Australian National University

PhD Thesis

Learning from Noisy Labels using Robust Structures

Australian National University
CSIRO

Yan Han

Supervisor: Prof. Daniel MacDonald, Dr. Lars Petersson, Prof. Mehrtash Harandi, Dr. Jing Zhang
ANU College of Engineering, Computing and Cybernetics
2025.3

Certificate of Original Authorship

I confirm that the work presented in this thesis has not been previously submitted for a degree, nor has it been used to fulfill the requirements of a degree, except where it forms part of a collaborative doctoral degree and/or is fully acknowledged within the text.

I also affirm that this thesis is my own work. Any assistance I have received in conducting my research or preparing the thesis has been duly acknowledged. Furthermore, I certify that all sources of information and literature referenced have been appropriately cited within the thesis.

Signature of Student: Yan Han

Date: 01/03/2025

Acknowledgement

Looking back at the completion of my thesis and research journey for my PhD, I find it difficult to fully grasp that this significant chapter has come to a close. The path to this accomplishment was far from smooth—it was one filled with moments of self-doubt, relentless hard work, and at times, what seemed like insurmountable challenges. However, it was also a journey filled with immense joy, personal growth, and profound learning. Each step was enriched by the support and encouragement of those around me, and I am deeply grateful to everyone who played a part in making this journey possible.

The research process itself was anything but mundane. What may seem from the outside as a monotonous life of endless coding, experiments, and literature review was, in fact, an intellectually stimulating adventure. It was filled with mysteries to solve, wonders to uncover, delightful new ideas to explore, and exciting new methods to test. Every challenge I faced was also an opportunity to grow as a researcher, and every moment of frustration was a stepping stone toward the eventual breakthroughs that made this work possible.

First and foremost, I must express my deepest gratitude to my supervisors: Prof. Daniel MacDonald, Dr. Lars Petersson, Prof. Mehrtash Harandi, and Dr. Jing Zhang. Your expertise, guidance, and constant support have been invaluable to me throughout this thesis journey. Your insightful feedback helped shape my work into what it is today, and your encouragement kept me motivated even when the path seemed unclear. You not only provided me with academic mentorship but also taught me the resilience and patience required in research.

To my family and friends, I cannot begin to express how grateful I am for your unwavering belief in me. You were my greatest source of strength and motivation, always reminding me that no challenge was too great to overcome. A special thank you to my parents, Mr. Jiagao Han and Ms. Ting guo, for their endless sacrifices and unconditional love—your belief in me has been a constant source of inspiration. And to my partner, Mr. Thanh Tony Huynh, words cannot capture how much your support has meant to me. Your patience, understanding, and constant presence throughout this process have been my lifeline, especially during the most difficult periods of this journey. You stood by me through every late night and every moment of doubt, and for that, I am eternally grateful.

I would also like to extend my thanks to my colleagues and classmates. Working alongside all of you in this collaborative environment has been one of the most

rewarding aspects of this journey. The academic discussions we shared, the exchange of ideas, and the collective problem-solving we engaged in made the process more enjoyable and enlightening. I have learned so much from each of you, and our shared experiences have made the challenges of this journey feel lighter. Your camaraderie has brought a sense of community that made my academic life not just fulfilling but also much less lonely.

This thesis also owes its success to the countless people who may not have been directly involved but whose impact was felt nonetheless. To the administrative staff, the technical teams, and everyone who worked behind the scenes to ensure that my research ran smoothly—thank you. Often, it is these unsung heroes who make the largest difference in a PhD candidate's journey.

In addition to these direct contributors, I also owe thanks to the broader academic community, whose work I have drawn inspiration from. Every paper I have read, every conference I attended, and every casual conversation about ideas has helped me along the way. Academia thrives on shared knowledge, and I am grateful to be part of such a supportive, intellectually enriching environment.

Finally, I would like to recognize and thank every person who supported me—whether directly or indirectly—in completing this thesis. From the smallest gestures of encouragement to the critical advice that shaped my research, each one played a part in this achievement. This accomplishment would not have been possible without your continued support, and for that, I am deeply appreciative.

Here is the list of people I would like to express my heartfelt thanks to: Prof. Daniel MacDonald, Dr. Lars Petersson, Prof. Mehrtash Harandi, Dr. Jing Zhang, Mr. Jiagao Han, Ms. Ting Guo, Mr. Thanh Tony Huynh, Dr. Ruitao Jin, Dr. Sitong He, Dr. Yizhou Yang, Dr. Soumava Kumar Roy, Ms. Xiaofeng Li, Mr. Jieming Zhou, Dr. Ali Cheraghian, Dr. Zeeshan Hayder, Dr. Xuesong Li, Miss Yanyu Zeng, Miss Yi Yu, Mr. Xingyi Zhang, Mr. Weiyao Zhang, Dr. Shi Qiu, Dr. Jie Hong, Dr. Peipei Song, Dr. Yifu Wang, Dr. Huihui Fang, Miss. Yao Wang, Miss. Shihui Peng.

Abstract

Modern machine learning is evolving toward increasingly complex models, such as deep neural networks, which rely on large volumes of well-annotated data. Crowdsourcing offers an efficient and cost-effective way to gather labels, but it often fails to provide a sufficient number of high-quality labels. This raises a critical question: How can we design robust structures to enhance learning from noisy labels?

Without such structures, labels obtained through crowdsourcing tend to be noisy, which negatively impacts model performance. Convolutional neural networks (CNNs) are particularly vulnerable, as they can overfit when trained on mislabeled data. To address this issue, we adopt the Co-Training framework, which consists of two parallel models that learn from each other and correct each other's errors.

The challenge, then, is to enhance this learning process. This thesis explores methods to improve Co-Training's ability to handle noisy labels, strengthening its effectiveness and robustness in real-world scenarios.

Therefore, in this thesis, we aim to develop a series of robust machine learning approaches, so that they can perfectly handle the difficulty from noisy supervision. Our works are summarized as follows:

Chapter 3: Recognizing that feature similarity in collaborative training setups can lead to suboptimal learning outcomes, we developed the Discriminate Co-Training model. This approach adapts the collaborative training framework by implementing discrepancy constraints between the feature extractors of two networks. These constraints allow each network to learn distinct, discriminative features, effectively pacifying the mimicry tendency often seen in homogeneous networks. Without these constraints, networks in co-training setups frequently replicate each other's features, which diminishes the model's resilience to noise. By addressing this mimicry problem, our model leverages the diversity of features to improve overall robustness and accuracy, particularly in noisy environments.

Chapter 4: Building on the insights from the previous chapter, we utilize Sinkhorn loss to enhance diversity further among models in a co-training setup. Traditional metrics like maximum mean discrepancy and Wasserstein distance are powerful but limited in their scope. Sinkhorn synthesizes these metrics, encapsulating their advantages while offering a more unified and efficient approach to measuring and encouraging diversity between models. This chapter explains the mathematical formulation of Sinkhorn and illustrates its benefits in promoting diver-

sity between deep learning networks. This approach fosters resilience against noise by preventing convergence on similar features, thereby boosting model accuracy and reliability in noisy data scenarios.

Chapter 5: To explore the potential of transformers in handling noisy datasets, we propose a novel Contrastive Co-Transformer model. Transformers have shown significant promise due to their capacity to learn complex dependencies, making them highly adaptable in situations with noisy labels. Our model leverages a combination of contrastive loss and classification loss to utilize all data samples, whether clean or noisy, without requiring pre-filtering or assumptions about data quality. This chapter discusses how transformers trained under this framework can harness the entirety of the dataset, achieving strong performance even when a substantial portion of the labels are noisy. We detail the architecture and training dynamics, demonstrating how transformers' inherent robustness can be amplified with this approach.

Chapter 6: Finally, to advance the robustness of learned representations in noisy datasets, we apply Sinkhorn in the context of deep embedding learning. Traditional embedding methods often falter in noisy environments, leading to unreliable representations that weaken model performance. By integrating Sinkhorn, our framework learns optimal embeddings that remain robust even with label noise. This chapter explains the practical applications and advantages of embedding learning using Sinkhorn, emphasizing its simplicity and effectiveness in maintaining model accuracy and feature diversity.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we outline our motivations, which naturally lead to several potential directions for improvement. We then introduce previous and related work on learning from noisy labels, followed by sections that discuss the limitations of these methods. We then detail our contributions to enhancing the structure, making it more robust and accurate for learning from noisy labels. The final two sections provide an overview of the thesis structure and my publications during my PhD studies.

1.1 Motivations: Why Learning From Noisy Labels?

Deep neural networks can be utilized in various real-world applications. However, before being employed for a particular task, the model must undergo a learning process. In the case of image classification, the model needs to learn from accurately labeled data. For research purposes, there are several clean datasets such as ImageNet, CiFAR10, CiFAR100, and MNIST available.

In real-world problems, clean datasets are not always accessible. Creating clean datasets for each specific application requires a significant amount of human resources, and even experts may struggle to accurately label images, particularly when identifying different birds. Consequently, low-quality labels can adversely affect the performance of learning models [47, 104, 134], reducing the accuracy of deep neural networks by 20% or more. Even with the rapid development of computer vision, the famous and widely-used clean datasets are still limited, the famous clean datasets are : MNIST: This dataset consists of 28x28 pixel grayscale images of handwritten digits (0-9), widely used for digit classification tasks. CIFAR-10 and CIFAR-100: CIFAR-10 contains 60,000 32x32 color images in 10 classes, while CIFAR-100 contains the same number of images but in 100 classes. ImageNet: ImageNet is one of the largest labeled image datasets, containing millions of images categorized into thousands of classes. The ILSVRC (ImageNet Large Scale Visual Recognition Challenge) subset is commonly used, which consists of over 1.2 million images across 1,000 classes. PASCAL VOC (Visual Object Classes): PASCAL

VOC datasets contain images with annotations for object detection, segmentation, and classification tasks. There are several versions, with PASCAL VOC 2007 and 2012 being the most commonly used. COCO (Common Objects in Context): COCO is a large-scale object detection, segmentation, and captioning dataset. It contains over 200,000 labeled images across 80 categories. Stanford Dogs and Stanford Cars: These datasets contain images of dogs and cars, respectively, with annotated bounding boxes. They are often used for fine-grained classification tasks. Caltech-256: This dataset consists of 30,607 images across 256 object categories, suitable for object recognition tasks. Oxford-IIIT Pet Dataset: This dataset contains images of cats and dogs, with annotations for fine-grained classification tasks. UC Merced Land Use Dataset: This dataset contains aerial images of 21 land use classes, suitable for land use classification tasks. Fashion MNIST: Similar to MNIST, this dataset contains grayscale images of fashion items like clothing and accessories, aimed at benchmarking computer vision algorithms.

Research conducted at institutions like MIT has scrutinized the prevalence of noisy labels within renowned datasets such as ImageNet and CiFAR. Despite the meticulous curation of these datasets, inherent challenges persist, stemming from factors such as human annotation errors, subjective labeling criteria, and inherent data ambiguities. These sources of label noise can detrimentally impact machine learning models by inducing overfitting, reducing generalization capabilities, and introducing biases into predictions. Consequently, there exists a pressing need for the development and refinement of methodologies aimed at mitigating the adverse effects of label noise. Proposed strategies encompass a spectrum of techniques including the formulation of robust loss functions, the employment of data augmentation schemes, as well as the exploration of semi-supervised and active learning paradigms. Continued investigation into the characterization and mitigation of label noise remains paramount for enhancing the reliability and robustness of machine learning models, particularly within real-world contexts where data quality may be variable.

Due to limited knowledge, understanding of the underlying task and random mistakes, crowd-source workers performing manual annotation of such large scale datasets cannot be expected to annotate with 100% accuracy.

Example of a dataset with noisy labels are shown in Figure 1.1. The example noisy dataset in Figure 1.1 is contributed by searching the key word ”panda” and ”red apple” on google respectively. As can be seen from Fig 1.1, there were both true panda images as well as red panda and other panda related images. For the other key word ”red apple”, the searching results included some random images as well. That is to say, if one wants to create image dataset without crowd-source (only by online searching), the dataset would contain large amount of noise. To deal with such noisy dataset, people can either spend time correct the corrupted labels or find methods to train from noisy dataset.

To address the issue of label noise, several techniques have been proposed, including data augmentation, active learning, and robust optimization. Data augmentation involves introducing variations in the input data to increase the size of the

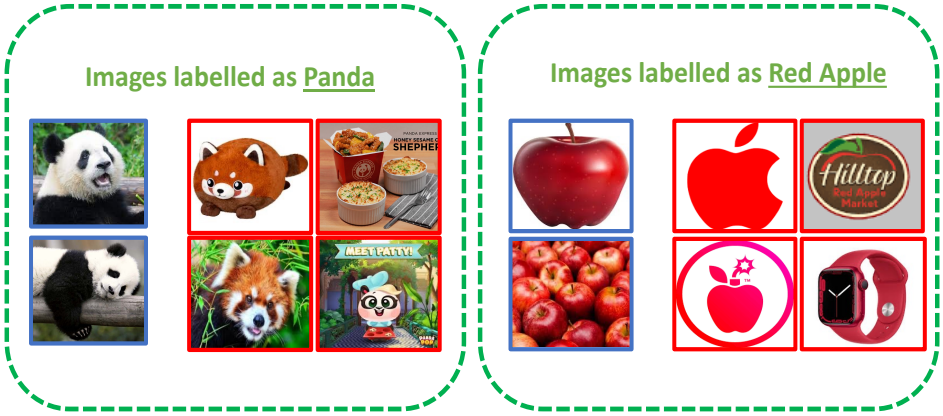


Figure 1.1: noisy-dataset. The above figure lists two different classes. Left is panda class, all images are labeled as panda. Right is red apple class. Images which are corrected labeled are marked blue; corrupted labeled are marked red.

dataset, which can help reduce overfitting. Active learning involves selecting the most informative samples to label, thereby reducing the number of noisy samples in the dataset. Robust optimization involves modifying the loss function to be more tolerant of label noise, which can help the model learn from the noisy data.

In addition to these techniques, several methods have been proposed for identifying and correcting label noise. One common approach is to use multiple annotators and measure the degree of agreement between them. Samples with low agreement can be flagged as potentially noisy, and the labels can be corrected or discarded. Another approach is to use semi-supervised learning, where the model learns from both labeled and unlabeled data. The unlabeled data can act as a source of additional information to help correct the noisy labels.

In summary, label noise is a common issue when working with large-scale datasets, and addressing it is crucial to ensure accurate predictions by deployed models. While it is possible to train models on noisy data, it is important to take steps to mitigate the negative impact of label noise on the model's performance. Techniques such as data augmentation, active learning, and robust optimization can help reduce overfitting, while methods such as multiple annotators and semi-supervised learning can help identify and correct noisy labels.

The corruption of data due to label noise can cause the model to make incorrect predictions, emphasizing the need to handle label noise in a systematic manner. Although there is a vast amount of readily available information online, accepting noisy data during training can result in neural networks overfitting to the noise [153, 90, 85]. [162] has shown that even random labels can be easily fit by deep

neural networks, particularly deeper and wider ones that have a greater capacity to memorize the training data. Additionally, research has found that a noise rate of 20% can have a significantly negative impact on test accuracy, with some cases showing a halving of the accuracy, as argued in [19].

The research community has realized the importance of learning from noisy labels. The research work can be dated back to 1990s. So far, there are lots of methods on the topic.

Learning from noisy labeled image datasets is motivated by several factors. Here are some potential motivations:

Cost-effectiveness: Acquiring large labeled datasets can be expensive and time-consuming. In some cases, obtaining accurate labels for images may require the expertise of human annotators, which can be costly. Noisy labeled datasets, which may contain errors in the labels, can be less expensive to acquire and can still provide useful information for training machine learning models.

Real-world scenarios: In many real-world scenarios, it may be difficult or impossible to obtain perfectly labeled datasets. For example, in medical imaging, some images may be labeled by non-experts or may have ambiguous labels due to the complexity of the underlying pathology. In such cases, learning from noisy labeled datasets can be a practical solution.

Robustness: Machine learning models that are trained on noisy labeled datasets may be more robust to label noise and other forms of perturbations. By exposing the model to different types of noise and errors during training, it can learn to generalize better and perform well on new, unseen data.

Generalization: Noisy labeled datasets can also help prevent overfitting and improve generalization performance. Since the model is exposed to more diverse examples and variations in the training data, it can learn to capture the underlying patterns and features of the images, rather than memorizing the labels of the training data.

Overall, learning from noisy labeled image datasets can be motivated by a combination of cost-effectiveness, real-world scenarios, robustness, and generalization.

1.2 Previous Work

(1) In numerous studies, architectural changes have been made to model the noise transition matrix of a noisy dataset [16], [75], [82]. These changes include adding a noise adaptation layer at the top of the softmax layer and designing a new dedicated architecture. The resulting architectures yield improved generalization through the modification of the DNN output based on the estimated label transition probability.

(2) Robust Regularization Regularization methods have been widely studied to improve the generalizability of a learned model in the machine learning community [23]–[26]. By avoiding overfitting in model training, the robustness to label noise improves with widely-used regularization techniques such as data augmentation [23], weight decay [24], dropout [25], and batch normalization [26]. These canonical

regularization methods operate well on moderately noisy data, but they alone do not sufficiently improve the test accuracy; poor generalization could be obtained when the noise is heavy [86]. Thus, more advanced regularization techniques have been recently proposed, which further improved robustness to label noise when used along with the canonical methods. The main advantage of this family is its flexibility in collaborating with other directions because it only requires simple modifications.

(3) Robust loss adjustment Loss adjustment is effective for reducing the negative impact of noisy labels by adjusting the loss of all training examples before updating the DNN [19], [62], [69], [107]–[111]. The methods associated with it can be categorized into three groups depending on their adjustment philosophy: 1) loss correction that estimates the noise transition matrix to correct the forward or backward loss, 2) loss reweighting that imposes different importance to each example for a weighted training scheme, 3) label refurbishment that adjusts the loss using the refurbished label obtained from a convex combination of noisy and predicted labels, and 4) meta learning that automatically infers the optimal rule for loss adjustment. Unlike the robust loss function newly designed for robustness, this family of methods aims to make the traditional optimization process robust to label noise. Hence, in the middle of training, the update rule is adjusted such that the negative impact of label noise is minimized. In general, loss adjustment allows for a full exploration of the training data by adjusting the loss of every example. However, the error incurred by false correction is accumulated, especially when the number of classes or the number of mislabeled examples is large [112].

(4) To avoid any false corrections, many recent studies [19], [70], [99], [112], [131]–[137] have adopted sample selection that involves selecting true-labeled examples from a noisy training dataset.

1.3 Limitations of Current Methods

1. Reducing Model Complexity:

To tackle overfitting, simpler architectures or regularization techniques (e.g., dropout or data augmentation) can be incorporated to reduce complexity while maintaining model robustness. Techniques like co-teaching or self-paced learning train simpler auxiliary models alongside the main model to handle noisy labels more effectively.

2. Lowering Computational Costs:

For resource-intensive methods, approximate or heuristic-based algorithms can help reduce computation. Techniques such as sample selection (choosing less noisy subsets) and gradient clipping (limiting parameter updates) have proven effective in noisy environments without significant computational overhead.

3. Reducing Dependence on Clean Data:

Semi-supervised learning and self-supervised techniques can leverage the noisy dataset itself, without requiring clean data. Consistency regularization and pseudo-

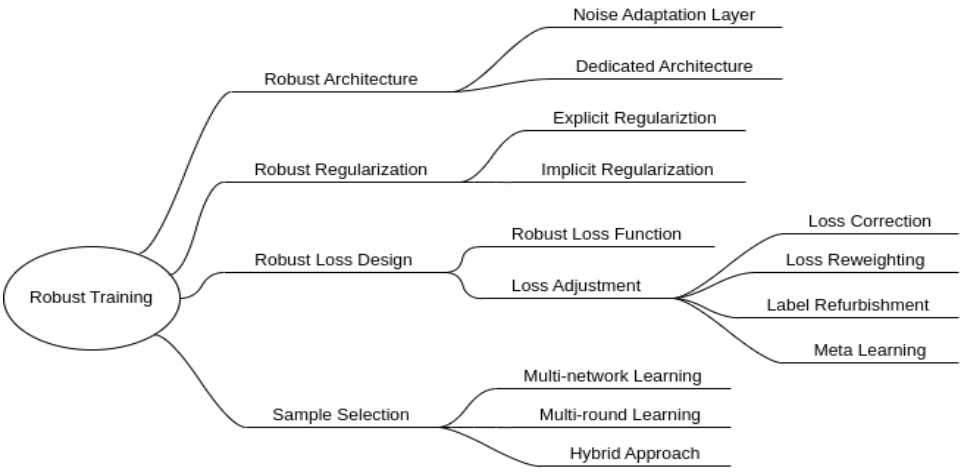


Figure 1.2: A high level research overview of robust deep learning for noisy labels. The research directions that are actively contributed by the machine learning community are categorized into five groups in blue italic.

labeling encourage the model to rely on its own predictions to refine learning over time, making these approaches feasible for situations where clean data is hard to obtain.

4. Increasing Cross-Domain Applicability:

Cross-domain noise-robust algorithms such as meta-learning or domain adaptation improve generalization across datasets with varying types of noise. Methods that do not rely on domain-specific assumptions can help expand applicability.

5. Adapting to Varied Noise Distributions:

To overcome assumptions about noise distributions, distribution-agnostic models (e.g., robust loss functions or adversarial training) are better suited for real-world variability in noise patterns. These methods aim to generalize across different types of noise distributions, rather than relying on specific assumptions.

6. Handling Sensitivity to Noise Rate:

Techniques such as noise-robust loss functions (e.g., mean absolute error or generalized cross-entropy) are less sensitive to noise levels and can maintain performance even when the noise rate fluctuates. Adaptive frameworks that can estimate and adjust to noise levels during training can also address this issue.

7. Simplifying Hyperparameter Tuning:

Automated tuning methods, like Bayesian optimization or grid search, can reduce the difficulty of finding optimal hyperparameters. Additionally, self-tuning algorithms that dynamically adjust hyperparameters based on training feedback are emerging as promising alternatives.

8. Improving Noise Detection:

Methods such as confidence-based filtering and ensemble-based approaches can help identify likely noisy labels. Explainable AI (XAI) techniques, like feature

importance measures, are also increasingly used to flag mislabeled samples for correction.

9. Enhancing Interpretability:

Incorporating attention mechanisms and layer-wise relevance propagation (LRP) can increase interpretability, allowing researchers and practitioners to understand how the model responds to noise and how it corrects label errors.

10. Minimizing Bias Introduction:

To mitigate the risk of bias, fairness-aware learning and adversarial training can be applied. Bias detection tools and rigorous validation against diverse datasets help prevent unintended biases introduced by noise-handling mechanisms. Each of these solutions contributes toward developing more robust, efficient, and versatile methods for handling noisy labels, particularly as these methods continue to evolve and adapt to real-world complexities across different applications and domains.

Based on the aforementioned limitations, the following research question arises:How can we design robust deep learning structures that maintain high performance and generalization ability when trained on datasets containing significant label noise? To elaborate further, the general research question can be specified through the following sub-questions: 1.How can collaborative or co-training frameworks be improved to reduce overfitting and enhance robustness when learning from noisy labels? 2. How can divergence-based metrics such as Sinkhorn be leveraged to enforce diversity between models and improve resistance to label corruption? 3. Can transformer-based architectures and contrastive objectives be combined to further enhance robustness and generalization under high noise ratios and in real-world noisy datasets?

1.4 Outline and Contributions

1.4.1 Outline

Chatper 1 To address the significant challenge of learning from noisy labels, we begin by establishing the real-world impact of label noise and the urgent need for robust methods capable of managing it. Label noise is a prevalent issue in datasets from various domains, such as medical imaging and autonomous driving, where errors in labeling are common due to the subjective or complex nature of the data. This chapter reviews existing approaches for handling noisy labels and emphasizes their limitations, such as inadequate generalization, computational inefficiencies, or susceptibility to specific types of noise. This discussion leads to the motivation for the new solutions proposed in this research, which aim to improve model robustness and accuracy. We also define the primary terminology and notations that will be used throughout the thesis, providing a clear foundation for the advanced discussions that follow.

Chapter 2 Recognizing that feature similarity in collaborative training setups can lead to suboptimal learning outcomes, we developed the Discriminate Co-Training model. This approach adapts the collaborative training framework by implementing

discrepancy constraints between the feature extractors of two networks. These constraints allow each network to learn distinct, discriminative features, effectively pacifying the mimicry tendency often seen in homogeneous networks. Without these constraints, networks in co-training setups frequently replicate each other's features, which diminishes the model's resilience to noise. By addressing this mimicry problem, our model leverages the diversity of features to improve overall robustness and accuracy, particularly in noisy environments.

Chapter 3 Building on the insights from the previous chapter, we utilize $\mathcal{S}_\epsilon$ to enhance diversity further among models in a co-training setup. Traditional metrics like maximum mean discrepancy and Wasserstein distance are powerful but limited in their scope. $\mathcal{S}_\epsilon$ synthesizes these metrics, encapsulating their advantages while offering a more unified and efficient approach to measuring and encouraging diversity between models. This chapter explains the mathematical formulation of $\mathcal{S}_\epsilon$ and illustrates its benefits in promoting diversity between deep learning networks. This approach fosters resilience against noise by preventing convergence on similar features, thereby boosting model accuracy and reliability in noisy data scenarios.

Chapter 4 To tackle the challenges of noisy data directly, we developed a Contrastive Co-Training framework, which combines the simplicity of contrastive learning with the strengths of co-training. This model, designed for both efficiency and high performance, outperforms many state-of-the-art techniques by providing a straightforward yet powerful means of differentiating noisy from clean data. In contrastive learning, samples are compared to learn effective representations, and in our model, this comparison helps the networks distinguish more effectively between signal and noise. The simplicity of this approach allows for faster training without sacrificing accuracy, making it a practical and robust solution for handling noisy labels.

Chapter 5 To explore the potential of transformers in handling noisy datasets, we propose a novel Contrastive Co-Transformer model. Transformers have shown significant promise due to their capacity to learn complex dependencies, making them highly adaptable in situations with noisy labels. Our model leverages a combination of contrastive loss and classification loss to utilize all data samples, whether clean or noisy, without requiring pre-filtering or assumptions about data quality. This chapter discusses how transformers trained under this framework can harness the entirety of the dataset, achieving strong performance even when a substantial portion of the labels are noisy. We detail the architecture and training dynamics, demonstrating how transformers' inherent robustness can be amplified with this approach.

Chapter 6 Finally, to advance the robustness of learned representations in noisy datasets, we apply $\mathcal{S}_\epsilon$ in the context of deep embedding learning. Traditional embedding methods often falter in noisy environments, leading to unreliable representations that weaken model performance. By integrating $\mathcal{S}_\epsilon$, our framework learns optimal embeddings that remain robust even with label noise. This chapter explains the practical applications and advantages of embedding learning using $\mathcal{S}_\epsilon$, emphasizing its simplicity and effectiveness in maintaining model accuracy and

feature diversity.

We present empirical results to validate the strength of this approach, highlighting its ability to deliver reliable, noise-resilient embeddings that improve downstream tasks.

1.4.2 Publications During PhD Study

1. Han, Y., ROY, S., Petersson, L. and Harandi, M., 2020. Learning from noisy labels via discrepant collaborative training. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 3169-3178).

2. Han, Y., Roy, S.K., Petersson, L. and Harandi, M., 2021. Discrepant collaborative training by Sinkhorn divergences. Image and Vision Computing, 112, p.104213.

3. Han, Y., Roy, S.K., Harandi, M. and Petersson, L., 2022. Learning Deep Optimal Embeddings with Sinkhorn Divergences. arXiv preprint arXiv:2209.06469.

4. Simon, C., Koniusz, P., Petersson, L., Han, Y. and Harandi, M., 2022. Towards a robust differentiable architecture search under label noise. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 3256-3266).

5. Han, Y., Roy, S.K., Harandi, M. and Petersson, L., 2025. Learning from Noisy Labels with Contrastive Co-Transformer. (Submitted to arXiv and currently under review/on hold.)

Chapter 2

Background

2.1 Computer Vision and Image Classification

Computer vision and machine learning strive to endow machines with the capability to interpret the world similarly to humans. This endeavor underpins various applications such as autonomous vehicles, humanoid robots, and unmanned aerial vehicles. Computer vision, as a field of study, is dedicated to developing techniques that enable computers to perceive and understand the content of digital images, encompassing both photographs and videos.

Within this domain, numerous methods empower machines to sense and comprehend their environment. Object detection[170, 168] is one such method, wherein machines analyze pixels in an image using machine learning or deep learning algorithms to identify distinct elements. A prevalent example of object detection is facial recognition[166], commonly used to secure access to smartphones.

Another pivotal technique is 3D scene reconstruction[52], which employs computer vision algorithms to construct realistic 3D objects from 2D imagery captured from multiple angles. This technology finds significant applications in areas such as architecture and interior design.

Moreover, computer vision facilitates advanced image and video processing[125] techniques that extend beyond the capabilities of traditional image processing algorithms. Neural networks, for instance, can execute image transformations like increasing the number of trees in an image without apparent alterations.

Scene segmentation[35] is another noteworthy technique in computer vision that isolates and examines pixels, resulting in an image with a stained-glass-like appearance. This technology is instrumental in autonomous navigation and radiology for detecting malignant tissue changes.

Machine learning models integrated into computer vision can automatically detect objects[168, 170] in photos, extract content[34], and generate tags[95]. Such technology is utilized in optimizing page rank ads and lead generation campaigns.

