taxonomy by Liu *et. al.* is accepted to be used to classify the SSL-based unsupervised image classification. We choose this taxonomy due to its comprehensiveness, clear hierarchy, and adherence to the principles of mutual exclusivity. Additionally, it includes contrastive-based representation learning, which is the pretext task followed by both methods emphasized in this study. Furthermore, a sub-category of constructive learning, namely context-instance discrimination, involves the class of pretext examples with mutual information (MI) objectives. MI-based pretext tasks hold significant importance, especially in the context of the joint approach clustering chosen for this study since the chosen joint approach clustering method, Invariant Information Clustering (IIC) [4], adopts this

type of SSRL approach.

The generative category consists of auto-regressive (AR) models, flow-based models, auto-encoding (AE) models, and hybrid generative models. As the name of the category suggests, this type of SSRL model is able to predict the distribution of the data which enables the use of them in the downstream tasks of classification and generation. However, they have limitations, such as being point-wise and potentially lacking robustness and high-level abstraction [29].

Contrastive-based SSRL can be categorized into two sub-categories: context-instance and instance-instance. Initially, representation learning was predominantly associated with generative architectures, and there was little consideration for discriminative ones. In Section 2.3, we discussed Sarker's taxonomy of architecture types, which didn't connect discriminative architectures with feature learning or data representation but rather with classification. However, with the advent of state-of-the-art contrastive-based methods like BYOL [35], SwAV [36], and Deep InfoMax [37], it has become clear that contrastive-based representation learning aligns closely with the principles of classification. These methods have demonstrated the effectiveness of such an alignment in achieving effective results.

The context-instance contrastive SSRL learns the representations by discovering the relationship between the local features of the instance and the global context that the instance belongs to. Both Predict Relative Position (PRP) and Maximize Mutual Information (MI) belong to this category, each with distinct objectives. PRP aims to understand the relationship between local features with global context as guidance, while MI (further details about MI-based pretext task and its usage in method IIC [4] is in Chapter 3) focuses on learning the direct relationship between local features and global context, excluding relationships between local parts. In IIC, context-instance contrast is applied, where instances are augmented versions of images, and context refers to common, distinct features shared by objects within the images, unaffected by the applied image augmentations.

Instance-instance contrast includes cluster discrimination and instance discrimination categories. In cluster discrimination, the representations of the images are grouped into clusters to produce the pseudo-labels and the pseudo-labels are used to discriminate the input images. Instance discrimination, on the other hand, aims to assign the same class to positive samples while differentiating negative samples. An example of instance discrimination is SimCLR [38], which is employed in the sample selection process of RUC. IIC does not fit the cluster discrimination category because it does not employ pseudo-labeling, nor does it fit the instance discrimina-

tion category because it doesn't involve positive and negative samples for learning differences between them.

Definition of Pattern Proximity Measure

In the clustering step of unsupervised classification, it's essential to define a distance function that quantifies the dissimilarity between patterns represented through a self-supervised pretext task. The definition of proximity measure step has a substantial impact on the performance of the model and the measure is dependent on the representation of the data [39]. In general proximity between two data samples could be interpreted as a distance if the data samples are assumed to be represented by a point in a multi-dimensional space [27]. Various distance measures are available in use while a popular metric is the Euclidian distance which is utilized in k-means after the data is represented by using the instance discrimination representation learning method SimCLR [38]. For IIC, the processes of pattern representation and clustering are implicitly coupled into each so there's no explicit need for a separate distance measure. A single objective function is responsible for both processes.

Clustering

Grouping criteria is another element which affects the output of the model. The taxonomy of clustering techniques in the literature varies, and one prominent viewpoint is that presented by Jain *et al.* [27] which will be used as a benchmark on clustering techniques for the rest of the study. The aforementioned taxonomy which is followed by subsequent overviews [40, 41] divides clustering into two main descendants categories: hierarchical and partitional. The primary differentiating factor between these two techniques is the number of memberships allowed for each sample. In hierarchical clustering, sub-clusters can exist within clusters, forming a nested structure that allows data points to have multiple memberships. In contrast, partitional clustering strictly assigns each data sample to a single cluster, with no overlap between clusters. The hierarchical algorithms generate dendrograms, tree-shaped clusters, which are produced by 2 opposite techniques; merging or splitting [41]. The scope of this study is limited to the algorithms based on the partitional technique to be aligned with the unsupervised image classification task. K-means clustering is one of the popular approaches which assigns the samples to the cluster with the nearest mean and is employed in RUC [9].

Assessment of Clusters

Outputted clusters are assessed by accuracy metric as a standard procedure while for unsupervised image classification during the procedure, the labels of the images are not shown to the machine. In order to measure the accuracy, the output clusters and the ground-truth labels should be compared. The Hungarian algorithm can be employed to solve the linear sum assignment problem, facilitating the matching of clusters produced by algorithms to the ground truth clusters. Both RUC [9] and IIC [4] utilize this algorithm.

Hungarian algorithm is an assignment which can be employed when the number of jobs and workers are the same. Each worker and job matching has a different cost associated. The algorithm finds the optimal matching between these values as such that the total cost is lowest or the total benefit is highest if the algorithm is structured with a benefit perspective. When the algorithm is used for an unsupervised classification task, a matching between the clusters formed by the unsupervised classification method and the true classes is found as such that the matching yields the highest accuracy for the method.

2.3.4 Approaches

The variety of clustering techniques arises from the diverse range of problems they aim to address. The specific goal of the problem, the data modality, its representation, distance metric, and grouping criteria all contribute to shaping the architecture of the clustering model. For the certain case of image data as the input modality and the task of unsupervised classification, samples require two main processes, representation learning and clustering.

That being said, another important factor that influences the structure of the model is the approach taken for the two main processes. The literature encompasses different categorizations of image clustering while in this study the categorization of the unsupervised image classification proposed by Park *et al.* [9] will be followed which is based on the approaches followed during training of these methods. The authors initially grouped the existing research into three categories: sequential, joint, and multi-step refinement approaches. This categorization has been further enriched by the inclusion of their recently proposed method, which falls into the category of add-on modules to improve unsupervised clustering.

Sequential Approach

The traditional way of clustering conforms to an ordered procedure. The procedure includes the execution of feature learning and clustering processes in a sequential manner. The feature learning phase corresponds to the mapping from the raw input images to the newly constructed vector space. During the clustering phase, the distance metric or grouping criteria are put into action on the recently learned features. The feature or representations could be learned by applying self-supervised pretext tasks or using embeddings to lower-dimensional subspaces. One of the applications in the field is to use SimCLR [38]to represent the input data. Projecting the raw input data into the reduced dimensional vector space formed by principle components of PCA could be also employed. The ease of implementation and being computationally advantageous with the complexity of $O(n)$ makes the k-means algorithm a popular choice among the clustering algorithms [27].

One of the applications of the sequential approach in the field of image processing is by Ranjan *et al.* [42], hyperspectral images are represented on a lower dimensional space by applying PCA and the feature learning step is subsequently followed by k-means clustering to realize the classification task. The research by Ding and He [43] uncover the relationship between these two widely used algorithms. principal components generated by PCA are the continuous version of the cluster assignment solution formed by k-means. The paper advocates that the dimensionality reduction by PCA is aligned with the objective of k-means clustering.

Even though the serial actions of feature learning and classification based on the features learned are intuitive and straightforward, according to [9], the decoupling of the steps can lead to problems while identifying the boundaries between clusters. The representation learning stage performs mappings from the original to the new representation space without considering the subsequent clustering step. The disassociation between the sequential steps results in disassociation in the alignment of the objectives of each step. Consequently, only minimal differentiation occurs between the sample representations belonging to different clusters, leading to poor separation and distinction between the clusters which might create a contradictory situation to the low-density assumption [20] of clustering. The assumption implies that the decision boundaries of a classifier should avoid passing through the high-density regions within the representation space.

Joint Approach

To avoid the issue of the previous approach, alternatively, unsupervised image classification methods which perform representation learning and clustering steps concurrently in an end-to-end fashion are proposed in the community. The objective function of the joint approach clustering methods is responsible for both the feature extraction and the grouping of the extracted features. The deep neural networks are used for the execution of generating the cluster assignments and the layered structure gets updated in each iteration according to the loss function calculated on the cluster assignments. For joint approach clustering methods, the CNNs are solely accountable for the feature extraction, so the hand-engineered feature extraction methods are disregarded.

Deep Cluster by Caron *et. al.* [33], jointly learns the parameters of the network and cluster assignments of the images by using clustering as an auxiliary step to generate pseudo-labels. The input data represented by the CNN is grouped into k distinct clusters. Each cluster represents the class assignment membership to one of the predefined classes. After each image is associated with a pseudo-label, the pseudo-labels are used to optimise the network parameters. Another method with the same approach is DEC [39], a joint approach clustering method based on k-means where an auxiliary target distribution is utilized to learn the feature representation and cluster assignments simultaneously. The raw data is projected into a lower dimensional feature space where each data point represented in the new vector space is assigned to a cluster of centroids by soft assignment. The network updating process is realized by minimizing the KL divergence between the soft assignment predictions and the target function. Both functions are examples of k-means clustering algorithms, a popular technique used in cooperation with DNNs. An issue of k-means-based clustering is the degenerate results (solutions).

Degenarate solutions occur when a classifier assigns all the input samples into a single cluster. During training, an unsupervised classifier tries to learn the best-fitted boundaries between the classes which could minimize the loss function. If the loss function doesn't imply any constraint to prevent empty clusters, the machine could stuck in a degenerate solution of having only one decision boundary [33]. The class assignment process could be dominated by a small number of clusters, this trivial solution could be observed commonly in joint approach clustering methods due to unbalanced datasets. Random initialisation is another key cause of the problem since an NN starts off by learning the low-level features and in the early stages of training the clustering can discriminate based on features such as colours, edges or

gradient orientations which creates a mismatch between the similar embeddings of the images and their belonging semantic classes.

As it is mentioned, to fix the issue a mechanism that will prevent the sub-optimal solution should be adopted. One of the ways to a mechanism is to regularise the model. The loss function impels the model to distribute the samples in a balanced way and IIC avoids the uninformative state of single-cluster dominance. The loss function includes mutual information terms for the class assignments of data augmentation to regulate and Sobel filters to prevent the network initialization limited on low-level features. The method embraces a different objective compared to the k-means-based algorithms which depend on CNNs to explore the general-purpose features and apply auxiliary objectives divorced from the semantic clustering objective and achieves better accuracy rates compared to the k-means-based algorithms (please refer to Section 3.1 for IIC's accuracy rate). Another method which employs information theory in its objective is DeepInfoMax [37] which aims to maximise the mutual information between the input and output of the deep encoder. A common problem that could be encountered with clustering models that have a mutual information-based objective is the overconfidence issue.

An overconfidence issue appears when the machine learning model exhibits predictions with high confidence even when the label assignment is false. The overconfidence of the model indicates that the model has low calibration skills which means that the confidence of the predictions of a class assignment does not align with the actual accuracy of the prediction. The issue could lead to follow-up problems such as the build-up of misclassification errors. If an incorrect prediction is made by the model in the beginning stages of the training, the misknowledge strengthens and the false predictions will be made with higher confidence rates in the following epochs [9]. The issue of overconfidence can be observed in any of the three conventional approaches(sequential, joint, multi-step refinement) and the multi-step refinement category of image clustering is rooted to address the issue of overconfidence.

Multi-step refinement Approach

Recent research on unsupervised clustering has shifted its focus towards addressing unintended solutions that the joint approach may encounter. One recent solution is the two-stage approach. In this approach, an initialization phase is employed, during which embedding learning is performed to reorganize the feature space in such a way that similar data points are positioned closely while dissimilar ones are separated. This new feature space becomes more suitable for the subsequent

clustering step. Unlike joint approach methods, which lack this initialization, the two-stage approach includes a second stage that iteratively refines both the class assignment process and the network's representation learning. Two contemporary state-of-the-art works falling under this category are TSUC [44] and SCAN [17].

The two-staged approach consists of two decoupled steps, which bear some similarity to the sequential approach but significantly deviate from it. In the two-staged approach, feature representation is constructed to serve as a prior for the second step, generating features that are semantically meaningful and valuable for the subsequent clustering step. In contrast, sequential approaches use a pretext task to construct suitable feature representations based on dataset characteristics, often overlooking the following clustering step. This oversight leads to inferior performance compared to multi-step refinement approaches [9]. According to Van Gansbeke *et. al.* [17], for the methods that involve iterative refinement during training, are sensitive to the initial embedding of the data. As mentioned earlier, the randomness of network initialization is a primary cause of the degeneracy issue observed in joint approaches. Moreover, attempting to unify the inherently different objectives of representation learning and class assignments makes the joint approach susceptible to degeneracy [44]. The two-stage approach avoids the sub-optimal results by disassociating the objectives and building the model with an awareness of the subsequent steps of representation learning and clustering.

SCAN [17] initially represents data samples in an embedding space through an instance discrimination task. It then identifies the nearest neighbours of each data sample in the embedding space based on similarity. Once this feature space initialization is established, which provides a suitable environment for clustering the represented data, a loss function is employed. This loss function compels the network to generate consistent and discriminative class predictions for both the images and their neighbours, thereby optimizing the network. While SCAN's loss function relies on the label invariance between nearest neighbours, TSUC [44] incorporates a loss term based on the label invariance between the original and augmented images.

Add-on module to improve unsupervised clustering

The last category, in which this study is rooted is the add-on modules which are designed as flexible structures that can be applied on top of any of the approaches (sequential, joint, multi-step refinement) mentioned above. They work by taking the predicted labels outputted by a previous clustering model, also known as the baseline model. This class of unsupervised clustering targets the problems of the

baseline clustering model it is applied to which might be preventing the model from achieving better unsupervised classification performance. More specifically, the RUC [9] model was proposed to fix the calibration of the original model resulting in higher accuracy rates for SCAN and TSUC on benchmark datasets (please refer to Section 3.2 for further details). The working principle of the add-on module is to detect and correct misaligned knowledge by applying robust training techniques.

The category is relatively new compared to others, leaving room for the development of new algorithms. A related work, the pseudo-semi-supervised model by Gupta et al. [45], is based on leveraging the semi-supervised learning paradigm to improve unsupervised classifiers, presenting a similar approach to RUC [9]. Both models begin with a sampling process that selects credible samples and aims to form a clean set with a high precision ratio, the ratio of truly predicted labels to falsely predicted labels in the clean set is high. The remaining samples, which are not deemed trustworthy by the selection method, are grouped separately to form the unclean set. The main difference between the pseudo-semi-supervised and RUC lies in how each model treats the predicted labels of the baseline model, often referred to as pseudo-labels. In the pseudo-semi-supervised model, the highly credible labels are treated as ground-truth labels, and the model is trained accordingly to them in the subsequent stages. In contrast, in RUC, even the pseudo-labels classified as clean are considered potentially noisy, leading to the development of a cautious method to handle misinformation at every stage.

2.3.5 Image Datasets

There is a considerable amount of benchmark datasets that have been widely employed by the image classification community to evaluate the performance of unsupervised image classification algorithms. Some of these datasets include MNIST [46], STL-10 [47], ImageNet [48], and CIFAR [49]. The MNIST [46] and Fashion-MNIST [50] datasets primarily serve as benchmarks for unsupervised image classification tasks related to handwritten digit and fashion item recognition respectively. Both datasets comprise 70,000 grey-scale images with a size of 28×28. Another notable dataset is the ImageNet [48] dataset, which is significantly larger, containing more than 14 million images in total which correspond to a thousand different categories. The STL-10 [47] dataset, derived from the ImageNet [48] dataset, is notably different as it contains both labelled and unlabelled data. The quantity of unlabelled data greatly surpasses the labelled portion. Some content in the unlabelled portion of the dataset does not correspond to any of the semantic classes represented in the

labelled part. The characteristics of the dataset make it suitable for testing classifiers in unsupervised and semi-supervised settings. Additionally, for both settings, the presence of unlabeled data functions as adversarial noise for the classifier which forces the classifier to be robust to achieve high accuracy rates. These datasets vary in terms of sample quantity, image size, number of colour channels, and content. This diversity provides different complexity levels, enabling evaluation of the image classification methods on a large scale from the perspectives of effectiveness and scalability.

The one that is chosen for the present study is the CIFAR-10 [49] dataset. The dataset consists of 60,000 colour images which include objects from 10 different semantic classes. These classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck where there are 6,000 samples per class. Although each sample in the dataset dataset has an associated label, the dataset is used for unsupervised classification by disregarding the corresponding labels during the network training process. Each of the aforementioned datasets besides CIFAR-10 [49] can be used for unsupervised image classification by applying the same procedure. The ground truth labels are only involved in the evaluation phase to be compared against the model's predictions and accuracy is calculated accordingly. There is an alternative version of the CIFAR-10 dataset, named CIFAR-100, with 100 classes which is grouped into 20 superclasses.

Considering the purpose of the present study and the available time and computation resources, we decided that the CIFAR-10 [49] dataset is the most suitable one due to its specific characteristics among the datasets. The MNIST [46] datasets, with their single-channel data and simplified illustrations of fashion items or digits, do not align well with the business case mentioned. On the other hand, although ImageNet [48] is able to represent a wide range of object and animal categories, the extensive size of the dataset poses a noteworthy computational obstacle. Similarly, the high-resolution STL-10 [47] dataset with 100,000 samples with a size of 96 is also disregarded due to the computational limitations (more could be referred to in Section 4) leaving CIFAR-10 [49] showing the closest alignment with the requirements and limitations of the study. The following Table 2.1, summarises the features of each dataset mentioned.

Dataset	Trainset size	Testset size	Image size	Number of channels	Unlabelled data
MNIST [46]	60,000	10,000	28×28	1	No
Fashion-MNIST[50]	60,000	10,000	28×28	1	No
CIFAR-10 [49]	50,000	10,000	32×32	3	No
ImageNet [48]	1.2M	50,000	Various	3	No
STL-10 [47]	500	800	96×96	3	Yes

Table 2.1: Benchmark Datasets and Characteristics, the highlighted row corresponds to the dataset of the study

2.4 Learning with Noisy Labels

The availability of good quality annotations for image data is low, as mentioned previously. The reasons behind this situation have been discussed before. To address the cost factor, non-expert sources such as Amazon's Mechanical Turk are often utilized. However, the reliability of labels provided by these platforms raises concerns, as noisy labels can be a substantial problem, especially considering that deep neural networks are capable of fitting even random data.