As a transformative technology, computer vision simplifies life and catalyzes new areas of development across various industries. The rapid advancement of this

technology has expanded its applications to include retail shelf analysis, RTG analysis, automatic video tagging, real estate valuation, security systems, ID verification, and industrial maintenance.

Image classification, which involves automatically categorizing an image based on its content, assigns a label or class to an image by analyzing its visual features. This process is extensively employed across fields such as photo organization, radiology, and autonomous driving.

However, image classification remains a complex process susceptible to influences from factors like lighting, scale, and orientation[94]. To enhance classification accuracy, researchers and practitioners have developed sophisticated methods and techniques, including deep learning models.

With the advent of deep learning, image classification has achieved significant strides across various sectors[94]. This progress is largely attributed to the availability of vast data volumes, which facilitate the training of deep learning models to discern patterns and features in images. Consequently, image classification is no longer viewed as a challenge but rather a resolved issue in numerous applications.

Image classification is the process of automatically categorizing an image or picture based on its content. It involves assigning a label, or class, to an image based on its visual features. Image classification is widely used in many fields, including photo organization, radiology, and autonomous driving.

However, image classification is a complex process that can be influenced by various factors, such as lighting, scale, and orientation[94]. To improve the accuracy of classification, researchers and practitioners have developed advanced approaches and techniques, including deep learning models.

With the help of deep learning, image classification has made significant progress in various domains. This success is due to the availability of large amounts of data, which can be used to train deep learning models to recognize patterns and features in images. As a result, image classification is no longer considered a challenge, but rather a solved problem in numerous applications.

2.2 Co-Training

Recent advancements in deep learning have propelled image classification to new heights, enabling more accurate and robust systems, a brief illustration is shown in Figure 2.1. Convolutional Neural Networks (CNNs) have become the cornerstone of modern image classification due to their ability to automatically learn hierarchical features from raw pixel data. Furthermore, architectures such as ResNet, DenseNet, and EfficientNet have pushed the boundaries of performance by addressing challenges related to vanishing gradients, model complexity, and computational efficiency. Despite these advancements, supervised learning models are often constrained by the availability of labeled data. This limitation has led to the exploration of semi-supervised and unsupervised learning methods, paving the way for innovative approaches such as Co-Training. By leveraging unlabeled data and learning

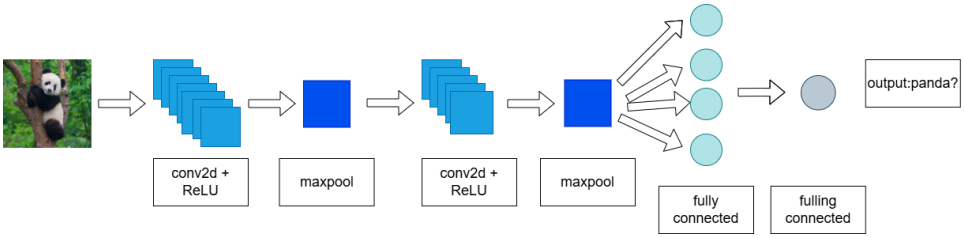


Figure 2.1: A brief illustration of how image classification is trained by using CNN.

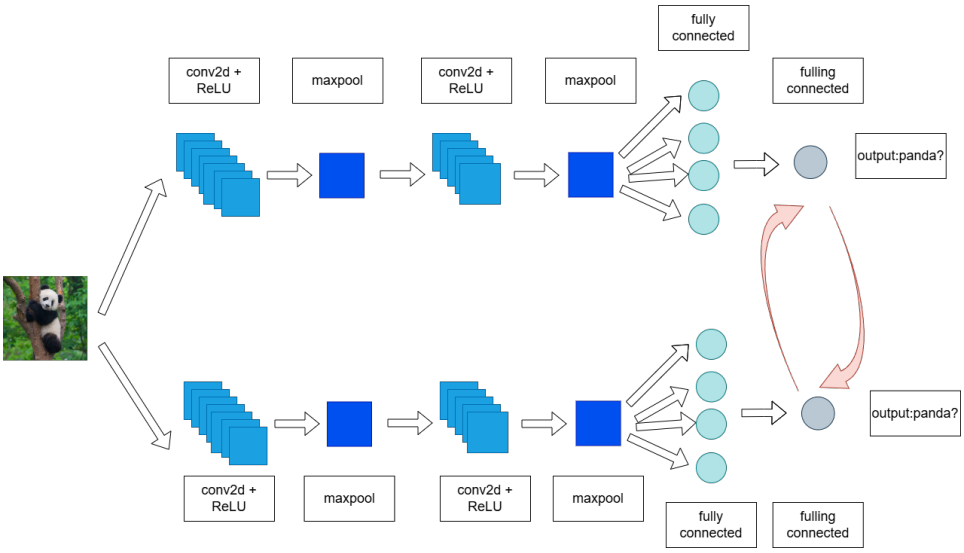


Figure 2.2: A brief illustration of how image classification is trained by using Co-training.

complementary feature representations, Co-Training enhances model generalization and robustness, making it a promising direction for complex tasks like image classification and domain adaptation, a brief illustration is shown in Figure 2.2.

The seminal work of Co-Training was proposed by Blum *et al.* [10], where they successfully demonstrated the use of a Co-Tr framework to learn different insights in a standard Web page classification problem. In general, a Co-Tr algorithm attempts to learn two or more distinct and divergent feature extractors and have been successfully used across a multitude of tasks ranging from domain adaptation [14], image classification [115, 48], data segmentation [12] to tag-based image search [41] and many more. One such well-known algorithm is Learning to Teach [56]. Similar to Co-Tr, LT co-evolves a teacher and a student network(s) to learn discriminative features thanks to an inherent feedback sharing mechanism between the two of them. Similarly, model distillation algorithms [164, 93, 56] use an additional *mimicry* loss to align the final class-specific posterior distributions of every student network.

Chen *et al*. [14] learns an Auto-encoder to benefit from using unlabeled data in the Co-Tr learning framework.

2.3 Learning from Noisy Labels

Learning from a clean dataset is not considered as a difficult task any longer [29, 119, 127]. Recently, there has observed a growing surge in the interest of studying the robustness of any machine learning algorithm against noisy labels. In this regard, MentorNet [63] trains an additional StudentNet network to select clean labels which is in-turn used to further guide the main training process. If a clean and unbiased validation set is not available, MentorNet will discover new data-driven sample-weight schemes from data which can be updated according to feedback from StudentNe. Ren *et al*. [116] follow a meta-learning paradigm and use a clean validation set to re-weight the training samples. Importance weights for training samples which result in the decrease of the loss in a clean validation set are increased, while the weights of those that result in the increase of the loss are decreased during the training process. One of the major drawbacks of [116] is the calculation of the clean validation set based importance weights after every gradient update of the network, which increases the time complexity of the overall algorithm. On the other hand, Decoupling [96] trains two different sub-networks with the examples that are confusing to both of them during the course of training[1]. Similarly, Co-Tr [48] trains two different networks with one selecting the clean examples, *i.e*. the examples with lower classification loss, for the other in an intertwined fashion. However, without any explicit discrepancy module between the networks that enforce the features learnt to be distinct and different, the solution learnt by the two aforementioned algorithm is not optimal. A breif illustration is shown in Fig 2.3.

2.4 Library and Preliminaries

In this section, we have listed all the notations and basic definitions used in this thesis. (It is arranged in alphabetical order, from A to Z.)

- **Adam** Adam (Adaptive Moment Estimation) is an advanced optimization algorithm used in machine learning, especially in training deep learning models. It combines the benefits of two other popular optimization techniques: Momentum and RMSProp. Adam adjusts the learning rate for each parameter individually by computing first and second moments of the gradients (i.e., the mean and the uncentered variance). This allows it to adaptively change the learning rate based on the recent gradient history.
- **CNN** A Convolutional Neural Network (CNN) is a type of deep learning model specifically designed for processing structured grid-like data, such

[1]Both the networks produce different predictions with high confidence.

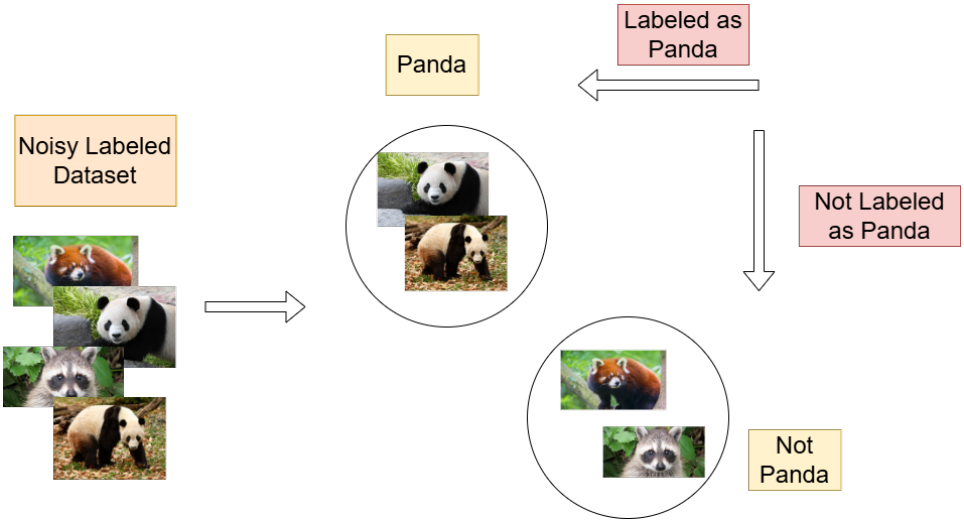


Figure 2.3: A brief illustration of how image classification is trained by using Co-training.

as images. CNNs are widely used in computer vision because they can automatically learn to recognize patterns, shapes, and objects in images without manual feature extraction. They are built upon layers that perform convolution operations, which apply filters (small matrices) to image regions, allowing the network to capture spatial hierarchies and features in an image, from edges and textures to more complex structures.

- **Contrastive Learning** Contrastive learning as proposed by Hadsell *et al*. [46] learned representations by contrasting positive pairs against negative pairs. The core idea of contrastive representation learning is to map semantically nearby points (positive pairs) closer together in the embedding space, while pushing apart the points that are dissimilar (negative pairs). Recently, contrastive learning is being used extensively in unsupervised learning [15, 65, 132, 3]. The standard approach uses different augmentation strategies such as horizontal and (or) vertical flips, random crop, colour jitter, random patches *etc*. to create multiple variations of every individual data point which are then considered positive pairs [154, 4, 15, 165, 61, 57]. Generating a distinct and an informative positive example for every data point is the fundamental bottleneck in contrastive learning.

- **Divergence** In computer vision, **divergence** typically refers to the measure of the difference or ”distance” between two distributions, such as the predicted image distribution and the ground truth distribution, or between the features of two images. Given two probability distributions P and Q defined over a space $\mathcal{X}$ (e.g., pixel values, feature space), the divergence $D(P\|Q)$ measures

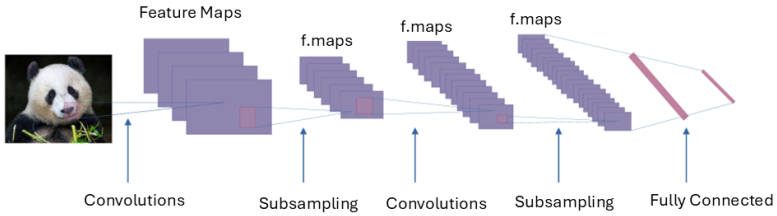


Figure 2.4: This figure illustrates a Convolutional Neural Network (CNN) model. It begins with an input image (panda), which undergoes layers of convolutions to extract feature maps. The feature maps then proceed through subsampling and additional convolutional layers. Finally, fully connected layers generate the model's output.

how much information is lost when Q is used to approximate P.

- **DNN** A Deep Neural Network (DNN) is an artificial neural network with multiple hidden layers between the input and output layers. DNNs are a powerful tool in machine learning, particularly for tasks requiring complex pattern recognition, such as image and speech recognition, natural language processing, and game playing. They consist of interconnected layers of neurons (or nodes) that process and transform input data through a series of mathematical operations to identify patterns and make predictions.
- **JS Divergence** A method of measuring the similarity between two probability distributions. It is a symmetrized and smoothed version of the Kullback-Leibler (KL) divergence
- **KL Divergence** KL Divergence (Kullback-Leibler Divergence) is a measure of how one probability distribution diverges from a second, expected probability distribution. It quantifies the difference between two distributions in terms of the amount of information lost when using one distribution to approximate another. KL Divergence is widely used in information theory, statistics, and machine learning, particularly for tasks involving probability distributions. Given two probability distributions P and Q over a discrete space $\mathcal{X}$, the **KL Divergence** from Q to P (denoted as $D_{\mathrm{KL}}(P\|Q)$) is defined as:

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)}{Q(x)}$$

For continuous distributions, the KL Divergence is defined as:

$$D_{\mathrm{KL}}(P\|Q) = \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)}\, dx$$

Where:

- $P(x)$ and $Q(x)$ are the probability mass functions (PMFs) or probability density functions (PDFs) of distributions P and Q, respectively,
- $\log$ denotes the natural logarithm.

KL Divergence is not a true distance because it is asymmetric and does not satisfy the triangle inequality. Instead, it measures how much one distribution deviates from another, which is crucial for tasks like distribution matching, model fitting, and information retrieval.

- **Learning Rate** The learning rate is a hyperparameter in machine learning algorithms, particularly in gradient-based optimization methods like stochastic gradient descent (SGD). It controls the size of the steps the model takes during training to update its parameters (e.g., weights) in response to the error or loss. A higher learning rate may lead to faster convergence but risks overshooting the optimal solution, while a lower learning rate ensures more precise updates but may result in slower convergence. Finding the right learning rate is crucial for effective training, as it directly influences the model's ability to learn efficiently and avoid getting stuck in suboptimal solutions.

- **MMD** MMD (Maximum Mean Discrepancy) is a statistical test used to measure the difference between two probability distributions. In machine learning, particularly in domain adaptation and generative models, MMD is used to quantify how much two distributions (such as the distributions of features in source and target domains) differ from each other. The goal is often to minimize this difference to make models more generalizable across domains.

  How MMD Works: MMD computes the distance between the mean embeddings of two distributions in a feature space, typically using a kernel function (like the Gaussian kernel) to map the data into a higher-dimensional space. The larger the MMD, the more the distributions differ, and the smaller the MMD, the more similar the distributions are.

  Key Components: 1. Kernel Function: Maps data into a higher-dimensional space where differences between distributions are more easily measurable. 2. Mean Embedding: Represents a distribution as the mean of the mapped data in the feature space. 3. Distance Measure: Typically, the squared distance between these embeddings is used as the MMD.

- **Noise Ratio** l to learn the true underlying patterns in the data.

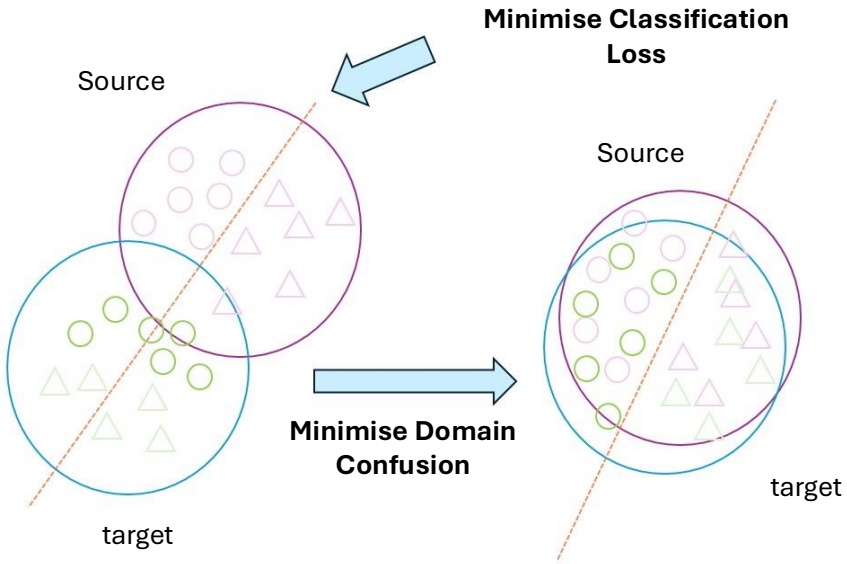
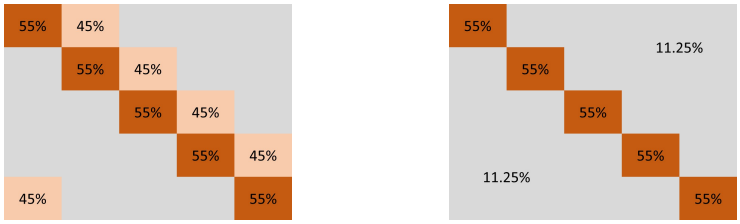


Figure 2.5: This image illustrates Maximum Mean Discrepancy (MMD) for domain adaptation. On the left, source and target domains are initially separated. By minimizing classification loss and maximizing domain confusion, MMD aligns the distributions, as shown on the right. This processreduces the discrepancy between source and target domains, improving model generalization.

- **Noise Transition** Throughout this thesis, we have consistently applied the following method to set up noisy data. Here we test our proposed algorithm on the task of noisy-supervised image classification. More specifically, and following the protocols of [48, 150], we manually corrupt all the clean datasets with the help of a noise transition matrix $\boldsymbol{Q} \in \mathbb{R}^{K \times K}$, where $\boldsymbol{Q}_{ij} = Pr(\tilde{\boldsymbol{y}} = j | \boldsymbol{y} = i)$ gives the probability that the label (*i.e.* $\tilde{y}$) is flipped to a value j from the clean label (*i.e.* y) having a value of i. In this chapter, we set up two different noise transition situations: (1) **Pair** flipping, and (2) **Symmetric** flipping. An exemplar noise transition matrix for each of the pair and symmetric flipping is shown in Fig2.6.

- **Robustness** In computer vision, robustness refers to a model's ability to accurately interpret and process images despite variations like noise, distortions, lighting changes, or occlusions. A robust model can handle input conditions without significant degradation in performance. Techniques such as data augmentation, which artificially increases the diversity of training data, and regularization, which prevents overfitting, are commonly used to improve robustness. Robustness is crucial in real-world applications like autonomous driving, facial recognition, or medical imaging, where slight variations in the



(a) Pair flip noise (b) Symmetric flip noise

Figure 2.6: Noise transition matrices [48] in the example of 5 classes.

data can otherwise lead to incorrect predictions or failures.

More specifically, In the context of learning from noisy labels, robustness refers to a model's ability to maintain accurate predictions despite the presence of incorrect or mislabeled training data. Noisy labels can arise from human errors or ambiguities in data annotation, which can severely affect model performance if not handled properly. A robust model is less sensitive to these noisy labels and can still generalize well by learning the underlying patterns in the data rather than memorizing the incorrect labels. Techniques like loss correction, regularization, and robust loss functions are commonly used to enhance robustness and reduce the impact of noisy labels.

- **Semi-supervised Learning** Semi-supervised learning is a type of machine learning that lies between supervised and unsupervised learning. In this approach, the model is trained on a small amount of labeled data and a large amount of unlabeled data. The labeled data provides the model with some guidance, while the unlabeled data allows the model to learn from the broader data distribution, improving its performance without requiring a fully labeled dataset. Semi-supervised learning is particularly useful when labeling data is expensive or time-consuming but acquiring a large set of unlabeled data is easy. By leveraging the unlabeled data, semi-supervised learning can achieve better generalization and performance than using just a small labeled dataset.

- **Sinkhorn Divergence** Sinkhorn divergence is a measure of the difference between two probability distributions that incorporates both optimal transport theory and regularization. It arises from the Sinkhorn distance, a regularized version of the Wasserstein distance (also known as the Earth Mover's Distance). The Sinkhorn divergence combines optimal transport with an entropy term to make the problem more computationally feasible.

  Given two probability distributions, μ nad ν, definition over a space $\mathcal{X}$, the Sinkhorn divergence is defined as the regularized Wasserstein distance:

$$S(\mu,\nu) = \inf_{\gamma \in \Pi(\mu,\nu)} \left(\sum_{(x,y)\in\mathcal{X}\times\mathcal{X}} c(x,y)\gamma(x,y) + \epsilon H(\gamma) \right) \tag{2.1}$$

Where: $\Pi(\mu, \nu)$ is the set of all joint distributions γ with marginals μ nad ν.

The entropy term makes the optimization easier by adding a smoothing effect, avoiding the combinatorial complexity of computing the optimal transport without regularization. The Sinkhorn divergence is computationally efficient and has found applications in machine learning, such as domain adaptation, generative models, and image matching.

- **SGD** Stochastic Gradient Descent (SGD) is an optimization algorithm used to minimize the loss function in machine learning models, particularly in neural networks. It is a variation of the traditional Gradient Descent (GD) algorithm, where the model's parameters (e.g., weights) are updated based on the gradient of the loss function. type of machine learning where the model is trained on labeled data. In this approach, the input data is paired with corresponding output labels, and the goal is for the model to learn the mapping between the inputs and the correct outputs. The training process involves minimizing the error between the predicted output and the true label. Supervised learning is commonly used for tasks like classification (e.g., spam detection, image classification) and regression (e.g., predicting house prices, stock market trends).

- **Transformer** The Transformer is a neural network architecture developed for tasks involving sequences, especially natural language processing (NLP). Unlike previous models like RNNs, Transformers process entire sequences at once using self-attention mechanisms, which allow each word in a sentence to relate to every other word, capturing context over long distances. This eliminates the need for sequential processing, making Transformers highly efficient and scalable.A Transformer consists of multiple encoder and decoder layers. Encoders process the input data, while decoders generate the output, each layer using self-attention to understand relationships within sequences. Positional encoding is added to each input to retain information about word order, since the model itself does not understand sequence positions.

- **Unsupervised Learning** A type of machine learning where the algorithm is trained on data that does not have labeled outputs. The goal is to find hidden patterns or structures in the data, such as clusters or relationships, without direct supervision. The model tries to understand the data's underlying structure without explicit guidance on what the correct output should be.

Table 2.1: Notation for Symbols

Symbols	Corresponding Notations
$\mathcal{S}_\epsilon$	Sinkhorn
O_1	Prediction output of first network

O_2	Prediction output of second network
ϵ	Smoothing parameter for Sinkhorn divergence
$\mathcal{U}$ and $\mathcal{V}$	positive random measures of unit mass over a metric space $\mathcal{X}$
$\{\boldsymbol{u}_i^{\mathcal{U}}\}_{i=1}^n$ and $\{\boldsymbol{v}_i^{\mathcal{V}}\}_{i=1}^m$	samples drawn from $\mathcal{U}$ and $\mathcal{V}$
$\mathbf{K}$	cost matrix
$\mathbf{N}$	noise transition matrix
$Vecy$	ground truth label
$\tilde{\boldsymbol{y}}$	corrupted label
α and β	finite discrete measures
σ_{x_i}	the probability of x_i to be at point α_i
π^*	Best transport plan
$\Pi(\boldsymbol{u}, \boldsymbol{v})$	Set of transportation plans
$c(\boldsymbol{u}, \boldsymbol{v})$	Cost function to move a unit of mass from $\mathcal{U}$ to $\mathcal{V}$
$\mathbf{C} \in \mathbb{R}^{n \times m}$	Matrix of cost function
$\otimes$	Product of the marginals
f and g	networks
θ and $\hat{\theta}$	parameters of networks
η	learning rate
T	epoch number
l	the layer number for $\mathcal{S}_\epsilon$
$\mathcal{D}$	Training set
$\bar{\mathcal{D}}$	Mini-batch of training set
$\mathcal{D}_1$ and $\mathcal{D}_2$	Discrepancy module
Loss_f	Total loss to update network f
Loss_g	Total loss to update network g
L_M^f	Conventional supervised loss for network f
L_M^g	Conventional supervised loss for network g
L_D	Diversity loss
L_C	Consistency loss
λ_D	Combination weights for the diversity loss
λ_C	Combination weights for the consistency loss
$\boldsymbol{X}_i$	Input data
$\boldsymbol{A}_i$	Features extracted from lth layer of network f
$\boldsymbol{B}_i$	Features extracted from lth layer of network g
$\boldsymbol{z}_i^f$	Softmax output of network f
$\boldsymbol{z}_i^g$	Softmax output of network g

Notation All notations used are shown in Table 2.1. Throughout this thesis, we use bold lower-case letters (*e.g.*, $\boldsymbol{x}$) and bold upper-case letters (*e.g.*, $\boldsymbol{X}$) to represent column vectors and matrices respectively. $[\boldsymbol{x}]_i$ denotes the ith element of the vector

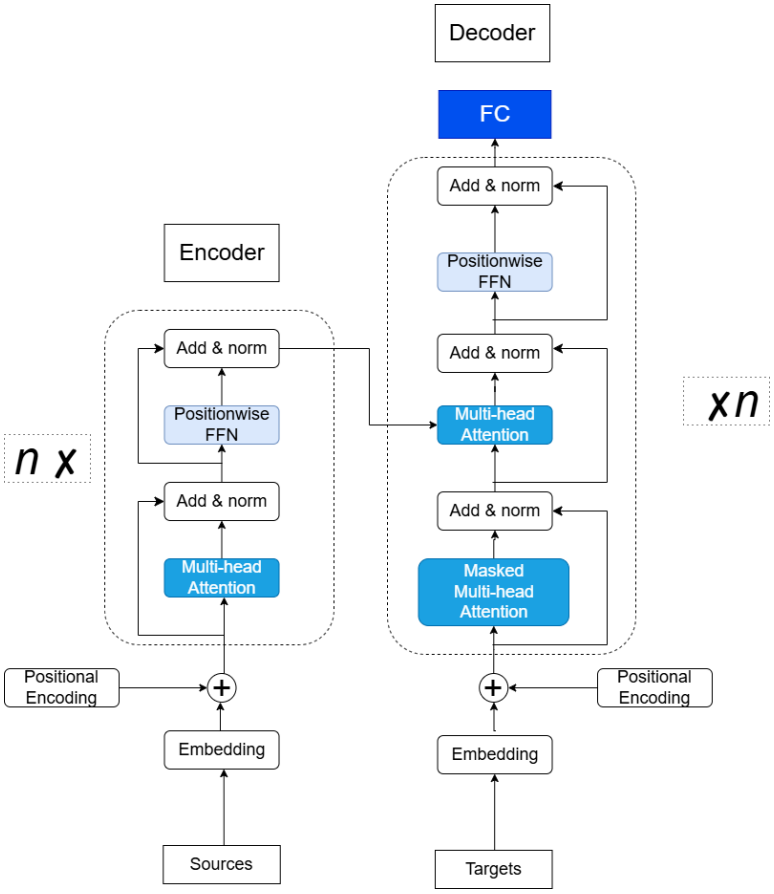


Figure 2.7: The architecture of a transformer

$\boldsymbol{x}$. $\mathbf{I}_n$ represents the $n \times n$ identity matrix. $\|\boldsymbol{X}\|_F = \sqrt{\mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X})}$ represents the Frobenius norm of the matrix $\boldsymbol{X}$, with $\mathrm{Tr}(\cdot)$ indicating the trace of the matrix $\boldsymbol{X}$. $\boldsymbol{X}^\top$ denotes the transpose of $\boldsymbol{X}$. $\mathcal{P}(\mathcal{U})$ and $\mathcal{P}(\mathcal{V})$ represent the set of probability measures on two metric spaces $\mathcal{U}$ and $\mathcal{V}$. $\{\boldsymbol{u}_i^{\mathcal{U}}\}_{i=1}^n$ and $\{\boldsymbol{v}_i^{\mathcal{V}}\}_{i=1}^m$ denote n and m i.i.d. samples drawn from $\mathcal{U}$ and $\mathcal{V}$, respectively.

In this work, we use $\mathcal{S}_\epsilon$ [38] which can be used over general spaces $\mathcal{X}$, instead of only the Euclidean space $\mathbb{R}^d$ and not only the 1-Wasserstein distance. We first calculate OT between two probability distributions $\mathcal{U}$ and $\mathcal{V}$ (assume $\mathcal{U}$ and $\mathcal{V}$ are positive random measures of unit mass over a metric space $\mathcal{X}$) which is defined as the solution of the, possibly infinite dimensional, linear program:

$$\mathcal{OT}_c(\boldsymbol{u}, \boldsymbol{v}) = \min_{\pi \in \prod(\boldsymbol{u}, \boldsymbol{v})} \int_{\mathcal{X} \times \mathcal{X}} c(\boldsymbol{u}, \boldsymbol{v}) d\pi(\boldsymbol{u}, \boldsymbol{v}) \tag{2.2}$$

where $\Pi(\boldsymbol{u}, \boldsymbol{v})$ is the set of transportation plans (or couplings) of the joint probability distribution over the product space $\mathcal{U} \times \mathcal{V}$ with marginals $\boldsymbol{u}$ and $\boldsymbol{v}$ respectively. Here, $c(\boldsymbol{u}, \boldsymbol{v})$ is the cost function to move a unit of mass from $\mathcal{U}$

to $\mathcal{V}$. The cost function c needs to be defined for every pair $\boldsymbol{u}_i^{\mathcal{U}}, \boldsymbol{v}_j^{\mathcal{V}}$; and thus can be represented as a matrix $\mathbf{C} \in \mathbb{R}^{n \times m}$. Therefore, the total cost incurred is $\langle \pi, \mathbf{C} \rangle := \sum_{ij} \pi_{ij} \mathbf{C}_{ij}$. When $\mathcal{X}$ is equipped with a distance $d_{\mathcal{X}}$, a typical choice is $c(\boldsymbol{u}, \boldsymbol{v}) = d_{\mathcal{X}}(\boldsymbol{u}, \boldsymbol{v})^p$ with $p > 0$ and thus we obtain the so-called p-Wasserstein distance between probability measures. However, because the above equation is not differentiable, we refer to the regularized optimal transport problem [37] defined by

$$\begin{aligned} \mathcal{W}_\epsilon(\boldsymbol{u}, \boldsymbol{v}) &:= \min_{\pi \in \Pi(\boldsymbol{u}, \boldsymbol{v})} \langle \pi, \mathbf{C} \rangle + \epsilon \, \mathrm{KL}(\pi \mid \boldsymbol{u} \otimes \boldsymbol{v}) \\ &= \min_{\pi \in \Pi(\boldsymbol{u}, \boldsymbol{v})} \int_{\mathcal{U} \times \mathcal{V}} \mathrm{c}(\boldsymbol{u}, \boldsymbol{v}) \, d\pi(\boldsymbol{u}, \boldsymbol{v}) + \epsilon \, \mathrm{KL}(\pi \mid \boldsymbol{u} \otimes \boldsymbol{v}) \ , \end{aligned} \tag{2.3}$$

where $\mathrm{KL}(\pi | \boldsymbol{u} \otimes \boldsymbol{v})$ denotes the KL divergence between π and $\boldsymbol{u} \otimes \boldsymbol{v}$. $\boldsymbol{u} \otimes \boldsymbol{v}$ represents the product of the marginals (or the probability measures) $\boldsymbol{u}$ and $\boldsymbol{v}$. When $\mathrm{c}(\boldsymbol{u}, \boldsymbol{v}) = \mathrm{D}^p(\boldsymbol{u}, \boldsymbol{v})$; we obtain the entropy regularized p-Wasserstein distance as shown below:

$$\mathcal{W}_\epsilon^p(\boldsymbol{u}, \boldsymbol{v}) = \Big(\min_{\pi \in \Pi(\boldsymbol{u}, \boldsymbol{v})} \int_{\mathcal{U} \times \mathcal{V}} \mathrm{D}^p(\boldsymbol{u}, \boldsymbol{v}) \, d\pi(\boldsymbol{u}, \boldsymbol{v}) \Big)^{1/p} + \epsilon \, \mathrm{KL}(\pi \mid \boldsymbol{u} \otimes \boldsymbol{v}) \tag{2.4}$$

Similar to [38], the *Sinkhorn* loss is defined as Eqn.2.5 and algorithm of finding best transport plan π^* is given in Algorithm.**??**.

$$\mathcal{S}_\epsilon(\boldsymbol{u}, \boldsymbol{v}) = \mathcal{W}_\epsilon(\boldsymbol{u}, \boldsymbol{v}) \ - \ \frac{1}{2} \big(\mathcal{W}_\epsilon(\boldsymbol{u}, \boldsymbol{u}) + \mathcal{W}_\epsilon(\boldsymbol{v}, \boldsymbol{v}) \big) \tag{2.5}$$

In this paper, we set $p = 2$. $\mathcal{S}_\epsilon$ has the following behavior in ϵ:

1. as $\epsilon \to 0, \mathcal{S}_\epsilon(\boldsymbol{u}, \boldsymbol{v}) \to 2\mathcal{W}_\epsilon(\boldsymbol{u}, \boldsymbol{v})_c$;
2. as $\epsilon \to \infty, \mathcal{S}_\epsilon(\boldsymbol{u}, \boldsymbol{v}) \to \mathcal{MMD}_{-c}(\boldsymbol{u}, \boldsymbol{v})$, where $\mathcal{MMD}_{-c}$ is the Maximum Mean Distance whose kernel is the cost function.

Maximum Mean Distance

An empirical estimate of MMD between $\mathcal{U}$ and $\mathcal{V}$ is obtained as

$$\mathcal{MMD}(\mathcal{U}, \mathcal{V}) = \Big\| \frac{1}{n} \sum_{i=1}^{n} \Phi(\boldsymbol{u}_i) - \frac{1}{m} \sum_{j=1}^{m} \Phi(\boldsymbol{v}_j) \Big\|_H^2 \tag{2.6}$$

Here, H denotes the induced Reproducing Kernel Hilbert space [1]) and $\| \cdot \|_H$ denotes its norm. $\mathcal{MMD}(\mathcal{U}, \mathcal{V})$ is a measure of overlap between $\mathcal{U}$ and $\mathcal{V}$ such that increase (or decrease) in overlap results in decrease (or increase) in $\mathcal{MMD}(\mathcal{U}, \mathcal{V})$. $\Phi(\boldsymbol{p})$ represents a functional mapping of the input $\boldsymbol{p}$ to a high-dimensional space. By the kernel trick, the form in Eqn. (2.9) can be written as;

$$\mathcal{MMD}(\mathcal{U},\mathcal{V}) = \frac{1}{n^2}\sum_{i}^{n}\sum_{i'}^{n} k(\boldsymbol{u}_i,\boldsymbol{u}_{i'}) - \frac{1}{nm}\sum_{i}^{n}\sum_{j}^{m} k(\boldsymbol{u}_i,\boldsymbol{v}_j) + \frac{1}{m^2}\sum_{j}^{m}\sum_{j'}^{m} k(\boldsymbol{u}_j,\boldsymbol{v}_{j'}) \tag{2.7}$$

In all our experiments, we have employed the Gaussian kernel

$$k(\boldsymbol{u},\boldsymbol{v}) = \exp(-\frac{\|\boldsymbol{u}-\boldsymbol{v}\|^2}{\sigma}) \ . \tag{2.8}$$

Notations: Throughout this paper, we use bold lower-case letters ($\boldsymbol{x}$) to show column vectors and bold upper-case letters (*e.g.*, $\boldsymbol{X}$) to show matrices.$[\cdot]_i$ is used to denote the i-th element of a vector and $\mathbf{I}_n$ shows the $n \times n$ identity matrix. The Frobenius norm of a matrix is shown by $\|\boldsymbol{X}\|_F = \sqrt{\mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X})}$, with $\mathrm{Tr}(\cdot)$ indicating the matrix trace.

In this work, we make use of the Maximum Mean Discrepancy ($\mathcal{MMD}$) [45] to measure the difference between two distributions $\mathcal{S}$ and $\mathcal{T}$. Let $\{\boldsymbol{X}_i^{\mathcal{S}}\}_{i=1}^n$ and $\{\boldsymbol{X}_i^{\mathcal{T}}\}_{i=1}^m$ denote i.i.d samples taken from $\mathcal{S}$ and $\mathcal{T}$, respectively.

An empirical estimate of $\mathcal{MMD}$ between $\mathcal{S}$ and $\mathcal{T}$ is obtained as

$$\mathrm{MMD}(\mathcal{S},\mathcal{T}) = \Big\| \frac{1}{n}\sum_{i=1}^{n} \Phi(\boldsymbol{X}_i^{\mathcal{S}}) - \frac{1}{m}\sum_{j=1}^{m} \Phi(\boldsymbol{X}_j^{\mathcal{T}}) \Big\|_H^2 \tag{2.9}$$

Here, H denotes the induced Reproducing Kernel Hilbert space [1]) and $\|\cdot\|_H$ denotes its norm. $\mathrm{MMD}(\mathcal{S},\mathcal{T})$ is a measure of overlap between $\mathcal{S}$ and $\mathcal{T}$ such that increase (or decrease) in overlap results in decrease (or increase) in $\mathrm{MMD}(\mathcal{S},\mathcal{T})$. $\Phi(\boldsymbol{p})$ represents a functional mapping of the input $\boldsymbol{p}$ to a high-dimensional space. By the kernel trick, the form in Eqn. (2.9) can be written as;

$$\mathcal{MMD}(\mathcal{S},\mathcal{T}) = \frac{1}{n^2}\sum_{i}^{n}\sum_{i'}^{n} k(\boldsymbol{X}_i^{\mathcal{S}},\boldsymbol{X}_{i'}^{\mathcal{S}}) - \frac{1}{nm}\sum_{i}^{n}\sum_{j}^{m} k(\boldsymbol{X}_i^{\mathcal{S}},\boldsymbol{X}_j^{\mathcal{T}}) + \frac{1}{m^2}\sum_{j}^{m}\sum_{j'}^{m} k(\boldsymbol{X}_j^{\mathcal{T}},\boldsymbol{X}_{j'}^{\mathcal{T}}) \tag{2.10}$$

In all our experiments, we have employed the Gaussian kernel

$$k(\boldsymbol{u},\boldsymbol{v}) = \exp(-\frac{\|\boldsymbol{u}-\boldsymbol{v}\|^2}{\sigma}) \ . \tag{2.11}$$

2.4.1 Datasets and Basic Information

In this subsection, we provide an overview of all the datasets mentioned and utilized in the experiments throughout this thesis.

We evaluate our proposed loss on the following well-studied image classification datasets:

1. Cifar-10 [73] (C-10)is a widely used image classification dataset designed for evaluating machine learning and deep learning algorithms, categorized into 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The dataset is evenly divided into 50,000 training images and 10,000 test images, with each class containing 6,000 images.

2. Cifar-100 [73] (C-100) consists of $32 \times 32 \times 3$ images from 100 different *fine* categories, which can be further grouped into 20 super categories. We have considered the 20 super categories in our experiments. Similar to Cifar-10 [73], the training and the test set consist of $50,000$ and $10,000$ images respectively.

3. SVHN SVHN is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting. It can be seen as similar in flavor to MNIST (e.g., the images are of small cropped digits), but incorporates an order of magnitude more labeled data (over 600,000 digit images) and comes from a significantly harder, unsolved, real world problem (recognizing digits and numbers in natural scene images). SVHN is obtained from house numbers in Google Street View images [105].

4. CUB200-2011 The *Caltech-UCSD Birds* (**CUBS-**200**-**2011**)** [145] consists of 11,788 images of birds from 200 different varieties. The first 100 categories are considered for training (5,864 images), while the rest 100 categories are considered for testing (5,924 images).

5. CARS196 The *Stanford Cars* (**CARS196**) dataset [71] consists of 16,185 images of cars from 196 different categories. The first 98 categories (8,054 images) are considered in the training of the network, while the next 98 categories (8,131 images) are used in the testing phase.

6. Clothing1M Clothing1M is a large-scale dataset with real-world noisy labels. It consists of 1 million training images collected from online shopping websites with labels generated from surrounding texts. More details about the datasets are listed in Table 2.2.

7. STL-10 [17] (S-10) is an image recognition dataset that consists of $96 \times 96 \times 3$ images from 10 different classes, each comprising $1,300$ examples. The training set consist of $5,000$ images, while the remaining $8,000$ images constitute the test set. It also consists of $100,000$ unlabeled images of the

Table 2.2: Details of the various datasets used in the thesis. Note that STL-10 includes 10,000 unlabeled images and 5,000 labeled images for training.

dataset	# of train images	# of test images	# of class	image size
MNIST	60,000	10,000	10	28×28
CIFAR10	50,000	10,000	10	32×32
CIFAR100	50,000	10,000	100	32×32
SVHN	73,257	26,032	10	32×32
CUB200-2011	5,864	5,924	200	227×227
CARS196	8,054	8,131	196	227×227
Clothing1M	50,000	10,000	14	224×224
STL-10	15,000	8,000	10	96×96

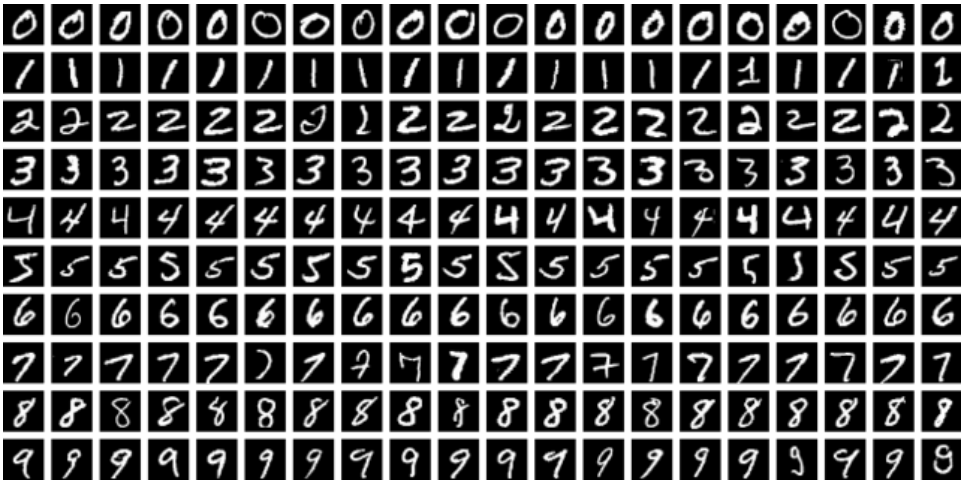


Figure 2.8: Sample images of MNIST datset

same resolution extracted from a similar but a wider distribution of images. However, we don't use these unlabeled images, neither during the training nor during the test phase in our proposed algorithm.

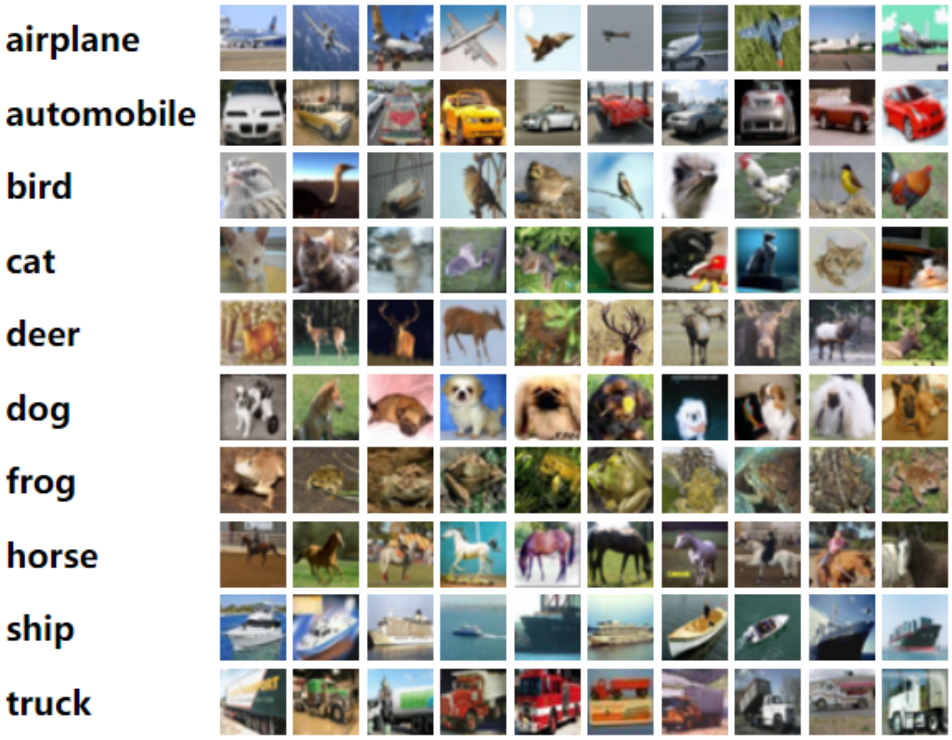


Figure 2.9: Sample images of Cifar10 datset

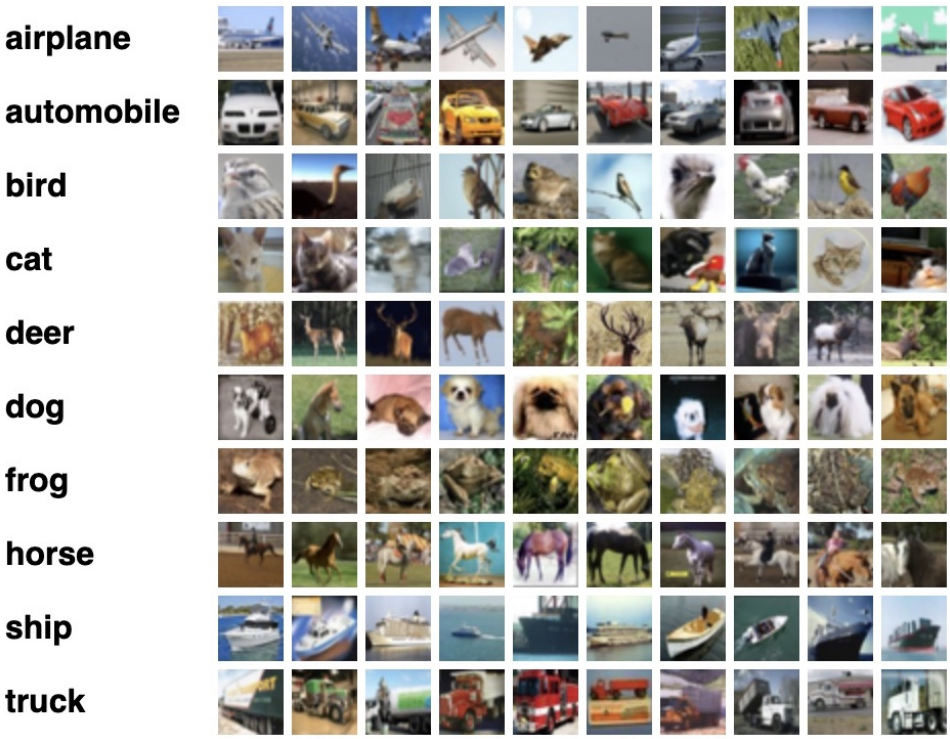


Figure 2.10: Sample images of Cifar100 datset

Figure 2.11: Sample images of SHVN datset

Figure 2.12: Sample images of Clothing1M datset

Figure 2.13: Sample images of Clothing1M datset

Figure 2.14: Sample images of Clothing1M datset

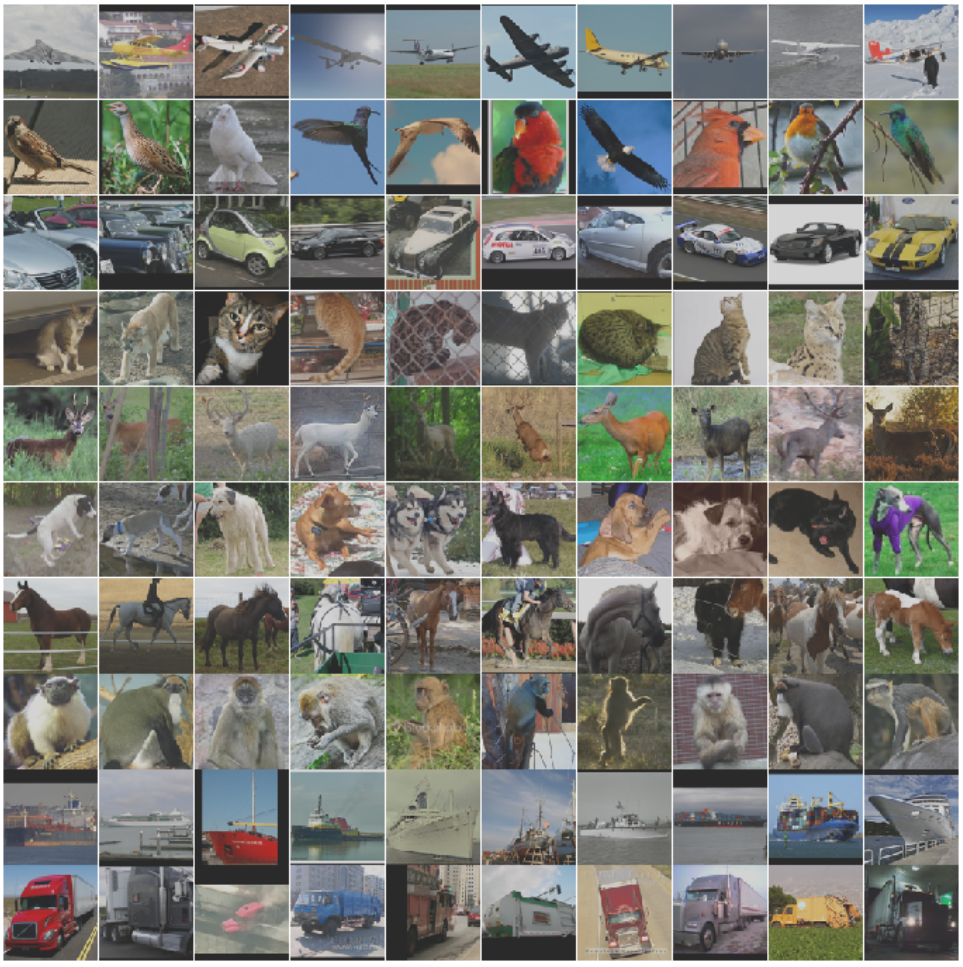


Figure 2.15: Sample images of STL-10 datset

Chapter 3

Learning Noisy labels with Discriminate Models

3.1 Introduction

Deep neural networks can easily fit random labels despite no apparent structure [162]. This phenomenon is particularly pronounced in deeper and wider networks due to their high capacity to memorize the training data, including noisy or incorrect labels. Such memorization leads to overfitting, where the model learns patterns that do not generalize to unseen data, ultimately degrading its performance. As highlighted in [19], even a modest noise rate of 20% can significantly impair model accuracy, in some cases halving the test performance. This vulnerability to noisy labels underscores the necessity for robust learning strategies that can effectively mitigate their adverse effects.

In this chapter, we propose a simple but effective learning paradigm called "Discrepant Collaborative Training". The proposed method allows us to make use of both clean and noisy examples, and improve the ability to select clean instances.

To identify clean samples from noisy ones, one can inspect the sample loss during training. Samples with a large loss value are more likely to be noisy and, hence, can be removed/weighted-down during the back propagation step.

The idea of Co-Training (Co-Tr) is to benefit from two (or more) networks which are complementary to one another and can help correct each other when they make mistakes. Naturally, the idea of Co-Tr suits the task of learning from a noisy dataset as well. A brief illustration of how co-training is used to train noisy labels can be found in Chapter 2 Figure 2.2. The underlying assumption, here, is that the two networks are substantially different to each other so their loss signal is complementary. To achieve a discrepancy between the loss signals, one can choose to use two different network architectures. This, however, brings a new level of difficulty as, if the networks are not similar in structure, one might adapt faster with the result of the iterative procedure failing. Moreover, use of two different network architectures may lead to potential mismatch between their individual

expressive powers, with a potential of one outperforming the other by a significant margin. As such, it may be more desirable in practice to use a specific structure for both networks and initialize them differently to achieve the necessary diversity [48, 63]. Unfortunately, even with this, there is no guarantee that the networks remain substantially different for the purpose of collaboration as required for the Co-Tr framework. In this work, we therefore propose a new method to ameliorate this shortcoming. A *max-discrepancy co-training framework* is proposed, where we achieve diversity by encouraging the networks to be statistically different. The notion of maximum mean discrepancy [49] is a simple yet powerful non-parametric criterion that measures the discrepancy of two distributions by mapping them to a Reproducing Kernel Hilbert Space (RKHS) [5].

Our main contributions are:

- A novel method to utilize diversity between networks in a co-training setting.
- Demonstrated improvement in identifying clean samples in a noisy dataset.
- Competitive or State-of-the-Art results with respect to comparable works on five different datasets, including two large fine-grained image recognition datasets (CUBS200-2011 [146] and CARS196 [72]).

3.2 Related Work

Discrepancy Measurement. Comparing and matching probability distributions between two different domains forms the fundamental building block for the design and development of several algorithms in the field of machine learning and computer vision [68, 42, 117, 89]. Several divergences such as Kullback-Leibler (KL, [75]), Jensen-Shannon [97], *etc*. have been successfully integrated in learning similar/dissimilar distributions across various domains. However, these diversity learning algorithms do not take into account the geometry of the distributions, thereby failing to either disentangle or join the distributions [31]. Fortunately, several algorithms such as Maximum Mean Discrepancy (MMD) [49, 89], Sinkhorn Divergence [31], Optimal Transport (OT) [9] *etc*. have been developed and successfully utilized to address the aforementioned drawback. MMD has been widely adopted as a discrepancy metric module across domain adaptation ([5]), unsupervised learning [156], generative models [28] and many more.

3.3 Methodology

In this section, we present our proposed methodology *i.e*. Discrepant Colaborative Training (DCT). We formulate DCT with a cohort of two networks (see Fig. 3.1). The two sub-networks used in DCT are represented as f and g with its learnable parameters θ and $\widehat{\theta}$, respectively. We first briefly describe the selection strategy for choosing the clean labels, followed by the description of the discrepancy MMD module and the overall definition of the loss function. A brief ilustration of the method is shown in fig3.1

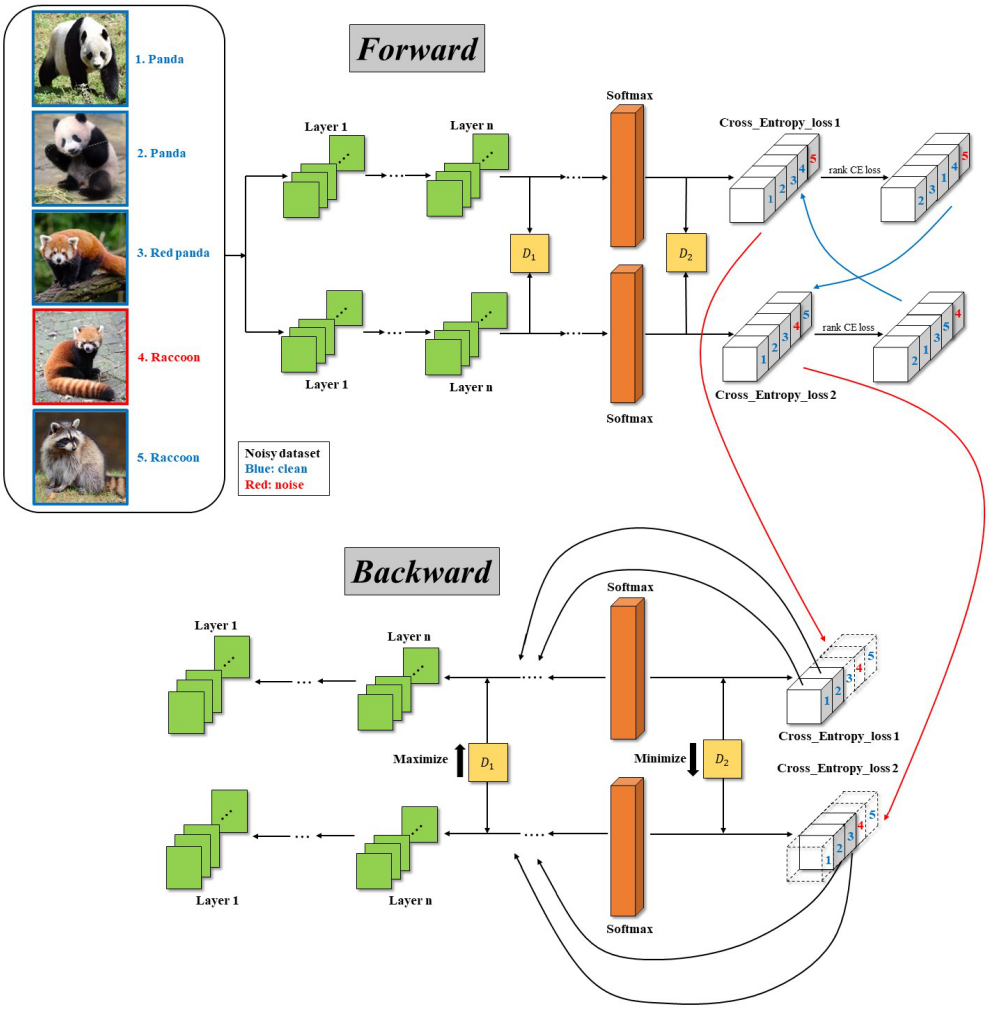


Figure 3.1: Forward (top) and backward (bottom) propagation of Discrepant Collaborative Training. **[FORWARD]** Five images are fed into sub-networks independently, four (blue) are correctly labeled and one (red) is corrupted with noise. The first discrepancy module is placed after *Layer n* between two networks. The second discrepancy module is placed after *softmax* layer. Then, the five images will be ranked according to its own cross entropy loss calculated by each network. Then the two networks exchange information of the ranking. **[BACKWARD]** According to the ranking information provided by its "peer" network, each network chooses only a few images with smaller loss value to update itself. D_1 and D_2 are Sinkhorn divergence modules. We maximize the diversity of the first discrepancy module to learn diverse features in each of the network, while the diversity of the second module is minimized so as to learn the same class distributions. [50]

3.3.1 Selection Strategy

Similar to [48, 63], we also employ the classification loss based selection strategy. More specifically, we obtain the classification loss of $\boldsymbol{X}_i$ for f and g as shown below:

$$\begin{aligned} \mathrm{L}_f(\boldsymbol{X}_i) &= -\log\left(\frac{\exp(\boldsymbol{z}_i^f)}{\sum_1^m \exp(\boldsymbol{z}_j^f)}\right) \\ \mathrm{L}_g(\boldsymbol{X}_i) &= -\log\left(\frac{\exp(\boldsymbol{z}_i^g)}{\sum_1^m \exp(\boldsymbol{z}_j^g)}\right) \quad , \end{aligned} \tag{3.1}$$

where $\boldsymbol{z}_i^f = f_\theta(\boldsymbol{X}_i)$ and $\boldsymbol{z}_i^g = g_{\widehat{\theta}}(\boldsymbol{X}_i)$ [1]. Similar to [48], we select R examples for each f and g within a mini-batch of size N that produces the R lowest L_g and L_f respectively. Thereafter the loss to update θ_f is given as

$$\mathrm{L}_1^f = \sum_{i=1}^{R} \mathrm{L}_f(\boldsymbol{X}_i) \quad \forall \boldsymbol{X}_i \in \mathcal{D}_g \quad , \tag{3.2}$$

where $\mathcal{D}_g$ represents set of images that results in the R lowest L_g calculated in Eqn 4.5. Similarly, we calculate the loss to update θ_g as

$$\mathrm{L}_1^g = \sum_{i=1}^{R} \mathrm{L}_g(\boldsymbol{X}_i) \quad \forall \boldsymbol{X}_i \in \mathcal{D}_f \quad , \tag{3.3}$$

where $\mathcal{D}_f$ represents set of images that results in the R lowest L_f as calculated in Eqn 4.5.