The experiments conducted by Zhang *et. al.* [10] revealed unexpected behaviour in DNNs and influenced how we should perceive the concept of generalizability. Using the same model with all hyperparameters unchanged, the DNN was first trained with true labels and then with random labels. The learning curve during training followed an expected shape, with the training error decreasing and eventually reaching zero as the epochs increased. The surprising result was that the training error when labels were completely replaced with random ones was the same as in the case with true labels. This result was counterintuitive, as randomizing the labels of images was expected to remove the relationship between the two variables, hindering the machine's learning and preventing convergence. A similar experiment was conducted by adding noise to the images instead of labels, yielding the same result.

The contribution of this experiment to this study is significant. It demonstrates that DNNs can overfit to data even when there is no relationship between the images and labels other than randomness. Therefore, when there is a possibility of adversarial noise in the labels, precautionary measures should be taken to prevent DNNs from overfitting to the noise. Even though this study primarily focuses on unsupervised image classification, it incorporates semi-supervised learning principles during the training of RUC [9], which involves pseudo-labels, including potentially

incorrect predictions outputted by a baseline model. This makes it crucial for RUC to incorporate mechanisms that enhance the algorithm's robustness against noise present in the labels.

DNNs' ability to overfit to corrupted labels during the training phase can lead to misinformed learning affecting the machine's later predictions in the test phase, resulting in poor generalization. This issue can be alleviated by robust training techniques. Explicit regularization techniques such as weight decay and dropout alone are insufficient to rectify the generalization error [10] when applied on DNNs. In literature [11], the robust training techniques are grouped into 4 main categories including robust architecture, robust regularization, robust loss design and sample selection. three of them except robust architectures are utilized in RUC [9]. Dropout and weight decay which are examples of explicit and label smoothing, an example of implicit regularization are used to ensure the robustness of the method. Label refurbishment and loss correction, under the category of robust loss design, are also implemented in RUC [9]. Finally, sample selection and multi-network learning are also adopted within the method. Further details on each robust training technique are included in Chapter 3.

Chapter 3

Methods

In this chapter, the unsupervised image classification methods; IIC [4] and RUC [9] that Chapter 4 is built upon are explained in detail. In this chapter, the experimental details section provides insights into the experiments conducted by the authors of [4], whereas specific details about our experiments are presented in Chapter 4.

3.1 IIC

IIC is an unsupervised DNN-based clustering method that can be employed for image classification and segmentation. The method is an example of the joint approach clustering where the network is trained in an end-to-end manner. The representation learning and class assignment steps are wrapped together inside the network. The objective of the network is to form clusters which will be aligned with the semantic classes within the dataset. IIC offers a distinct advantage and novelty compared to several sequential approach-based methods, which require post-processing steps like k-means [51] after an intermediate representation learning step [33]. This advantage stems from IIC's capability to optimize the network with its clustering objective while simultaneously extracting relevant features during the training process.

	CIFAR10
IIC(lowest loss sub-head)	61.7
IIC(avg. sub-head $\pm$ STD)	57.6 ± 5.01

Table 3.1: Accuracy results of IIC. Table is based on the results achieved and published in the paper [4].

The method exhibits robustness to two significant challenges: clustering de-

generacy, a common issue in clustering tasks, and noisy data containing distractor classes. The degenerate solutions are mitigated through the entropy maximization component of the objective function, as shown in Eq. 3.1. Additionally, an auxiliary layer is introduced to over-cluster the dataset, enhancing the representation learning process. By incorporating this auxiliary overclustering head alongside the main head in the network, IIC effectively leverages all samples from noisy datasets without necessitating the removal of irrelevant (distractor) samples. The accuracy rates of the best and average head (refer to Section 3.1.2) of IIC when applied to the CIFAR10 dataset are as in Table 3.3.

3.1.1 Loss Function

The objective function of IIC relies on mutual information. While the utilization of criteria related to information and entropy has been observed in the deep learning community by the precedents [37, 52], IIC follows a distilled approach from other methods by focusing on only the mutual information between the class assignments of paired data without combining multiple criteria. The formulation of the loss function is as follows:

$$\max_{\Phi} I(\Phi(x), \Phi(x')) \tag{3.1}$$

where Φ equals the network to be optimised and (x, x') is the paired input data for any sample x and its random transformed pair x'. The function of network Φ could be defined as $\Phi : \mathcal{X} \rightarrow \mathcal{Y}$ with $\mathcal{X}$ which symbolises the input dataset and the output is symbolised by $\mathcal{Y}$ which takes a value within the range of $[1, C]$ for C classes. Mutual information in the objective function could be calculated exactly as a consequence of the discrete distribution of class assignments. Eq. 3.1 could be written as the formula below:

$$\max I(z, z') = I(\mathbf{P}) = \sum_{c=1}^{C} \sum_{c'=1}^{C} \mathbf{P}_{cc'} \cdot \ln \frac{\mathbf{P}_{cc'}}{\mathbf{P}_{c} \mathbf{P}_{c'}} \tag{3.2}$$

where z is the discrete random variable which equals to $\Phi(x)$, $\mathbf{P}_{cc'}$ equals to $\mathbf{P}(z = c, z' = c')$, $\mathbf{P}_{c}$ and $\mathbf{P}_{c'}$ refers to the marginal probability of random variables $z = c$ and $z' = c'$ respectively. For further details on the computation, refer to the IIC paper [4]. To maximise the mutual information in Eq. 3.1, the conditional entropy of class assignments should be minimised and the entropy of individual

class assignments should be maximised. This can be observed by expanding the mutual information as follows:

$$I(z, z') = H(z) - H(z|z') \tag{3.3}$$

The first component on the right-hand side of the equation, $H(z)$, is maximised when the probability distribution of clusters is equal while the equation is equivalent to 0 if all samples are classified into a single cluster. Consequently, maximising the mutual information requires the maximisation of individual entropy. Maximisation of individual entropy guarantees to prevent the degenerate solutions which occur when the samples of a dataset are assigned to the same cluster. The conditional entropy term, $H(z|z')$ is the predictability of z when z' is given. z' is a random transformation of z, in an ideal case the conditional entropy is minimum if the network is optimised. If the network is optimised, the predictability of the class assignment for any sample is at its maximum in the presence of its augmented pair's class assignment. In other words, the information necessary to predict the class assignment of z should be zero when the class assignment of z' is known. The conditional entropy component enforces the network to classify augmented pairs of an image as members of the same cluster.

3.1.2 Overall Pipeline

The pipeline of IIC has the following elements which contribute to the overall performance of the method:

1. Sample Repeats
2. Sobel Filter
3. Multiple Sub-heads
4. Auxiliary Overclustering Layer

Authors of IIC [4] have tested the method on several image datasets with different characteristics including STL, CIFAR, MNIST, COCO-Stuff and Postdam, for both classification and segmentation tasks. Consequently, IIC exhibits a variety of configuration details, such as the number of sample repeats within a batch and sub-heads, for each dataset and task. For simplicity and relevance to this study, only the configuration setting specified for the CIFAR-10 dataset for the classification task is mentioned below.

The overall pipeline with CIFAR10 dataset configurations is illustrated in Figure 3.1. Each element could be distinguished from its separate color-filled frame.

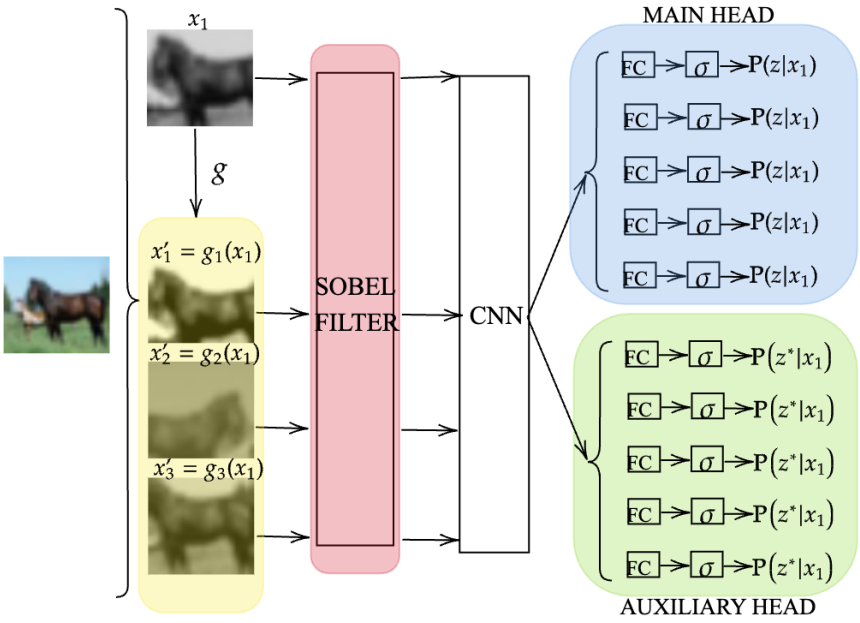

Figure 3.1: IIC model illustration. Each frame belongs to a different element: yellow, red, blue and green correspond to sample repeats, sobel filter, multiple sub-heads and auxiliary overclustering respectively. The figure is composed based on the paper [4] and code

Sample Repeats

As mentioned above, during the network optimisation, the network is fed by paired input that consists of a dataset sample and its augmented version. The Eq. 3.1 could be re-formulated as:

$$\max_{\Phi} I(\Phi(x), \Phi(g(x))) \tag{3.4}$$

where g is the combination of the transformations at the bottom left of Table 3.2 during training. Table 3.2 shows the spatial and colour distortions that are applied to both pairs in the training and testing phases. The primary objective of training the network using a paired data structure is to instruct the network to emphasize common features that remain consistent across pairs, even in the presence of spatial distortions, while disregarding instance-specific features. Data augmentation

is utilized to mitigate overfitting and enhance robustness during the representation learning phase.

The advantages of data augmentations are further amplified by the repetition of samples, with the aim of expanding the dataset and facilitating improved learning by exposing the machine to more data. In the case of IIC, each sample is paired with different transformed counterparts multiple times. For the CIFAR10 dataset, three sample repeats are included in each batch. This means that an image x is paired with three augmented versions, $x_1^{'}$, $x_2^{'}$ and $x_3^{'}$ which corresponds to $g_1(x)$, $g_2(x)$ and $g_3(x)$ respectively. The transformation g_i respectively. It's important to note that each transformation introduces randomness, resulting in a different output with each application. This approach provides the network with a multitude of examples to learn the invariant features of images.

Transformation List		
Pair	Train	Test
x	Random crop Resize Greyscale	Center crop Resize Greyscale
$g(x)$	Random crop Resize Random horizontal flip Color jitter Greyscale	-

Table 3.2: Transformations applied to paired input x and $g(x)$ during train and test time

Sobel Filter

These filters work by amplifying the amplitude of high spatial frequency regions within the image. High spatial frequency, in this context, relates to the rate of change in image intensity within the spatial domain of an image. In image data, edges are characterized by high-frequency spikes, as there is a significant difference in pixel values before and after an edge. The filter operates a convolution of 3×3 kernel on the image and computes the magnitude of the gradient of pixel intensities along the image. Sobel filter combines two kernels with perpendicular orientations: horizontal and vertical. Figure 3.2 illustrates both kernels.

Images are transformed into grayscale before filtering. This allows the representation learning process to concentrate on high-dimensional features, such as shape,

-1	0	+1
-2	0	+2
-1	0	+1

+1	+2	+1
0	0	0
-1	-2	-1

Figure 3.2: Sobel filter: Horizontal and vertical convolution kernels

instead of focusing on trivial features, such as colour, which might lead the model to make classification decisions based on less relevant information.

The phenomenon of co-training adopts the idea of increasing the effectiveness of learning through the collaboration of multiple learners which are inputted with multiple views (transformations) of image data.

Multiple Sub-heads

In the IIC method, there are two layers: the main head and the auxiliary head. Both of these heads update the ResNet backbone in turns, which also correspond to layers in the network. However, the architecture does not strictly adhere to the co-training principle (please refer to Section 3.2.2 for details), due to its shared single ResNet backbone that bifurcates into two heads. In Figure 3.1, the parallel two layers could be differentiated by the blue and green coloured frames. Each head incorporates a final fully connected layer and softmax activation function. Additionally to the main and auxiliary heads, IIC employs multiple sub-heads on each layer which are called concrete instantiations sub-heads. Each output head comprises 5 duplicates which are randomly initialised. In every epoch, the main head outputs five probability distributions. The advantage of having multiple sub-heads is the additional robustness. The IIC loss in Eq. 3.1 is computed for each five sub-heads and the average of them is used to update the network's parameters.

Auxiliary Overclustering

The parallel overclustering layer, which operates alongside the main network layer, categorizes input images into a more extensive set of clusters compared to the main head. While the primary output head generates predictions within the relevant class space, the auxiliary head produces predictions across a larger number of clusters,

surpassing the count of relevant classes. In Figure 3.1, the random variable of class assignment of the auxiliary head is symbolised by z^* while the main head is by z to highlight the fact that main and auxiliary heads output probability distributions from different sets of possible outcome. Despite the different output spaces for predictions, both heads utilize the same IIC loss 1 during training. The auxiliary layer is constructed to make the method versatile to input datasets that incorporate samples from unknown or distractor classes. An example of these datasets is the STL10 dataset which is a benchmark dataset for unsupervised classification tasks. By employing an overclustering layer, IIC utilises the content supplied by the irrelevant classes. Additionally, for datasets without any irrelevant classes, Ji *et. al* [4] argues that the network still benefits from the auxiliary layer.

3.1.3 Experimental Details

IIC uses the full dataset during both training and testing which corresponds to 60000 samples for dataset CIFAR10. Authors run the code in a total of 2000 epochs. Experimental details about network architecture, training and evaluation phases are mentioned below and Table 3.3 summarises all setting details for CIFAR10 together.

	CIFAR10
Train samples	60000
Test samples	60000
# of Epochs (t)	2000
# of Batch (b)	660
Sub-heads (h)	5
Main head Output channels (k_{gt})	10
Auxiliary head Output channels (k_{out})	70
Sample repeats (r)	3
Data Input channels (k_{in})	2
Optimiser	Adam
Optimiser learning rate	10^{-4}

Table 3.3: The experimental details of IIC for the dataset CIFAR10. The table is composed based on the paper and the code.

Network Architecture

The network backbone is ResNet34 and was randomly initialised before training. The backbone network could be terminated by 2 heads: main and auxiliary. Both

heads consist of 5 duplicates which we call sub-heads and each sub-head is a combination of a linear layer and a following softmax function. Each sub-head is randomly initialised. The auxiliary overclustering layer performs over-clustering, resulting in the auxiliary head forming a larger number of clusters than the ground truth classes. For instance, in the CIFAR10 dataset, there are 10 ground truth classes, while the auxiliary head forms 70 clusters. These two layers run in parallel and share the same backbone. The ResNet34 architecture is displayed in Table.

Training

During training, firstly the paired train samples are computed according to the transformation in Table 3.2. The sample repeats within each batch is 3, so the random transformation, $g(z)$, is repeated three times to compute augmented data. The batch size is 660. The transformations include grey-scale conversion of the images resulting in single color channel input. After the image pairs are converted to greyscale, the Sobel filter is applied which increases the input channel of images to 2: one for the vertical and one for the horizontal orientation. The IIC loss is gathered from sub-heads and averaged to train the network, during training IIC loss is the only criterion in the optimisation of the network and the ground truth labels are not used. Adam optimiser is used with the learning rate of 10^{-4} to optimise the network. At each epoch, both layers train the network, at first the auxiliary layer and the main layer follow as second. The pseudo code that is composed based based the code is attached as Algorithm 1.

Evaluation

During the evaluation process, the auxiliary layer becomes obsolete, and only the main head is utilized. Data samples are transformed in accordance with the test transformations outlined in Table 3.2 before being input into the backbone. After collecting predictions from each sub-head, accuracies are computed concerning the best sub-head and the average of sub-heads. The Hungarian matching algorithm is employed to identify the optimal mapping between the clusters predicted by the method and the ground truth classes. This step involves the use of ground truth labels; however, these labels are not exposed to the network. Consequently, the principle of unsupervised learning remains intact within the framework of the method.

Algorithm 1: Information invariant clustering algorithm

Input: Network with two heads $f_\theta^{(1)}$, $f_\theta^{(2)}$, Sobel filter ϕ_S, dataset $\mathcal{X}$, data transformation g, number of sample repeats R, number of epochs T, number of data transformations R, number of batches B, number of sub-heads H, mutual information I

$\mathcal{X} = \{x_b : b \in (1, ..., B)\}$
for $t \in \{1, ..., T\}$ **do**
  for $b \in \{1, ..., B\}$ **do**
    for $r \in \{1, ..., R\}$ **do**
      `/* Data transformations for each sample repeats */`
      $x'_{b,r} = g(x_b)$
    end
    $x_b, x'_b = \phi_S(x_b), \phi_S(x'_b)$; `/* Apply Sobel filter */`
    for $k \in \{1, 2\}$ **do**
      `/* Main and auxiliary heads are trained iteratively */`
      for $h \in \{1, ..., H\}$ **do**
        `/* Employ multiple sub-heads for each head */`
        $\mathcal{L}_h = I(f_{\theta_h}^{(k)}(x_b, x'_b))$
        $\bar{\mathcal{L}} = \text{average}(\mathcal{L}_h)$
        $\theta_h^{(t)} \leftarrow \text{Adam}(\bar{\mathcal{L}}, \theta_h^{(t-1)})$
      end
    end
  end
end

3.2 RUC

Inspired by robust learning, RUC is an unsupervised image classifier with an innovative approach. Its main feature that establishes novelty lies in its flexible structure, which allows it to be used on top of any existing unsupervised classifier. RUC serves as an add-on module, differentiating itself from other unsupervised image classifiers. RUC functions as a module for improving the performance of an existing unsupervised classifier, which it works in conjunction with. The prior clustering model which can also be called the baseline model could follow sequential, joint, or multi-step refinement approaches.

The cluster assignments outputted by the baseline model are referred to as pseudo-labels. The pseudo-labels are treated as a noisy dataset by the add-on model, and the model aims to revise the predictions to increase the classification performance of the baseline model. This innovative clustering model does not contradict the concept of unsupervised learning, even though the labels are inputted into the

model. The labels are pseudo-labels generated by the prior clustering model, rather than ground truth labels. So, the whole procedure (baseline clustering model and add-on improvement model) does not involve the ground-truth labels either in generating pseudo-labels (generated by baseline model) or in the revision part (by add-on improvement model).