3.3.2 Discrepancy Loss

In order for f and g to learn diverse features, we propose to explicitly insert a MMD module in-between them. The loss calculated is shown as:

$$\mathrm{L}_2 = \mathrm{MMD}(\boldsymbol{A}_i, \boldsymbol{B}_i), \tag{3.4}$$

where $\boldsymbol{A}_i = f_{\theta(1:l)}(\boldsymbol{X}_i)$ and $\boldsymbol{B}_i = g_{\widehat{\theta}(1:l)}(\boldsymbol{X}_i)$, l denotes the layer where the MMD module is used, and $\theta(1:l)$ and $\widehat{\theta}(1:l)$ represent the parameters of the two networks f and g till the layer l. It is to be noted that L_2 is calculated irrespective of the presence or absence of noisy labels within the mini-batch of the images.

3.3.3 Consistency Loss

Even though we want both f and g to learn diverse and distinct features, the final class probability distribution learnt by f and g should not be very different from

[1] Usually it is the output of the softmax layer for each of f and g.

Table 3.1: Structure of Network trained by MNIST, CIFAR10 and CIFAR100. The slope of all all LReLU functions in the network are set to 0.01. K denotes the number of training classes for the dataset used.

#Layer	Input Image
1	3×3 conv, 128 LReLU
2	3×3 conv, 128 LReLU
3	3×3 conv, 128 LReLU
	2×2 max-pool, stride 2
	dropout, p = 0.25
4	3×3 conv, 256 LReLU
5	3×3 conv, 256 LReLU
6	3×3 conv, 256 LReLU
	2×2 max-pool, stride 2
	dropout, p = 0.25
7	3×3 conv, 512 LReLU
8	3×3 conv, 256 LReLU
9	3×3 conv, 128 LReLU
	avg-pool
10	fc-layer $128 \rightarrow K$
	softmax

each other. Thus we use another MMD module to explicitly reduce the discrepancy between the $\boldsymbol{z}_i^f$ and $\boldsymbol{z}_i^g$ for every $\boldsymbol{X}_i$ as shown below:

$$\mathrm{L}_3 = \mathrm{MMD}(\boldsymbol{z}_i^f, \boldsymbol{z}_i^g). \tag{3.5}$$

The final loss for DCT is given below:

$$\mathrm{Loss}_f = \mathrm{L}_1^f + \lambda_3 \mathrm{L}_3 - \lambda_2 \mathrm{L}_2 \tag{3.6}$$

$$\mathrm{Loss}_g = \mathrm{L}_1^g + \lambda_3 \mathrm{L}_3 - \lambda_2 \mathrm{L}_2, \tag{3.7}$$

λ_3 and λ_2 are the combination weights for the consistency and the diversity loss respectively. Eqn 3.6 and 3.7 are used to update f and g respectively using stochastic gradient optimizers. Algorithm 1 provides the pseudo code of our propose DTC algorithm.

3.4 Experiments

Dataset. We verify the effectiveness of our approach on five benchmark datasets: MNIST [80], CIFAR10 [73], CIFAR100 [73], CUB200-2011 [146] and CARS196 [72]. More details can be found in section 2.4.1.

Table 3.2: Comparison of our proposed DCT against several baseline algorithms. We report the average accuracy (%) after 5 runs for DCT.

noise	Dataset	F-C [112]	DP [96]	MN [63]	CT [48]	DCT
pairflip-45%	MNIST	0.24	58.03	80.88	87.63	**88.54**
symmetric-50%	MNIST	79.61	81.15	90.05	91.32	**94.21**
symmetric-20%	MNIST	**98.82**	95.70	96.70	97.25	98.54
pairflip-45%	CIFAR10	6.61	48.80	58.14	72.62	**72.91**
symmetric-50%	CIFAR10	59.83	51.49	71.10	74.02	**78.50**
symmetric-20%	CIFAR10	84.55	80.44	80.76	82.32	**85.41**
pairflip-45%	CIFAR100	1.60	26.05	31.60	34.811	**35.33**
symmetric-50%	CIFAR100	41.04	25.80	39.00	41.37	**42.11**
symmetric-20%	CIFAR100	**61.87**	44.52	52.13	54.23	56.11

Noise Type. We test our design on noisy-supervised image classification task. Since all datasets are clean, following [48, 112], we corrupt these datasets manually by the noise transition matrix $\boldsymbol{Q} \in \mathbb{R}^{K\times K}$, where $\boldsymbol{Q}_{ij} = Pr(\tilde{\boldsymbol{y}} = j|\boldsymbol{y} = i)$ gives the probability that noisy $\tilde{y}$ is flipped from clean y. In this chapter, we test our methods on two different noise transitions matrix: (1)Symmetry flipping; (2) Pair flipping. Noise transition matrices are can be found in Chapter 2.

In our experiments, we test three different noise conditions: (**1**) 45 pair flip noise; (**2**) 50 symmetry flip noise; (**3**) 20 symmetry flip noise. Since this chapter mainly focuses on the value of noisy data and robustness of our proposed DCT, we choose high noise rate values *i.e*. 45 and 50. It is to be noted that, for pair flip noise, ϵ should be lower than 50, otherwise the neural network will need additional information to learn discriminative features. In order to verify the robustness of our DCT method under lower noise condition, we also test on 20 symmetry flip noise.

Network and optimizer. For experiments on MNIST, CIFAR10 and CIFAR100, we use the CNN architecture as shown in Table 4.1. We follow the settings of "Temporal Ensembling" ([77]) and "Virtual Adversarial Training" ([100]) which is the well-acknowledged standard test bed for weakly-supervised learning. Here, we use Adam [67] optimizer with momentum and initial learning rate set to 0.9 and 0.001 respectively. The batch size is fixed to 128 and we run DCT for 600 epochs. We choose [48] as our baseline and we follow their detailed settings. For experiments on CUB200-2011 and CARS196, we use Inception-V1 [136] architecture pretrained on Imagenet [21]. Here, we use RMSProp and Adam optimizer for CUB200-2011 and CARS196 dataset respectively. The initial learning rate for both the optimizers is set to 0.0001. The batch size is fixed to 32 and we report the results after 50 epochs of training.
Note: We report the optimal value of the hyper-parameters λ_2, λ_3 and σ used in DCT for all the datasets in ablation study.

Table 3.3: Comparison of our proposed DCT against several baseline algorithms for large fine-grained image recognition datasets in terms of accuracy on the test set (%).

noise	Dataset	Cross Entropy	Co-Teaching	DCT (ours)
symmetric-50%	CUB200-2011	40.80	54.64	**57.24**
symmetric-20%	CUB200-2011	63.78	72.34	**74.57**
symmetric-50%	CARS196	38.86	66.75	**67.80**
symmetric-20%	CARS196	71.76	86.00	**86.62**

Table 3.4: Study of the importance of using noisy samples. The average accuracy (%) is reported after 5 different runs of DCT.

noise	Dataset	Co-Teaching	DCT-clean	DCT
symmetric-50%	MNIST	91.32	92.92	**94.21**
symmetric-20%	MNIST	97.25	98.14	**98.54**
symmetric-50%	CIFAR10	74.02	76.81	**78.50**
symmetric-20%	CIFAR10	82.32	84.47	**85.41**
symmetric-50%	CIFAR100	41.37	41.55	**42.11**
symmetric-20%	CIFAR100	54.23	55.54	**56.11**

Baselines. For MNIST, CIFAR10 and CIFAR100, we compare our results with the baseline methods: (i) F-correction [112], which corrects the prediction by using a noise transition matrix; (ii) Decoupling [96],which updates the parameters only using the samples where the two networks are not confident in their predictions; (iii) MentorNet [63], where an extra teacher network is pre-trained and used to filter out the noisy instances for its student network to learn robustly under noisy labels. (iv) Co-Teaching ([48]), which trains two identical sub-networks with one selecting the possible clean labels for the other. Results of the above baselines are reported from [48]. Furthermore, in order to verify the robustness of the performance of DCT method on the fine-grained image recognition datasets such as CUB200-2011 and CARS196; we consider two baselines model trained with the **(a)** conventional cross-entropy loss and **(b)** Co-Teaching algorithms.

3.4.1 Analysis on MNIST, CIFAR10 and CIFAR100

MNIST. As observed in Table 4.2, all the baseline methods obtain competitive results against each other for a low value of noise rate (*i.e*. symmetry 20%). Our DCT method obtains competitive 98.54% against the best method *i.e*. F-correction (which achieves 98.82%) accuracy on the test set. A drop in performance for F-correction and Decoupling baseline methods is observed when the noise rate is increased to 50%. On the other hand, MentorNet and Co-Teaching are pretty reboust in dealing

with a higher noise rate. Our proposed DCT method outperforms all the baselines and obtains the state-of-the-art accuracy of 94.21%, outperforming the current best algorithm (*i.e*. Co-Teaching) by 2.89%. Further increase in the complexity of the nosie (*i.e*. pair-flip noise with $\epsilon = 45\%$) F-correction and Decoupling fail to classify images. Again, one can evidently observe that DCT outperforms all the baselines by a significant margin. More specifically, we outperform MentorNet by 7.66% in terms of accuracy on the test-set.

CIFAR10. From Table 4.2, it is observed that all the baseline methods obatin similar results in terms of accuracy on the test set for symmetry flip noise with $\epsilon = 20\%$. Unlike MNIST dataset, DCT outperforms all the baselines including F-correction by 0.86% for CIFAR10 dataset. On further increasing the complexity of the noise properties, it is easily observed that DCT is the best performing method against all the baselines, thereby validating the design choices of using two different MMD modules to learn distinct and discriminative features.

CIFAR100. It is observed from Table 4.2 that for symmetric noise with $\epsilon = 20\%$, DCT outperforms all the competitive baseline algorithms except F-correction. It is evident that F-correction is a reliable approach to learn from noisy labels for a lower value of noise rate. However, one can definitely observe that F-correction lacks robustness when the noise rate is increased. On the other hand, it is evident that DCT is more robust against the increase in noise percentages in comparison to the baseline methods; thereby reinforcing the choice of our algorithmic design.

3.4.2 Analysis on CUB200-2011 and CARS196

Details of the two datasets can be found in section 2.4.1.

CUB200-2011 The results are reported in Table 4.3. The baseline model trained with the conventional cross-entropy classification loss achieves a test accuracy of 63.78% for the symmetric noise with $\epsilon = 20\%$. An increase of 8.56% is observed over the cross-entropy baseline for the Co-Teaching algorithm. Morever, DCT outperforms all the baseline methods and achieves an accuracy of 74.57% for the same noise setting. Further increase in the ϵ to 50% evidently results in the decrease of the classification accuracy, however DCT still outperforms the rest by a significant margin. These results clearly demonstrate the effectiveness and robustness of DCT for large scale fine-grained image recognition dataset.

CARS196 As seen by the results obtained in Table 4.3, it is observed that our proposed DCT algorithm outperform both the Cross-Entropy baseline by a significant margin in terms of accuracy on the test set. It is however also observed that the performance gain over Co-Teaching is not substantial for CARS196 dataset in comparison to the performance gain obtained in CUBS200-2011. One plausible

explanation that can be attributed to this trend is that CARS196 is a difficult dataset to train in comparison to CUBS200-2011.

According to the results shown in Table 4.2 and 4.3, we verify the effectiveness and robustness of our DCT method, irrespective of the properties of the noise present in the datasets.

Note: One of the key aspects of the fine-grained image recognition datasets is that the inter-class variance is low and intra-class variance is high, and therefore are more vulnerable to noise. From the results obtained in Table 4.3, it is noted that the performance of Cross-Entropy baseline is substantially inferior against Co-Teaching and DCT. This observation clearly demands the need of two (or more) feature extractors in order to learn discriminative features in presence of noise with the fine-grained dataset.

3.4.3 Ablation

In this section, we perform extensive ablation studies regarding the various design choices that have been considered for DCT.

Importance of the selection strategy

As mentioned in § 3.3.2, L_2 in Eqn. 5.5 is not influenced by the selection strategy mentioned in § 4.4.1 as the entire mini-batch of is used to calculate L_2. In order to obtain more insights into the importance of the Diversity discrepancy module in the overall learning framework of DCT, we employ the selection strategy based on the ranking of the cross entropy loss to choose the samples for maximizing the divergence between the networks. In other words, we prune the noisy samples for the networks and attempt to increase the difference between the two networks [2]. We refer to this setting as *DCT-clean*. Similar to Eqn. (5.5), this operation is shown as follows:

$$\begin{aligned} & \mathrm{L}_2 = \mathrm{MMD}(\widehat{\boldsymbol{A}}_i, \widehat{\boldsymbol{B}}_i) \\ s.t. \quad & \widehat{\boldsymbol{A}}_i = f_{\theta(1:l)}(\boldsymbol{X}_i) \quad \forall \boldsymbol{X}_i \in \mathcal{D}_f \\ & \widehat{\boldsymbol{B}}_i = g_{\widehat{\theta}(1:l)}(\boldsymbol{X}_i) \quad \forall \boldsymbol{X}_i \in \mathcal{D}_g, \end{aligned} \tag{3.8}$$

The results are shown in Table 3.4. As observed, one can obtain better performance without such pruning of noisy labels, thereby successfully demonstrating the need of noisy labels to learn diverse features. One plausible explanation for this observation is that noisy labels perturb the overall decision boundary learnt by the two networks, thereby leading to optimal solution.

[2] It is to be noted that the clean labels of one network is chosen by the other network and vice-versa.

Table 3.5: Study of the importance of the two discrepancy modules used in DCT. w denote either of the feature extraction networks *i.e.*, f and g. (please refer to § 4.5.3 for more details.) The average accuracy (%) is reported after 5 different runs of DCT.

noise	**Dataset**	Loss_w^D	Loss_w^C	**DCT**
symmetric-50%	MNIST	93.14	91.35	**94.21**
symmetric-20%	MNIST	97.89	97.04	**98.54**
symmetric-50%	CIFAR10	77.11	74.35	**78.50**
symmetric-20%	CIFAR10	84.08	82.42	**85.41**
symmetric-50%	CIFAR100	41.89	41.29	**42.11**
symmetric-20%	CIFAR100	55.48	54.13	**56.11**

Importance of the Discrepancy Modules

In this section, we evaluate the importance of the two discrepancy modules, namely Diversity loss (*i.e.*, Eqn. (5.5)) and Consistency loss (*i.e.*, Eqn. (4.4)) in the DCT learning framework. We train the networks with the following loss functions

$$\text{Loss}_w^D = \text{L}_1^w + \text{L}_3 \quad \forall \; w = f, g \tag{3.9}$$

$$\text{Loss}_w^C = \text{L}_1^w - \text{L}_2 \quad \forall \; w = f, g, \tag{3.10}$$

for the former and the later case respectively. The results are shown in Table 4.8. It is clearly observed that without use of diversity loss, the performance of DCT drops significantly. Surprisingly, with the removal of the consistency loss, no such drastic drop in the performance is observed. This clearly indicates the need of the Diversity loss module in DCT to learn distinct and discriminative features.

Importance of the Position of the Diversity Module

In this section, we study the effect of the position l in calculating the Diversity loss (please refer to Eqn. 5.5). In the original DCT algorithm, we have added the discrepancy module after the 5^{th} layer of the CNN. As an additional experiment, we also study the effect of fixing the discrepancy module after the 7^{th} layer. The results are shown in Table 3.6. As observed, fixing the discrepancy module after the 5^{th} layer leads to better results in comparison to the 7^{th}, although the difference in performance is not significant.

Robustness of DCT

In this Section, we study the robustness of our DCT design with respect to its parameters, namely λ_2 and λ_3 (see Eq. (4.12), below). We will first report the value of the hyper parameters used in our experiments. Then we will analyze and investigate the effect of the hyper-parameters over the performance of the DCT algorithm.

Table 3.6: Study of position l for the Diversity loss module in the DCT framework. (please refer to § 3.4.3 for more details.) The average accuracy (%) is reported after 5 different runs of DCT.

noise	**Dataset**	**DCT**(7^{th})	**DCT**(5^{th})
symmetric-50%	MNIST	93.27	94.21
symmetric-20%	MNIST	97.49	98.54
symmetric-50%	CIFAR10	77.69	78.50
symmetric-20%	CIFAR10	84.67	85.41
symmetric-50%	CIFAR100	41.90	42.11
symmetric-20%	CIFAR100	55.72	56.11

Table 3.7: The accuracy (%) of DCT for various setups (using CIFAR10 dataset contaminated by symmetric noise with rate of 50%.)

λ_2	λ_3	**DCT(%)**	**Descriptions**
0	0	74.02	Baseline
0	0.0001	74.35	Consistency loss only
0.001	0	77.11	Diversity loss only
0.001	0.00001	77.88	parameter search
0.001	0.0005	78.32	parameter search
0.001	0.001	77.79	parameter search
0.01	0.0001	77.94	parameter search
0.005	0.0001	78.24	parameter search
0.0001	0.0001	77.23	parameter search
0.001	0.0001	**78.50**	Optimal case

Hyper-Parameters

We recall that the loss of the DCT algorithm as:

$$\text{Loss} = \text{L}_1 + \lambda_3 \text{L}_3 - \lambda_2 \text{L}_2. \tag{3.11}$$

Here, L_1 is the classification loss (for each network), L_2 and L_3 are diversity and consistency losses, respectively. Furthermore, λ_2 and λ_3 denote the associated weights of the diversity and consistency loss, respectively. In all of our experiments, we set $\lambda_2 = 1e-3$. We set $\lambda_3 = 1e-3$ for MNIST, CUB200-2011 and CARS196. For CIFAR10 and CIFAR100, we set $\lambda_3 = 1e-4$. In all our experiments, we used a Gaussian kernel for MMD with $\sigma = 0.05$.

We stress that the hyper-parameters reported above are used across all noise settings.

Robustness of the DCT algorithm

In this part, we analyze the robustness of the DCT algorithm with respect to its parameters. By doing so, we evaluate the performance of the DCT algorithm on the CIFAR10 dataset by varying the values of λ_2 and λ_3. Table 3.7 shows the accuracy of the DCT algorithm for the 50% symmetric noise (which is a challenging setup) for various values of λ_2 and λ_3. First note that for $\lambda_2 = \lambda_3 = 0$, we recover the vanilla co-training framework with an accuracy of 74.02%. Setting $\lambda_3 = 1e-4$ and $\lambda_2 = 0$ results in adding only the consistency loss and a modest increase in the performance (0.33% to be exact). Interestingly, by just adding the diversity loss ($\lambda_2 = 1e-3$ and $\lambda_3 = 0$), the accuracy soars to 77.11%, a significant improvement over the vanilla co-training solution. This confirms the premise of our work, *i.e.*, the importance of diversity in co-training.

Another observation in favor of the DCT algorithm is its robustness to the variation of λ_3 and λ_2. For example, by fixing $\lambda_2 = 0.001$ and varying λ_3 in a wide range from $1e-5$ to $1e-3$, the accuracy varies in the range $[77.79\%, 78.50\%]$. Similar trends can be observed if we pick λ_2 reasonably. For example, with $\lambda_3 = 1e-4$, changing λ_2 from $1e-4$ to $1e-2$ results in accuracies in the range of $[77.23\%, 78.50\%]$.

We also want to emphasize that, as long as we incorporate the DCT module on top of the baseline algorithm, it becomes possible to achieve significantly improved performance with minimal additional effort. This demonstrates the ease with which enhancements can be integrated into our existing framework to boost effectiveness. Moreover, it highlights the inherent efficiency and robustness of our algorithm, as it remains capable of delivering better results even with slight modifications. These qualities underscore the practical applicability and reliability of our approach in various scenarios, making it a versatile and dependable solution.

3.5 Summary

In this chapter, we presented a novel yet effective method (*i.e.* Co-Tr with Discrepancy) for training deep neural networks in the presence of noise. Specifically, we equipped the Co-Tr framework with two different discrepancy modules, **(a)** Diversity and **(b)** Consistency. The former was aimed at enforcing discrepancy between the two feature extraction networks in the Co-Tr learning module, thereby learning distinct features; while the latter enforced the networks to learn similar class probability distributions. Our empirical evaluation across several datasets for different complex noise conditions clearly demonstrated the need of using such discrepancy modules in DCT. As an extension, the performance of different discrepancy modules other than MMD can be studied. Furthermore, noisy multi-label classification tasks is an area where similar approaches may prove successful. In the next chapter, we will expand the current method into a broader, more general version by employing a composite loss function that includes MMD. We also demonstrate that the co-training framework, enhanced with a discrepancy module,

can be effectively applied to a variety of tasks.

Chapter 4

Discrepant Collaborative Training using Sinkhorn Divergences

In the previous chapter, we explored using Maximum Mean Discrepancy (MMD) to encourage distinct feature learning for handling noisy labels. Building on that work, we now extend our approach by integrating Sinkhorn divergence into the co-training framework. This enhancement facilitates the learning of diverse representations, enabling the networks to capture better, complementary views. Moreover, our framework is generalized to serve not only in scenarios involving noisy labels but also in semi-supervised learning, broadening its applicability.

4.1 Introduction

In this chapter, we address the issue of learning diverse yet distinct models in a general Co-Training [10] (Co-Tr) framework. More specifically, we equip the Co-Tr learning paradigm with a novel discrepancy module that results in learning different yet complementary views of the input; thereby enhancing the overall discriminative ability of the entire Co-Tr module.

Designing diverse models is a long-standing problem in machine learning despite several breakthroughs [11, 2, 106]. The prime example is the boosting algorithm and its variants, where a number of different, weak classifiers are learned sequentially. However, in the era of deep learning, the capacity of boosting algorithms to produce strong classifiers have been vastly superseded, and more appropriate alternative methods to enforce diversity in a deep network need to be investigated.

To address the aforementioned need, what started as a simple learning of two conditionally independent views of classifying web-data[10], has now been employed across a wide variety of tasks such as image classification [48, 115], text classification [106], email classification [69], and natural language processing [135] *etc*.. In our preliminary study, we have employed Co-Tr framework with MMD to learn from noisy labels. Our work proved that improved Co-Tr could learn better from noisy labels. Co-Tr has also been used in a semi-supervised setting [115],

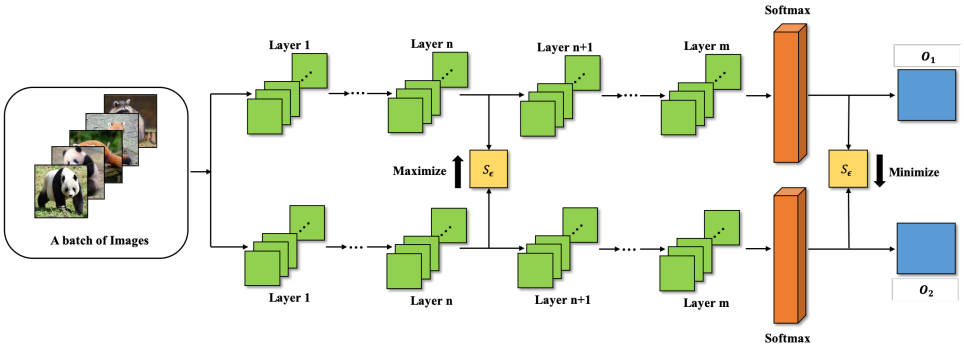


Figure 4.1: Deep Co-Training with Discrepancy schematic. Propagation of Discrepant Collaborative Training. A batch of images are fed into homogeneous networks independently. The first $\mathcal{S}_\epsilon$ module (orange box on left) is placed after *Layer n* between two networks. The second $\mathcal{S}_\epsilon$ (orange box on right) module is placed after *softmax* layer. We maximize the first $\mathcal{S}_\epsilon$ module to learn diverse features in each of the network, while the second $\mathcal{S}_\epsilon$ is minimized so as to learn the same class distributions.

We stress that the diversity is a requirement for the success of the Co-Tr framework [6]. The predominant techniques to induce diversity in Co-Tr are **(a)** use of different network architectures [164], **(b)** random initialization schemes [131] in each of the individual networks, and **(c)** training each network with different sets of samples [48, 115]. While these tactics have achieved significant improvements in learning different but complementary views of the input, they suffer from a few setbacks. First, two or more networks with different architectures must be highly compatible with each other in order to jointly learn different yet complementary features given the same input data. Moreover, random initialization of the networks does not guarantee that the learnt features will indeed be diverse, distinct and informative enough for the task at hand. As a remedy, the recent work of Qiao *et al*. utilized adversarial examples during training along with random initialization [115]. Nevertheless, one cannot guarantee the learning of complementary features as the distribution of the adversarial examples is very similar to the original training images. Interestingly, several recent works used identical homogeneous networks as the feature extractor units for co-training [48, 96]. Nonetheless, these methods do not explicitly enforce a discrepancy between the homogeneous networks and may thereby fail in capturing inherent discriminative features across the different views.

In this chapter, we address the above concern by explicitly enforcing a diversity constraint in the Co-Tr framework such that the underlying networks do learn different yet complementary views (or features) of the input even though they have the same architecture and different initialization. Both the two networks f and g have the same structure (Fig 4.1). They are fed with same batch of input data and are trained on the same task: image classification. However, they will

be updated by different losses. Our proposed module (yellow boxes in Fig 3.1) helps to keep the homogeneous networks different from each other and improve the image classification accuracy. O_1 and O_2 are prediction outputs from networks. For the test stage, we gather the outputs of network f and g for each test image. We then select the larger value (most confident) as the final prediction of the test image. More details are explained in Section 4.4. We apply $\mathcal{S}_\epsilon$ divergence [38] to measure diversity between networks. One can simply tune $\mathcal{S}_\epsilon$ into Maximum Mean Discrepancy (MMD) or Optimal Transport (OT) distance by changing ϵ. The $\mathcal{S}_\epsilon$ module can be seamlessly added to any part of a Co-Tr framework. This module drives the networks to robustly learn from different views but keeping the consistency of the outputs at same time. Figure 4.1 provides a brief schematic of our proposed methodology.

Our major contributions are as follows:

- An explicit method to disentangle learnt representations via an $\mathcal{S}_\epsilon$ based discrepancy module.
- An empirical study of the effectiveness of two well known variants of $\mathcal{S}_\epsilon$ divergence measures, including Maximum Mean Discrepancy and Wasserstein Distance.
- An extensive set of experiments demonstrating the advantage of such a $\mathcal{S}_\epsilon$ module across two different settings; *(i)* noisy labeled image classification, and *(ii)* semi-supervised image classification.

4.2 Extension of Previous Work

We extended the method in Chapter 3 to a more general module by applying more tasks, datasets and divergences. Our previous work can be considered as a special case of our proposed work here. Our theoretical contribution is to formulate the problem using Sinkhorn divergence. Sinkhorn divergence is a family of divergences where MMD and Wasserstein distance are considered as instances of this family. Our new work aims to cover more general notion of distance measures under the family of Sinkhorn divergences. As shown in Section 4.5, the new method outperforms previous work by a large margin. Moreover, we also show how our contribution can be used for semi-supervised learning.

1. We extend the specific MMD divergence to Sinkhorn divergence which is a family of divergences, which can be further tuned to MMD or Wasserstein distance by tuning the hyper parameter which is more general and flexible compared to our previous work. For applying our module to different tasks, one can always tries to achieve better performance by tuning the hyper parameter easily. And our experiment results shows that Sinkhorn Divergence provides better and robust performance than MMD divergence in most cases

which means one can always achieve competitive results without tuning hyper parameters.

2. More datasets. We tested our methods on more and lager datasets such as Clothing1M [92] and ImageNet [23]. In our previous work, we focused on more common datasets for noisy labels problems such as MNIST, CIFAR10, CIFAR100, CUB200 and CARS196. In this work we not only include all those datasets but also studied on large scale datasets Clothing1M and ImageNet, results are shown in Section 4.5.

3. Extension to semi-supervised learning task. By applying the module to semi-supervised task, we prove that our module is a general solution for the co-training framework. Since the co-training framework has been applied to numerous task (domain adaptation [14], image classification [115], data segmentation [12], tag-based image search [41] and many more), one could apply our module to any tasks using the co-training framework and expect improvements.

4. Experiments with different settings are provided. Here, we also considered extreme noise rates (such as 80%) on ImageNet. By setting the noise rate to 80%, one expects an extremely difficult condition to learn from labels. Even under this stringent noise condition, our proposed method outperformed the other baseline by a large margin.

4.3 Related Work

Discrepancy Measurement Our main objective in this work is to utilize discrepancy between the homogeneous networks. Hence, we need to measure the dissimilarity between two sets of samples, *i.e.*, the generated features of homogeneous networks. Here, this is achieved by statistical measures. For the design and development of several algorithms of machine learning and computer vision, comparing and matching probability distributions between two different domains is a fundamental building block [68, 42]. Several divergences such as Kullback-Leibler [75], Jensen-Shannon [97], *etc*. have successfully been used in learning similar/dissimilar distributions across various domains. They are appreciated for their computational simplicity, but they suffer from the major shortcoming of not metrizising weak-convergence [31]. It has been shown earlier that MMD [84, 50] is a highly nontrivial choice as well as the Optimal Transport (OT) distance which has been long known to be a powerful tool to compare probability distributions with non-overlapping supports. Both MMD and OT have the ability to metrizise weak-convergence, but they enjoy different characteristics. MMD can be efficiently and robustly estimated from a small number of samples of the measures, as a closed form solution exists for MMD. OT on the other hand, takes into account the underlying geometry while is computationally expensive. Thanks to [38], this cost is

largely mitigated by settling for cheaper approximations obtained through strongly convex regularizers, in particular entropy. [38] introduce the Sinkhorn loss $\mathcal{S}_\epsilon$ with a smoothing parameter ϵ. When $\epsilon \to 0$, $\mathcal{S}_\epsilon$ is reduced to a pure Wasserstein distance, and conversely when $\epsilon = +\infty$, it leads to MMD.

Learning from noisy datasets and Semi-supervised learning Learning from a clean dataset is not considered as a difficult task any longer. Recently, there has observed a growing surge in the interest of studying the robustness of any machine learning algorithm against noisy labeled images and unlabeled images. With regard to learning from noisy labeled images, MentorNet [63] trains an additional StudentNet network to select clean labels which is in-turn used to further guide the main training process. If a clean and unbiased validation set is not available, MentorNet will discover new data-driven sample-weight schemes from data which can be updated according to feedback from StudentNet. Ren *et al*. [116] follow a meta-learning paradigm and use a clean validation set to re-weight the training samples. Importance weights for training samples which result in the decrease of the loss in a clean validation set are increased, while the weights of those that result in the increase of the loss are decreased during the training process. One of the major drawbacks of [116] is the calculation of the clean validation set based importance weights after every gradient update of the network, which increases the time complexity of the overall algorithm. On the other hand, Decoupling [96] trains two different sub-networks with the examples that are confusing to both of them during the course of training[1]. Similarly, Co-Tr [48] trains two different networks with one selecting the clean examples, *i.e*. the examples with lower classification loss, for the other in an intertwined fashion. However, without any explicit discrepancy module between the networks that enforce the features learnt to be distinct and different, the solution learnt by the two aforementioned algorithm is not optimal. Motivation of semi-supervised learning is similar to learning from noisy labels. Regard of semi-supervised learning tasks, [121] presented the mutual-exclusivity loss and [101] presented the entropy minimization which are applied in one of the most famous semi-supervised learning methods – self-training technique.

4.4 Methodology

Here, we present our proposed methodology, *i.e*., Discrepant Collaborative Training (DCT+). We formulate DCT* as a cohort of two homogeneous networks f and g with their learnable parameters represented as θ and $\widehat{\theta}$ respectively (see Fig. 3.1 for more details.). Our total loss is contributed by three parts which are: L_M, L_D and L_C. L_M is the basic loss function for different tasks (for example, cross entropy loss for image classification task). L_D is discrepancy loss to increase diversity. L_C is the consistency loss which keeps consistency between networks.

[1]Both the networks produce different predictions with high confidence.

We now define our total loss function as follows:

$$\text{Loss}_f = \text{L}_M^f - \lambda_D \text{L}_D + \lambda_C \text{L}_C \tag{4.1}$$

$$\text{Loss}_g = \text{L}_M^g - \lambda_D \text{L}_D + \lambda_C \text{L}_C \tag{4.2}$$

λ_D and λ_C are the combination weights for the diversity and the consistency loss (*i.e.* L_D and L_C) respectively. Eqn 4.1 and 4.2 are used to update f and g using stochastic gradient decent optimizer.

The first part of the loss function, L_M, is different for each specific network. For example, if we are considering a supervised learning task, L_M will be a conventional supervised loss that trains the network to predict the correct labels for the training instances. To increase the diversity between f and g, we apply our $\mathcal{S}_\epsilon$ in the middle of our two networks (see Fig.3.1) which forms the second part of the total loss which we call discrepancy loss.

$$\text{L}_D = \mathcal{S}_\epsilon(\boldsymbol{A}_i, \boldsymbol{B}_i) \tag{4.3}$$

where $\boldsymbol{A}_i = f_{\theta(1:l)}(\boldsymbol{X}_i)$ and $\boldsymbol{B}_i = g_{\widehat{\theta}(1:l)}(\boldsymbol{X}_i)$ for the ith input $\boldsymbol{X}_i$, l denotes the layer where the discrepant $\mathcal{S}_\epsilon$ module is inserted, and $\theta(1:l)$ and $\widehat{\theta}(1:l)$ represent the parameters of the two networks f and g up until layer l. It is to be noted that L_D is calculated irrespective of the presence or absence of noisy labels within the mini-batch of images. Even though we want both f and g to learn diverse and distinct features, the final class probability distribution learnt by f and g should not be very different from each other. Thus, we use another S_ϵ to explicitly reduce the discrepancy between $\boldsymbol{z}_i^f$ and $\boldsymbol{z}_i^g$ for every $\boldsymbol{X}_i$ as shown below:

$$\text{L}_C = \mathcal{S}_\epsilon(\boldsymbol{z}_i^f, \boldsymbol{z}_i^g), \tag{4.4}$$

where $\boldsymbol{z}_i^f = f_\theta(\boldsymbol{X}_i)$ and $\boldsymbol{z}_i^g = g_{\widehat{\theta}}(\boldsymbol{X}_i)$ [2].

4.4.1 Extension to Learning from Noisy Labels

Settings We refer to the settings of [48] for the task of learning from noisy labels. The input for f and g is always the same mini-batch of training/testing set. Labels of all samples in mini-batch are randomly corrupted based on noise rate. The outputs of f and g for each sample are predictions of the class (each network will predict one most possible class where the sample is from). During the test step, we gather the outputs of f and g for each sample. For example, if the dataset is CIFAR10, final outputs of f and g are both $1 * 1$ for each test image. We select the larger value as the final prediction of the test image.

Now, we show how we introduce our DCT+ design to the Co-Tr framework to deal with noisy labels Fig.3.1. As elaborated in the previous section, we employ the

[2] Usually it is the output of the softmax layer for each of f and g.

total loss defined in Eqn. 4.1 and 4.2. Similar to [63, 48, 50], networks will select possible clean examples based on classification loss. More specifically, classification loss for $\boldsymbol{X}_i$ for f and g are shown below:

$$\begin{aligned} \mathrm{L}_f(\boldsymbol{X}_i) &= -\log\left(\frac{\exp(\boldsymbol{z}_i^f)}{\sum_1^m \exp(\boldsymbol{z}_j^f)}\right) \\ \mathrm{L}_g(\boldsymbol{X}_i) &= -\log\left(\frac{\exp(\boldsymbol{z}_i^g)}{\sum_1^m \exp(\boldsymbol{z}_j^g)}\right) \quad , \end{aligned} \tag{4.5}$$

where $\boldsymbol{z}_i^f = f_\theta(\boldsymbol{X}_i)$ and $\boldsymbol{z}_i^g = g_{\widehat{\theta}}(\boldsymbol{X}_i)$ [3]. Similar to [48], within each mini-batch examples which produce the $\mathbf{R}$ lowest L_g and L_f will be selected for f and g separately. The loss to update θ_f is given as

$$\mathrm{L}_M^f = \sum_{i=1}^{R} \mathrm{L}_f(\boldsymbol{X}_i) \quad \forall \boldsymbol{X}_i \in \mathcal{D}_g \quad , \tag{4.6}$$

where $\mathcal{D}_g$ represents a set of images that results in the R lowest L_g calculated in Eqn 4.5. Similarly, we calculate the loss to update θ_g as

$$\mathrm{L}_M^g = \sum_{i=1}^{R} \mathrm{L}_g(\boldsymbol{X}_i) \quad \forall \boldsymbol{X}_i \in \mathcal{D}_f \quad , \tag{4.7}$$

where $\mathcal{D}_f$ represents a set of images that results in the R lowest L_f as calculated in Eqn 4.5. L_D and L_C are the same as discussed in the previous section. Algorithm 1 provides the pseudo code of our proposed DCT+ algorithm for the noisy label task.

4.4.2 Extension to Semi-Supervised Learning

Settings We choose [115] as our baseline for the task of semi-supervised learning. Here, we denote $\mathcal{D} = \mathcal{S} \bigcup \mathcal{U}$ as our dataset where images in $\mathcal{S}$ are labeled and images in $\mathcal{S}$ are not. For the training of semi-supervised task, both networks have access to $\mathcal{S}$ and $\mathcal{U}$. We feed two networks with same mini-batch which contains both labeled and unlabeled samples. The network is trained to make similar predictions on training samples (no matter labeled or not) and make different predictions on adversarial samples. We simply add the divergence modules to the co-training framework to force the networks to extract different features but similar softmax outputs on training samples.

We also define f_{fc} and g_{fc} as the final fully-connected layer that classify the softmax outputs $\boldsymbol{z_f}$ and $\boldsymbol{z_g}$ to one of the categories in $\mathcal{S}$. L_1 is the semi supervised learning loss [115] defined as:

$$\mathrm{L}_{\mathrm{M}}^{\mathrm{f}} = \mathrm{L}_{\mathrm{M}}^{\mathrm{g}} = \mathrm{L}_{\mathrm{sup}} + \mathrm{L}_{\mathrm{cot}} + \mathrm{L}_{\mathrm{diff}}. \tag{4.8}$$

[3] Usually it is the output of the softmax layer for each of f and g.

For the above equation, we define:

$$\mathrm{L}_{\mathrm{sup}}(\boldsymbol{X}, y) = \mathrm{H}\Big(y, f_{fc}(p_f(\boldsymbol{X}))\Big) + \mathrm{H}\Big(y, g_{fc}(p_g(\boldsymbol{X}))\Big), \tag{4.9}$$

$p_f(\boldsymbol{x}) = \boldsymbol{z_f}$ and $p_g(\boldsymbol{x}) = \boldsymbol{z_g}$ (see Fig.4.1), for any data $(\boldsymbol{X}, y)$ in $\mathcal{S}$ where y is the label for $\boldsymbol{X}$ and $\mathrm{H}(\boldsymbol{m}, \boldsymbol{n})$ is the cross entropy between distribution $\boldsymbol{m}$ and $\boldsymbol{n}$.

$$\mathrm{L}_{\mathrm{cot}}(x) = \mathrm{H}\Big(\frac{1}{2}(p_f(\boldsymbol{X}) + p_g(\boldsymbol{X}))\Big) - \frac{1}{2}\mathrm{H}\Big(p_f(\boldsymbol{X}) + p_g(\boldsymbol{X})\Big), \tag{4.10}$$

where $x \in \mathcal{U}$ and $\mathrm{H}(\boldsymbol{m})$ is the entropy of $\boldsymbol{m}$. $\mathrm{L}_{\mathrm{cot}}$ is the Jensen-Shannon divergence between $f_{fc}(p_f(\boldsymbol{X}))$ and $g_{fc}(p_g(\boldsymbol{X}))$.

We then generate a set of adversarial examples [42] $\mathcal{D}' = \{\mathrm{h}(\boldsymbol{X}) | \boldsymbol{X} \in \mathcal{D}\}$. We employ several constraints for the adversarial examples: **(i)** $f_{fc}(p_f(\mathrm{h}(\boldsymbol{X}))) \neq g_{fc}(p_g(\mathrm{h}(\boldsymbol{X})))$, **(ii)** $p_f(\mathrm{h}(\boldsymbol{X})) = p_f(\boldsymbol{X})$ but $p_g(\mathrm{h}(\boldsymbol{X})) \neq p_g(\boldsymbol{X})$, which implies that $\mathrm{h}(\boldsymbol{X})$ is an adversarial example of g that fools network g but not network h (and vice versa), and **(iii)** $\mathrm{h}(\boldsymbol{X}) - \boldsymbol{X}$ is small. $\mathrm{L}_{\mathrm{diff}}$ is finally defined as:

$$\mathrm{L}_{\mathrm{diff}}(x) = \mathrm{H}\Big(p_f(\boldsymbol{X}), p_g(\mathrm{h}(\boldsymbol{X}))\Big) - \mathrm{H}\Big(p_f(\mathrm{h}(\boldsymbol{X})), p_g(\boldsymbol{X})\Big). \tag{4.11}$$

Similarly, L_D and L_C will be the same discussed in the previous section. Algorithm 2 provides the pseudo code of our proposed DCT+ algorithm for the task of semi-supervised learning.

Algorithm 1 Discrepant Collaborative Training on Learning from Noisy labels referred to [48]

Data: θ and $\hat{\theta}$ (parameters of network f and g), learning rate η, epoch number T, l the layer number for $\mathcal{S}_\epsilon$.

Algorithm `DCT+ with noisy labels`

for $t = 1, \ldots, T$ **do**
  Shuffle training set $\mathcal{D}$;
  Fetch mini-batch $\bar{\mathcal{D}}$ from $\mathcal{D}$;
  Obtain $\mathcal{D}_f = \arg\min_{\bar{\mathcal{D}}} l(f, \bar{\mathcal{D}})$; sample R(T) instances with small cross entropy loss
  Obtain $\mathcal{D}_g = \arg\min_{\bar{\mathcal{D}}} l(g, \bar{\mathcal{D}})$; sample R(T) instances with small cross entropy loss
  Update $\theta = \theta - \eta \Big[\lambda_M \nabla_\theta \mathrm{L}_{\mathrm{M}}^{\mathrm{f}}(\theta, \mathcal{D}_g) - \lambda_D \nabla_\theta \mathcal{S}_\epsilon(f_{\theta(1:l)}(\bar{D}), g_{\hat{\theta}(1:l)}(\bar{D})) + \lambda_C \nabla_\theta \mathcal{S}_\epsilon(f_\theta(\bar{D}), g_{\hat{\theta}}(\bar{D}))\Big]$;
  Update $\hat{\theta} = \hat{\theta} - \eta \Big[\lambda_M \nabla_{\hat{\theta}} \mathrm{L}_{\mathrm{M}}^{\mathrm{g}}(\hat{\theta}, \mathcal{D}_f) - \lambda_D \nabla_{\hat{\theta}} \mathcal{S}_\epsilon(f_{\theta(1:l)}(\bar{D}), g_{\hat{\theta}(1:l)}(\bar{D})) + \lambda_C \nabla_{\hat{\theta}} \mathcal{S}_\epsilon(f_\theta(\bar{D}), g_{\hat{\theta}}(\bar{D}))\Big]$;
end

Algorithm 2 Discrepant Collaborative Training on Semi-Supervised Learning [115]

Data: θ and $\hat{\theta}$ (parameters of network f and g), learning rate η, epoch number T, l the layer number for $\mathcal{S}_\epsilon$.

Algorithm `DCT+ for semi-supervised learning`

for $t = 1, \ldots, T$ **do**

Shuffle training set $\mathcal{D}$;

Fetch mini-batch $\bar{\mathcal{S}}$ from $\mathcal{S}$;mini-batch $\bar{\mathcal{U}}$ from $\mathcal{U}$;$\bar{\mathcal{D}} = \bar{\mathcal{S}} \bigcup \bar{\mathcal{U}}$;

Create Adversarial Examples Compute the adversarial examples $h_1(x)$ and $h_2(x)$ for $x \in \bar{\mathcal{D}}$ using FGSM [43]

Calculate $\mathrm{L_M}$ $\mathrm{L^f_M} = \mathrm{L^g_M}$ (see Eq.4.8);

Update $\theta = \theta - \eta\Big[\lambda_M \nabla_\theta \mathrm{L^f_M}(\theta, \bar{D}) - \lambda_D \nabla_\theta \mathcal{S}_\epsilon(f_{\theta(1:l)}(\bar{D}), g_{\hat{\theta}(1:l)}(\bar{D})) + \lambda_C \nabla_\theta \mathcal{S}_\epsilon(f_\theta(\bar{D}), g_{\hat{\theta}}(\bar{D}))\Big]$;

Update $\hat{\theta} = \hat{\theta} - \eta\Big[\lambda_M \nabla_{\hat{\theta}} \mathrm{L^g_M}(\hat{\theta}, \bar{D}) - \lambda_D \nabla_{\hat{\theta}} \mathcal{S}_\epsilon(f_{\theta(1:l)}(\bar{D}), g_{\hat{\theta}(1:l)}(\bar{D})) + \lambda_C \nabla_{\hat{\theta}} \mathcal{S}_\epsilon(f_\theta(\bar{D}), g_{\hat{\theta}}(\bar{D}))\Big]$;

end

4.5 Experiments

Dataset. We verify the effectiveness of our approach on six benchmark datasets: MNIST [80], CIFAR10 [73], CIFAR100 [73], SVHN [105], CUB200-2011 [146], CARS196 [72], Clothing1M [23] on different tasks. More information about the datasets can be found in section 2.4.1.

Setup. Table 4.1 shows the CNN architecture used for experiments on MNIST, CIFAR10, CIFAR100 and SVHN. For experiments in the weakly-supervised learning setup, we follow the well-acknowledged standard settings of "Temporal Ensembling" [77] and "Virtual Adversarial Training" [100]. Further, we use Adam optimizer with details of hyper-parameters such as learning rate and weight decay are provided in abalation study. We choose [48] as our baseline for the Noisy Label task and we follow their detailed settings. For the task of Semi-Supervised learning, we follow the settings of [115]. For experiments on two fine-grained image recognition datasets (*i.e*. CUB200-2011 and CARS196), we use the Inception-V1 [136] architecture, pretrained on Imagenet [22]. Here, we use RMSProp and the Adam optimizer for the CUB200-2011 [146] and CARS196 [72] datasets respectively. The initial learning rate for both optimizers is set to 0.0001.

Note: We report the optimal value of the hyper-parameters λ_2 and λ_3, batch size, and number of epochs used in DCT+ for all the datasets in abalation study. The method trains two networks simultaneously and out of the two predictions, we select the prediction that has the higher confidence.

Table 4.1: Network Structure trained by MNIST, CIFAR10, CIFAR100 and SVHN. The slope of all LReLU functions in the network are set to 0.01. Number of training classes for the dataset is K.

#Layer	Input Image
1	3×3 conv, 128 LReLU
2	3×3 conv, 128 LReLU
3	3×3 conv, 128 LReLU
	2×2 max-pool, stride 2
	dropout, p = 0.25
4	3×3 conv, 256 LReLU
5	3×3 conv, 256 LReLU
6	3×3 conv, 256 LReLU
	2×2 max-pool, stride 2
	dropout, p = 0.25
7	3×3 conv, 512 LReLU
8	3×3 conv, 256 LReLU
9	3×3 conv, 128 LReLU
	avg-pool
10	fc-layer $128 \rightarrow K$
	softmax

4.5.1 Learning from Noisy Labels

Noise Transition. Here we test our proposed algorithm on the task of noisy-supervised image classification. More specifically and following the protocols of [48, 112], we manually corrupt all the clean datasets by a noise transition matrix $\boldsymbol{Q} \in \mathbb{R}^{K \times K}$, where $\boldsymbol{Q}_{ij} = Pr(\tilde{\boldsymbol{y}} = j | \boldsymbol{y} = i)$ gives the probability that noisy label (*i.e.* $\tilde{y}$) is flipped to a value j from clean label (*i.e.* y) value of i. In this chapter, we set up two different noise transition situations: (1) **Pair** flipping, and (2) **Symmetric** flipping. Exemplar noise transition matrix for each of the pair and symmetric flipping is shown in Fig. 2.6(a) and 2.6(b), where the value of noise rate ζ is set 45% and 50% respectively.

In our experiments, we run and test our proposed DCT+ in three different noise conditions, which are specified as follows: (**1**) 45% pair flip noise; (**2**) 50% symmetric flip noise; (**3**) 20% symmetric flip noise. We choose high noise rate values, *i.e.*, 45% and 50%, as this chapter mainly focuses on the robustness of DCT+ and the value of working with noisy data. However one should definitely set the value ϵ lower than 50% for pair flip noise, otherwise the neural networks will need additional information to learn discriminative features. We further test on 20% symmetric flip noise so as to verify the robustness of our proposed DCT+ algorithm under lower noise conditions.

Baseline Algorithms. For MNIST, CIFAR10 and CIFAR100, we compare

Table 4.2: Comparison between our proposed DCT+ algorithm with popular baseline algorithms. We report the average accuracy in (%) from 5 runs of DCT+. F-C stands for F-correction, M-Net stands for MentorNet, and Co-T stands for Co-Teaching.

noise	Dataset	F-C [112]	Decoupling [96]	M-Net [63]	Co-T [48]	DCT+($\epsilon = +\infty$)	DCT+($\epsilon = 0$)
pairflip-45%	MNIST	0.24	58.03	80.88	87.63	88.54	**90.85**
symmetric-50%	MNIST	79.61	81.15	90.05	91.32	94.21	**95.85**
symmetric-20%	MNIST	98.82	95.70	96.70	97.25	98.54	**99.07**
pairflip-45%	CIFAR10	6.61	48.80	58.14	72.62	72.91	**74.34**
symmetric-50%	CIFAR10	59.83	51.49	71.10	74.02	78.50	**80.4**
symmetric-20%	CIFAR10	84.55	80.44	80.76	82.32	85.41	**87.2**
pairflip-45%	CIFAR100	1.60	26.05	31.60	34.811	35.33	**36.00**
symmetric-50%	CIFAR100	41.04	25.80	39.00	41.37	42.11	**43.4**
symmetric-20%	CIFAR100	**61.87**	44.52	52.13	54.23	56.11	57.8

our results against the following baseline methods: **(i)** F-correction [112], which makes use of a noise transition matrix to correct the softmax predictions; **(ii)** Decoupling [96], which select those samples where the underlying networks are highly non-confident of their predictions and use them to update the parameters; **(iii)** MentorNet [63], where noisy instances for the student networks are filtered out by a pre-trained additional teacher network to learn robustly under noisy labels. **(iv)** Co-Teaching [48], which trains two identical homogeneous networks where one selects the best possible clean instances for the other and vice-versa. Results of the above baselines are reported from [48]. Furthermore, we consider two baseline models trained with **(a)** conventional cross-entropy loss and **(b)** Co-Teaching algorithms so as to verify the robust performance of our proposed DCT+ method on CUB200-2011 and CARS196 datasets.

Results Analysis and Discussion

Here, we provide a detailed insight regarding the performance of DCT+ and compare it against the several baseline algorithms mentioned before across the different datasets for different settings of noise.

MNIST. As observed in Table 4.2, all baseline methods obtain competitive results against each other in the case of a low noise rate (*i.e.* symmetric 20%). Our DCT+ method achieve a competitive 99.07% against the second best F-correction method (which achieves 98.82% accuracy on the test set). Further one can observe a drop in performance when the noise rate is increased to 50% for the F-correction and the Decoupling baseline methods. However both MentorNet and Co-Teaching are quite robust when dealing with large values of the noise rate. Further our proposed DCT+ algorithm obtains the state-of-the-art accuracy of 95.85%, thereby outperforming all the baselines and the current best algorithm (*i.e.* Co-Teaching) by 4.53%. A further increase in the complexity of the noise (*i.e.* pair-flip noise with $\zeta = 45\%$) results in a considerable drop in the classification accuracy for both F-correction and Decoupling methods. On the other hand, we again observe that DCT+ outperforms all the baselines by a significant margin. More specifically, we

Table 4.3: Comparison between our proposed DCT+ algorithm with baseline algorithms for CUB200-2011 and CARS196 in terms of accuracy on the test set (%). Co-T stands for Co-teaching [48].

noise	Dataset	Cross Entropy	Co-T [48]	DCT+($\epsilon = +\infty$)	DCT+($\epsilon = 0$)
symmetric-50%	CUB200-2011	40.80	54.64	59.24	**60.56**
symmetric-20%	CUB200-2011	63.78	72.34	74.57	**75.44**
symmetric-50%	CARS196	38.86	66.75	67.80	**69.15**
symmetric-20%	CARS196	71.76	86.00	86.62	**87.51**

outperform MentorNet by 9.97% in terms of accuracy on the test set. Moreover, we notice that when $\epsilon = 0$ ($\mathcal{S}_\epsilon$ becomes the Wasserstein Distance) the performance is improved somewhat compared to when $\epsilon = +\infty$ (the $\mathcal{S}_\epsilon$ loss becomes MMD). The same conclusion can be drawn from the experiments on the other datasets as well.

CIFAR10. From Table 4.2, one can observe that similar results (in terms of classification accuracy) is obtained by all the baseline methods for symmetric flip noise with 20%. Unlike for the MNIST dataset, on CIFAR10, DCT+ outperforms all the baselines including F-correction by 2.65%. With further increase in the level and the complexity of the noise, it is easily seen that DCT+ outperforms all the baseline algorithms, thus further validating the design choices incorporated within DCT+ to learn distinct and discriminative features.

CIFAR100. It is observed from Table 4.2 that for symmetric noise with 20% noise rate, DCT+ outperforms all the competitive baseline algorithms apart from F-correction. It is evident that F-correction is indeed a reliable method below a certain moderate level of noise corruption. However as the complexity of noise in increased, F-correction is no longer able to cope. On the other hand in complex noise settings, DCT+ is significantly more robust in comparison to the baseline methods, thereby reinforcing the choice of our algorithmic design.

Table 4.4: Comparison between our proposed DCT+ algorithm with popular baseline algorithms. We report the average accuracy (%) from 5 runs using DCT. CIFAR100+ means it is trained using data augmentation, otherwise not. S-T stands for Stochastic Transformations. T-E stands for Temporal Ensembling. D-C stands for Deep Co-Training. "-" means the original papers did not report the corresponding accuracy.

dataset	GAN[121]	S-T[121]	π Model[77]	T-E[77]	D-C[115]	DCT+($\epsilon = +\infty$)	DCT+($\epsilon = 0$)
CIFAR10	81.37	88.71	87.64	87.84	90.97	91.45	**91.89**
CIFAR100	-	-	56.57	-	61.23	61.77	**61.98**
CIFAR100+	-	-	-	-	65.37	65.84	**66.16**
SVHN	91.89	-	95.18	95.58	96.39	96.52	**96.78**

CUB200-2011 The results are reported in Table 4.3. In the 20% symmetric noise setting, the baseline model trained with the conventional *cross-entropy* classification loss achieves a test accuracy of 63.78%. Co-Teaching algorithm outperforms this baseline by 8.56%. Moreover, DCT+ outperforms all the baseline methods and achieves an accuracy of 74.57% and 75.44% with different values of noise rate for the same noise setting. A decrease in the classification accuracy is observed when

the noise rate is increased to 50%. Nonetheless, DCT+ still outperforms the rest by a significant margin. These results clearly demonstrate the effectiveness and robustness of DCT+ for large scale fine-grained image recognition tasks against different value of noise corruption percentages.

CARS196 It is observed in Table 4.3 that our proposed DCT+ algorithm outperforms both cross-entropy baselines by a significant margin in terms of accuracy on the test set for different values of 20% and 50%. However unlike the CUBS200-2011 dataset, the performance gain observed over Co-Teaching is not substantial for the CARS196 dataset. One plausible justification that can be attributed to this observation is that CARS196 is a difficult dataset to train on in comparison to CUBS200-2011.

In summary, according to the results shown in Table 4.2 and 4.3, we have successfully verified and demonstrated the effectiveness and robustness of our proposed DCT+ method, irrespective of the kind and the level of noise present in the datasets.

Note: Fine-grained image recognition datasets usually consist of labeled images which have low (and high) intra (and inter) class variances, thereby making it difficult to train models on such datasets as they are more vulnerable to noise. From the results obtained in Table 4.3, it is observed that both Co-Teaching and DCT+ significantly outperforms the cross-entropy baseline in terms of classification accuracy on the test set. This observation clearly demonstrates the need for two (or more) feature extractors in order to learn distinct and discriminative features from a fine-grained dataset in the presence of noisy labels.

Clothing1M

We use the noise corrupted version of the Clothing1M[4] to infer the noise rate by the clean validation set. The batch size is set to 64. Further, we report the classification accuracy after training for 51 epochs for all methods. As shown in Table 4.5, our DCT+ performs better than Co-Teaching [48] and JoCoR[150]; thus demonstrating the effectiveness of DCT+ over Co-Teaching and JoCoR for a large dataset.

Table 4.5: Comparison of our proposed CCT against two different baselines on Clothing1M dataset: Co-Teaching [48] and JoCoR [150]. Accuracy($\%$) is reported as the average of 5 runs.

method	**Co-Teaching**	**JoCoR**	**DCT+(OURS)**
accuracy	68.5	69.8	**70.1**

ImageNet

Jiang et al.[62] proposed to use mini-ImageNet and Stanford Cars to introduce both web and symmetric indistribution noise in a controlled manner with different noise ratios. We adopt the mini-ImageNet web noise dataset for evaluation in a real scenario with several ratios, which consists of 100 classes with 50K (5K) samples

[4]We have already used the noise corrupted version of the Clothing1M dataset in our experiments.

for training (validation). For fair comparison, we use a standard ResNet-18 [55] and follow the work of [110, 62].

Table 4.6: Comparison of our proposed CCT against two different baselines on Mini-Imagenet: Co-Teaching [48] and JoCoR [150]. Accuracy(%) is reported as the average of 5 runs.

noise rate	**20%**	**40%**	**80%**
Co-Teaching	54.1	45.3	37.0
JoCoR	57.2	49.1	40.4
DCT+(OURS)	58.4	50.8	42.1

Table 4.6 illustrates the superior performance of our DCT+ when training in the presence of web label noise in mini-ImageNet [92]. The results demonstrate that DCT are robust to web noise and the improvements are consistenct across noise levels.