The revision process of the pseudo-labels is based on common issues encountered in unsupervised learning, such as overfitting and faulty overconfident predictions. The method aims to alleviate the issue of overconfidence by applying two essential steps. The first step involves splitting the supplied dataset, classified by the prior classifier, into two categories: clean and unclean. Following the data sampling procedure, the network is trained according to the groups constructed. This way, the network effectively utilizes the dataset by applying distinct approaches to each set (clean and unclean) during the retraining process, thereby reducing overconfidence in the results and correcting the machine's misinterpretations.

With improved calibration, it is argued by Park *et. al.* [9] that RUC mitigates overconfident predictions and becomes robust against adversarial noise, leading to an enhanced unsupervised classification accuracy compared to the sole baseline classifier.

3.2.1 Loss Function

As mentioned above, the pseudo-labels are treated as a noisy dataset and RUC tries to refine the classification process by correcting the misclassified pairs. The loss function comprises 3 elements. The first element includes a loss function based on the clean set, $\mathcal{X}$, the second targets the strongly augmented clean set, $\mathcal{X}^{\mathcal{S}}$, and the third targets the unclean set, $\mathcal{U}$. The strongly augmented clean set, $\mathcal{X}^{\mathcal{S}}$, is the data-augmented version of the clean set, $\mathcal{X}$. The complete objective is as:

$$\mathcal{L}(\theta; \mathcal{X}, \hat{\mathcal{X}}, \hat{\mathcal{U}}) = \mathcal{L}_{\mathcal{X}^S} + \mathcal{L}_{\hat{\mathcal{X}}} + \lambda_{\mathcal{U}} \mathcal{L}_{\hat{\mathcal{U}}} \tag{3.5}$$

where $\lambda_{\mathcal{U}}$ is the control hyperparameter for the unsupervised i.e., unclean data (See Section 3.2.2 for further detail) loss. The first and second components of the right-hand side of the equation are computed by applying entropy minimization to clean and strongly augmented clean samples while the third component corresponds to consistency regularization loss term of unclean samples. Expansions of the first and third components are as in Eq. 3.6 and 3.7 respectively where $\hat{\mathcal{X}}, \hat{\mathcal{U}} = \text{MixMatch}(\mathcal{X}, \mathcal{U})$ and f_θ corresponds to RUC's network. Eq. 3.8 is the

formula of the third loss component where ϕ_A is a strong data augmentation transformation. The objective function relies on the predictability of clean samples and consistency within the class assignment of unclean samples through various augmentations that are applied to both.

$$\mathcal{L}_{\hat{\mathcal{X}}} = \frac{1}{|\hat{\mathcal{X}}|} \sum_{\hat{\mathbf{x}},\hat{\mathbf{y}} \in \hat{\mathcal{X}}} H(\hat{\mathbf{y}}, f_\theta(\hat{\mathbf{x}})) \tag{3.6}$$

$$\mathcal{L}_{\hat{\mathcal{U}}} = \frac{1}{|\hat{\mathcal{U}}|} \sum_{\hat{\mathbf{u}},\hat{\mathbf{q}} \in \hat{\mathcal{U}}} ||\hat{\mathbf{q}} - f_\theta(\hat{\mathbf{u}})||_2^2 \tag{3.7}$$

$$\mathcal{L}_{\mathcal{X}^S} = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x},\tilde{\mathbf{y}} \in \mathcal{X}} H(\tilde{\mathbf{y}}, f_\theta(\phi_A(\mathbf{x}))) \tag{3.8}$$

The details on the sampling strategies for dividing the dataset into clean and unclean sets and the consecutive retraining processes specified for each set are mentioned in the following section.

3.2.2 Overall Pipeline

The overall pipeline of the method consists of the following elements:

1. Data Sampling
2. Label Smoothing
3. Semi-supervised Network
4. Co-training
5. Co-refurbishment

Data Sampling

Data sampling is the first step of two main steps. The latter retrains a network in a semi-supervised way by utilizing the data that has been separated into two disjoint sets of clean and unclean. The sampling strategy has a significant role since the network learns in a semi-supervised manner, the credible pseudo-labels which form the clean set are considered as labelled while the unclean samples which don't meet

the criterion of reliability are regarded as unlabelled. The labels of unclean pairs are discarded before the retraining process. There are three sampling methods proposed in paper [9] that are implemented to extract clean samples from unclean including confidence-based, metric-based and hybrid approaches.

Confidence-based sampling selects clean samples based on the baseline model's confidence for the predictions. The confidence of a prediction is the highest value in the output probability distribution over all possible classes. If the confidence level surpasses a predefined threshold for a given prediction, the corresponding sample is classified as a clean sample and its label is used in the retraining process. The effectiveness of this data sampling method relies on the calibration capabilities of the baseline clustering model. If the model has low calibration i.e., the output predictions are over- or underconfident, the clean set which is deemed to be reliable may include a substantial number of inaccurately predicted pseudo-labels.

Metric-based sampling is based on employing a distinct representation learner and clustering method to filter out unclean samples. In this sampling strategy, the images are embedded into new representations within a different representation space by the feature learner model. Subsequently, a non-parametric classifier performs clustering based on these new representations. The obtained class assignment results are then compared with the pseudo-labels. If the pseudo-label and the class assignment determined by the auxiliary classifier are the same, the sample and its pseudo-label are put into the clean set to be used during the re-training procedure. Conversely, if a discrepancy arises, the pseudo-label is disregarded due to its lack of credibility and the unlabeled sample is added to the unclean set. For the CIFAR10 dataset, the feature learner model employed is SimCLR [38], while the clustering model utilized is the k-Nearest Neighbor (k-NN) algorithm.

The hybrid approach is straightforward in its implementation. If a pseudo-label is determined to be reliable according to both of the aforementioned sampling strategies, it is assigned to the clean set. Conversely, if the pseudo-label does not meet the criteria of both sampling strategies, it is categorized as unclean.

Class-wise Small-loss Selection (CSS) is utilized in [13]. It is not a sampling strategy employed in RUC [9] but rather proposed by us to test different sampling strategies. The concept of favouring samples with smaller loss values is a common principle in robust training methods [11]. With a slight modification, the CSS takes into account class-wise loss variance. Given that different classes can vary in difficulty, comparing loss values across classes may pose a challenge, potentially leading to an imbalance where some classes dominate the clean set. At the start,

pseudo-labels are grouped according to their respective classes, resulting in C classes. Later, cross-entropy loss is calculated for each sample class by class. From each label, we select k samples with the smallest loss values to be included in the clean set. Following the methodology presented in [13], the number of samples chosen from each label, denoted as k, is determined as $\min\left(\lceil \frac{n}{C} \times R \rceil, |S_j|\right)$, where $S_j = (\mathbf{x}_i, \tilde{y}_i) \in \mathcal{D} | \tilde{y}_i = j$, and n represents the total number of samples subjected to the sampling strategy. Different R values from the set $\{0.7, 0.5\}$ are tested.

Label Smoothing

A regular issue of DNNs is their susceptible nature to overfitting as evidenced by observations from experiments conducted by [10]. The success of the deep networks heavily relies on the accurate labels of the input training set if the learning approach involves the use of labels [11]. Without proper regularisation, networks with a large number of parameters can easily fall into the trap of memorizing the training data, leading to limited generalization on unseen data. The issue should be avoided to increase classification performance on unseen data, especially in the instance of dealing with a noisy dataset and weak data samplers. In this context, we define a strong data sampler as one which could divide the noisy data into clean and unclean, with the clean set containing accurately labelled data samples, achieving 100% precision. Conversely, a weak data sampler exhibits a lower sampling accuracy rate, resulting in the inclusion of faulty predictions in the clean set. This problem is addressed by RUC and elements from robust learning are applied to enhance the classification performance on unseen data.

RUC adjusts a semi-supervised learning setting and combines multiple robust training elements to enhance the model's resilience against adversarial noise. The robust training elements have been mentioned previously in Section 2.4. The first robust training technique that is utilized in RUC is sample selection. Another one is label smoothing, a regularization mechanism that helps to prevent overconfident faulty prediction error caused due to the potential noise in the clean set. The principle behind label smoothing involves adding random noise, drawn from a uniform distribution $\mathcal{N} \sim U(0, 1)$, to the labels within the clean set and converting the one-hot vector of the pseudo-label to a soft prediction vector. The formulation for the robust training feature described above can be expressed as follows:

$$\tilde{\mathbf{y}} = (1 - \epsilon) \cdot \mathbf{y} + \frac{\epsilon}{(C - 1)} \cdot (1 - \mathbf{y}) \tag{3.9}$$

where $\mathbf{y}$ represents the one-hot vector corresponding to the label of the image vector $\mathbf{x}$ within a pair $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}$. The variable ϵ is defined as a random variable sampled from a uniform distribution with a minimum value of 0 and a maximum value of 1. C denotes the total number of classes.

The smoothed labels, obtained through the label smoothing technique, are inputted into the Eq. 3.8 additional to the label guesses of the strongly augmented images, $\phi_A(\mathbf{x})$.

Semi-supervised Network

As mentioned above, the data clustered from the baseline model undergoes a sample selection process. Two disjoint sets are formed as the result of the procedure and the reliable samples and their class assignments are separated into a clean set to be used later. The labels of the rest of the samples are not acceptable for retraining so they are discarded. As a result, prior to retraining, two sets exist, a clean set containing labelled data and an unclean set where the samples require labelling. Subsequently, the nature of the problem shifts to a semi-supervised context, with the labels in the clean set serving as target labels. It's important to note that, the labelled data in the clean set does not represent ground truth labels, marking a significant distinction with conventional semi-supervised learning.

For the vanilla semi-supervised model, the baseline employed is MixMatch [53]. The losses associated with the MixMatch model are Eq. 3.6 and 3.7. More information about the baseline algorithm is included in the subsequent paragraph.

MixMatch [53] is a state-of-the-art semi-supervised model which aims to leverage unlabeled data in addition to the labelled samples for the classification task. The model integrates prevailing approaches that show a leading position in the benchmark techniques in the field of semi-supervised learning. These techniques combined within the loss function of MixMatch [53] incorporate entropy minimization, consistency regularisation and lastly traditional regularisation. By joining three different losses in a single one, the method can generate consistent low-entropy predictions for unlabeled data while ensuring alignment with conventional regularization techniques. This comprehensive approach allows the model to effectively utilize the unlabeled data, and enhance model generalisability thus improving the model's classification performance.

MixMatch's [53] loss term includes a p-Norm computation on unlabelled data which is responsible for the consistency regularisation and a cross-entropy computation on labelled data while sharpening function on unlabelled data predictions

enforces the network to minimise the entropy of label distribution. Lastly, weight decay and MixUp augmentation are employed to further regularise the network.

MixUp is a data augmentation technique used in MixMatch [53] to create artificial data by combining two submitted input data samples, unlabelled or labelled (See Eq. 3.10). In the case of image data, the output image is a weighted summation of the pixel content of the pair, potentially belonging to different classes. The augmented data, $\hat{\mathbf{x}}$, is then inputted into the network, f_θ, to obtain the class prediction. The class predictions are later inserted in either cross-entropy (Eq. 3.6) or p-norm (Eq. 3.7) formulas with the augmented labels.

$$\begin{aligned} &\lambda \sim \text{Beta}(\alpha, \alpha) \\ &\lambda^{'} = \max(\lambda, 1-\lambda) \\ &x^{'} = \lambda^{'} x_1 + (1-\lambda^{'}) x_2 \\ &p^{'} = \lambda^{'} p_1 + (1-\lambda^{'}) p_2 \end{aligned} \tag{3.10}$$

In Eq. 3.10, the value of λ is sampled from a Beta distribution, with α as its parameter. The λ and its complement to 1 (1-λ) are computed and the larger value is denoted as λ'. Later the set of weights $(\lambda, 1-\lambda)$ is multiplied with the set consisting of the pair of samples and their associated labels $((x_1, p_1), (x_2, p_2))$. The larger weight, λ', is used to scale the first sample and its label, while the smaller weight scales the second. This step is necessary to identify the dominant sample after the weight multiplication process. If the first pair of samples and its label (x_1, p_1), in other words, the dominant pair, originates from the labelled dataset, it will remain considered labelled after the MixUp augmentation. Conversely, if it was unlabelled initially, it will continue to be considered unlabeled. The resulting augmented pair (x', p') is used to calculate the loss function based on its label status.

The augmented labels, $\hat{\mathbf{y}}$, are obtained similarly to the process of augmenting images. The associated labels of the input images are blended together using the same weight applied in the image augmentation process. Both labelled and unlabelled data are utilized in MixUp augmentation. However, since the target labels are not initially available for the unlabeled images, they need to be matched with semantic classes before applying the augmentation. To achieve this, a specific procedure is applied before the augmentation process begins.

Firstly, labelled and unlabelled data are randomly perturbed by weak random data augmentation (horizontal flip and random crop) which can change the image's content while leaving the semantics of the classes untouched. After the unlabelled images are transformed m times, the expanded dataset is fed into the network and

the average of m predictions are sharpened by using the sharpening function (See Eq. 3.11). On the other hand, labelled images undergo weak transformation only once, so averaging and sharpening functions are not applied to them. At this stage, both the initially unlabeled and labelled data have class assignments, making the application of MixUp augmentation possible to be applied on data.

$$\text{Sharpen}(p, T)_i = \frac{p_i^{\frac{1}{T}}}{\sum_{j=1}^{L} p_j^{\frac{1}{T}}} \tag{3.11}$$

where p is a probability distribution and the T is the hyperparameter called "temperature". Temperature values smaller than 1 increase the difference in magnitude between the probability values of the distribution.

This is the part where MixMatch reshapes the MixUp augmentation formulation to align with its objective. While prior works in the semi-supervised field applied MixUp on only unlabelled data, MixMatch mixes the labelled and unlabeled images in the process of streaming synthetic images and incorporates their associated labels in the loss function as the target label. The same weights ($\lambda^{'}$ and $1 - \lambda^{'}$) of the weighted summation in the synthesis of augmented images are also applied to their labels to create augmented labels. The augmented labels combine labels that could be from the clean set and/or the guessed labels from the unclean set. The augmented labels are the target labels for the clean loss term, Eq. 3.6 and the unclean loss term, Eq. 3.7.

The complete formulation is as in Eq. 3.10, where the λ is the weight, $x^{'}$ denotes the mixed image and $p^{'}$ represents the mixed label. The complementary weight is computed and the greater weight from the pair determines the loss term which the MixUp augmented labels are used for as the target labels. If the weight for the labelled data is larger compared to its pair, the outputted augmented label is used in the entropy minimization loss term if else in the consistency regularisation loss term. The consistency regularisation factor which restricts the network from producing inconsistent class assignments for different augmentations of an image is:

$$||\mathbf{p}_{\text{model}}(\mathbf{y}|\text{Augment}(\mathbf{x}); \theta) - \mathbf{p}_{\text{model}}(\mathbf{y}|\text{MixUp}(\mathbf{x}); \theta)||_2^2$$

where the $\mathbf{p}_{\text{model}}(\mathbf{y}|\mathbf{x}; \theta)$ is the model network which generates a prediction distribution $\mathbf{y}$ over the given number of classes accordingly to the network parameters θ when input $\mathbf{x}$ is given and Augment($\mathbf{x}$) is a weak data transformation.

The complete algorithm is attached below as excerpted from the MixMatch [53]

paper.

Algorithm 2: MixMatch algorithm

Input: Batch of labeled examples and their one-hot labels $\mathcal{X} = ((x_b, p_b)); b \in (1, ..., B)$, batch of unlabeled examples $\mathcal{U} = u_b; b \in (1, .., B)$, sharpening temperature T, number of augmentations M, Beta distribution parameter α for MixUp.

for $b \in \{1, ..., B\}$ **do**
  $\hat{x_b} = \text{Augment}(x_b)$
  for $m \in \{1, ..., M\}$ **do**
    $\hat{u}_{b,m} = \text{Augment}(u_b)$
  end
  $\bar{q}_b = \frac{1}{M} \sum_m \mathbf{p}_{\text{model}}(y|\hat{u}_{b,m}; \theta)$ `/* Compute average predictions */`
  $q_b = \text{Sharpen}(\bar{q}_b, T)$ `/* Apply temperature sharpening */`
end
$\hat{\mathcal{X}} = ((\hat{x}_b, p_b); b \in (1, .., B))$
$\hat{\mathcal{U}} = ((\hat{u}_{b,m}, q_b); b \in (1, .., B), m \in (1, ..., M))$
`/* Combine and shuffle labelled and unlabeled data */`
$W = \text{Shuffle}(\text{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}}))$
`/* Apply MixUp to labelled data and entries from` W `*/`
$\mathcal{X}' = (\text{MixUp}(\hat{\mathcal{X}}_i, W_i, \alpha); i \in (1, ..., |\hat{\mathcal{X}}|))$
`/* Apply MixUp to unlabeled data and the rest of` W `*/`
$\mathcal{U}' = (\text{MixUp}(\hat{\mathcal{U}}_i, W_{i+|\hat{\mathcal{X}}|}, \alpha); i \in (1, ..., |\hat{\mathcal{U}}|))$

The integration of MixMatch as the baseline algorithm complements RUC's objective by leveraging the benefits of semi-supervised learning and incorporating the strengths of MixMatch. RUC aims to boost classification accuracy while simultaneously enhancing the robustness and reliability of the baseline classifier.

The semi-supervised MixMatch algorithm is fully adopted by RUC albeit with an additional step for higher robustness. Both labelled and unlabelled sets are refined by applying a co-training step before MixUp augmentation which could be represented:

$$\bar{\mathcal{X}}^{(1)}, \bar{\mathcal{U}}^{(1)} = \text{Co-refinement}(\mathcal{X}, \mathcal{U}, \theta^{(1)}, \theta^{(2)})$$

$$\hat{\mathcal{X}}^{(1)}, \hat{\mathcal{U}}^{(1)} = \text{MixMatch}(\bar{\mathcal{X}}^{(1)}, \bar{\mathcal{U}}^{(1)})$$

Rather than relying on the predictions of a single network, the samples avail the guesses from two networks instead of one during the re-labelling. The process introduces greater variance and enables the network to be less vulnerable to noise accumulation. This process, known as label refinement, is a key component of robust

training, which will be discussed in more detail in the next section.

Co-training

Co-training is another robust training element adopted by RUC to prevent overfitting. In co-training, two networks run in parallel and output predictions from both are used to refine the labels. This operation is called co-refinement. The refinement process is applied to both labelled and unlabelled samples. For labelled pairs $(\mathbf{x}, \mathbf{y})$ in clean set $\mathcal{X}$, the refinement process is held by the formula:

$$\bar{\mathbf{y}} = (1 - w^{(2)}) \cdot \mathbf{y} + w^{(2)} \cdot f_{\theta^{(2)}}(\mathbf{x}) \tag{3.12}$$

where $w^{(2)}$ is the confidence value of the second network in the prediction of the label for the given input $f_{\theta^{(2)}}(\mathbf{x})$. The same equation (Eq. 3.12) is used for the second network, with the confidence value and the predicted label are replaced by the counter network's values.

For unlabeled images, $\mathbf{u}$, in unclean set $\mathcal{U}$, a different approach is employed for the refinement process. Due to the absence of target labels for the unclean set, co-training works as an ensemble of both networks' learning where predictions of both are used jointly. A surrogate label is obtained by merging both networks' guesses for each weakly augmented set. Taking the average of label guesses from multiple networks fed up with weakly augmented versions of data yields diversity for set $\mathcal{U}$ and adds robustness. The formulation of the procedure is:

$$\bar{\mathbf{q}} = \frac{1}{2M} \sum_{m} (f_{\theta^{(1)}}(\mathbf{u}_m) + f_{\theta^{(2)}}(\mathbf{u}_m)) \tag{3.13}$$

In this equation, M represents the number of transformations applied to an unlabeled image, and $\bar{\mathbf{q}}$ denotes the average of the label prediction.

Co-refurbishment

The training clean set is expanded by the refurbishment stage. In each epoch, by applying the above steps, the labels for unclean samples are predicted by the network. If the confidence value of the predicted label is higher than the threshold, the unclean data sample is added to the clean set with its one-hot encoded prediction as its label. Since two networks are trained during each epoch, there will be two label predictions for an unclean sample in every epoch. It is considered sufficient if either of the two networks has a confidence level exceeding the threshold value for

the respective label. The threshold value is 0.9 for the CIFAR10 dataset.

3.2.3 Experimental Details

RUC adopts a train/test split methodology for its experimental setup. In order to comply with the standard principle approved by the DL community, the train/test split methodology is followed for both methods that are utilised in this study. The length of the train and test sets are 50000 and 10000 respectively. The number of epochs is equal to 200. The details regarding the experiment can be found in Table 3.4.

Network Architecture

As mentioned above, the network architecture of RUC exhibits flexibility, which means it adapts the network architecture of the chosen baseline model. RUC is applied to both SCAN and TSUC in paper [9]. In the case of experiments conducted with the SCAN model, the backbone consists of a ResNet18 with a 2x2x2x2 layered structure. A 3x3 convolution filter is applied to the input with a stride and padding value set to 1 before entering the layered structure. Batch normalization is applied after each convolutional layer and before the activation function, ReLU. The network's layered structure concludes with an adaptive average pooling of size (1,1). Projection shortcuts are implemented in ResNet architecture to match the dimensions of two consecutive layers. The configuration of the convolution layers follows the same approach as IIC: weight initialization uses He initialization with a fan-out mode.

Training

The training procedure of RUC involves multiple elements and three distinct loss terms. To present the sequence clearly, we have organized the steps in a list structure and indicated the associated loss functions for those steps that include a loss function. During training, Stochastic Gradient Decent (SGD) is utilised as the optimiser of the algorithm.

1. Data Sampling
    (a) Confidence
    (b) Metric
    (c) Hybrid

- Output: Clean and unclean set

2. Re-training

   ! Steps concerning 2 separate loss term calculations happen in parallel!

   i. Loss term Eq. 3.8

      A. Strong Augmentation: applied to clean images

      B. Label Smoothing: applied to clean labels

      - Output: $\mathcal{L}_{\mathcal{X}^s}$

   ii. Loss terms Eq. 3.6 and 3.7

      ! For the 3 below, succeeding steps get affected by the previous steps, the output of the prior is the input of the subsequent

      A. Refinement and sharpening: applied to clean labels

      B. Label guessing and sharpening: applied to unclean samples

      C. MixMatch: applied to clean and unclean set

      - Output: $\mathcal{L}_{\hat{\mathcal{X}}}$ and $\mathcal{L}_{\hat{\mathcal{U}}}$

Evaluation

During the evaluation, the test samples are inputted into the network and predictions are outputted for each. The total number of batches is 100. Hungarian match is employed to match the predicted clusters to the ground truth classes, similarly to IIC.

	CIFAR10
Train samples	50000
Test samples	10000
# of Epochs (t)	200
# of Train batches (b)	200
# of Test batches (b_{test})	100
# of networks (k)	2
# of weak augmentations (m)	2
Sharpening temperature T	0.5
Weight of unsupervised loss (λ_u)	25
Confidence threshold (τ_1)	0.99
Embedding model	SimCLR [38]
# of neighbors (n)	100
Refurbishment threshold (τ_2)	0.9
Optimiser	SGD
# of optimisers	2
Optimiser learning rate	10^{-2}
Weight decay	5×10^{-4}
Momentum	0.9

Table 3.4: The experimental details of RUC for the dataset CIFAR10. The table is composed based on the paper and the code.

Chapter 4

Experiments

In this chapter, the previously mentioned objectives are fulfilled by conducting the appropriate experiments. Detailed discussions of each experiment are provided in their respective sections. To address the primary objective of this study, which is to assess the effectiveness of robust learning training in unsupervised image classification tasks, we must thoroughly investigate the performance differences between using and not using robust learning methods. This involves carefully formulating the problem, designing suitable experiments, and evaluating the results using appropriate metrics. Before delving into the main experiments for unsupervised image classification using the IIC and RUC methods, it is crucial to establish a baseline by replicating the results of two benchmark methods in the field. This baseline is covered in the following section.

4.1 Experiment 0: Preliminary Work

The initial step is to reproduce the performance results of each method. This step serves to validate the correctness and consistency of our implementation in comparison to the original work. Furthermore, the results obtained from this preliminary step serve as a reference point for making comparisons and evaluating the performance of the proposed methods. Below, we provide details regarding the configuration and the reproduced results for each algorithm.

4.1.1 Reproduction of IIC

We aim to reproduce the results reported in the original paper by following the same implementation details and evaluation metrics. The dataset used for this experiment

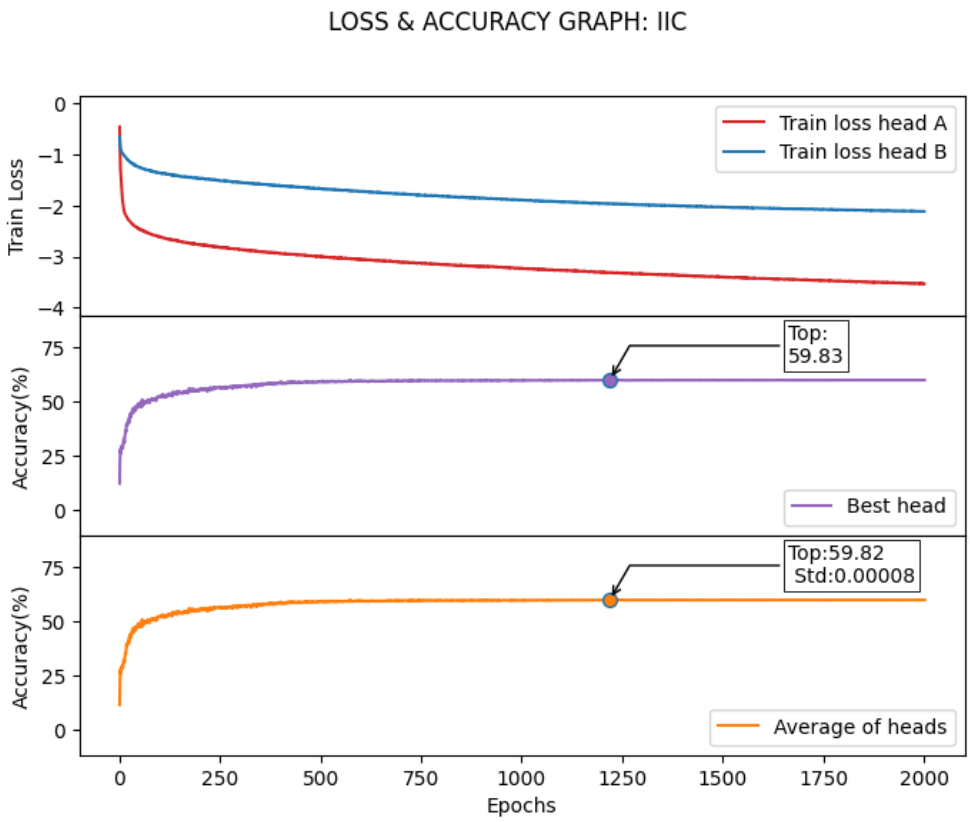

Figure 4.1: Loss and Accuracy Graph of IIC: The graph shows the train loss of both heads and the accuracy curves belonging to the best and average sub-heads. The results are reproduced according to the original code.

is the standard benchmark dataset CIFAR10 which is one of the datasets that is also used in the original paper for the evaluation of the method. The original accuracy performance of the method is compared to the accuracy achieved with our implementation in Table 4.1 which is constructed based on Figure 4.1.

	Original	**Our Implementation**
Lowest loss sub-head	61.7	59.83
Avg. sub-head ± STD	57.6 ± 5.01	59.82 ± 8.26 $\times 10^{-5}$

Table 4.1: IIC accuracy performance. The original paper evaluates the method by accuracy metric according to the lowest loss head and average head with the standard deviation of the sub-heads

According to the lowest loss sub-head, the results are reproduced with a variance of 3%. The reproduction of Benchmark Method IIC involves the following steps:

1. Preprocessing: The dataset is preprocessed using standard techniques such as resizing, normalization, and data augmentation, as described in the original paper. The details of each transformation are performed as it is described in detail in the previous section.

2. Training: The model is trained on full dataset using the same architecture and hyperparameters as specified in the original paper. The training process follows the recommended protocol, including the optimization algorithm, learning rate schedule, and epoch number. It is specifically important to match the setting to the original, to verify that our implementation faithfully represents the original algorithm and allows for a fair and meaningful comparison between our implementation to the authors' implementation in [4].

3. Evaluation: The trained model is evaluated on the complete set as reported in the original paper. The only metric used in the paper is accuracy, the same approach is followed to be consistent.

Although we followed a nearly identical procedure to the original code during the above steps, we opted to convert method IIC, originally developed in Python 2, into Python 3. The primary reason for this conversion was the compatibility issues encountered during the compilation process. Most online platforms and the computing clusters we used primarily or exclusively support the newer version of the language and its libraries. We have included a comparison table that outlines the library versions we used and the ones employed in the original code, which can be found in A.1.

By reproducing the results of ICC with a slight reproduction variance of approximately 3%, we can ensure the reliability of our experimental setup and validate the consistency of our implementation with the state-of-the-art.

As mentioned earlier, IIC is a method sensitive to network initialization. The initialisation of the network parameters, data augmentation and shuffling process of the data involves randomness. As a result, the subsequent training process is inherently stochastic. This variability may account for the differences observed in the results.

Furthermore, differences in software environments, especially variations in library versions and functions, may have contributed to the deviation of reproduction. During the Python conversion, several functions were replaced, as they had been discontinued, with substitutes that offer the same functionality but have different implementations

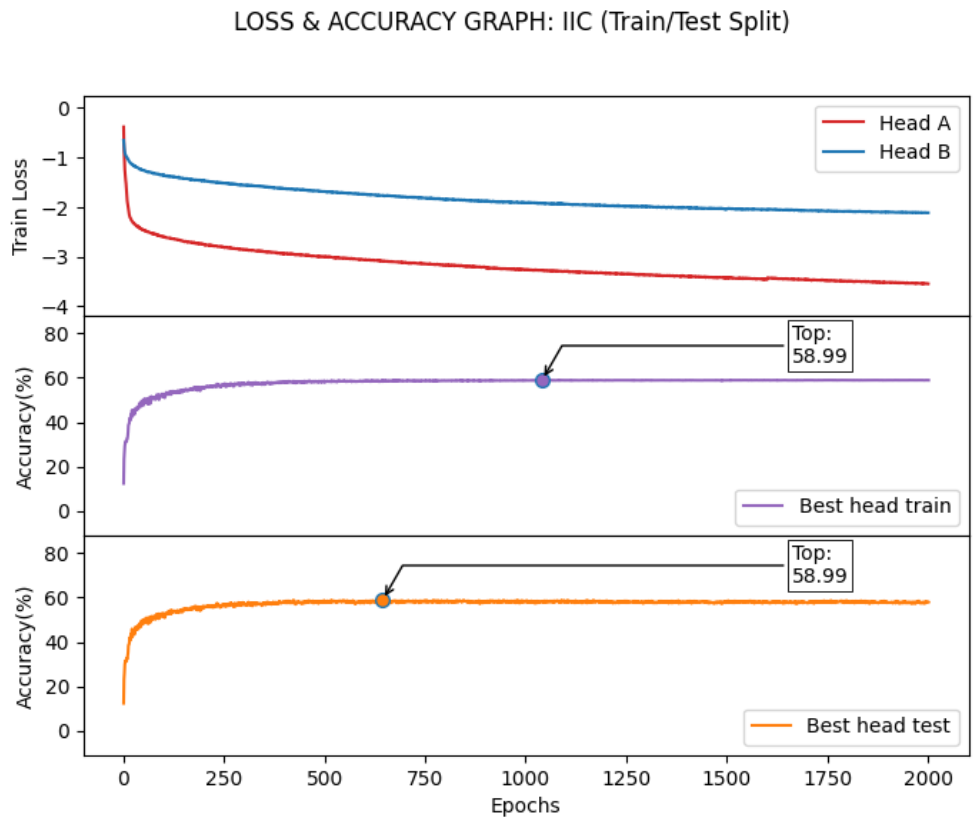

Figure 4.2: Loss and Accuracy Graph of IIC with train-test split principle: The graph shows the train loss of both heads and the accuracy curves belonging to the best and average sub-heads. The results are produced according to the principle of train-test split principle and the code is altered accordingly.

Train-test Split for IIC

Train-test split is applied before conducting further experiments (Exp. 4.2, 4.3, 4.4). As mentioned, IIC uses the whole dataset to train the network and establishes the train accuracy as test accuracy as well. Following [17], CIFAR10 dataset is split into train and test and the test data is kept unseen from the network during training. This way, the generalization skills of the models can be studied. Figure 4.2 illustrates the curves of loss and accuracy of IIC when the principle is applied. The results presented in the figure are compared to the original accuracy rates in Table 4.2. The accuracy and head B's loss curves are gathered in a single plot in Figure **??**. This figure depicts the relationship between the accuracy and loss curves better. Loss of head B decreases as the epoch increases and simultaneously both accuracy curves show an increase.

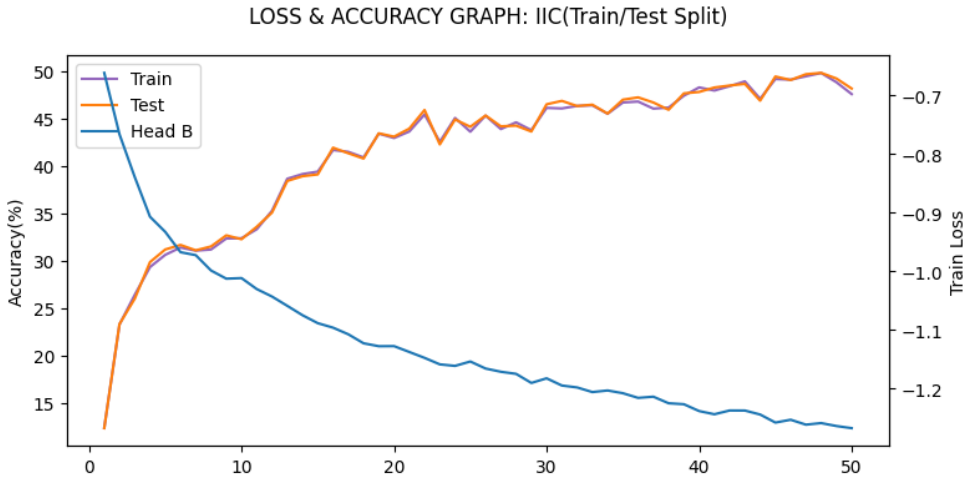

Figure 4.3: Loss and Accuracy Curves of IIC during the first 50 epochs: Head B's train loss and accuracy curves are displayed in the same plot. The results are excerpted from IIC with the train-test split. The figure shows the similarity between the train and test curves clearly.

	Original		**Our Implementation with split**
Best sub-head	61.7	Best sub-head for train	58.99
Avg. sub-head	57.6	Best sub-head for test	58.98

Table 4.2: IIC accuracy performance with train-test split.

4.1.2 Reproduction of RUC

The experiment is conducted aiming to reproduce the results presented in the original paper of RUC using the CIFAR10 dataset. Comparison table below displays the accuracies achieved in the original paper compared to those obtained with our implementation: The reproduction procedure involves:

1. Preprocessing: The dataset is preprocessed using the same preprocessing techniques employed in the original paper. The training data is augmented by using random horizontal flips and crops before being normalized and inputted into the network. For the strongly augmented clean set, Randaugment [54] is employed.

2. Training: The model is trained using the specified architecture, hyperparam-

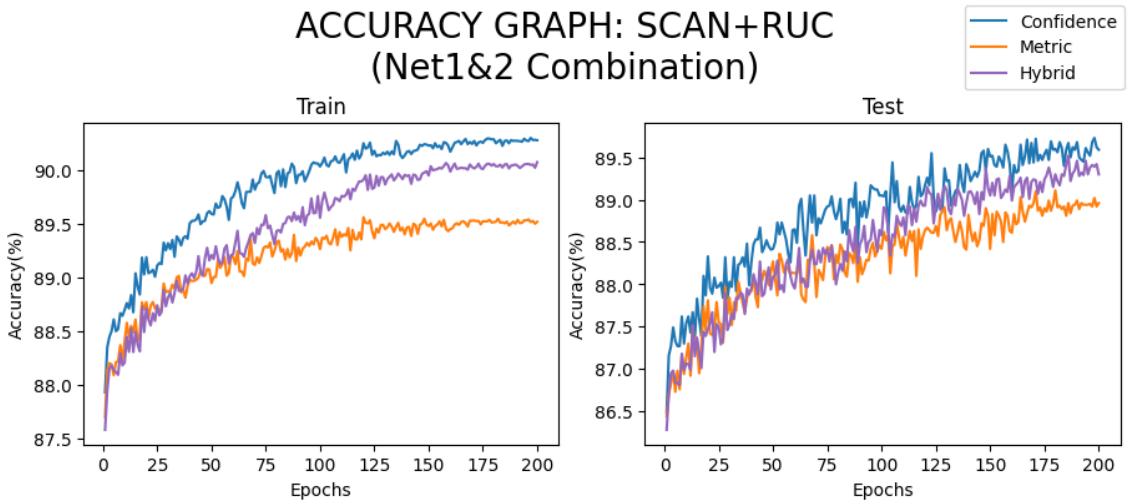

Figure 4.4: Reproduced SCAN+RUC: Accuracy Graphs for both training and testing phases. Showcasing each of the implemented sampling strategies.