4.5.2 Semi-supervised Learning

We also investigate the performance of our algorithm in the semi-supervised setting, in particular, on the SVHN, CIFAR10 and CIFAR100 datasets. Following [77], for SVHN, we only use $1,000$ images out of the available $73,257$ training images as the supervised set $\mathcal{S}$, the remaining $73,257-1,000$ images goes in to the unsupervised set $\mathcal{U}$. For CIFAR10, we only use $4,000$ images out of the $500,000$ available training images in the supervised set $\mathcal{S}$ and the remaining $46,000$ images form the unsupervised set $\mathcal{U}$. For CIFAR100, we use $10,000$ images out of the $50,000$ available training images in the supervised set $\mathcal{S}$ and the remaining $40,000$ images make up the unsupervised set $\mathcal{U}$. We use the full set of test images for evaluation across all datasets. For the baseline, we follow the work by [115]. We also compare our design with: GAN [122], Stochastic Transformations [121], $\prod$ Model [77], Temporal Ensembling [77], Mean Teacher [139] and Deep Co-training 2-views D-C, [115]. To support a fair comparison, we only use D-C-2-views as more views increase the number of network parameters.

Baselines Baseline methods are listed in Table 4.4. All of the baselines methods, as well as ours, require the evaluation of multiple models, meaning that even though DCT+ has two homogeneous networks, it is not at a computational disadvantage.

Discussion of Results From Table 4.4 it is observed that for CIFAR10, all baseline methods reach 80% accuracy on the test set. Our DCT+ outperforms other methods and achieves 91.89% which beats previous state-of-the-art, Deep-Co-Training [115], by 0.92%. The same conclusion can be drawn for CIFAR100, CIFAR100+ and SVHN where our DCT+ outperforms previous state-of-the-art by $0.39-0.79\%$. Also, $\mathcal{S}_{\epsilon=0}$ works better than $\mathcal{S}_{\epsilon=+\infty}$, which is consistent with our results on the tasks using noisy labels. According to the results in Table 4.4, Deep Co-Training [115] is indeed a very good method for semi-supervised learning.

Whenever we are injecting $\mathcal{S}_\epsilon$ into the Deep Co-Training framework, we see improvements. Moreover, these improvements do not come at the expense of adding more parameters to the network and, thanks to [38], the computational burden is affordable. Details of running times and ablation studies are provided in the next section.

4.5.3 Abalation

In this section, we provide details of our implementation and study the robustness of our DCT+ design with respect to its parameters, namely λ_2 and λ_3 (see Eq. (4.12), below). We will first report the value of the hyper parameters used in our experiments. Then we will analyze and investigate the effect of the hyper-parameters over the performance of the proposed DCT+ algorithm.

Hyper-Parameters

- For all experiments on MNIST, CIFAR10, CIFAR100 and SVHN, we set initial learning rate and weight decay as 0.001 and 0 respectively. The models are trained for 600 epochs (with a batch of size 128 at each iteration).
- For CUBS200-2011 and CARS196, we use RMSProp and Adam optimizer respectively. The initial learning rate for both optimizers is set to 0.0001. The models are trained for 80 and the batch size is set to 32.

Robustness of the DCT algorithm

We recall that the loss of the DCT+ algorithm as:

$$\text{Loss} = \text{L}_1 - \lambda_2 \text{L}_2 + \lambda_3 \text{L}_3. \tag{4.12}$$

Here, L_1 is the classification loss (for each network), L_2 and L_3 are diversity and consistency losses, respectively. Furthermore, λ_2 and λ_3 denote the associated weights of L_2 and L_3 respectively. Here, we analyze the robustness of the DCT+ algorithm with respect to its parameters. By doing so, we evaluate the performance of the DCT+ algorithm on the CUB200-2011 dataset by varying the values of λ_2 and λ_3. Table 4.7 shows the accuracy of the $\mathcal{S}_{\epsilon=0}$ (*i.e.*, 2-Wasserstein Distance) for the 50% symmetric noise (which is a challenging setup) for various values of λ_2 and λ_3. Some of the observations are as follows:

When $\lambda_2 = \lambda_3 = 0$, we recover the vanilla Co-Tr framework with an accuracy of 54.64%. Setting $\lambda_3 = 0.0001$ and $\lambda_2 = 0$ results in adding only the consistency loss and a modest increase in the performance (1.88% to be exact) is observed. Interestingly, by just adding the diversity loss ($\lambda_2 = 0.001$ and $\lambda_3 = 0$), the accuracy soars to 57.44%, a significant improvement over the vanilla Co-Tr. This confirms the premise of our work, *i.e.*, the importance of diversity in Co-Tr. For example, by fixing $\lambda_2 = 0.001$ and varying λ_3 in a range from 0.0005 to 0.005, the accuracy

varies in the range $[58.25\%, 60.56\%]$. Similar trends can be observed if we pick λ_2 reasonably. For example, with $\lambda_3 = 0.001$, changing λ_2 from 0.0001 to 0.005 results in accuracies in the range of $[58.22\%, 59.45\%]$.

Overall, we have observed a very robust performance with DCT+, outperforming vanilla Co-Tr framework consistently with ease. Here, we recommend a simple and general way to set λ_2 and λ_3 which is to set both of them as 0.001. However, one can fine tune them to achieve even better results and our suggestion is to set λ_2 a bit smaller than λ_3.

Table 4.7: The accuracy (%) of $\mathcal{S}_{\epsilon=0}$ (*i.e.* 2-Wasserstein Distance) for various setups (using CUB200-2011 dataset contaminated by symmetric noise with rate of 50%.)

λ_2	λ_3	**DCT+(%)**	**Descriptions**
0	0	54.64	Co-Teaching [48]
0.001	0	57.44	Diversity loss only
0	0.001	56.52	Consistency loss only
0.001	0.005	58.75	parameter search
0.001	0.0005	58.25	parameter search
0.005	0.001	58.22	parameter search
0.0005	0.001	59.45	parameter search
0.001	0.001	60.56	Optimal Case

Effect of ϵ

In this part, we study the importance of ϵ in our DCT+ algorithm for the CUB200-2011 dataset in the presence of 50% symmetric noisy labels. The results are shown in Table 4.8. It is clearly observed that with increasing the ϵ, the performance drops. As seen, the performance drops as ϵ is increased from 0 (*i.e.* 2-Wasserstein Distance) to $+\infty$ (MMD). Thereby $\mathcal{S}_{\epsilon=0}$ (*i.e.* 2-Wasserstein Distance) clearly outperforms $\mathcal{S}_{\epsilon=+\infty}$ (MMD).

Table 4.8: Study of the importance of ϵ for $\mathcal{S}_\epsilon$ for CUB200-2011 dataset in the presence of 50% symmetric noisy labels. The average accuracy (%) is reported after 5 different runs of DCT+. WD stands for Wasserstein Distance. We fix $\lambda_2 = \lambda_3 = 0.001$

Condition	$\epsilon = 0$ (2-WD)	$\epsilon = 1$	$\epsilon = 5$	$\epsilon = 1000$	$\epsilon = \infty$ (MMD)	Co-Teaching [48]
Accuracy	60.56	60.25	59.85	59.41	59.24	54.64

4.6 Summary

A novel and effective method (*i.e*. Co-Tr utilizing $\mathcal{S}_\epsilon$) for training deep neural networks was presented. It was demonstrated to be outperforming all other baseline methods in the vast majority of settings influenced by noisy labels, as well as in a semi-supervised setting. $\mathcal{S}_\epsilon$ divergence was used, in the middle of the network, to drive the networks to learn distinct features, while the $\mathcal{S}_\epsilon$ at the end enforces the networks to still learn similar class probability distributions. Multi-label classification tasks is another area with related challenges where similar approaches may prove successful. In the next chapter, we continue our exploration of improving the co-training framework for learning from noisy labels by investigating a fundamentally different network architecture: the transformer. We will discuss the advantages of replacing conventional CNNs with transformers, highlighting their unique learning properties and potential benefits in handling noisy data.

Chapter 5

Learning from Noisy Labels with Contrastive Co-Transformer

In the previous two chapters, we have discussed how to leverage the co-training framework with CNN architectures to effectively learn from noisy labels, leading to significant performance improvements. Building on these results, and given the demonstrated robustness of transformers in handling noisy data, this chapter explores the application of the co-training framework using transformer architectures to further enhance learning from noisy labels. By integrating the strengths of transformers with the co-training approach, we aim to achieve improved generalization and resilience against label noise. We propose a novel method called Contrastive Co-Transformer (CCTran) and demonstrate its ability to learn from image datasets with noisy labels.

5.1 Introduction

As a result, such corruption of the data leads to incorrect predictions by the deployed model. This highlights the necessity to deal with label noise in an ordered way.

Transformers have recently started to receive significant attention in the field of machine learning and computer vision research. The seminal work in the field of transformers was proposed in [144]. Even though the transformers were fundamentally designed to learn long range dependencies in natural language processing [66, 18, 78], they have recently been widely applied to learning discriminative features from images [91, 149, 13]. The vast amount of research on vision transformer and self-attention based models show the importance of an efficient architectural design that can learn discriminative, distinct, yet complementary visual representations[86, 132, 161, 149, 163]. As a result of introducing great flexibility in modeling long-range dependencies in vision tasks while introducing less inductive bias, vision transformers have already achieved performances competitive with their CNN counterparts [55, 137]. They can naturally process multi-modal input data including images, video, text, speech and point clouds. Previous studies [103]

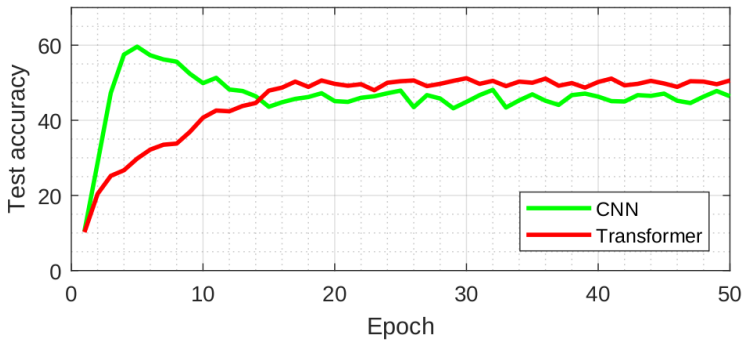


Figure 5.1: Test accuracy of plain CNN and transformer based architectures with 45% of pairflip noise on CIFAR10.

have shown that transformers are highly robust to severe occlusions. However, there is, to the best of our knowledge, no study discussing the robustness of transformers with respect to label noise.

Our work studies the robustness to label noise of transformers via experiments on different datasets and settings. We also propose a new Co-Tr method which introduces a contrastive loss module which improves the robustness of the transformer against label noise.

We begin by showing the importance of introducing transformers, over a standard CNN, as a backbone of Co-Teaching [48] with the help of a toy experiment. Co-Teaching [48] is a well-known method which uses two concurrent CNNs that share information of possibly clean samples during training. In our toy experiment, we utilise a 3-layer MLP in the case of a CNN based backbone, while using the transformer proposed in [53] in the transformer based version. To our surprise, we noticed (as shown in Fig. 5.1) that transformers are more resilient to noise and are able to reach superior performance in terms of accuracy compared to using an MLP as the backbone of Co-teaching. We set the noise rate of pairflip to 45% (Details of the experiment are attached in the supplementary material). We also observe in Fig. 5.1 that CNNs attain a very high accuracy on the test set during the initial epochs, while eventually over-fitting to the noise present in the data. On the other hand, and unlike CNNs, the learning curve of the transformer is slow but steady during the initial epochs, and eventually plateauing as the learning progresses whilst outperforming the CNNs in terms of accuracy on the test set.

Co-Tr is one of the well known algorithms aiming at automatically classifying samples as clean or noisy. Blum *et al*. [10] proposed the general Co-Tr learning framework where they successfully addressed the problem of website classification. Recently, Co-Tr has been applied extensively to the noisy label problem[48, 150, 159]. The main idea behind the Co-Tr algorithm is to simultaneously train two or more sister networks together to learn distinct yet complementary features for the task at hand, thus constraining the need to have two or more different aspects of the data. In our work, we apply transformers to Co-Tr framework and we call it

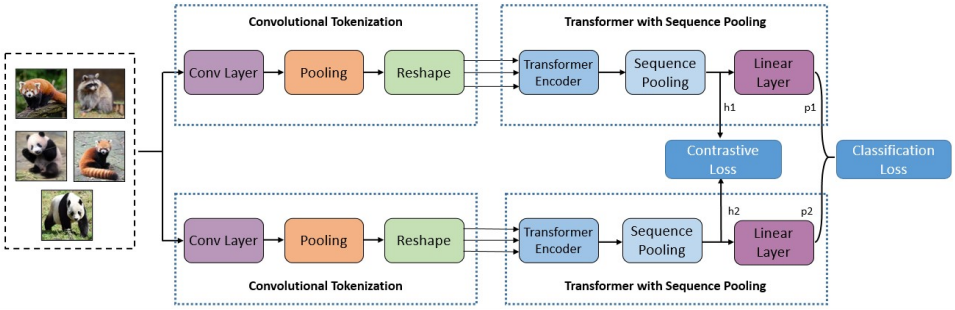


Figure 5.2: Schematic diagram of CCT. The noisy dataset is fed into two transformers in parallel. Features ($\boldsymbol{h}_1$ and $\boldsymbol{h}_2$) are extracted before the linear layers. Simultaneously, both $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ are passed through classifiers for calculating the classification loss ($\boldsymbol{p}_1$ and $\boldsymbol{p}_2$).

Co-Transformer.

In order to achieve better performance and robustness against the various degrees of noise present in the data, we also introduce the *contrastive loss* module to the Co-transformer framework. Contrastive learning on visual representations can be dated back to [46] and has recently been applied to unsupervised learning [15, 158, 24], which is formulated as the task of finding similar and dissimilar features for a machine learning model. It aims to pull similar examples closer to each other and push dissimilar ones further away which is also needed when learning distinct yet complementary features in a Co-Tr framework in the presence of noisy labels. Generally, for contrastive learning algorithms, data-augmentation techniques are used to create similar examples [15, 154, 4, 15, 165, 61, 57]. However, for our Co-Transformer we overcome such (in general dataset specific) augmentation techniques by using the outputs of the same input image by Co-transformers as two similar examples. After having extracted the features using two homogeneous transformers, we calculate the contrastive loss for every single example even if the examples have been labeled as clean or noisy by the underlying transformers based on its loss value[1]. As a result, and unlike [48, 150] which only make use of samples that are labeled as clean, we utilize all the samples within the mini batch in the back propagation of the gradients, resulting in stable and efficient learning. A diagrammatic schematic of our proposed approach is illustrated in Fig 5.2.

Our major contributions are:

- A novel method to improve the divergence between two homogeneous transformers in the Co-Training framework.
- A novel method to extract more information from a noisy mini batch compared to other state-of-the-art methods.

We have evaluated our proposed CCTran on six common image classification datasets MNIST [81], CIFAR10 [73], CIFAR100 [73], CUBS200-2011 [145],

[1]This selection of samples based on their loss value is also famously known as the "small-loss" principle. [140]

CARS196 [71] and Clothing1M [155]. Our empirical evaluation clearly demonstrates the effectiveness of CCT over the five state-of-the-art algorithms tested, namely Decoupling [96], F-correction [112], Mentor-Net [64], Co-Teaching [48], JoCor [150] and Jo-SRC [157] in terms of classification accuracies on the test set of these datasets.

5.2 Related Work

Contrastive Learning. More details about definition and current research trends of contrastive learning can be found in Chapter 2. (see introduction of contrastive learning). In our proposed CCT, the features extracted from two encoder networks are considered to be a positive pair. As a result, no additional data augmentation is needed, thereby saving the time and the resources needed to find the right augmentation strategy for every dataset.

Transformers More background information about transformers can be found in Chapter 2. The recent success of ViT [91, 53] have provided us with more reliable and excellent performance transformer backbones. [25] argued that Transformers lack some of the inductive biases inherent to CNNs, such as translation equivariance and locality, and therefore do not generalize well when trained on insufficient amounts of data. However, [53] has shown for the first time that with the right size and tokenization, transformers can perform head-to-head with state-of-the-art methods on small datasets, often with better accuracy and fewer parameters. In this work, we apply transformers from [53] for small datasets: MNIST, CIFAR10 and CIFAR100. For CU200-2011, CARS196 and Clothing1M we apply transformers from [91].

5.3 Methodology

As mentioned above, we aim to utilize all the examples in the noisy datasets and to introduce the contrastive learning algorithm within the Co-Transformer framework. Inspired by the recent success of contrastive learning and Co-training algorithms (as mentioned in the previous section), CCT takes advantages of the small-loss sample selection strategy to boost the diversity when training the two transformers by exchanging information between them. Moreover, samples with labels marked as clean or noisy by the transformers need not to be correct, therefore leading to further erroneous predictions when separating the original noisy samples from the clean ones. The features extracted from the two transformers are different from each other, and thus perfectly fit with the realm of contrastive learning. We therefore add a contrastive loss module in our proposed methodology which considers all the samples within the mini batch, giving a significant performance boost over the current state-of-the-art methods. Our algorithm is illustrated in Algorithm 3.

Algorithm 3 Contrastive Co-Transformer

Input: Network $f(\mathbf{\Theta}_1)$ and $f(\mathbf{\Theta}_2)$, $t(\mathbf{\Theta}_1)$ and $t(\mathbf{\Theta}_2)$, $g(\mathbf{\Theta}_1)$ and $g(\mathbf{\Theta}_2)$, learning rate η, constant λ, epoch number $\mathrm{T}_{\max}$ and max iteration number $\mathrm{I}_{\max}$, batch size K

For t=1,2,...,$\mathrm{T}_{\max}$
Shuffle training set $\mathcal{D}$;
For n=1,2,...,$\mathrm{I}_{\max}$ do
Fetch mini batch $\mathcal{D}_n$ from $\mathcal{D}$;
For k=1,2,...,K do
$\boldsymbol{h}_k^1 = f(\boldsymbol{x}_k, \mathbf{\Theta}_1)$
$\boldsymbol{h}_k^2 = f(\boldsymbol{x}_k, \mathbf{\Theta}_2)$
$\boldsymbol{z}_k^1 = g(\boldsymbol{h}_k^1, \mathbf{\Theta}_1)$
$\boldsymbol{z}_k^2 = g(\boldsymbol{h}_k^2, \mathbf{\Theta}_2)$
$\boldsymbol{p}_k^1 = t(\boldsymbol{h}_k^1, \mathbf{\Theta}_1)$
$\boldsymbol{p}_k^2 = t(\boldsymbol{h}_k^2, \mathbf{\Theta}_2)$
End For
Obtain R samples with smaller loss for $\mathcal{D}_n^1$ and $\mathcal{D}_n^2$. Selection strategy depends on different baselines.
Obtain L_{ce} on $\mathcal{D}_n^1$ and $\mathcal{D}_n^2$ according to 2.10, 2.11.
Obtain L_{con} on D_n according to 5.5.
Obtain $\mathrm{L} = \mathrm{L}_{ce} + \lambda \mathrm{L}_{con}$
End For
Update f, g, t to minimize loss L (g is updated only if using linear or non-linear projection head)
End For
Return encoder network f and t.

For multi-class classification containing M classes, a dataset with N samples is represented as $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$, where $\boldsymbol{x}_i$ is the i^{th} instance with its observed label as $\boldsymbol{y}_i \in \{1, ..., M\}$. We train networks with mini batches of size K.

5.3.1 Transformer Architecture

Based on the conventional Co-Training framework, we formulate the proposed CCTran approach with two transformer based encoders denoted by $f(\boldsymbol{x}, \mathbf{\Theta}_1)$ and $f(\boldsymbol{x}, \mathbf{\Theta}_2)$, where $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$ are the parameters of the underlying transformer network respectively. One fundamental advantage of our proposed algorithm is that it is independent of the choice of transformer backbone. Thus, we opt for simplicity and adopt two commonly used transformer backbones from [53, 91] (Details of the transformers are shown in Section § 5.4. We use different configurations of transformers for different datasets, while the configuration of both $f(\boldsymbol{x}, \mathbf{\Theta}_1)$ and $f(\boldsymbol{x}, \mathbf{\Theta}_2)$ remains the same for each separate dataset.) to obtain features $\boldsymbol{h}_i^1 = f(\boldsymbol{x}_i, \mathbf{\Theta}_1)$ and $\boldsymbol{h}_i^2 = f(\boldsymbol{x}_i, \mathbf{\Theta}_2)$ where $\boldsymbol{h}_i^1, \boldsymbol{h}_i^2 \in \mathbb{R}^d$. We denote the prediction probabilities of $\boldsymbol{x}_i$ as $\boldsymbol{p}_i^1 = t(\boldsymbol{h}_i^1, \mathbf{\Theta}_1)$ and $\boldsymbol{p}_i^2 = t(\boldsymbol{h}_i^2, \mathbf{\Theta}_2)$ such that $\boldsymbol{p}_i^1, \boldsymbol{p}_i^2 \in \mathbb{R}^M$

(t is the linear layer.)

5.3.2 Selection Strategy and Classification Loss

Similar to [48, 159, 150], we also employ the cross entropy classification loss based selection strategy. Before going into details, we first introduce the relationship between small-loss samples and clean samples. According to [48, 150], there is a consensus that samples with a smaller loss are more likely clean, while those samples with larger loss values are more likely to be noisy and corrupted. We obtain the classification loss of the sample $\boldsymbol{x}_i$ for two transformers as shown below:

$$\mathcal{L}^1_{ce}(\boldsymbol{x}_i, \boldsymbol{y}_i) = -\Sigma^M_{m=1} \boldsymbol{y}_i \log(\boldsymbol{p}^m_1(\boldsymbol{x}_i)) \quad , \quad \mathcal{L}^2_{ce}(\boldsymbol{x}_i, \boldsymbol{y}_i) = -\Sigma^M_{m=1} \boldsymbol{y}_i \log(\boldsymbol{p}^m_2(\boldsymbol{x}_i)) \tag{5.1}$$

We then select R examples according to $\mathcal{L}^1_{ce}$ and $\mathcal{L}^2_{ce}$ within the mini batch that produces the R lowest cross entropy losses respectively. Thereafter, for each mini batch the classification loss that will be back propagated to update $f(\boldsymbol{x}, \boldsymbol{\Theta}_1)$ and $f(\boldsymbol{x}, \boldsymbol{\Theta}_2)$ are:

$$\mathrm{L}^1_{ce} = \Sigma^R_{i=1} \mathcal{L}^1_{ce}(\boldsymbol{x}_i) \quad \forall \boldsymbol{x}_i \in \mathcal{D}_1 \quad , \quad \mathrm{L}^2_{ce} = \Sigma^R_{i=1} \mathcal{L}^2_{ce}(\boldsymbol{x}_i) \quad \forall \boldsymbol{x}_i \in \mathcal{D}_2 \tag{5.2}$$

where $\mathcal{D}_1$ and $\mathcal{D}_2$ represent the set of examples that is used to update the parameters of the network. We follow the work of [48] and two transformers select the R lowest $\mathcal{L}_{ce}$ calculated in Eqn 5.1 for one another. Differently, [150] uses the joint classification loss to select examples such that $\mathcal{D}_1 = \mathcal{D}_2$.

5.3.3 Contrastive Loss Calculation

We calculate the conventional contrastive loss as used in [128, 57, 154, 15]. A mini batch of K examples is randomly sampled and we define the contrastive loss on features extracted from two transformers. That is to say, we have $2K$ features. Following the work of [16], for each sample $\boldsymbol{x}_i$ (or $\boldsymbol{h}^1_i$), we consider the corresponding feature of $\boldsymbol{x}_i$ from the other network (*i.e.* $\boldsymbol{h}^2_i$) as the positive sample and the remaining $(2K-2)$ features as the negative samples. The contrastive loss for the extracted feature $\boldsymbol{h}^1_i$ is defined as below:

$$\begin{aligned} \mathrm{L}_{con}(\boldsymbol{h}^1_i) &= -\log \frac{\exp(\mathrm{sim}(\boldsymbol{h}^1_i, \boldsymbol{h}^2_i)/\phi)}{\Sigma^{2K}_{j=1} \mathbb{1}_{[j \neq i]} \exp(\mathrm{sim}(\boldsymbol{h}^1_i, \boldsymbol{c}_j)/\phi)} \quad , \\ \boldsymbol{c}_j &= \begin{cases} \boldsymbol{h}^1_k & \text{if } j \mod 2 == 1 \\ \boldsymbol{h}^2_k & \text{if } j \mod 2 == 0 \end{cases}, \; k = \mathrm{floor}((j+1)/2), \end{aligned} \tag{5.3}$$

where $\mathbb{1}_{[j \neq i]}$ is an indicator function evaluating to 1 when $j \neq i$, ϕ denotes a temperature parameter which is set to 0.5 for all experiments, and "sim" denotes the standard cosine similarity measure. Similarly for h^2_i, the contrastive loss is defined as

$$\mathrm{L}_{con}(\boldsymbol{h}^2_i) = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{h}^2_i, \boldsymbol{h}^1_i)/\phi)}{\Sigma^{2K}_{j=1} \mathbb{1}_{[j \neq 2i]} \exp(\mathrm{sim}(\boldsymbol{h}^2_i, \boldsymbol{c}_j)/\phi)}. \tag{5.4}$$

Thus, the overall contrastive loss for a mini batch is given below:

$$\mathrm{L}_{con} = \Sigma_{i=1}^{K}[\mathrm{L}_{con}(\boldsymbol{h}_i^1) + \mathrm{L}_{con}(\boldsymbol{h}_i^2)]. \tag{5.5}$$

5.3.4 Comparison and Application

The final loss of CCT for each transformer is defined as follows:

$$\mathrm{L}_1 = \mathrm{L}_{ce}^1 + \lambda\, \mathrm{L}_{con} \quad , \quad \mathrm{L}_2 = \mathrm{L}_{ce}^2 + \lambda\, \mathrm{L}_{con}, \tag{5.6}$$

where λ is a hyper parameter whose value is fixed to 0.0001 for every experiment.

Our CCTran approach binds the contrastive learning within the noisy learning setting with the result that all examples in a noisy dataset are utilized, such that none of the noisy samples are fully discarded during the training phase. While straightforward and fast, CCT still obtains superior performance over the baseline Co-Tr algorithm. The main differences between CCT and a general Co-Tr framework are as follows:

1. CCTran makes use of transformers to robustly handle label noise.
2. CCTran treats examples in both a supervised and an unsupervised manner.
3. When calculating the contrastive loss, all the other $2K - 1$ features are considered in calculation of the loss (as shown in Eqn 5.3 and 5.4), thereby utilizing all the samples in the mini batch irrespective of whether they are labeled as clean or corrupted.
4. By adding the contrastive loss, the regularization between two transformers increases as it encourages the two extracted features of the same input to be similar; without any introduction of additional learnable parameters in the training phase.

It is to be noted that Wei *et al*. [150] also employed "contrastive loss" in their proposed method, however there still exist major differences between CCT and [150]. CCT considers all the samples within the mini batch; while Wei *et al*. considers one single input from the two base encoder networks and only similarity between positive pairs in the calculation of the contrastive loss. That is to say, for each sample in a mini-batch, CCT's contrastive loss consists of two parts: (a) distance between the two features extracted from the same image (which are minimized); (b) distances between the feature (from one transformer) of the sample and features (from the other transformer) of all other samples in the mini-batch (which are maximized). Moreover, Wei *et al*. [150] selects samples based on agreement between the two networks while we consider all the samples within the mini batch in calculation of the contrastive loss irrespective of whether they are labeled as clean or noisy by the underlying transformer based on the "small-loss" principle.

Our proposed CCT is easy to integrate into a conventional Co-Tr framework while boosting its performance. When dealing with a noisy labeled dataset, the "small-loss" principle is widely used [48, 150, 159]. In different baseline algorithms,

Table 5.1: Structure of CNN backbone for the toy experiment. The slope of all LReLU functions in the network are set to 0.01. K denotes the number of training classes for the dataset used.

#Layer	Input Image
1	3×3 conv, 128 LReLU
2	3×3 conv, 256 LReLU
3	3×3 conv, 128 LReLU
4	avg-pool fc-layer $128 \rightarrow K$ softmax

various sample selection strategies are used ($\mathcal{D}_i$ in Equation 5.2). In these small-loss selection strategies, samples with a large cross entropy loss are ignored. However, by using our CCT framework, one does not only keep the principle of selecting clean examples but also utilizing all data when calculating and back propagating the contrastive loss to update the parameters of the underlying networks.

5.4 Experiments

Details of Implementation of Toy Experiment We use the CIFAR10 dataset with 45% pairflip noise. We use the 3-layer MLP as the backbone of CNN (Refer to Table 5.1 for the details of its architecture); we use *Compact Convolutional Transformer*$-2/3 \times 2$ [53] as the backbone for the transformer. We use the Adam optimizer [67] with the initial learning rate set to 2×10^{-3}, and training is performed over 50 epochs.

Datasets We verify the effectiveness of our approach on six benchmark datasets: MNIST [80], CIFAR10 [73], CIFAR100 [73], CUBS200-2011 [145], CARS196 [72] and Clothing1M [155] with different settings. More details about the datasets can be found in Section 2.4.1.