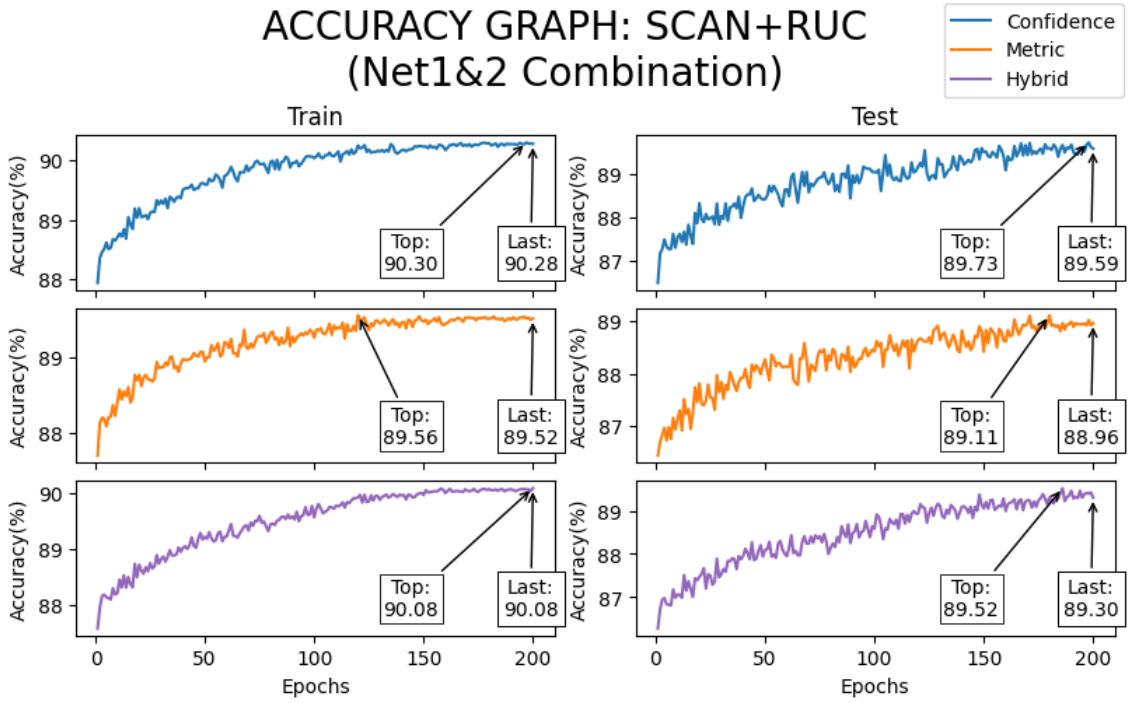

Figure 4.5: Reproduced SCAN+RUC: Accuracy Graph. SCAN+RUC accuracy rates are shown with respect to epochs for each sampling strategy. Repeated for both train and test phases. The top and last accuracies on each subgraph are included.

	Original	**Our Implementation**
SCAN	88.7	88.7
SCAN + RUC (Confidence)	90.3 / 90.3	90.3 / 90.3
SCAN + RUC (Metric)	89.5 / 89.5	89.6 / 89.5
SCAN + RUC (Hybrid)	90.1 / 90.1	90.1 / 90.1

Table 4.3: RUC improvement performance. The accuracy metric is shown in percentage. The baseline model is SCAN. The best and last accuracies are reported.

eters, and training protocol from the original paper. According to the paper, SCAN and SimCLR are used. To obtain the predicted pseudo-labels for each image SCAN pre-trained model is utilised while RUC is trained for 200 epochs to revise these pseudo-labels. SimCLR pre-trained model is employed in the metric-based sampling strategy.

3. Evaluation: During the evaluation, only center-crop and normalisation are applied to the images, resulting in the test set. The trained model is evaluated on the test set using the same evaluation metrics as reported in the original paper.

By reproducing the results of RUC with a reproduction variance of 0.04, we establish a solid foundation for testing the hypotheses and evaluating the performance of the IIC and RUC methods in the following experiments.

4.2 Experiment 1: Entropy-based Clustering and Confidence Relationship

As mentioned in Section 2.3.4, the joint approach clustering is susceptible to generating degenerate solutions and often requires mechanisms to alleviate this problem. Even when these mechanisms are applied, sensitivity to initialization remains a concern. For instance, IIC has been shown to be sensitive to the initial network initialization [17]. This sensitivity can lead to the accumulation of misknowledge in the early epochs, potentially resulting in false predictions with increasing confidence values in later epochs [9], ultimately leading to overconfident false labels.

Related Objective: *Identify the relationship between entropy-based joint approach clustering and confidence of labels.*

4.2.1 Experimental Setup

- Dataset: CIFAR10 dataset is utilized.
- Preprocessing: No change applied to the preprocessing steps.
- Training: As a more appropriate and usually accepted way in the community, IIC method is trained on the trainset (as described in Section 4.1.1) following the recommended architecture, hyperparameters, and training protocol instead of the full dataset.
- Testing: During testing the test set is used.
- Calibration Analysis: The predictions and their associated confidence values are obtained from the best sub-head, which corresponds to the sub-head with the highest accuracy rate during test time. The model's confidence and calibration are assessed using the predictions generated by that sub-head.

4.3 Experiment 2: Performance Gain of RUC applied on IIC

In this experiment, our primary objective of investigating the potential performance enhancements attainable through the application of the robust learning training approach proposed in RUC to the joint approach clustering algorithm, specifically IIC is tested. By implementing RUC with IIC, we aim to discern the extent to which this integration can bolster the overall performance of the clustering algorithm. In RUC paper, it is asserted that the robust training when applied to overconfident results generated by entropy-based models, both the calibration and performance of the baseline model can increase [9].

Related Objective: *Identify the performance gain achievable when robust learning training approach proposed in RUC is applied to the joint approach clustering algorithm, namely IIC.*

4.3.1 Experimental Setup

- Dataset: CIFAR10 dataset is utilized.
- Preprocessing: The preprocessing steps are changed with the IIC's preprocessing steps which involve grey scale transformation and sobel filtering addition

to the preprocessing steps embraced in the experiment in Section 4.1.2. This step is necessary to be compatible with the network.

- Training: RUC was built upon SCAN's network architecture in the original paper. This network architecture changed from SCAN's (ResNet18) to IIC's (ResNet34) network architecture. Later, the network parameters which are obtained from the experiment in Section 4.2 are used as the initial state of the network before training. The trainset is inputted into the network after the preprocessing step. Even though the network architecture changed, the loss function and the model are constructed in alignment with the RUC as in Section 4.1.2) following the hyperparameters, and training protocol at first. Since the auxiliary layer and multiple sub-heads present in IIC model are not compatible with the loss functions and model of RUC, the auxiliary layer is discarded and only a single head from the 5 sub-heads is used. The single head is re-initialised following RUC to avoid the same local optimum. Due to the results obtained the hyperparameter, namely the learning rate is changed to observe its effect on the accuracy.

- Testing: During testing the test set is used as usual, following the train-test split principle. Grey scaling and Sobel filtering are applied to the test set.

- Performance Gain: The loss and accuracies of different sampling strategies are plotted. The procedure is executed for both training and testing time with 2 different learning rates.

4.4 Experiment 3: Sampling Strategy of RUC

In the paper by Park et al. [9], a parallel analysis to ours is presented, focusing on the performance of sampling strategies. Following this analysis, the authors emphasize that a comprehensive exploration of sample selection can contribute significantly to enhancing the overall RUC method. This is because establishing a reliable set of labelled data within a semi-supervised network structure is known to exert a substantial positive influence on performance.

Related Objective: *Search sampling strategies in the literature, apply and determine the sampling strategy which has the potential to form a rich set of clean samples.*

4.4.1 Experimental Setup

- Dataset: CIFAR10 dataset is utilized.
- Preprocessing: The preprocessing steps kept the same as the previous experiment.
- Training: The network architecture kept the same as the previous experiment. The training set is sampled after the preprocessing step. Sampling is made according to different data selection strategies. These strategies involve confidence-based, metric-based and hybrid sampling strategies which are from the original paper, additionally, we have implemented the CSS strategy introduced in 3.2.2. All the data selection strategies are applied on the training data and obtained clean and unclean sets are provided to the network for training.
- Testing: During testing the test set is used as usual, following the train-test split principle. Grey scaling and Sobel filtering are applied to the test set.
- Sample Selection Analysis: Precision, recall and F1 scores of the sets formed by each sampling strategy are listed and compared to the accuracy rates achieved.

Chapter 5

Results and Discussion

Each experiment which is conducted in the previous chapter delivers a result which is used to (dis)prove the associated hypothesis introduced in Chapter 1. This chapter is organized into three sections, each dedicated to the outcomes of the experiments from the previous chapter. Within each section, we identify the hypothesis associated with the deliverable, discuss the results, and share our insights.

5.1 Result 1: Confidence Analysis of IIC

Referring to hypothesis 1, testing whether an entropy-based joint approach clustering generates overconfident results; we plotted the confidence distribution of clustered data, displayed the calibration graph and calculated the Expected Calibration Error (ECE) of IIC.

5.1.1 Confidence Distribution

We plotted the probability distribution of IIC's predictions. During the class assignment process, the model labels the sample as the class which has the highest probability among these class probabilities. The highest probability is the confidence level of that prediction.

When plotted as a line graph, the probability distribution of IIC is extremely skewed to the right, displaying a spike at the top value where the confidence is around 1.0 and a plateau with a value close to zero elsewhere. Due to the low expressiveness of that plot, instead, we displayed the distribution of the probability by a pie chart. The pie chart in Figure 5.1 shows that the model, in general, predicts the label of a sample (during the testing phase) with a very high confidence rate,

between the range of $(0.99, 1.0]$.

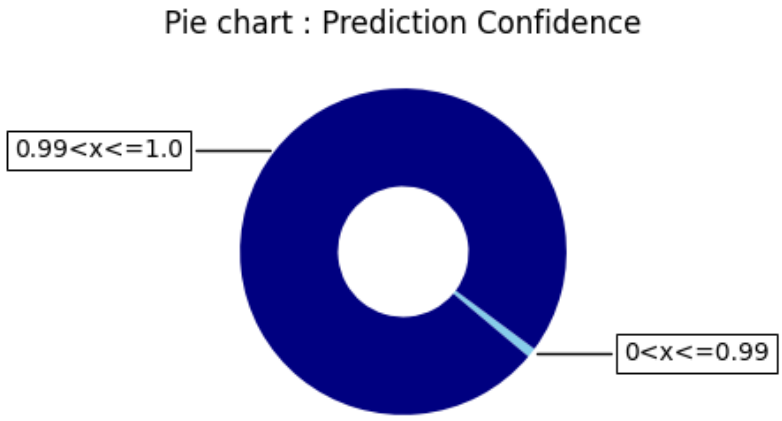

Figure 5.1: IIC (w/ train-test split) Confidence Pie Chart: The predictions made with a confidence value larger than 0.99 are indicated by the dark blue colour while the slice with the light blue colour is the portion of predictions made with a confidence less than or equal to 0.99.

5.1.2 Calibration Skills

In addition to the confidence distribution, ECE is calculated and displayed in Figure 5.2 which displays the calibration skills of the model in a graph. The calibration plot shows the confidence levels associated with the predictions on the x-axis and the empirical probability of the model making correct predictions on the y-axis. We preferred to show the plot in a bar graph for easier interpretation where each bin corresponds to a 0.1 range in the confidence axis. The dashed line is used as a reference line. The confidence bars follow this line when the model is well calibrated in other words when the prediction probability is equal to the true prediction accuracy of that prediction. The bars below the reference line indicate overconfident predictions while bars above the line shows that the model is underconfident for those predictions. ECE shows the miscalibration in percentages and its formula is given as:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|$$

where n is the number of data, B_m corresponds to the samples of m-th equally spaced buckets in total of M and $acc(B_m)$ and $conf(B_m)$ are the average accuracy and confidence of B_m respectively.

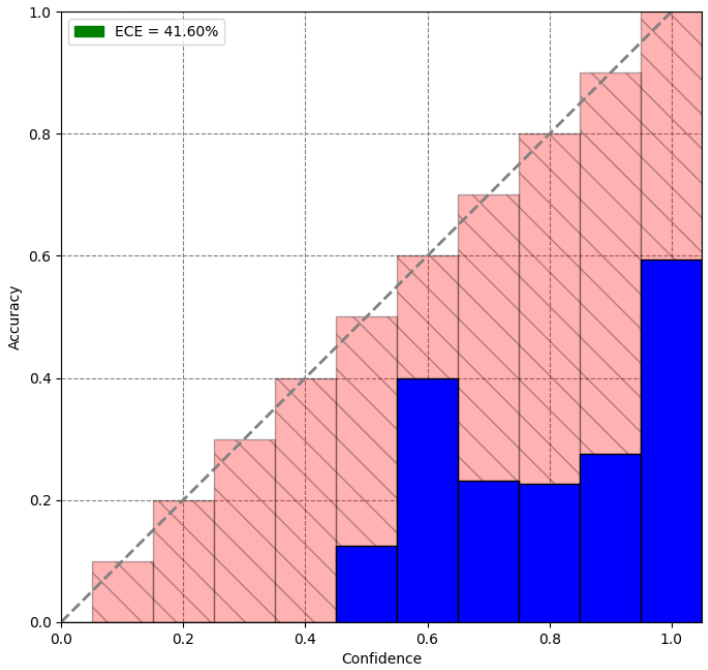

Figure 5.2: IIC (w/train-test split) Calibration Plot: The dashed line and the red blocks belong to the ideally calibrated model whereas blue blocks represent the calibration of IIC. ECE is 41.60%.

Discussion: The first experiment, rooted in the Hypothesis 1, focuses on assessing the calibration skills of the IIC. The relationship between entropy-based joint approach clustering models and overconfident results are observed in line with [9]. As an example of an entropy-based joint clustering model, IIC generates overconfident results. We observed in Figure 5.1 that IIC make predictions with a confidence rate 1.0 while having an overall accuracy performance of 58.98 (results from our implementation). The calibration plot in Figure 5.2 shows a parallel result where the predictions with a confidence level higher than 0.5 have an accuracy rate lower than their confidence rate which indicates that IIC has a low calibration skills and generates overconfident results. As an example, only 60% of the predictions with a confidence of 1.0 are true positives. Since almost all predictions are made by IIC with 100% confidence, the last bin also shows an approximation of the model's overall accuracy rate.

5.2 Result 2: Performance Analysis IIC+RUC

Referring to hypothesis 2, to test whether RUC can boost the performance of IIC, we applied the RUC model to the IIC network and analysed the performance achieved from two perspectives: calibration and accuracy. We refer to the concatenation of IIC and RUC as IIC+RUC. We plotted the confidence distribution, and calibration graph to observe the change in calibration after RUC is applied to IIC. The accuracy curves are plotted and the rates of IIC and IIC+RUC are listed to be compared.

5.2.1 Calibration Skills of IIC+RUC

Confidence distribution and calibration graphs are displayed to compare the results of IIC+RUC with the performance of IIC alone.

Confidence Distribution

The confidence distribution is plotted again as a pie chart (Figure 5.3) as in the previous section. The main difference between these two figures is the increase in predictions which have a confidence value lower than 0.99. Concurrently, the predictions with a confidence larger than 0.99 decreases.

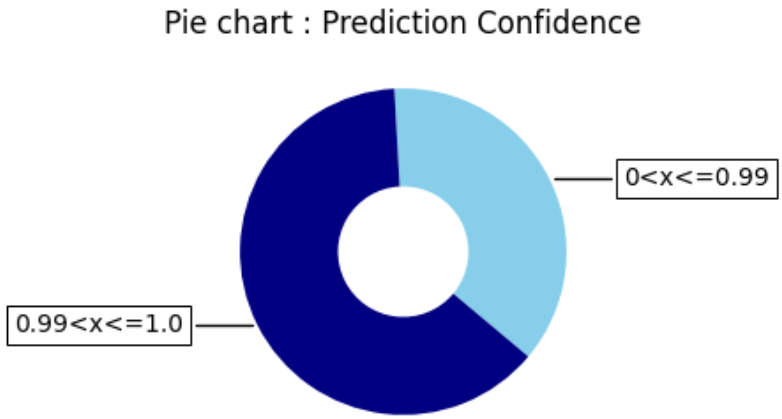

Figure 5.3: IIC+RUC Confidence Pie Chart: The predictions made with a confidence value larger than 0.99 are indicated by the dark blue colour while the slice with the light blue colour is the portion of predictions made with a confidence less than or equal to 0.99.

Calibration Skills

We repeated the experiments made on IIC. The calibration skills of IIC+RUC are displayed in Figure 5.4. This time, the blue bars are closer to the red ones in a visible degree, except for the last bin. The visible difference in the graphs can also be shown by the decrease in ECE values. ECE decreases from the value of 41.6% to 28.72% after RUC is applied.

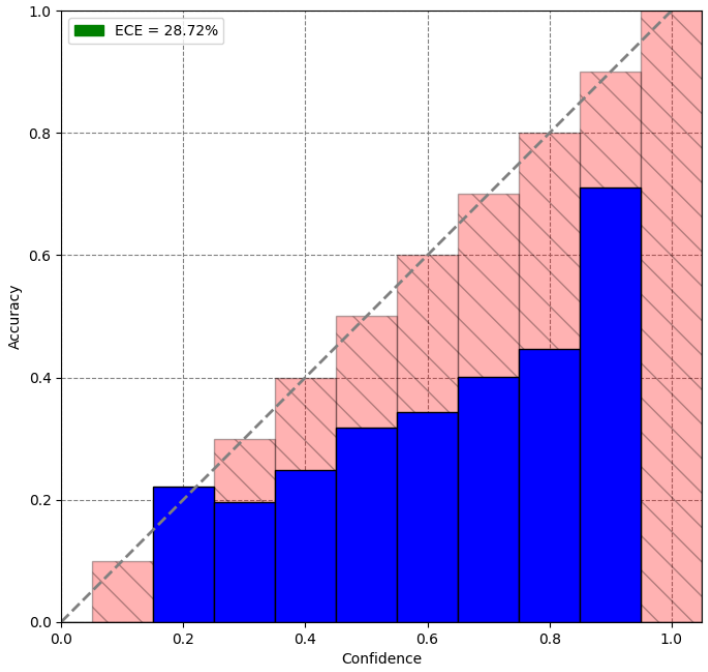

Figure 5.4: IIC+RUC Calibration Plot: The dashed line and the red blocks belong to the ideally calibrated model whereas blue blocks represent the calibration skills of IIC+RUC.

Discussion: The results (in Figure 5.4 and 5.3) suggest that the calibration skills of the baseline model, IIC, are improved when it is compared to the initial state (Figure 5.2). IIC+RUC is less confident during the process of classification. The predictions with very high confidence(higher than 0.99) are far less. In Figure 5.4, the confidence buckets are below (except the bin with 0.2 confidence) the diagonal line which means that the model still generates overconfident results, however, the significant change in calibration error, namely 12.88%, indicates an improvement in calibration. ECE shows the difference between IIC+RUC and an ideally calibrated model and it summarises the calibration skills graph numerically. The overall calibration of IIC+RUC compared to sole IIC is higher, proving that RUC is capable of improving the calibration of the model it is applied to and mitigating

the problem of overconfident results.

5.2.2 Accuracy Performance of IIC+RUC

The accuracy performance of the model IIC (with the train-test split) is included in the previous section. The results of the accuracy performance when RUC is applied on IIC (with the train-test split) are shown below with three figures to be compared.

Loss and Accuracy Change

Figure 5.5 consists of four subplots, a subplot which displays the train losses is followed by three accuracy subplots. We plotted the train loss curves belonging to networks 1 and 2. The train losses show a sharp decrease during (approximately) the first 10 epochs and converges around 5. The curves of train losses of both networks follow similar shape by overlaping each which indicates a high degree of similarity in their learning processes. This similarity is expected as the only difference between them is the initialisation of the parameters in their last final fully connected layer.

Conversely to the decreasing train losses, as the number of epochs increases, train and test accuracy rates increase. While during the initial epochs we observe fluctuations in the accuracy curves, the fluctuations slow down as process of training continues, resulting in a stabilisation in accuracy. The decreasing train loss and increasing accuracy curves exhibit a healthy learning process of the model. Train and test curves (purple and orange lines) in the last three subplots overlap each other by displaying resemblance in values. This resemblance in train and test values is also observed with the accuracy curve of sole IIC in Figure 4.3.

The middle two subplots belong to the first and second network accuracy curves while the last one is the combination of both. For this figure, the accuracy curves are shown separately to display the learning process of each network with respect to the epochs. In the following graphs, the performance of networks is not observed individually, the total performance of the model is emphasized.

For each sampling strategy, the accuracy curves during training and testing time with respect to epochs are plotted in Figure 5.6. The hybrid sampling approach achieves better accuracy rates compared to other approaches after convergence for both training and testing phases. In training, the hybrid approach achieves a top value of 60.91% and in testing a top value of 60.47%. Towards epoch 200, accuracy decreases slightly. We annotated the top and last accuracy rates on each accuracy curve in Figure 4.3. These best (top) and last accuracy rates are gathered in Table

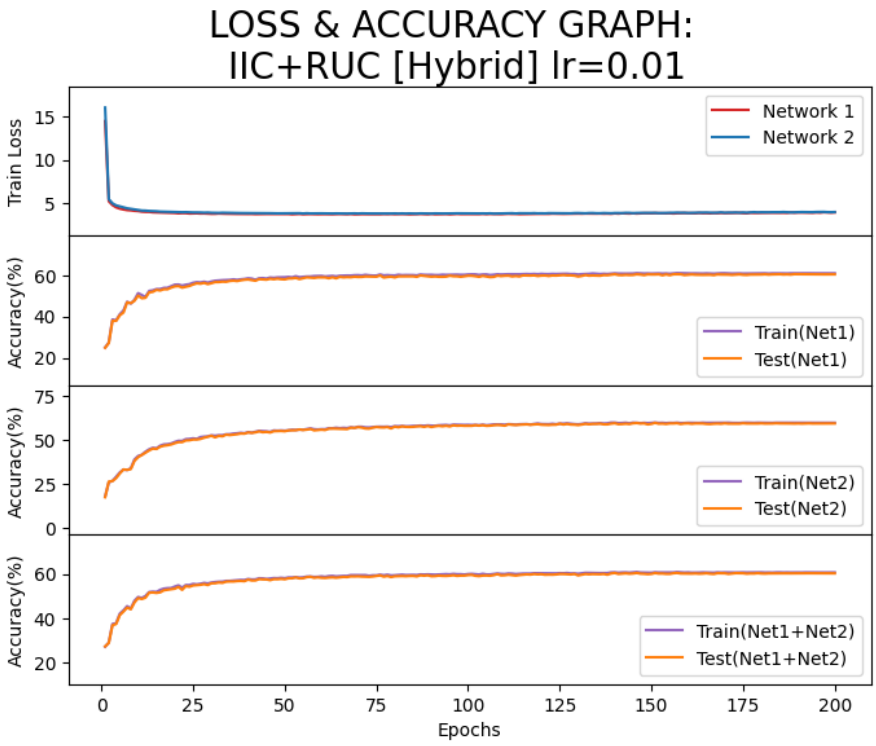

Figure 5.5: IIC+RUC Loss and Accuracy Graph: The sampling strategy is hybrid and the lr. (learning rate) is 0.01. Both networks' train loss curves are plotted in the first subplot. Accuracy curves belonging to each individual network and their combination are plotted for the train and test phases.

5.1 to be compared to the best accuracy achieved by sole IIC (w/ train-test split in our implementation).

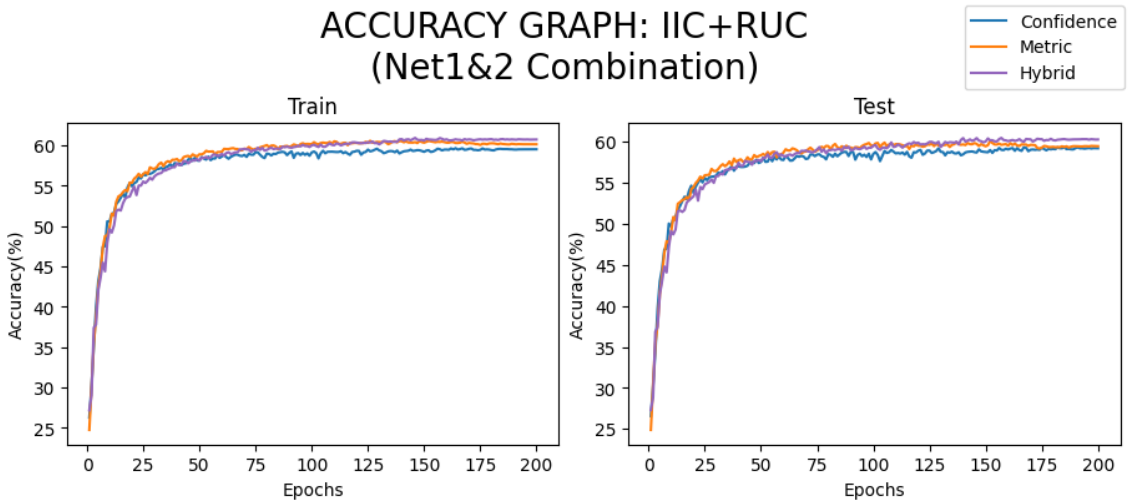

Figure 5.6: IIC+RUC Accuracy Graph: Showcasing each sampling strategy from the original paper. Plotted for both training and testing phases.

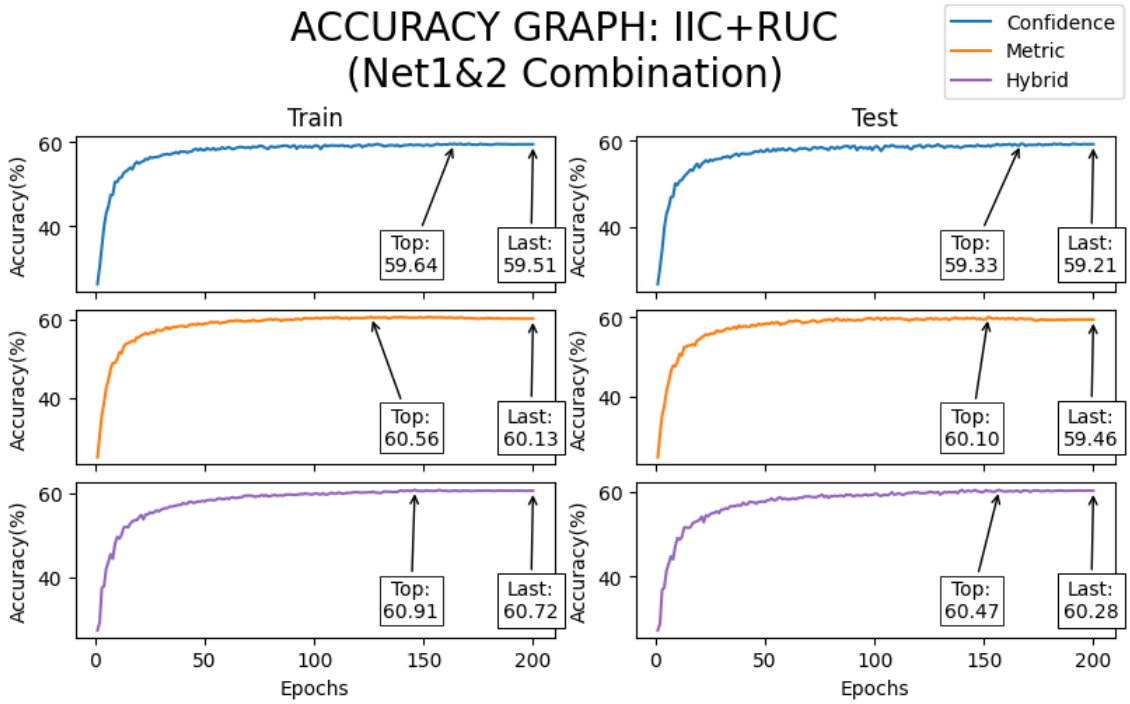

Figure 5.7: IIC+RUC Accuracy Graph with best/last accuracy annotation: Accuracy curves belonging to each sampling strategy are plotted separately. The top and last accuracy rates are shown on each sub-graph. Repeated for both train and test phases.

Discussion: When we compare the accuracies achieved by the IIC+RUC model to IIC's (as shown in Table 5.1), we observe that an improvement is present when RUC is employed for each sampling strategy. Notably, when employing the hybrid approach, a substantial improvement of 1.92 percentage points (pp) is attained. From the perspective of accuracy, it is evident that utilising RUC to IIC leads to enhanced prediction accuracy.

	Best / Last
IIC	58.99
IIC + RUC (Confidence)	59.64 / 59.51
IIC + RUC (Metric)	60.56 / 60.13
IIC + RUC (Hybrid)	**60.91 / 60.72**

Table 5.1: RUC Accuracy Improvement. The accuracy metric is shown in percentage. The baseline model is IIC with the train-test split. The best and last accuracy rates are reported. The highest rate is highlighted in bold.

Learning Rate Effect

The accuracy curve of IIC+RUC in Figure 5.6 starts from approximately 25% while the latest accuracy rate of IIC is 58.99%. The accuracy rate shows a substantial amount of decrease at the start of the epochs. Before the training starts the head is randomly initialised as described in Section 4.3.1 to avoid local minima achieved by sole IIC. Since the parameters that comprise the head is randomised, a decrease is expected. A decrease in the accuracy is also observed in Figure 4.4, at the start of the re-training process of SCAN+RUC. The final accuracy rate of SCAN is 88.7%, however, due to the re-initialisation of the final fully connected layer, it decreases to 87.5%. Even though, for both of the models, IIC and SCAN, a decrease in the accuracy at the start of the re-training process is observed, the decrease is more apparent for IIC+RUC. The significant decrease can be related with the the learning rate. To study the possibility, the behaviour of the model accuracy when the learning rate is changed is investigated.

We plotted the accuracy curves of IIC+RUC with two different learning rates. In the original implementation, the learning rate is 0.01 while we have decreased it by 10%. A high learning rate can cause the low starting accuracy observed during re-training. When the learning rate is decreased by 10%, we observed that the starting accuracy value becomes 58.24%. This value is closer to 58.99% which is the final accuracy achieved by sole IIC. Consequently, decreasing the learning rate adjusts the initial accuracy, creates a similar case that was observed in the accuracy curves of SCAN+RUC. The highest accuracy achieved with a learning rate of 0.001 is 59.89, whereas with a learning rate of 0.01, it reaches 60.91.

Discussion: We questioned the underlying cause of the notable accuracy drop observed during the initial epoch of re-training of IIC+RUC. Changing the learning rate improved the starting accuracy value which revealed a connection between the starting accuracy and the learning rate. On the other hand, since we have decreased the learning rate significantly, the network became susceptible to getting stuck in a local minimum. Consequently, while lower learning rates improved the starting accuracy, they also resulted in a lower overall top accuracy achieved. With the exception of this experiment, the learning rate kept the same as stated in the original paper for all others.

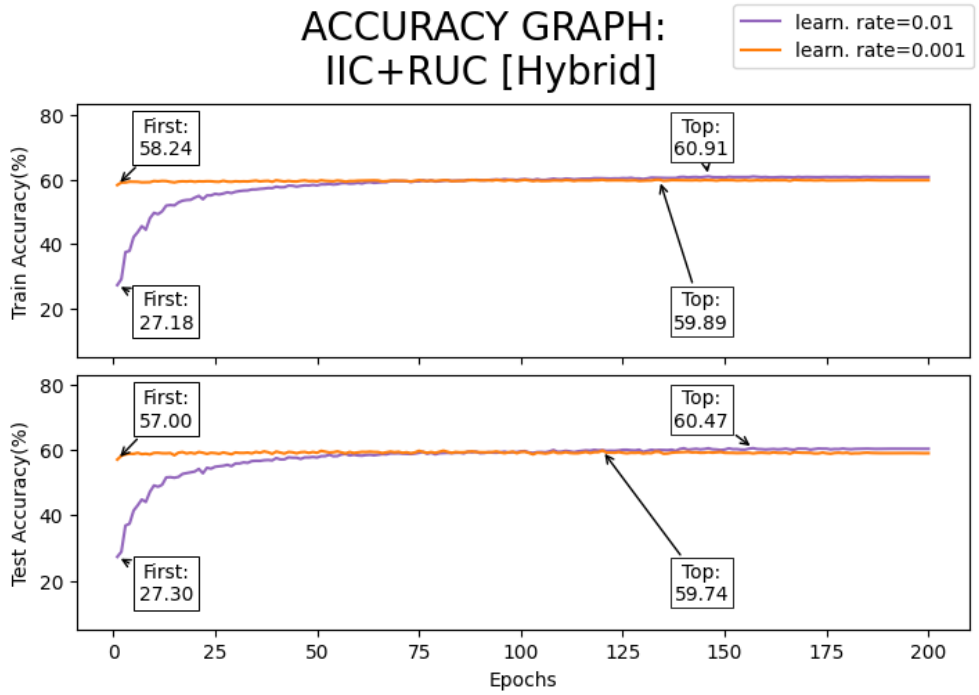

Figure 5.8: IIC+RUC Accuracy Curves with different learning rates: The accuracy curves are plotted for two learning rates: 0.01 (original) and 0.001. Repeated for training and testing phases.

5.3 Result 3: Performance Analysis of Sampling Strategies

Referring to Hypothesis 3, to observe the performance change when a different sampling approach is used in the data sampling part of the RUC model, we proposed to implement CSS. The effect of the new sampling strategy on the accuracy is compared to the ones achieved by the strategies in the original paper [9]. The confidence-based sampling strategy is substituted with CSS and combined with the metric approach to create a second hybrid approach. In order to differentiate this new combination from the original hybrid approach in the original paper, it will be referred to as 'Hybrid-2'. To further investigate the relationship between the sampling strategy and accuracy curves, the behaviour of SCAN+RUC is revisited and the patterns observed in SCAN+RUC are compared to IIC+RUC.

5.3.1 Base Case: SCAN+RUC

Referring to the previous chapter where the results of SCAN+RUC are reproduced, in Figure 4.4 and 4.5, we observed that the sampling strategy has a substantial

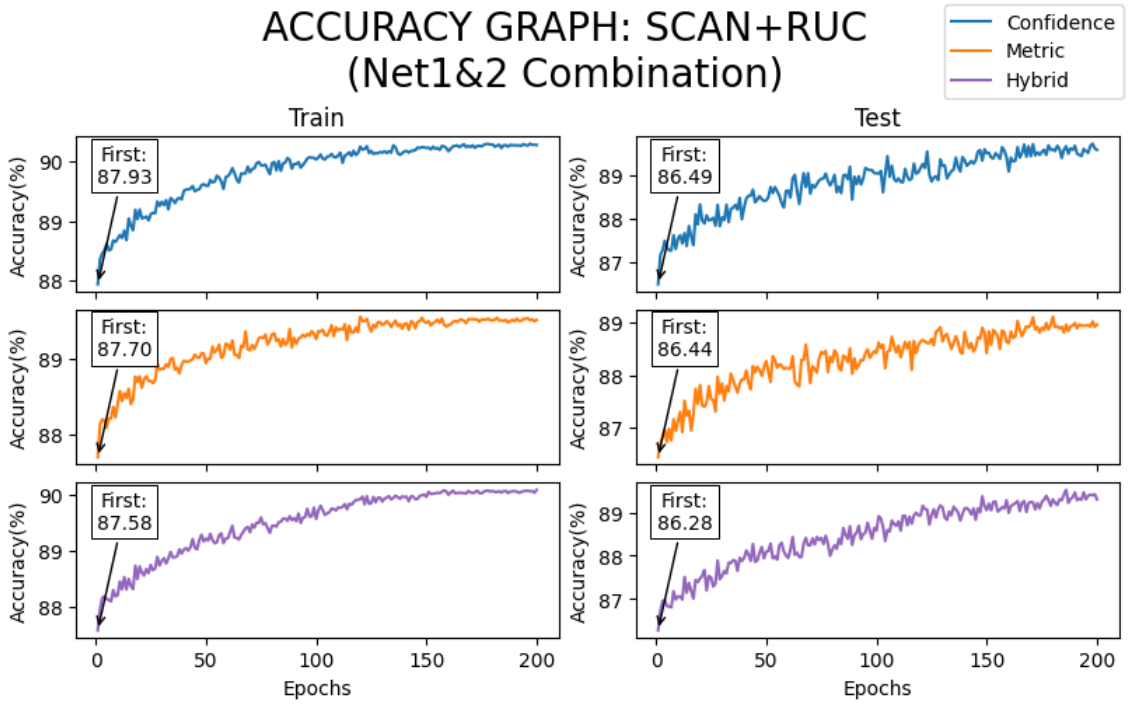

Figure 5.9: Reproduced SCAN+RUC Accuracy Graph. The initial accuracy rates are annotated.

impact on the final accuracy rates. There is a difference of almost 1 pp in the final train accuracy achieved and 0.63 pp in the final test accuracy achieved between confidence-based (the sampling strategy with the best accuracy performance for SCAN+RUC) and metric-based (the sampling strategy with the worst accuracy performance for SCAN+RUC) approaches.