Network Architecture and Optimizer. We provide the details of the experiments performed in Section 4 in the chapter. For a fair comparison, we implement all methods with default parameters using PyTorch. For MNIST, CIFAR10 and CIFAR100, we use a *Compact Convolutional Transformer*-$7/3 \times 2$ architecture from [53] as it is a well-established network architecture for these two datasets and does not need pre-training. For this architecture, we use the Adam optimizer [67] with an initial learning rate of 3×10^{-4}. For the CNN backbone, we use a 9-layer CNN architecture following [77, 150, 48, 99] as it is a well-established network architecture for weakly-supervised learning (Refer to Table 5.2 for details). For experiments

Table 5.2: Structure of the network trained using MNIST, CIFAR10 and CIFAR100. The slope of all LReLU functions in the network are set to 0.01. K denotes the number of training classes for the dataset used.

#Layer	Input Image
1	3×3 conv, 128 LReLU
2	3×3 conv, 128 LReLU
3	3×3 conv, 128 LReLU
	2×2 max-pool, stride 2
	dropout, p = 0.25
4	3×3 conv, 256 LReLU
5	3×3 conv, 256 LReLU
6	3×3 conv, 256 LReLU
	2×2 max-pool, stride 2
	dropout, p = 0.25
7	3×3 conv, 512 LReLU
8	3×3 conv, 256 LReLU
9	3×3 conv, 128 LReLU
	avg-pool
10	fc-layer $128 \rightarrow K$
	softmax

on CUBS200-2011 and CARS196 datasets, we use the Inception-V1 [136] network pre-trained on ImageNet [21] as the CNN backbone; where as we use Swin-L [91] pre-trained on ImageNet-22K as the transformer backbone for these two datasets. We use the RMSprop optimizer and the Adam optimizer for the CUBS200-2011 and CARS196 dataset respectively, each with the initial learning rate set to 1×10^{-4}. The total number of epochs for every experiment is set to 200. For the Clothing1M dataset, we follow the previous work of Xiao *et al*. [155] and use ResNet-18 [55] pre-trained on ImageNet [21] as the CNN backbone, and consider the Swin-B [91] pre-trained on ImageNet-22K as the transformer backbone. Similar to the learning protocol of [150], we have used the Adam optimizer for both networks with the learning rate scheduling as follows: 8×10^{-4}, 5×10^{-4} and 5×10^{-5} after every 17 epochs of training.

Experimental setup For results shown in Table 5.3, 5.4 and 5.5, we assume the noise rate τ is known. This is a common assumption in recent studies on label noise [48, 150, 157]. Please refer to Chapter 2 for more details on transformer structure, optimizer and learning rate used for different datasets. Details regarding the noise transition matrices are also provided in Chapter 2.

Table 5.3: Comparison of our proposed CCT against several baseline algorithms. We report the average accuracy (%) after 5 runs for CCT. The best and the second best results are shown in **red** and **blue** respectively

noise	Dataset	F [112]	DC [96]	MN [64]	CT [48]	JC[150]	CCT OURS
pairflip-45%	MNIST	0.24	58.03	80.88	87.63	**93.3**	**94.1±0.14**
symmetric-80%	MNIST	13.35	28.51	66.58	79.73	**84.9**	**85.6±0.11**
symmetric-50%	MNIST	79.61	81.15	90.05	91.32	**95.8**	**96.2±0.07**
symmetric-20%	MNIST	98.82	95.70	96.70	97.2	**97.5**	**97.9±0.02**
pairflip-45%	CIFAR10	6.61	48.80	58.14	72.62	**74.4**	**74.9±0.17**
symmetric-80%	CIFAR10	15.88	15.31	22.15	26.58	**27.8**	**27.4±0.20**
symmetric-50%	CIFAR10	59.83	51.49	71.10	74.2	**79.4**	**80.1±0.12**
symmetric-20%	CIFAR10	84.55	80.44	80.76	82.32	**85.7**	**86.0±0.07**
pairflip-45%	CIFAR100	1.60	26.05	31.60	**34.8**	32.2	**34.9±0.21**
symmetric-80%	CIFAR100	2.1	3.89	10.89	15.2	**15.5**	**15.5±0.24**
symmetric-50%	CIFAR100	41.04	25.80	39.00	41.4	**43.5**	**44.3±0.15**
symmetric-20%	CIFAR100	**61.9**	44.52	52.1	54.2	53.0	**54.2±0.12**

Table 5.4: Comparison of our proposed CCT against several baseline algorithms for large fine-grained image recognition datasets for a **symmetric noise** setup with different values of noise rate τ. We report the average accuracy on the test set (%) after 5 runs for CCT. CCT runs with a batch size of 64).

	CUB200-2011				CARS196			
τ	CE	CT [48]	JC [150]	CCT Ours	CE	CT [48]	JC [150]	CCT Ours
80%	12.3	15.2	15.9	**16.8±0.25**	7.1	13.0	14.1	**15.2±0.23**
50%	43.7	55.6	56.3	**57.2±0.18**	39.7	67.7	68.3	**69.6±0.16**
20%	64.9	73.0	73.6	**74.6±0.11**	72.0	86.1	86	**86.5±0.08**

5.4.1 Analysis on Different Datasets

MNIST. Rows 1-4 in Table 5.3 report the accuracy on the test set of MNIST [81]. MNIST is a relatively simple dataset compared to the others considered in our experiments. As observed, in the symmetry case with $\tau = 20\%$, which is also the simplest experimental setup, the performance of every algorithm is similar to each other. CCT manages to outperform all the other baseline and state-of-the-art methods, especially when the value of noise rate is high. More specifically, when it comes to more difficult cases, such as symmetric noise with a τ set to 50% and 80%, the performances of F-correction (F) [112], Decoupling (DC) [96] and MentorNet (MN) [64] drop significantly and are not robust to such extreme levels of corruption. However, Co-Teaching (CT) [48] and JoCoR (JC)[150] work well under all these noise rates. Co-Teaching [48] is a clean-sample-selection strategy based method and JoCoR [150] is based on maximizing the agreement between the underlying homogeneous networks. Among all the experimental conditions for MNIST, our

CCT achieves the best performance. Especially when the noise setup is 45% pair flip and 80% symmetric, we beat the next best method JoCoR [150] by 0.8% and 0.7% respectively; which is impressive considering those baselines are well over 90% or 80% in terms of test accuracy respectively.

CIFAR10 Rows 5-8 of Table 5.3 show the results of the state-of-the-art methods and our CCT on the noisy CIFAR10 [73] dataset. Similar to MNIST, we outperform the baselines in all the experimental setups except when τ is set to 20%. When τ is set to 80%, CCT still works well but falls behind JoCoR by a negligible value of 0.4%. However, for all other noise rates, we consistently achieve the best results. We also observe that F-correction, Decoupling and MentorNet work well when the noise rate is low. With the increase in the noise rate, the performance of F-correction drops dramatically. Decoupling and MentorNet still work well but their performances are not on par with our CCT results.

CIFAR100 The performances of other baseline algorithms and CCT on CIFAR100 [73] dataset are presented in Rows 9-12 of Table 5.3. We note that the performances on CIFAR100 are much lower than MNIST and CIFAR10, as there are 100 classes in CIFAR100 in comparison to 10 in MNIST and CIFAR10. We observe that CCT performs better than Co-Teaching and JoCoR (which are relatively robust compared to others) under all evaluated experimental setups. CCT only loses to F-correction when the noise rate is symmetric 20%

CUBS200-2011 We show the experimental results on CUBS200-2011 [145] in Table 5.4. For this dataset, we compare our CCT with three baselines: Cross Entropy classification (CE)[2], Co-Teaching (CT) [48] and JoCoR (JC) [150]. Note that, CUBS200-2011 is a more difficult dataset compared to MNIST, CIFAR10 and CIFAR100 as it contains images from 200 fine-grained classes of birds. We observe that our CCT outperforms all the three baseline algorithms considered across several different experimental setups, thus demonstrating that CCT is more robust to various noise conditions against the other methods.

CARS196 Table 5.4 shows results of the performance of CCT on the CARS196 [71] dataset. Similar to the experimental setup of CUBS200-2011, we compare CCT against Cross Entropy (CE), Co-Teaching (CT) [48] and JoCoR (JC) [150]. CCT outperforms all other baseline methods for several experimental setups. Cross entropy loss fails to perform competitively against the other algorithms when τ is increased to 50% and 80%. In a less challenging setup (*i.e.* 20% symmetric noise), CCT achieves a similar performance as Co-Teaching and JoCoR. However, CCT outperforms both Co-Teaching and JoCoR when τ is over 50%. More specifically, when τ is 50%, CCT beats the next best algorithm JoCoR by 1.3%. This is an impressive difference as 50% noise rate on CARS196 is considered to be a very challenging setup. A similar conclusion can be drawn with $\tau = 80\%$.

[2]The network is trained with the cross entropy classification loss using corrupted (or noisy) labels.

Table 5.5: Comparison of our proposed CCT against two different baselines on Clothing1M: Co-Teaching [48] and JoCoR [150]. Accuracy(%) is reported as the average of 5 runs.

method	**Co-Teaching** [48]	**JoCoR** [150]	**CCT(OURS)**
accuracy	68.5	69.8	**72.0**

Clothing1M We use the noisy version of the Clothing1M [155][3] and follow the work of [90, 160] to infer the noise rate by the clean validation set. The batch size is set to 64. Further, we report the classification accuracy after training for 51 epochs for all methods. As shown in Table 5.5, Our CCT performs better than Co-Teaching [48] and JoCoR[150]; thus demonstrating the effectiveness of CCT over Co-Teaching and JoCoR for a real world corrupted label dataset.

Discussion of Contrastive Loss Here, we study the difference in the calculation of the contrastive loss between JoCoR [150] and CCT. In order to calculate the contrastive loss of each sample, JoCoR calculates Jensen-Shannon (JS) divergence between the two softmax outputs of the sample; while only considering $(100 - \tau)\%$ of the entire samples in the loss calculation step. On the other hand, CCT makes use of all samples in the mini-batch to calculate contrastive loss for each samples such that

1. the cosine similarity between the positive samples (outputs of the two networks for the *anchor*[4] sample) is maximized, and

2. the cosine similarity between the feature of the anchor sample (output of one network of the *anchor* sample) and all other negative features within the mini-batch is minimized.

5.4.2 Ablation

Robustness of Transformers Against Label Noise As mentioned in Section 1 in the chapter, Figure 5.1 shows the robustness of varying noise settings. For a fair comparison, we use the CIFAR10 dataset and test on two different noise rates τ: symmetric 50% and pairflip 45%. The backbone used for Co-teaching is a 9-convolution layer (CCN_9) network which is commonly used for Co-Tr frameworks [48, 150, 115]. In order to show the robustness of transformers against label noise, we only swap the backbone from CCN_9 to transformer [53] (without adding any contrastive loss module or other strategies such as [150, 157]). More details of the CCN_9 and the transformer are provided in the supplementary material. As observed from the test accuracy curve, the test accuracy of CCT increases at a slower rate than the Co-teaching baseline. However, soon after 10 epochs of training, Co-teaching overfits to label noise and its accuracy decreases drastically. Further, as the training progresses, CCT outperforms Co-teaching.

[3] We have used the already noise corrupted version of the Clothing1M dataset in our experiments.

[4] The sample considered for calculating the loss is referred to as the *anchor* sample.

Table 5.6: Evaluation of our proposed method CCT with different values of temperature which is shown in Eqn 3 on different datasets.

noise	Dataset	CCT (0.3)	CCT (0.5)	CCT (0.7)	CCT (0.9)
pairflip-45%	MNIST	93.2	94.1	92.1	91.1
symmetric-80%	MNIST	84.3	85.6	83.6	81.1
symmetric-50%	MNIST	94.5	96.2	95.4	92.1
symmetric-20%	MNIST	95.8	97.9	95.4	93.2
pairflip-45%	CIFAR10	72.7	74.9	73.1	70.4
symmetric-80%	CIFAR10	25.8	27.4	25.4	24.1
symmetric-50%	CIFAR10	77.9	79.9	78.2	76.1
symmetric-20%	CIFAR10	85.6	86.0	84.3	82.6
pairflip-45%	CIFAR100	34.2	34.9	33.9	32.7
symmetric-80%	CIFAR100	13.4	15.5	13.6	12.7
symmetric-50%	CIFAR100	44.0	44.3	43.6	40.8
symmetric-20%	CIFAR100	52.5	54.2	53.1	52.7

Study on Temperature The temperature plays a role in controlling the strength of penalties on the negative samples. Specifically, contrastive loss with small temperature tends to penalize much more on the negative samples [147]. In all the experiments considered so far, the value of the temperature parameter ϕ in Eqn 3 is set to 0.5. Here we study the overall effect of ϕ in the calculation of the contrastive loss as shown in Eqn 3. Results are shown in Table 5.6. Our contrastive loss module is relatively non-sensitive to temperature value. The best performances are always achieved by 0.5; when the temperature is set to 0.9, the performance drops more.

Batch Size Contrastive learning benefits from a large batch size [15, 154, 54]. Table 5.7 shows the results of varying the training batch size K from 128 to 1024. It illustrates that CCT benefits from a large batch size. Even with the same batch size as other baselines (*i.e*. 128), our CCT achieves a similar or better performance as compared to the state-of-the-art methods. For example, with an 80% corrupted CIFAR10 dataset, CCT with a 128 batch size falls behind JoCoR by 0.1%. When it comes to a batch size of 1024, CCT achieves a 32.6% accuracy on the test set which beats the next best JoCoR [150] by 4.8%. We conjecture that this is because the contrastive loss benefits from a large batch size; as a batch size of K will give us $2K - 2$ negative samples. This indeed will increase the robustness of sample selection strategy while identifying the clean and noisy samples based on the loss values. We also noticed that gain with larger batch size varies for different settings. As an example, on the CIFAR10 dataset, when noise rate is 20% symmetric, the test accuracy increases from 82.4% to 84.8% as the batch size is increased from 128 to 1024. For 80% symmetric noise, test accuracy increases from 27.7% to 32.6% on the CIFAR10 dataset as the batch size is increased from 128 to 1024; which is admirable due to complex noisy conditions. But for CIFAR100, increasing batch

Table 5.7: Evaluation of our proposed method CCT for different values of training batch size (*i.e*. 128, 256, 512 and 1024).

noise	Dataset	CCT (128)	CCT (256)	CCT (512)	CCT (1024)
pairflip-45%	MNIST	92.3	93.5	94.4	95.9
symmetric-80%	MNIST	83.2	85.3	91.1	93.2
symmetric-50%	MNIST	95.2	95.4	95.8	96.2
symmetric-20%	MNIST	97.9	98.2	98.5	98.9
pairflip-45%	CIFAR10	73.4	73.8	74.6	75.2
symmetric-80%	CIFAR10	27.7	29.1	31.4	32.6
symmetric-50%	CIFAR10	77.8	78.6	79.2	79.9
symmetric-20%	CIFAR10	82.4	83.1	83.9	84.8
pairflip-45%	CIFAR100	33.6	33.9	34.4	34.9
symmetric-80%	CIFAR100	15.3	15.4	15.5	15.5
symmetric-50%	CIFAR100	41.4	42.5	43.1	44.3
symmetric-20%	CIFAR100	54.5	55.5	56.0	56.2

size does not help much when noise rate is 80%.

Discussion of different projection head In this section, we study the importance of including the projection head $g(\Theta)$ in our proposed CCT. Table 5.8 shows evaluation results using three different architectures for the projection head: identity mapping; nonlinear projection and linear projection. For nonlinear projection, following the work of [4], we apply a Multi Layer Perceptron (MLP) with one additional hidden layer followed by a ReLU activation. We set the batch size to 128 and the noise rate to $\tau = 50\%$ and perform experiments on the CIFAR10 dataset. The dimension of the hidden layer in the non-linear projection head and output dimension of the linear projection head is set to $2d$. In our experiments, we observe that the identity mapping outperforms non-linear projection and obtains comparable performance against linear projection. We conjecture that both the linear and identity mapping keeps more information than the non-linear projection. [128, 4] suggested that MLP is a better choice when applying contrastive loss. However, when dealing with noisy labels, we may lose information due to the hidden layers. We believe that might be the reason as to why a non-linear projection does not perform well compared to the other two projections. More experiments of using different projection heads are shown in Supplementary material.

Different values of λ: In the experiments performed in Table 5.3, 5.4, 5.8, we have set the value of λ in Eqn. 5.6 to 0.001. Table 5.9 shows that the CCT approach is relatively robust to the value of λ. We believe that one can get even better results by further tuning the value of λ.

How do we balance between filtering noisy labels and utilizing the data? As discussed in Chapters 3-5, there is actually a wealth of useful information in the

Table 5.8: Classification Accuracy (%) for different types of projection head: linear projection, non-linear projection and identity projection.

noise	linear	non-linear	identity
symmetric-80%	27.5	26.4	**27.7**
symmetric-50%	77.2	75.5	**77.8**
symmetric-20%	82.4	80.5	**82.4**

Table 5.9: Classification Accuracy (%) for different values of λ which is shown in Eqn 5.6. λ is a hyper-parameter that controls the balance between the classification loss and the contrastive loss in the total loss function.

noise	0.01	0.001	0.0001
symmetric-80%	25.3	**27.7**	26.8
symmetric-50%	74.9	**77.8**	75.8
symmetric-20%	81.8	**82.4**	82.3

images even if their labels are completely incorrect. Our approach is that if a label is deemed highly unreliable—based on the confidence score from the model's output—we discard the label but keep the image as unlabeled data. This is the key application demonstrated in Chapter 5, which also forms the core idea of contrastive learning (unsupervised learning). For images with confident labels, we retain the labels and proceed with standard supervised learning.

However, the challenge lies in determining the confidence threshold, which is a critical factor. In our experiments, we tested different confidence levels across various datasets. While the optimal confidence threshold varied slightly to achieve the best results, we observed that these values generally fell within a small range. This suggests there is a common confidence level suitable for datasets with similar noise levels. Moreover, the optimal confidence level appears to correlate with the noise rate; for example, if the noise rate is around 40%, the best confidence level is also roughly 40%.

This insight provides a practical guideline; when applying our method to new datasets, one can estimate the confidence threshold based on the known or expected noise level. Nevertheless, for optimal performance, we recommend tuning this hyper-parameter through additional experiments.

5.5 Summary

In this chapter, we presented a straightforward, novel and yet a powerful framework, Contrastive Co-Transformer (CCTran), which was designed to learn from data with noisy labels. The key motivational idea was to combine contrastive learning with transformers in order to identify clean and noisy labels. Unlike previous methods that only propagate the gradients based on samples identified as clean,

CCTran makes use of the entire mini batch of samples using an unsupervised contrastive learning module. Our empirical results on MNIST, CIFAR10, CIFAR100, CUBS200-2011, CARS196 and Clothing1M showed that CCTran performs well across different complex and very challenging noisy experimental setups. In the future, we aim to integrate and study the effect of several optimal transport based loss functions such as Wasserstein distance, Maximum Mean Discrepancy etc. in the learning framework of CCTran.

In Chapters 3, 4, and 5, we have explored learning from noisy labels using the co-training framework with three different approaches: Maximum Mean Discrepancy (MMD), Sinkhorn Divergence, and Transformers, respectively. Each method has demonstrated its effectiveness in enhancing model robustness against noisy labels. In the next chapter, we will shift our focus to how Sinkhorn divergence could be used for metric learning problem, where we will investigate its performance under both noisy and clean data conditions. By addressing noise in metric learning, we aim to further advance the reliability and generalization using Sinkhorn divergence.

Chapter 6

Learning Deep Optimal Embeddings with Sinkhorn Divergences

In the previous chapter, we extensively explored learning from noisy labels, leveraging the strengths of Sinkhorn Divergence to enhance model robustness and accuracy. Building on these insights, this chapter applies the knowledge gained from Sinkhorn Divergence to the domain of metric learning. We will investigate its effectiveness on both noisy and clean datasets, aiming to improve representation learning and similarity measurement while maintaining robustness against label noise.

6.1 Introduction

Deep Metric Learning algorithms aim to learn an embedding space that preserves similarity relationships. However, they often neglect class-wise similarity constraints, leading to sub-optimal metrics, and are rarely tested with noisy labels. To address this, we propose Deep Class-wise Discrepancy Loss (DCDL), which segregates similarity distributions between classes. Experiments on multiple datasets show that incorporating class-wise relationships improves embedding space discrimination, even with noisy labels.

Existing metric learning algorithms overlook class-wise similarity and dissimilarity, leading to sub-optimal embedding spaces. To address this, we propose Deep Class-wise Discrepancy Loss (DCDL), which separates class-wise probability distributions using Maximum Mean Discrepancy and Wasserstein Distance. Experiments across five datasets demonstrate that DCDL learns a more discriminative embedding space compared to baseline algorithms. Notably, DCDL remains robust against noisy labels, effectively managing inter-class and intra-class variances to enhance embedding space discrimination.

In this chapter, we address the issue of learning an efficient, effective and discriminative non-linear embedding space that results in compact class-specific

clusters of the embedded points. Towards this aim, we propose and develop a novel loss function that comprehensively incorporates *class-wise* similarity attributes of the embedded points to simultaneously minimize and maximize the intra-class and inter-class variances respectively.

Irrespective of their immense successes recently, these *local* deep metric learning algorithms suffer from two serious drawbacks. First, they only incorporate and maintain local geometrical or statistical constraints in their respective algorithms; thereby failing to integrate any comprehensive geometrical or statistical characteristics of the embedding space. Second, these algorithms [142, 76] do not encapsulate *class-wise* similarity/distance relationships between the embedded points, thereby failing to learn a superior discriminative embedding space to enforce the intra-class and inter-class separation constraints. A majority of the aforementioned algorithms explicitly consider class information by either pre-training [124, 130, 79, 148] or simultaneously training [30, 143] the network with *partially global* cross entropy loss. This subtle, yet important practice aims to assist the network in finding a superior starting point in the loss landscape [82] such that an optimal solution can be attained within a minimal number of descent steps. Moreover, Horiguchi *et al*. [59] also successfully demonstrated the need of employing a softmax function along with learning a discriminative metric. Even though cross-entropy loss is able to obtain class separating sub-spaces; it cannot guarantee that the resultant embedding points will be tightly knit within each of the sub-spaces. Generative modeling based algorithms [87, 169] attempt to comprehensively encompass the entire embedding space by generating hard negatives [124] for every embedded point. However, such generation also limits their applicability as these hard negatives need to be within the ϵ neighbourhood of the anchor; thus limiting their overall outreach in the embedding space. Distribution based statistical methods [142, 76] fail to consider such class-wise relationships and enforce a weak disparity measure between the pairwise distributions; thereby limiting their capacity to form well separated class-wise clusters.

Keeping these drawbacks in mind, we propose and design a novel loss function known as Deep Class-wise Discrepancy Loss (DCDL) which takes into account the *class-wise* similarity relationships between the embedded points. Specifically, we compute the probability distributions of the embeddings that belong as well as do not belong to a particular class (thus *class-wise*) and enforce a discrepancy constraint between the two distributions and **push them away** from each other with the aim to increase/decrease the intra/inter class similarities in the embedding space simultaneously. Unlike the algorithms mentioned before that do not contemplate such similarity constraints in their loss functions; DCDL attempts to cluster the class-wise embeddings within certain specific regions of the embedding space. Thus, DCDL learns to maximize the separation between the embedded points belonging to each and every distinct class while preserving their class-wise relationships. Towards achieving this objective, we employ Optimal Transport [38] based Sinkhorn Divergences [20] to enforce discrepancy constraints between the probability distributions of the class-wise embeddings.

The major contributions are as follows:

1. A novel loss function that exploits the class-wise pairwise similarity relationships to learn a discriminative embedding space.

2. An augmentation to the conventional *local* metric learning loss functions to learn and enforce similarity constraints in the **presence** and **absence** of noisy labels.

3. We further show that the proposed loss function can be combined with the cross-entropy loss to further boost the overall performance of the network.

We evaluate our proposed loss function DCDL on the Cifar-10 [73], Cifar-100 [73] and STL-10 [17] datasets in the **presence** and **absence** of noisy labels. We also evaluate DCDL on *Caltech-UCSD Birds* (CUBS-200-2011) [145] and *Stanford Cars* (CARS196) [71] datasets in the Deep Metric Embedding Learning (DMEL) framework in both the noisy and clean labels settings. Our empirical evaluations of DCDL outperform the *local* deep-metric learning algorithms in both noisy and clean label settings across the various datasets; which undoubtedly demonstrate the generic need of exploiting and learning such class-wise discrepancy constraints in order to learn an effective and discriminative embedding metric space.
Notation: Throughout this chapter, we use bold lower-case letters (*e.g.*, $\boldsymbol{x}$) and bold upper-case letters (*e.g.*, $\boldsymbol{X}$) to represent column vectors and matrices respectively. $[\boldsymbol{x}]_i$ denote the ith element of the vector $\boldsymbol{x}$. $\mathbf{I}_n$ represents the $n \times n$ identity matrix. $\|\boldsymbol{X}\|_F = \sqrt{\mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X})}$ denote the Frobenius norm of the matrix $\boldsymbol{X}$, with $\mathrm{Tr}(\cdot)$ indicating the trace of the matrix $\boldsymbol{X}$. $\omega(\boldsymbol{r})$ represents a diagonal matrix with diagonal elements as $\boldsymbol{r}$. $\boldsymbol{X}^\top$ denotes the transpose of $\boldsymbol{X}$. $\mathcal{U}$ and $\mathcal{V}$ represent two distinct probability distributions on the metric space $\mathcal{M}$. $\{\boldsymbol{u}_i\}_{i=1}^n$ and $\{\boldsymbol{v}_j\}_{j=1}^m$ denote n and m i.i.d samples (or **support points**) drawn from $\mathcal{U}$ and $\mathcal{V}$ respectively.

6.2 Related Work

In this section we provide a brief overview of the several baseline Deep Metric Learning algorithms which have been developed in the recent past.
Metric Learning In general, metric learning algorithms aim to learn a parametric-functional representation of a distance metric $\boldsymbol{M} \succeq \mathbf{0}$ [151] to obtain distinct and compact clusters in the embedding space. Traditional algorithms either attempt to learn the feature extractors or $\boldsymbol{M}$ separately, thus neglecting the need for learning both holistically. Interested readers are referred to [7, 74] for more detailed insights into such algorithms. Deep Learning algorithms, boosted by their recent remarkable success across a variety of research areas in computer vision and machine learning, have been proposed to overcome these limitations of the traditional algorithms. These algorithms can be broadly categorized into **(a) Structure** based methods [124, 130, 79, 148, 119] which preserve the local structure of the learnt

embedding space, **(b) Sampling** based methods [129, 152, 26] to efficiently sample informative samples, **(c) Statistical** based methods [142, 76, 118] and **(d) Generative** modeling based methods [87, 27, 169] to explicitly model the intra-class variances and reduce the inter-class variances. These algorithms have been successfully employed across a variety of well-established computer vision applications such as person re-identification [30, 143], zero/few shot learning [58, 107, 70, 141], object tracking [138, 8], image retrieval [129, 130, 44] to name a few.

Structure based methods: The seminal work in DML was undertaken by Schroff *et al*. [124]; where they proposed a *semi-hard* triplet mining algorithm which guarantees that for a given *anchor* ($\boldsymbol{z}^a$), its closest *negative* ($\boldsymbol{z}^-$) is further away from the *positive* ($\boldsymbol{z}^+$). Similar constraints in the angle formed at $\boldsymbol{z}^-$ within a triplet in the embedding space was proposed by Wang *et al*. [148]. Hierarchical Triplet loss proposed by Weifeng *et al*. [36] constructed a class-level tree and dynamically learnt a margin such that the *semi-hard* triplet mining criterion is fulfilled. Song *et al*. [130] proposed a structured prediction loss to enforce global geometrical constraints in the embedding space to prevent outlier clusters. Roy *et al*. [119] imposed rotation-invariant orthogonality constraints in the embedding space to truncate the search space for learning an efficient metric. Law *et al*. [79] exploited spectral clustering concepts to obtain tight but well separated clusters in the embedding space. Sanakoyeu *et al*. [123] splits the embedding space into P consecutive parts; and learnt a local discriminative loss over each of the individual parts with the aim to learn distinct and disjoint features in each of them. SoftTriple loss [114] additionally models and learns multiple clusters per class as an additional layer and is trained with a smoothened version of the triplet loss.

Sampling based methods: The objective of these algorithms is to learn efficient sampling strategies with the aim to mine important and informative samples. Lifted-Structure proposed by Song *et al*. [107] constructs a mini-batch of samples by subsequent addition of importance-sampled hard negatives. Sohn [129] proposed a multi-class *NPairs* loss by utilizing all the negatives for every anchor point $\boldsymbol{z}^a$. They also proposed an efficient batch construction scheme that uses all the available negative pairs in order to form the informative triplets. Wu *et al*. [152] clearly demonstrated the importance of a steady sampling function to mine informative negative samples to reduce noisy back-propagated gradients.

Statistical based methods aim to model (parametric or non-parametric) the statistical distributions in the embedding space. Histogram loss [142] enforces an overlap constraint minimizing the overlap between the histograms of the cosine similarities for the positive and the negative pairs respectively. Kumar *et al*. [76] formulates the pairwise distances between the matching and non-matching pairs as two different Gaussians; and propose the Distribution loss to reduce the overlap between them. Roth *et al*. [118] learnt an auxiliary encoder to reduce the intra-class variances while separating the intra-class means of the class-wise embedding points.

Generative modeling based methods: Deep Adversarial Metric Learning [27] learns a Generative Adversarial Network [42] to generate hard negatives to enable efficient learning of the discriminative embedding space. Lin *et al*. [87] proposed

Deep Variational Metric Learning that makes use of variational inference to generate robust discriminative samples to reduce the intra-class variances. Zheng *et al*. [169] proposed to control the hardness level of the generated negatives by linear interpolation over the entire embedding space, thereby resulting in well spread out clusters of the embedding samples.