The sampling strategy directly affects the initial quality of the clean and unclean sets. Consequently, the accuracy rates at the beginning. The initial training accuracy rates are compared to the quality of the sets to indicate a relationship. In this regard, Figure 4.5 is reconstructed with annotated initial epoch accuracy rates (please see Figure 5.9). When the initial accuracy rates are considered, the best is achieved by the confidence-based and the worst performance belongs to the hybrid approach. Between the best and worst sampling strategy for the initial accuracy, there is a 0.45 pp difference which increases towards the final epochs.

For SCAN+RUC, the confidence-based sampling strategy starts with the highest accuracy value of 87.93. Through the epochs, it maintains this lead, achieving the highest accuracy of 90.28 at the end of the 200 epochs compared to the accuracy rates by the metric and hybrid approaches.

	Accuracy	**Sampled Set Quality**		
	Initial	Precision	Recall	F1 score
SCAN + RUC (Confidence)	**87.93**	0.93	**0.94**	**0.93**
SCAN + RUC (Metric)	87.70	0.92	0.93	0.92
SCAN + RUC (Hybrid)	87.58	**0.94**	0.89	0.91

Table 5.2: SCAN+RUC Initial Accuracy and Quality of Sampled Sets: A comparison of initial accuracy rates and the quality of the sampled sets is presented for each sampling strategy. The highest values are highlighted in bold.

Discussion: Comparing the accuracy rates to the quality of the sets (as shown in Table 5.2), we observe that the sampling strategies are competent at distinguishing clean samples from unclean samples, achieving commendable scores in both precision and recall. We further compared the initial accuracy to the F1 score of the sets, which combines precision and recall. This comparison reveals that sets with a higher F1 score exhibit a higher accuracy rate at the beginning. Also the sampling strategy with the highest initial accuracy remains its position after 200 epochs, resulting in the highest final accuracy.

5.3.2 Our Case: IIC+RUC

The choice of sampling strategy has a substantial effect on the top accuracy. For our choice of the dataset as the baseline, CIFAR10, the effect of sampling strategy on the best accuracy varies in between 59.64 (confidence-based) and 60.91 (hybrid), the characteristics of the noisy dataset and the compliance and capability of the strategy to distinguish the clean from the unclean samples within that chosen dataset could vary. The result is highly correlated with the noisy dataset and naturally the baseline model which outputs the noisy dataset. We can observe it in Figure 5.7, for the same dataset, CIFAR10, when the baseline model changes from SCAN to IIC, the confidence-based sample selection drops down to the bottom of the list in accuracy.

By taking into consideration the nature of IIC which is susceptible to generate overconfident predictions, we proposed to change the confidence-based approach with the CSS approach. Firstly, the proposed sampling strategy, CSS, is implemented and the hybrid-2 is composed. The new hybrid approach with CSS is compared to the original hybrid approach. Two different values for the hyperparameter R in the construction of CSS are used which are 0.7 and 0.9. The initial quality

of the sets formed and their impact on the initial accuracy are investigated for our case: IIC+RUC. The Table 5.3 summarises the comparison.

Secondly, Figure 5.10 shows the accuracy curves achieved by different sampling strategies both from the original paper and the newly proposed one. Lastly, two sampling strategies, CSS and confidence-based, are compared in depth by exhibiting the initial states of clean and unclean sets and the accuracy curves.

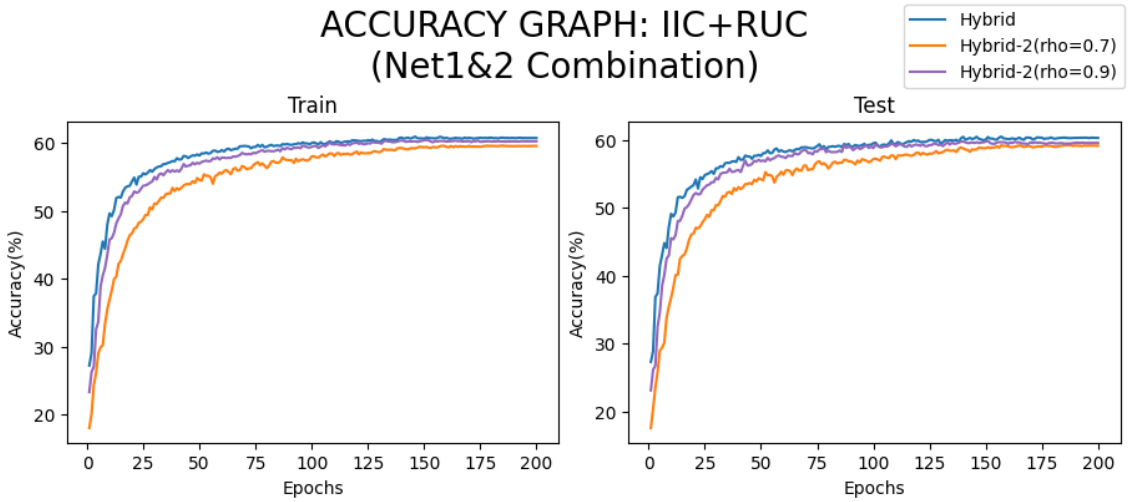

Figure 5.10: IIC+RUC Accuracy Graph: Showcasing sampling strategies Showcasing both the sampling strategies from the original paper and the newly proposed ones. Plotted for both training and testing phases.

	Accuracy	**Sampled Set Quality**		
	Initial	Precision	Recall	F1 score
IIC + RUC (Hybrid)	**27.18**	0.67	**0.95**	**0.79**
IIC + RUC (Hybrid-2, R=0.7)	17.97	0.67	0.68	0.68
IIC + RUC (Hybrid-2, R=0.9)	23.28	**0.68**	0.87	0.76

Table 5.3: IIC+RUC Initial Accuracy and Quality of Sampled Sets: For each sampling strategy, the initial accuracy rates are compared to the quality of the sets sampled. The highest values are highlighted in bold.

Table 5.3 displays a similar relationship between the F1 score and initial accuracy which has been observed in the base case: SCAN+RUC. In between the hybrid and hybrid-2 approaches, the highest initial accuracy rate is achieved by the hybrid approach from the original RUC strategies. In parallel, this sampling strategy has the highest F1 score. Once again, the set with the highest F1 score has the highest initial accuracy.

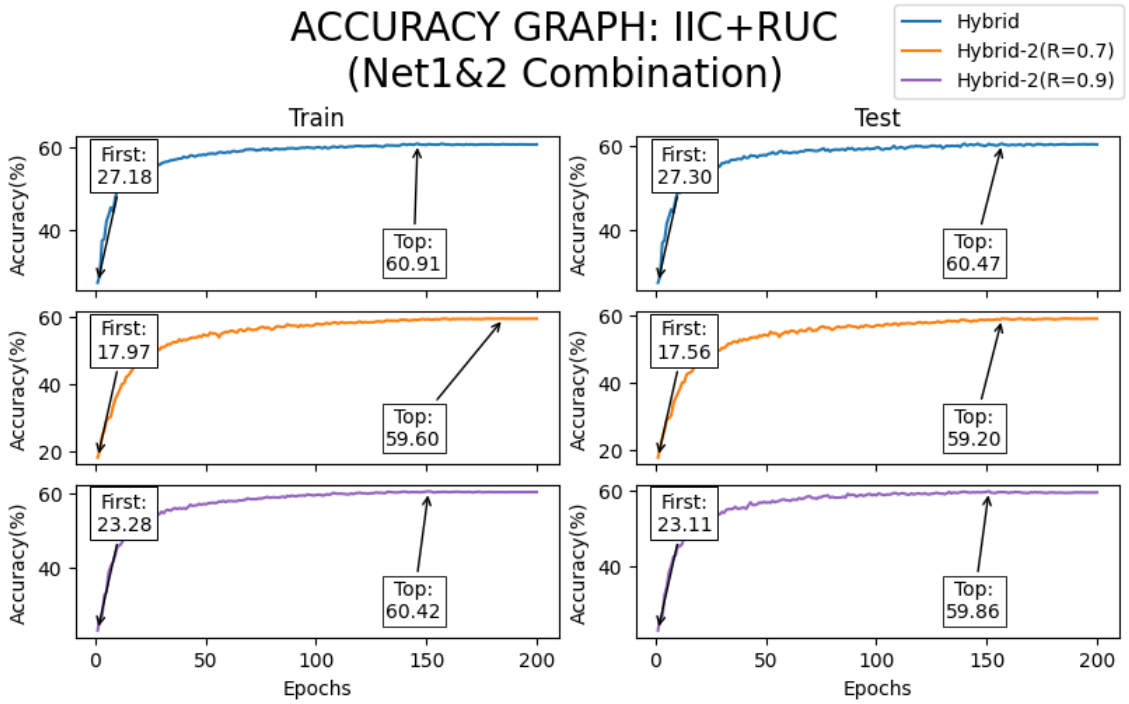

Figure 5.11: IIC+RUC Accuracy Graph with initial/best accuracy annotation: Showcasing sampling strategies from the original paper (Hybrid) and newly proposed ones (Hybrid-2 with R values 0.7 and 0.9).

The final accuracy rates achieved by our experiments in addition to the ones achieved by other state-of-the-art methods are collected in Table 5.4.

In Figure 5.13, the ratios of the clean and unclean samples within the total train samples are given. We observed the parallel result as discussed in Section 5.1. When a confidence threshold is applied as selection criteria on the noisy dataset formed with overconfident predictions, 99% of the train samples are labelled as clean whereas the true positives (a sample with a true predicted label sampled to the clean set) are only 58.7% of the total.

For the CSS approaches with R value set to 0.7 and 0.9, the clean set is relatively smaller with an approximation of 70% and 80% of the total train samples respectively. For CSS when R is 0.9, true positives have a ratio of 53.1% in total which corresponds to 0.59 precision. From the perspective of sensitivity, the strategy has a recall of 92%. The clean set has a high rate in the completeness of the true positives while the correctness of the clean set is lower, resulting in an F1 score of 0.79. Table 5.5 shows precision, recall and F1 score for each sampling strategy.

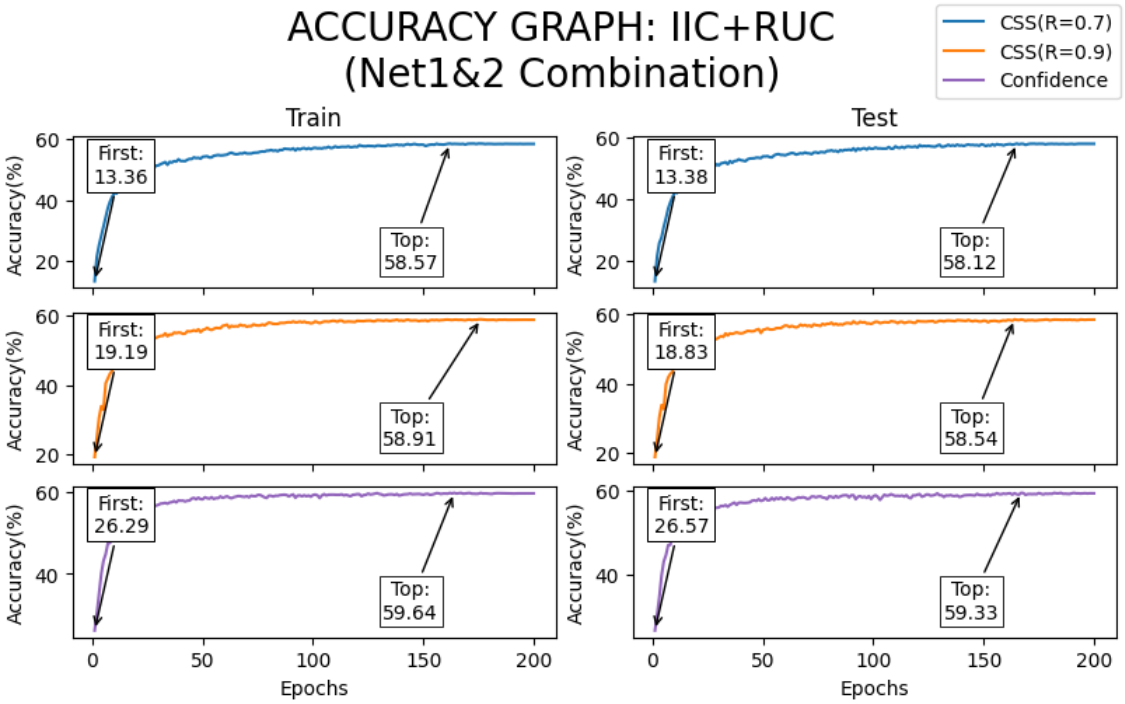

Figure 5.12: IIC+RUC Accuracy Graph with initial/best accuracy annotation: Showcasing sampling strategies from the original paper (Confidence) and newly proposed ones (CSS with R values 0.7 and 0.9).

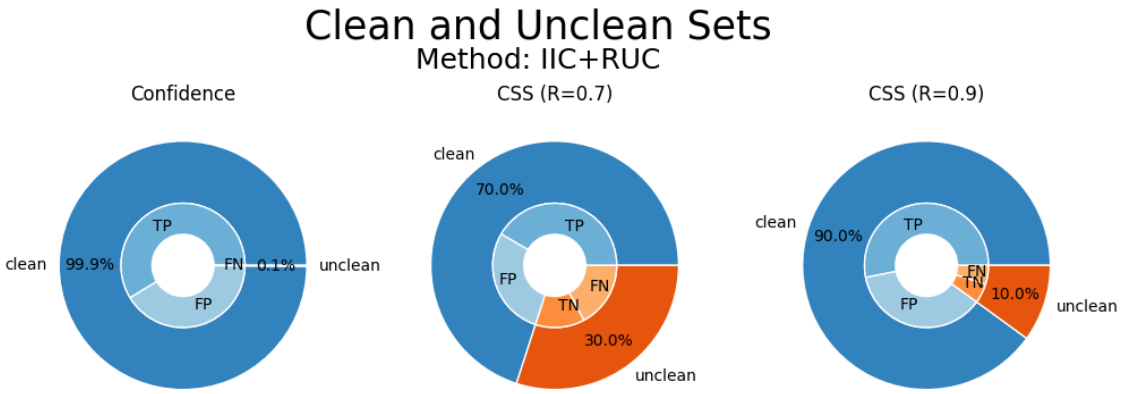

Figure 5.13: Clean and Unclean Sets of IIC+RUC: pie charts are formed by each Sampling Strategy included in [9]. The ratios of clean and unclean are indicated for comparison.

Discussion: When the accuracy curves are compared in Figure 5.10, the hybrid approach from the original paper has the highest initial and final accuracy rates compared to the newly proposed hybrid-2 strategies. The goal of proposing the CSS strategy was to boost the final performance of the network considering its low calibration. When the confidence-based sampling strategy is replaced by CSS and

Method	CIFAR-10
k-means [51]	22.9
Spectral Clustering [55]	24.7
Triplets [56]	20.5
Autoencoder(AE) [57]	31.4
Variational Bayes AE [58]	29.1
GAN [59]	31.5
JULE [60]	27.2
DEC [39]	30.1
DAC [61]	52.2
DeepCluster [33]	37.4
ADC [62]	32.5
TSUC [44]	80.2
IIC‡ [4]	58.99
SCAN‡ [17]	88.7
SCAN + RUC (Confidence)‡ [9]	90.28
SCAN + RUC (Metric)‡ [9]	89.52
SCAN + RUC (Hybrid)‡ [9]	90.08
IIC + RUC (Confidence)	59.64
IIC + RUC (Metric)	60.56
IIC + RUC (Hybrid)	**60.91**
IIC + RUC (CSS, R=0.7)	58.57
IIC + RUC (CSS, R=0.9)	58.91
IIC + RUC (Hybrid-2, R=0.7)	59.60
IIC + RUC (Hybrid-2, R=0.9)	60.42

Table 5.4: State-of-the-art Methods and Accuracy Performance: Performance improvement with RUC when applied to SCAN and IIC are displayed. Accuracy rates are presented in percentages. Baseline results are excerpted from [9]. ‡Results obtained from our experiments with official code. Results in red are lower compared to their baseline results. The rates in the last division are our contribution. The highest rate is highlighted in bold.

combined with the metric approach, the resulting hybrid-2 approach performs better neither for the R value of 0.7 nor 0.9.

In parallel to the results obtained for SCAN+RUC (in section 5.3.1), the sets formed by the sampling strategy with the highest accuracy rates have the highest F1 score. Namely, the hybrid approach starts with a 27.18% accuracy rate which is achieved by training on the sampled set with an F1 score of 0.79 rate and the accuracy increases to 60.91%. Both hybrid-2 approaches have lower initial and final accuracy rates during both training and test phases.

Strategy	C	M	H	S	M	H2
Precision	0.59	0.67	0.67	0.59	0.67	0.68
Recall	0.99	0.95	0.95	0.92	0.95	0.87
F1 score	0.74	0.79	0.79	0.72	0.79	0.76

Table 5.5: IIC+RUC Initial Quality Metrics of Sets: Quality of the sets formed by each sampling strategy is assessed by metrics of precision, recall and F1 score. The left side shows the strategies from the original paper and the right side shows the proposed new set of strategies. Acronyms: Confidence (C), Metric (M) Hybrid (H), Class-wise Small loss with R value equal to 0.9 (S), Hybrid-2 with R value equal to 0.9 (H2)

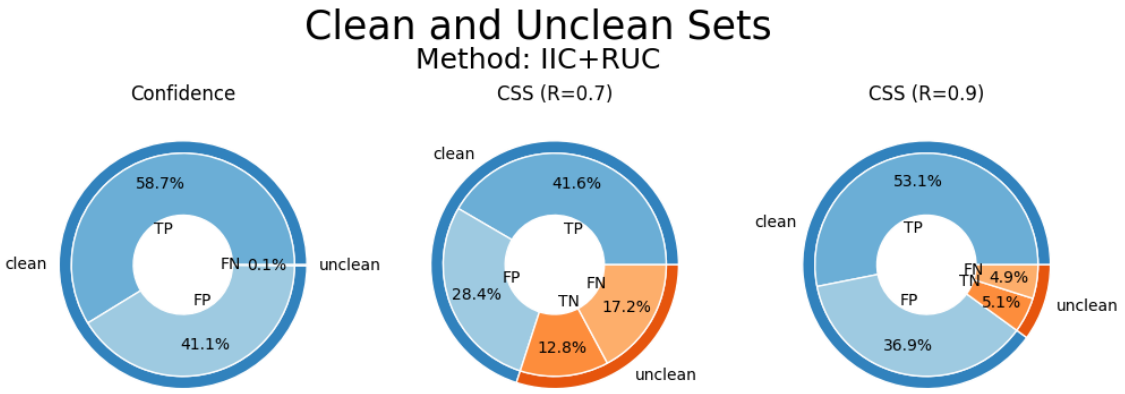

Figure 5.14: IIC+RUC Prediction Ratios of Clean and Unclean Sets: Pie charts are formed by each sampling strategy in [9]. Truly and falsely sampled samples are shown in percentages for each set.

The sets formed after the sampling process are still considered by RUC as noisy so mechanisms of co-training and refurbishment are also implemented in RUC to decrease the confirmation bias and continue improvements on each new epoch. However, it doesn't change the fact that the initial state is crucial since the DNN continue learning based on the clean and unclean sets formed. The best-performing sampling strategy, the hybrid approach, has the highest F1 score of 0.79. This indicates that the quality of the clean set has an effect on the initial accuracy and, consequently the final accuracy.

Table 5.4 is partitioned into three, the first section includes the state-of-the-art methods excerpted from [9], the middle section is dedicated to the reproduced results and the final section summarises our contribution to this research. Comparing the accuracy rates achieved by employing different sampling strategies, the combination

of confidence and metric-based approaches achieves 60.91% accuracy performance, achieving the top rank along the set of original strategies and compared to the new ones. Specifically with the hybrid strategy, applying RUC to IIC increases the accuracy rate by 1.92 pp.

From the list of strategies which are proposed in this paper (the last four rows), the hybrid-2 approach exhibits a noteworthy increase of 1.43 in accuracy when the R value is set to 0.9 during training, and this improvement in accuracy is sustained at test time. On the other hand, when data is sampled according to the sole CSS strategy, a decrease in accuracy is observed compared to the baseline result of IIC. The baseline accuracy rate of 58.99% of IIC decreases to 58.57% and 58.12% when R is equal to 0.7 and 0.9, respectively. The effectiveness of the choice of data selection process can be observed with this experiment.

An observed relationship between the F1 score and the initial state of the sets formed at the beginning of re-training holds for both cases: SCAN+RUC and IIC+RUC. However with the comparison made between CSS and confidence-based reveals that F1 score alone is not the sole determining factor for achieving an accuracy increase when RUC is applied to a baseline model. Referring to Table 5.3, there is only a difference of 0.02 in the F1 scores of confidence-based and CSS, yet RUC is not able to boost the accuracy of IIC when CSS sampling is employed instead of confidence-based.

When the precision and recall metrics are compared between the confidence-based and CSS strategies, an immense dissimilarity between these metrics is not observed. So, to gain more insight into why IIC+RUC with CSS has lower accuracy than IIC, the clean and unclean sets are displayed as a pie chart. Figure 5.14 illustrates the proportion of true positives for each strategy, revealing a significant difference in the number of true positives in clean sets between the two strategies. Before retraining, the true positives in the clean set formed by the confidence approach is 58.7% of the total train set, which is equivalent to 29,350 samples, while the ratio for CSS is 41.6% and it corresponds to 20,800 samples. In addition to the quality metrics, such as precision, recall, and F1 score, the number of true positives in the clean set may be a crucial parameter for RUC to achieve its objectives. Since all the quality metrics mentioned are in ratios, when comparing them for two different sampling strategies, they don't infer the quantity for the number of true positives directly.

The difference in the quantity of true positives can be the reason of the performance degradation of RUC. A lower limit in sample quantity can be required for the

semi-supervised structure of RUC to work properly. When considering the capabilities of DNN to fit to data with many types of noisy in different ratios which has been displayed by the experiments in [10], the data sampling strategies and the sets they form should be further explored to understand their effect and consequences on the mechanisms of RUC (see Section 6.1).

Chapter 6

Conclusion and Future Work

In this study, we focused on unsupervised image classification, with a specific focus on the robust training techniques and their possible effects on the joint approach clustering algorithms. While an add-on module based on robust training techniques, namely RUC, has been tested on clustering methods, these methods are limited to the multi-step refinement approach. When RUC is applied to methods of multi-step refinement approach such as SCAN and TSUC, the combined models achieve higher accuracy rates with better calibration. As claimed by the authors of [9], RUC can be utilised by any type of unsupervised clustering due to its flexible architecture and can boost the baseline model from both of the performance criteria.

Our research question is built upon this possible performance gain when robust training techniques are applied to the joint approach clustering. In this regard, we conducted three experiments which had the following goals: the first experiment was dedicated to illustrate the IIC's issue of overconfident results, and the second experiment proved that RUC is capable of boosting the accuracy performance and increasing the existing calibration of IIC. Lastly, the data selection process of RUC is investigated and an alternative strategy is proposed which leverages the class-wise small loss. However, this new strategy achieved lower final accuracy as it was not capable of forming a more decent clean set compared to the original RUC strategies.

In addition to our experiments, several concepts which remain open to further exploration and development are mentioned below.

6.1 About Sampling Strategy

In the discussion held in Section 5.3, we compared the performance of the newly proposed strategy with the original set of strategies. The sets formed by the class-wise

small loss strategy achieved a lower F1 score in contrast to the original strategies, leading to a lower accuracy compared to the confidence strategy. The choice of sampling strategy used in RUC has a significant impact on accuracy performance. Both our experiments and those conducted by the authors of [9] have shown that the best-performing sampling strategy is highly dependent on the baseline model. The experiments have shown the effects of the choice of sampling strategy on the accuracy achieved. Further exploration into categorizing different types of data selection models and identifying which baseline models can work best with each type is a promising area of research which is beyond the scope of this study.

6.2 About Learning Rate

Through the discussions held in Section 5.2.2, the effect of the learning rate on both the initial and final accuracy rates are obviously shown. Since the effect of this hyperparameter on the accuracy rate is beyond the interest of the study, no experiments with the aim of finding the optimal value for this hyperparameter are investigated. Rather the presence of a relationship between the learning rate and accuracy is displayed. The optimal value for the learning rate can vary depending on the baseline model so even though a value of 0.01 is suitable for SCAN+RUC [9], for IIC+RUC a different rate can yield better performance. Further investigation to explore the correlation might help to find the maximum achievable accuracy rates.

Appendix A

A.1 Package version Comparison Table

	Original code	My Implementation
matplotlib	2.2.3	3.7.1
numpy	1.15.1	1.23.5
scikit-learn	0.20.0	1.2.2
scipy	1.1.0	1.10.1
torch	0.4.1	2.0.1
torchvision	0.2.1	0.15.2
Implementation Language	Python2	Python3

Table A.1: Package versions comparison table. The left column shows the original implementation setup while the right side displays the versions we have incorporated during the reproduction process.

A.2 Loss Accuracy Curves of Sampling Strategies

In Section 5.2.2, only the loss and accuracy curve belonging to the IIC+RUC with the hybrid approach is displayed since the other approaches follow a similar pattern in the learning process. Below the other two sampling strategies (Metric and CSS) which constitute the hybrid-2 approach are included.

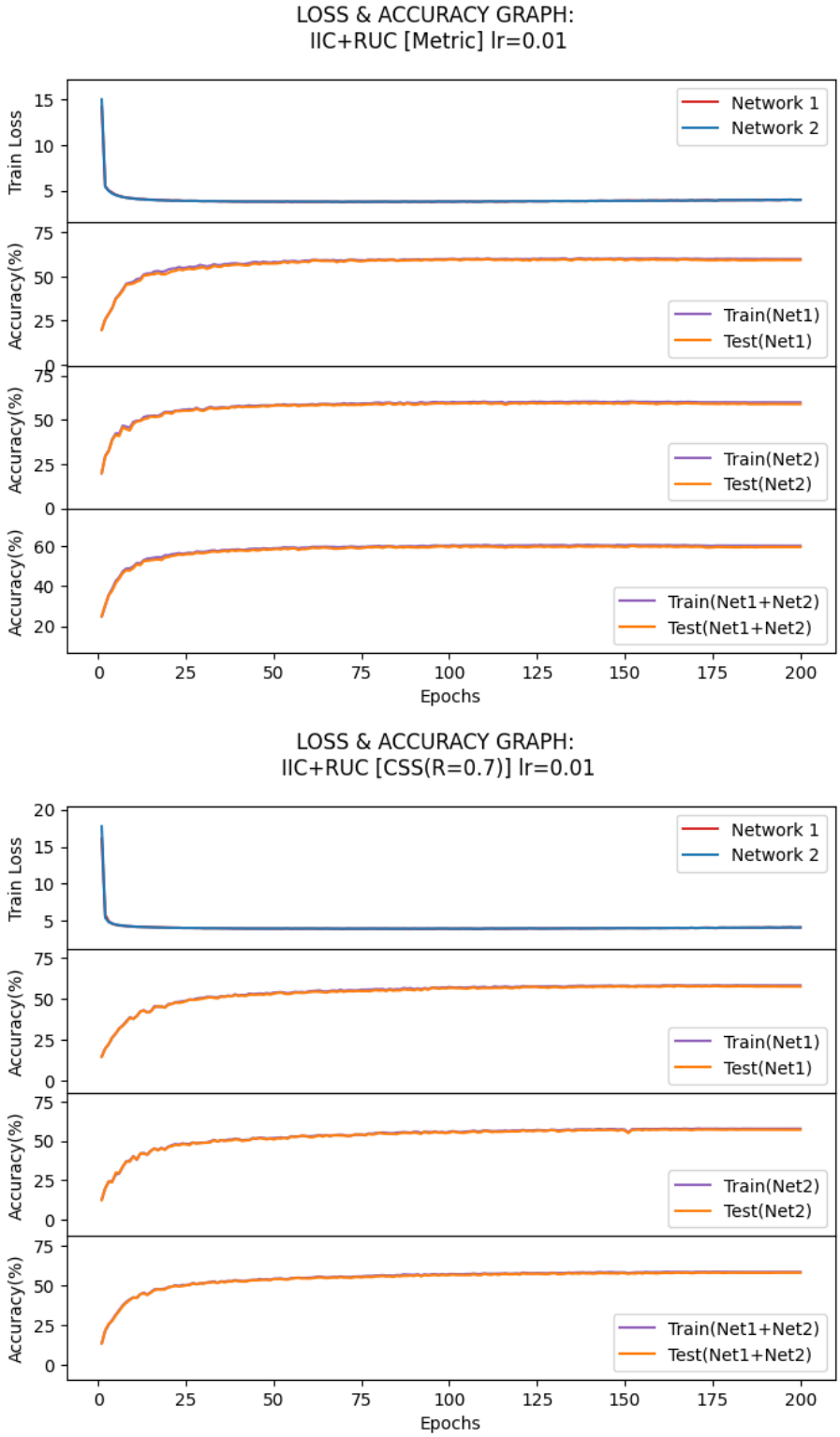

Figure A.1: IIC+RUC Loss and Accuracy Graphs: The sampling strategy are metric (at top) and CSS with R equals 0.7 (at bottom) and the lr. (learning rate) is 0.01. Both networks' train loss curves are plotted in the first subplot. Accuracy curves belonging to each individual network and their combination are plotted for the train and test phases.

Bibliography

[1] L. Chen, S. Lin, X. Lu, D. Cao, H. Wu, C. Guo, C. Liu, and F.-Y. Wang, "Deep neural network based vehicle and pedestrian detection for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3234–3246, 2021.

[2] V. Sharma, M. Gupta, A. Kumar, and D. Mishra, "Video processing using deep learning techniques: A systematic literature review," *IEEE Access*, vol. 9, pp. 139489–139507, 2021.

[3] F. Wang, T. Kong, R. Zhang, H. Liu, and H. Li, "Self-supervised learning by estimating twin class distributions," *arXiv preprint arXiv:2110.07402*, 2021.

[4] X. Ji, J. F. Henriques, and A. Vedaldi, "Invariant information clustering for unsupervised image classification and segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9865–9874, 2019.

[5] A. Gupta, A. Anpalagan, L. Guan, and A. S. Khwaja, "Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues," *Array*, vol. 10, p. 100057, 2021.

[6] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[7] E. Moebel and C. Kervrann, "Towards unsupervised classification of macromolecular complexes in cryo electron tomography: challenges and opportunities," *Computer Methods and Programs in Biomedicine*, p. 107017, 2022.

[8] J. D. Weihermann, M. P. Ferreira, L. G. de Castro, F. J. F. Ferreira, and A. M. Silva, "Retrieving geological units with unsupervised clustering of gamma-ray spectrometry data," *Journal of Applied Geophysics*, vol. 184, p. 104225, 2021.

[9] S. Park, S. Han, S. Kim, D. Kim, S. Park, S. Hong, and M. Cha, "Improving unsupervised image clustering with robust learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12278–12287, 2021.

[10] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

[11] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[12] B. Han, Q. Yao, T. Liu, G. Niu, I. W. Tsang, J. T. Kwok, and M. Sugiyama, "A survey of label-noise representation learning: Past, present and future," *arXiv preprint arXiv:2011.04406*, 2020.

[13] H. Wang, R. Xiao, Y. Dong, L. Feng, and J. Zhao, "Promix: Combating label noise via maximizing clean sample utility," *arXiv preprint arXiv:2207.10276*, 2022.

[14] I. H. Sarker, "Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions," *SN Computer Science*, vol. 2, no. 6, p. 420, 2021.

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[16] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[17] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, and L. Van Gool, "Scan: Learning to classify images without labels," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X*, pp. 268–285, Springer, 2020.

[18] L. Ericsson, H. Gouk, C. C. Loy, and T. M. Hospedales, "Self-supervised representation learning: Introduction, advances, and challenges," *IEEE Signal Processing Magazine*, vol. 39, no. 3, pp. 42–62, 2022.

[19] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 11, pp. 4037–4058, 2020.

[20] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine learning*, vol. 109, no. 2, pp. 373–440, 2020.

[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, pmlr, 2015.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[23] M. Placek, "Global autonomous vehicle market size 2030," Jan 2023.

[24] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixao, F. Mutz, *et al.*, "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113816, 2021.

[25] H. Gajjar, S. Sanyal, and M. Shah, "A comprehensive study on lane detecting autonomous car using computer vision," *Expert Systems with Applications*, vol. 233, p. 120929, 2023.

[26] S. Du, H. Guo, and A. Simpson, "Self-driving car steering angle prediction based on image recognition," *arXiv preprint arXiv:1912.05440*, 2019.

[27] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[28] V. Bolon-Canedo and B. Remeseiro, "Feature selection in image analysis: a survey," *Artificial Intelligence Review*, vol. 53, no. 4, pp. 2905–2931, 2020.

[29] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 857–876, 2021.

[30] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2536–2544, 2016.

[31] G. Larsson, M. Maire, and G. Shakhnarovich, "Colorization as a proxy task for visual understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6874–6883, 2017.

[32] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*, pp. 69–84, Springer, 2016.

[33] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 132–149, 2018.

[34] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," *Technologies*, vol. 9, no. 1, p. 2, 2020.

[35] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21271–21284, 2020.

[36] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," *Advances in neural information processing systems*, vol. 33, pp. 9912–9924, 2020.

[37] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," *arXiv preprint arXiv:1808.06670*, 2018.

[38] 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.

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

[40] T. S. Madhulatha, "An overview on clustering methods," *arXiv preprint arXiv:1205.1117*, 2012.

[41] M. G. Omran, A. P. Engelbrecht, and A. Salman, "An overview of clustering methods," *Intelligent Data Analysis*, vol. 11, no. 6, pp. 583–605, 2007.

[42] S. Ranjan, D. R. Nayak, K. S. Kumar, R. Dash, and B. Majhi, "Hyperspectral image classification: A k-means clustering based approach," in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 1–7, IEEE, 2017.

[43] C. Ding and X. He, "K-means clustering via principal component analysis," in *Proceedings of the twenty-first international conference on Machine learning*, p. 29, 2004.

[44] S. Han, S. Park, S. Park, S. Kim, and M. Cha, "Mitigating embedding and class assignment mismatch in unsupervised image classification," in *European Conference on Computer Vision*, pp. 768–784, Springer, 2020.

[45] D. Gupta, R. Ramjee, N. Kwatra, and M. Sivathanu, "Unsupervised clustering using pseudo-semi-supervised learning," in *International Conference on Learning Representations*, 2019.

[46] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[47] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223, JMLR Workshop and Conference Proceedings, 2011.

[48] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[49] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[50] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[51] J. Wang, J. Wang, J. Song, X.-S. Xu, H. T. Shen, and S. Li, "Optimized cartesian k-means," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 1, pp. 180–192, 2014.

[52] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama, "Learning discrete representations via information maximizing self-augmented training," in *International conference on machine learning*, pp. 1558–1567, PMLR, 2017.

[53] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," *Advances in neural information processing systems*, vol. 32, 2019.

[54] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020.

[55] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," *Advances in neural information processing systems*, vol. 17, 2004.

[56] M. Schultz and T. Joachims, "Learning a distance metric from relative comparisons," *Advances in neural information processing systems*, vol. 16, 2003.

[57] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, 2006.

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

[59] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[60] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5147–5156, 2016.

[61] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, "Deep adaptive image clustering," in *Proceedings of the IEEE international conference on computer vision*, pp. 5879–5887, 2017.

[62] P. Haeusser, J. Plapp, V. Golkov, E. Aljalbout, and D. Cremers, "Associative deep clustering: Training a classification network with no labels," in *Pattern Recognition: 40th German Conference, GCPR 2018, Stuttgart, Germany, October 9-12, 2018, Proceedings 40*, pp. 18–32, Springer, 2019.