Remark 1 *It is to be noted that none of the above mentioned algorithms comprehensively model class-wise similarity relationships while learning the embedding space. Thus, it seems that the embedding points are neither tightly knit nor well spread over the entire embedding space. Unlike these algorithms,* DCDL *learns class-wise distributions belonging to each and every distinct class and maximizes the separation between them to group the class-wise embeddings within certain specific regions of the embedding space while retaining their class-wise relationships.*

Remark 2 *Statistical based methods learn a discriminative embedding by enforcing similarity (or dissimilarity) constraints on the probability distributions of all the positive and negative pairs within the dataset. However, as before, these methods do not consider class-wise similarity/dissimilarity constraints between the positive and the negatives pairs for each individual class within the dataset. Moreover, they enforce a weak discrepancy measure to reduce the overlap between the probability distributions and hence are not able to decrease (increase) intra-class (inter-class) variances within the dataset, thus obtaining a sub-optimal embedding space. On the other hand,* DCDL *considers the class-wise samples within each mini-batch to calculate such class-wise probability distributions. Thus, apart from maintaining class-wise relationships,* DCDL *also employs the well-known Optimal Transport probability diversity measures as a discrepancy function to reduce the overlap between these distributions in order to increase the inter-class variances within the embedding space. Moreover, we also add a local metric learning loss function to* DCDL *in order to reduce the intra-class variances within each of the probability distributions (Refer to Eqn.* (6.6) *in Section § 6.3) Further, the statistical methods mentioned above are highly sensitive to their respective parameters (such as number of histograms bins in [142], dimensionality of the encoder in [118]* etc.*) which are used to empirically estimate the probability distributions; while Optimal Transport is less sensitive to its corresponding hyper-parameters [20].*

6.3 Methodology

Estimating the parameters (*i.e*. $\theta \in \Theta$) of probability density functions chosen from a family of parametric distributions p_Θ in order to fit the observed data in the best possible and meaningful way is the fundamental approach in several machine learning algorithms. One such popular algorithm is Maximum Likelihood Estimation (MLE) [33]; which aims to obtain the optimal parameters *i.e*. θ^* that

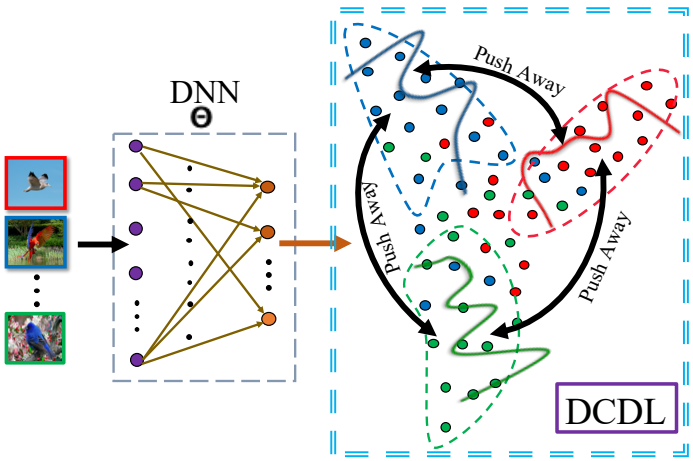


Figure 6.1: A schematic diagram of our proposed algorithm DCDL is shown here.

maximizes the likelihood of the observed data as shown below:

$$\theta^* := \max_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \ln \, \mathrm{p}_\theta(\boldsymbol{X}_i) \;\; , \tag{6.1}$$

where $\{\boldsymbol{X}_i\}_{i=1}^N$ denote the N observed data samples. Though highly popular and successful, MLE fails in those instances where the density of the desired distribution is singular on the observation space. One such example being generative modeling algorithms such as Generative Adversarial Networks (GANs) [42]; Variational Auto-Encoders (VAEs) [68] which rely on a sampling mechanism to map a low dimensional random vector to a high dimensional space. Nonetheless, GANs attempt to overcome this limitation by utilizing classification accuracy as a proxy for similarity between two parameterized distributions; whereas VAEs use Kullback–Leibler divergence (KL) [75] to reduce the distance between two parametric distributions. However, such practices fail to consider and respect the underlying geometry of the observation space. Therefore, GANs suffer from mode collapse and vanishing gradients [133]; whereas VAEs fail to provide any meaningful representation due to mismatch between two non-overlapping distributions.

Let $\boldsymbol{X}_i$ represent an image within the image space $\mathcal{X} \subset \mathbb{R}^{\mathrm{H} \times \mathrm{W} \times \mathrm{C}}$. H, W and C denote the number of rows, columns and channels of $\boldsymbol{X}_i$ respectively. The corresponding label of $\boldsymbol{X}_i$ in the label space $\mathcal{Y} \in \{1, 2, ..., \mathrm{K}\}$ is represented by $\boldsymbol{y}_i$, where K denotes the total number of classes in $\mathcal{X}$. The mini-batch B of size $\mathrm{N_B}$ is represented as $\big\{\big(\boldsymbol{X}_i, \boldsymbol{y}_i\big)\big\}_{i=1}^{\mathrm{N_B}}$. In order to avoid any degenerate solutions, we ensure that there are at least r samples belonging to each class within every mini-batch. A Deep Neural Network with its learnable parameters Θ is represented as $\mathcal{F}_\Theta$, and learns a generic non-linear mapping from $\mathcal{X}$ onto a latent feature space $\mathcal{Z} \in \mathbb{R}^n$; *i.e.* $\boldsymbol{z}_i = \mathcal{F}_\Theta(\boldsymbol{X}_i)$. Moreover, the images belonging to a particular class label $k \in \mathcal{Y}$ are denoted as $[\boldsymbol{X}]_k = \{\boldsymbol{X}_i \mid \boldsymbol{y}_i = k\}$. Furthermore, the ℓ_2 norm of

every embedded point is constrained to be 1 (*i.e.* $\|\boldsymbol{z}_i\|_2 = 1 \; \forall i = 1 \cdots N$). The aim of any deep embedding learning algorithm is to learn Θ such that the resulting embeddings have lower (higher) intra (inter) class variances respectively.
In the recent past, most deep learning approaches either (i) design a new objective function in the embedding space [130, 79, 152], (ii) design efficient batch sampling techniques [107, 129], or (iii) use generative models [87, 27, 169] to simultaneously decrease and increase intra-class and inter-class variances. Our algorithm falls in the first category where we design a novel, yet effective loss function that aims to decrease the overlap between the pairwise similarity distributions of the positive and the negative pairs for every class within the mini-batch.

Proposed Approach

We pass the input images $\boldsymbol{X}_i$ through the DNN to obtain their embedding points $\boldsymbol{z}_i$. For a particular class label $k \in \mathcal{Y}$, we obtain two different sets of features embeddings $[\boldsymbol{Z}]_k^+$ and $[\boldsymbol{Z}]_k^-$ as shown below:

$$[\boldsymbol{Z}]_k^+ = \{\boldsymbol{z}_i \mid \boldsymbol{y}_i = k\} \qquad [\boldsymbol{Z}]_k^- = \{\boldsymbol{z}_i \mid \forall\, \boldsymbol{y}_i \;\; \& \;\; \boldsymbol{y}_i \neq k\} \tag{6.2}$$

$[\boldsymbol{Z}]_k^+$ and $[\boldsymbol{Z}]_k^-$ denote the embeddings that are class-specific and class-distinct for a particular class k. The overall DCDL loss is given as follows:

$$\mathrm{L}^{\varphi} = -\sum_{k=1}^{\mathrm{K}} \mathrm{L}_k \qquad \text{where} \quad \mathrm{L}_k = \varphi\big([\boldsymbol{Z}]_k^+, [\boldsymbol{Z}]_k^-\big) \quad , \tag{6.3}$$

where φ denotes the probability discrepancy calculated using MMD with Laplacian or Gaussian kernels.

An overall schematic of DCDL (*i.e.* L^{φ}) is shown in Fig. 6.1, where DNN denotes a Deep Neural Network with learnable parameters Θ. As an illustration, we have shown only 3 classes in the input images and each have been assigned a different color, *i.e.* red, blue and green respectively. **Black Double** arrows represent the main notion of DCDL; *i.e.* to enforce and increase the distance between the probability distributions of the class-wise embeddings for every single class (Eqn. (6.2)).
Compact Class-wise Distributions: L^{φ} enforces a disparity constraint between the class-wise distributions, thereby ensuring that these class-wise distributions are well spread in the embedding space. However, it is to be noted that the class-wise distributions are non-parametric and are not explicitly modeled by L^{φ}. Hence it becomes difficult to make such class-wise distributions dense and compact. Thus, along with L^{φ}, we have incorporated several well-studied *local* loss functions (a) Triplet loss $\mathrm{L}_{\mathrm{Trip}}$ [124], (b) NPairs loss L_{NP} [129], (c) Angular loss $\mathrm{L}_{\mathrm{Ang}}$ [148] and (d) $\mathrm{L}_{\mathrm{Ang\_NP}}$ [148]; to reduce the variances of each of the class-wise distribution so as to obtain tightly knit and well separated class-wise clusters. The various *local* loss functions used are defined below:

$$
\begin{aligned}
\mathrm{L}_{\mathrm{Trip}} &= \frac{1}{|P|}\sum_{m=1}^{|P|}\Big[\big\|\boldsymbol{Z}_m^a-\boldsymbol{Z}_m^+\big\|^2-\big\|\boldsymbol{Z}_m^a-\boldsymbol{Z}_m^-\big\|^2+\tau\Big]_+ \ , \\
\mathrm{L}_{\mathrm{NP}} &= \frac{1}{N}\sum_{\boldsymbol{Z}^a\in\mathrm{B}}\Big\{\log\Big[1+\sum_{\substack{\boldsymbol{Z}^n\in\mathrm{B}\\ \boldsymbol{y}^n\neq\boldsymbol{y}^a,\boldsymbol{y}^p}}\exp(\boldsymbol{Z}^{a^\top}\boldsymbol{Z}^- - \boldsymbol{Z}^{a^\top}\boldsymbol{Z}^+)\Big]\Big\} \ , \\
\mathrm{L}_{\mathrm{Ang}} &= \frac{1}{N}\sum_{\boldsymbol{Z}^a\in\mathrm{B}}\Big\{\log\Big[1+\sum_{\substack{\boldsymbol{Z}^-\in\mathrm{B}\\ \boldsymbol{y}^-\neq\boldsymbol{y}^a,\boldsymbol{y}^+}}\exp\big(4\tan^2\alpha(\boldsymbol{Z}_a+\boldsymbol{Z}^+)^\top\boldsymbol{Z}^- \\
&\qquad\qquad -2\,(1+\tan^2\alpha)\boldsymbol{Z}^{a^\top}\boldsymbol{Z}^+\big)\Big]\Big\} \ ,
\end{aligned}
\tag{6.4}
$$

where (i) $[y]_+ = \max(0, y)$ is the hinge loss, (ii) $\tau > 0$ is a user-specified margin, (iii) $|P|$ represents the number of triplets of the form $(\boldsymbol{Z}^a, \boldsymbol{Z}^+, \boldsymbol{Z}^-)$ such that $\boldsymbol{y}^n \neq \boldsymbol{y}^a$ and $\boldsymbol{y}^a = \boldsymbol{y}^p$, (iv) α is a predefined parameter constraining the angle formed at $\boldsymbol{Z}^-$ by the triplet $(\boldsymbol{Z}^a, \boldsymbol{Z}^+, \boldsymbol{Z}^-)$ and (v) B denotes the batch of the samples. Additionally, as per the suggestion in [148], we also train our model with a combination of L_{NP} and $\mathrm{L}_{\mathrm{Ang}}$ as shown below:

$$
\mathrm{L}_{\mathrm{Ang\_NP}} = \mathrm{L}_{\mathrm{NP}} + \lambda_{Ang}\,\mathrm{L}_{\mathrm{Ang}} \quad , \tag{6.5}
$$

where λ_{Ang} is set to 2 in our experiments.

Training Loss: As observed, the aforementioned local loss functions enforce *local* constraints in their respective formulation and fail to enforce the dissimilarity constraints in learning a discriminative embedding space. Therefore, we augment these loss functions with L^{φ} so as to holistically integrate these constraints in terms of a class-wise discrepancy measure within the mini-batch B. The final training loss is defined as follows:

$$
\mathrm{L}_{\mathrm{Train}} = \mathrm{L}_{\mathrm{Local}} + \lambda\,\mathrm{L}^{\varphi}, \tag{6.6}
$$

where $\mathrm{L}_{\mathrm{Local}}$ is either $\mathrm{L}_{\mathrm{Trip}}$, L_{NP}, $\mathrm{L}_{\mathrm{Ang}}$ or $\mathrm{L}_{\mathrm{Ang\_NP}}$ corresponding to Triplet, NPairs, Angular and the combination of NPairs and Angular loss respectively[1].

6.4 Experiments

Implementation Details

We implemented DCDL in PyTorch [111]. We used VGG-9 [82] as the backbone network with a fully-connected, embedding layer at the end. We randomly initialize ($\mathrm{U}\,[0, 1]$)) all the layers in the network. The dimension of the embedding layer is fixed to 64. A dropout layer with a dropout rate of 0.1 is added before the embedding layer. When $\mathrm{L}_{\mathrm{Local}}$=$\mathrm{L}_{\mathrm{Trip}}$, we ensure that there are at least 10 data samples from every class; thereby resulting in a mini-batch of 100 for the Cifar-10

[1] Similar to $\mathrm{L}_{\mathrm{Train}}$, [87] also make use of a combination of various metric loss functions along with KL divergence loss to learn well separated but compact clusters

and STL-10 datasets, and 200 for the Cifar-100 dataset respectively. Likewise, when $\mathrm{L}_{\mathrm{Local}} = \{\mathrm{L}_{\mathrm{NP}}, \mathrm{L}_{\mathrm{Ang}}, \mathrm{L}_{\mathrm{Ang\_NP}}\}$, we guarantee that there are at-least 2 data samples from every class within a mini-batch. This leads to a mini-batch of size 20 for the Cifar-10 and STL-10 datasets, and 40 for the Cifar-100 dataset respectively for these loss functions. We have not used any data augmentation techniques for the train and test sets across all the datasets[2]. In our empirical evaluations with L_{NP}, we have observed that the Stochastic Gradient Descent (SGD) optimizer with Momentum attained the best accuracy on the test set in the Cifar-10 dataset; whereas for the others we have used the Adam [67] optimizer to fine-tune the parameters of the network. The initial learning rate for the Adam and the SGD optimizer is set to 5×10^{-4} and 1×10^{-2} respectively. We decay the learning rate by a factor of 0.1 after every 50 epochs. We compare and evaluate our proposed algorithm (*i.e*. DCDL) against several baseline algorithms, namely (i) Trip [124], (ii) NPairs (NP) [129], (iii) Angular (Ang) [148] and (iv) Angular-Npairs (Ang_NP) [148]. In all of our experiments, we train our models for 100 epochs with $\mathrm{L}_{\mathrm{Train}}$ as defined in Eqn. (6.3). Further, we evaluate the discriminative ability of an embedding space by learning a linear classifier on the embeddings of the training set and report the classification accuracy on the test set. We set the value of ϵ to 2.5×10^{-3} [32] for experiments with $\mathrm{L}^{\mathcal{W}}$ for Sinkhorn divergence. Further, we set the value of **(a)** τ in $\mathrm{L}_{\mathrm{Trip}}$ to 0.5, **(b)** α in $\mathrm{L}_{\mathrm{Ang}}$ and $\mathrm{L}_{\mathrm{Ang\_NP}}$ to $30°$ and $45°$, respectively. We report the best results (*i.e*. the classification accuracy in % on the test set) obtained for all the hyper-parameter settings after training the models for 100 epochs for all the datasets across different experimental settings (*i.e*. Table 6.1 to Table 6.8).

6.4.1 Analysis of Coarse-Grained Datasets

Notation: While reporting the quantitative results, we have used the following notation to avoid any confusion. We represent the calculation of L^{φ} with Laplacian and Gaussian kernels for MMD as L^{L} and L^{G} respectively; with the value of σ fixed to 0.05 across all the experiments [51]. Similarly, $\mathrm{L}^{\mathcal{W}}$ denotes the calculation of L^{φ} with $\mathcal{W}_{\epsilon}^{p}(\mathcal{U}, \mathcal{V})$ such that $p=2$ and $\boldsymbol{D}^{p}(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2} \|\boldsymbol{u} - \boldsymbol{v}\|_2^2$).

We report the test accuracy for two primary experimental setups; **(a) clean** labels; where the ground truth labels are not corrupted manually by any noise-transition matrix, and **(b) noisy** labels; where we artificially corrupt the ground truth labels of the training set across all the datasets, while the test labels are unaltered. The transition of labels is parameterized by $\delta \in [0, 1]$ such that the probability of the true and the noisy class labels are δ and $1 - \delta$ respectively. Noisy labels can be further categorized as follows:

- **Symmetric (Sym) Noise:** Similar to [83], in this experimental setup we corrupt the ground truth train labels with a random one-hot vector with a probability of $1 - \delta$. An exemplar noise transition matrix for the symmetric

[2]The training and test images of STL-10 are resized to $32 \times 32 \times 3$.

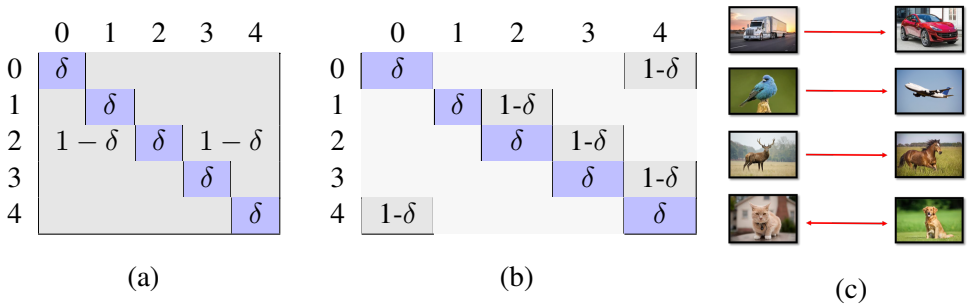


Figure 6.2: Exemplar Noise Transition matrix for 5 classes for (a) **Symmetric** and (b) **Asymmetric** Noise. δ denotes the percentage of noise for both settings. (c) Transition class pairs for Asymmetric noise.

noise is shown in Fig. 6.2 **(a)**. We evaluate DCDL for two values of δ, *i.e.*, $\{0.1, 0.3\}$ for each of the three datasets.

- **Asymmetric (Asym) Noise:** Asymmetric noise takes into account realistic real life mistakes, thus confusing similar classes within the dataset. Similar to the asymmetric noise protocol of [83], we also create the following class transition pairs for the Cifar-10 and STL-10 datasets[3]; (i) TRUCK $\rightarrow$ AUTOMOBILE, (ii) BIRD $\rightarrow$ AIRPLANE, (iii) DEER $\rightarrow$ HORSE and (iv) CAT $\leftrightarrow$ DOG. An exemplar noise transition matrix for the asymmetric noise is shown in Fig. 6.2 **(b)**. The value of δ is set to $\{0.1, 0.2\}$ in our experiments for this setting.

Table 6.1: Importance of L^{L}, L^{G} and $\mathrm{L}^{\mathcal{W}}$ (final training loss Eqn. (6.6)) in clean and noisy labels settings with different datasets Cifar10 and STL-10.

		Clean	Symmetric		Asymmetric	
Dataset	Method	δ=0.0	δ=0.1	δ=0.3	δ=0.1	δ=0.2
S-10	L^{L}	42.8	40.5	33.6	41.0	40.3
	L^{G}	42.9	39.6	33.5	41.3	40.7
	$\mathrm{L}^{\mathcal{W}}$	47.2	46.2	44.6	46.4	45.5
C-10	L^{L}	78.2	74.4	52.4	71.5	69.5
	L^{G}	78.0	73.5	52.7	71.5	69.3
	$\mathrm{L}^{\mathcal{W}}$	80.7	77.1	55.0	74.3	71.6

6.4.2 Ablation

Importance of L^{L}, L^{G} and $\mathrm{L}^{\mathcal{W}}$: We first begin by providing a detailed insight into the effectiveness of using MMD with Laplacian and Gaussian kernels, as well

[3]We do not evaluate on the Cifar-100 dataset for the Asymmetric Noise setting.

Table 6.2: Classification results obtained from different values of λ in the clean labels setting for S-10 dataset in the clean labels setting.

	λ	0.0	0.01	0.1	0.2	0.5	0.7	1.0	5.0
$\mathrm{L_{Trip}}$	$+\mathrm{L^L}$	59.9	59.4	60.9	**61.6**	61.2	61.6	60.8	58.5
	$+\mathrm{L}^{\mathcal{W}}$		64.1	63.8	66.1	**66.1**	65.8	65.3	63.6
$\mathrm{L_{NP}}$	$+\mathrm{L^L}$	44.8	42.7	45.7	**45.4**	43.4	43.1	42.9	41.9
	$+\mathrm{L}^{\mathcal{W}}$		46.9	48.5	49.7	**49.8**	49.4	48.3	45.5

as the Wasserstein Distance in enforcing a class-wise discrepancy loss in both the clean and the noisy labels experimental setups. Specifically, we train our models with $\mathrm{L_{Train}} = \mathrm{L}^{\varphi}$; where $\mathrm{L}^{\varphi} = \left\{\mathrm{L^L}, \mathrm{L^G}, \mathrm{L}^{\mathcal{W}}\right\}$ by discarding $\mathrm{L_{Local}}$ from Eqn. (6.6) and setting the value of λ to 1.0. The results for STL-10 and Cifar-10 datasets are reported in Table 6.1. It is observed that across all the settings; the Wasserstein Distance based discrepancy measure (*i.e.* $\mathrm{L}^{\mathcal{W}}$) outperforms MMD based $\mathrm{L^L}$ and $\mathrm{L^G}$ in terms of classification accuracy, thus being able to exploit the underlying geometry to a better extent over MMD, as, unlike Wasserstein distances, the norm of MMD scales up as the batch size is increased with a low sample complexity [32]. Moreover, MMD generally seems to suffer from the drawback of vanishing gradient, while such difficulty rarely occurs in learning with Wasserstein distances [113].. Empirically, it is also observed in the majority of the cases that $\mathrm{L^L}$ obtains similar or outperforms $\mathrm{L^G}$ across both the datasets. Thus, from here onwards we only report the results of MMD with the Laplacian kernel (*i.e.* $\mathrm{L^L}$).

Importance of λ: Table 6.2 shows the classification results obtained for various values of λ on STL-10 dataset in the clean labels setting. The optimal value of λ is 0.2 and 0.5 for $\mathrm{L^L}$ and $\mathrm{L}^{\mathcal{W}}$ respectively; which have been chosen to report the results in the subsequent sections. One can observe that the classification accuracy increases when λ is increased from 0.01 with the optimum accuracy being attained at their respective chosen values of λ. Thereafter it drops when λ is increased beyond 1.0. One can also see improvements over $\mathrm{L_{Trip}}$ and $\mathrm{L_{NP}}$ (especially for $\mathrm{L}^{\mathcal{W}}$) even when the value λ is outside its optimal range. There also exists a large performance between $\mathrm{L_{Trip}}$ and $\mathrm{L_{NP}}$ for λ=0. One plausible explanation is that $\mathrm{L_{NP}}$ considers all the negative samples for a given pair of anchor and positive sample in calculating $\mathrm{L_{NP}}$ (Refer to Section § 6.3 for more details regarding $\mathrm{L_{NP}}$). As a result, some of these negative samples might be detrimental to the optimization procedure as they might be too difficult to separate given the anchor-positive pair. Further, $\mathrm{L_{NP}}$ can form only one fixed positive pair per anchor sample within the mini-batch, thereby resulting in less variability in the positive pair of samples so as to overcome the adverse effects caused by the presence of large number of negative samples within the training batch. On the other hand, $\mathrm{L_{Trip}}$ enforces semi-hard triplet mining to select the most informative triplets in training the model; thereby outperforming $\mathrm{L_{NP}}$ by a significant margin.

Embedding Size: We have performed experiments with different dimensions of the embedding space on the STL-10 dataset with $\mathrm{L_{Local}}$ set to $\mathrm{L_{Trip}}$ in the clean labels

setting. When trained with L^{L}, we obtain 61.1%, 61.6%, 61.7% and 60.5% accuracy for 32, 64, 128 and 256 dimensional embedding space respectively. Similarly for $\mathrm{L}^{\mathcal{W}}$, we observed 64.2%, 66.1%, 64.7% and 64.3% accuracy respectively. This verifies that a 64 dimensional embedding space obtains superior results for $\mathrm{L}^{\mathcal{W}}$, while achieving competitive performance against a 128 dimensional embedding space for L^{L}. We have therefore set the dimensionality of the embedding space to 64 for all the datasets in the forthcoming sections.

More samples per class: We have also conducted experiments with 1000 samples for STL-10 dataset (*i.e.* 100 samples per class) and obtained 63.7% accuracy on its test set against 66.1% when trained with 100 samples (*i.e.* 10 samples per class) per mini-batch respectively with $\mathrm{L}_{\mathrm{Local}}$ set to $\mathrm{L}_{\mathrm{Trip}}$ in the clean labels setting. Similar to [126], this study clearly demonstrates that DCDL is able to approximate the underlying distribution with fewer samples per mini-batch. In general, assuming that larger batches should always lead to better results does not hold as shown in [40, 98]. One intuitive way of understanding this is to think of gradients of small batches as noisy gradients which can help escaping from bad local minima.

Note: In the subsequent section, σ and ϵ are set to 0.05, 2.5×10^{-3} respectively. The dimension of the embedding is set to 64. The value of λ is set to 0.2 and 0.5 for all the experiments with L^{L} and $\mathrm{L}^{\mathcal{W}}$ respectively. We also sample 10 samples per class for every dataset.

Table 6.3: Experimental results demonstrating the importance of incorporating L^{L} and $\mathrm{L}^{\mathcal{W}}$ in Eqn. (6.6) across the three datasets in the **clean** labels setting.

Method \ Dataset	C-10	C-100	S-10
$\mathrm{L}_{\mathrm{Trip}}$	85.9	68.4	59.9
$\mathrm{L}_{\mathrm{Trip}}$ + L^{L}	87.5	69.2	61.6
$\mathrm{L}_{\mathrm{Trip}}$ + $\mathrm{L}^{\mathcal{W}}$	87.4	70.1	66.1
L_{NP}	68.6	42.6	44.8
L_{NP} + L^{L}	69.9	44.6	45.4
L_{NP} + $\mathrm{L}^{\mathcal{W}}$	70.2	45.5	49.8
$\mathrm{L}_{\mathrm{Ang}}$	85.7	63.0	54.6
$\mathrm{L}_{\mathrm{Ang}}$ + L^{L}	86.6	63.6	57.0
$\mathrm{L}_{\mathrm{Ang}}$ + $\mathrm{L}^{\mathcal{W}}$	87.2	65.2	59.4
$\mathrm{L}_{\mathrm{Ang\_NP}}$	84.5	61.6	53.4
$\mathrm{L}_{\mathrm{Ang\_NP}}$ + L^{L}	85.3	63.6	55.9
$\mathrm{L}_{\mathrm{Ang\_NP}}$ + $\mathrm{L}^{\mathcal{W}}$	87.1	66.1	57.4

Clean Labels Dataset Image Classification:

Here, we assess the effectiveness of our proposed algorithm DCDL in the clean labels settings. The results on the three datasets are shown in Table 6.3 (δ set to 0.0). As observed, enforcing discrepancy between the distributions of the class-wise embeddings leads to an enhancement in the discriminative ability of the learnt embedding space. Further, $\mathrm{L}^{\mathcal{W}}$ achieves significant performance gains over L^{L} for different assignments of $\mathrm{L}_{\mathrm{Local}}$ losses (Refer to Eqn. (6.6)). Thus it is indeed evident that one must enforce a class-wise discrepancy loss that aims to separate the probability distributions of the embedded points for every class in order to learn well-separated, yet compact clusters in the embedding space.

Table 6.4: Experimental results demonstrating the importance of incorporating L^{L} and $\mathrm{L}^{\mathcal{W}}$ in Eqn. (6.6) across the three datasets for different values of δ in the **symmetric** noisy label setting.

	δ=0.1			δ=0.3		
Method \ Dataset	C-10	C-100	S-10	C-10	C-100	S-10
$\mathrm{L}_{\mathrm{Trip}}$	80.4	58.9	48.4	71.1	33.8	39.8
$\mathrm{L}_{\mathrm{Trip}}$ + L^{L}	82.2	60.9	51.8	72.7	34.7	41.2
$\mathrm{L}_{\mathrm{Trip}}$ + $\mathrm{L}^{\mathcal{W}}$	82.8	61.9	52.5	74.3	36.7	43.6
L_{NP}	65.5	36.9	39.8	51.1	23.3	32.9
L_{NP} + L^{L}	65.4	37.5	44.0	50.9	26.2	35.2
L_{NP} + $\mathrm{L}^{\mathcal{W}}$	67.1	38.1	48.4	51.4	27.4	43.2
$\mathrm{L}_{\mathrm{Ang}}$	80.8	49.9	46.8	63.2	25.1	35.2
$\mathrm{L}_{\mathrm{Ang}}$ + L^{L}	81.4	51.1	48.1	64.8	26.3	37.8
$\mathrm{L}_{\mathrm{Ang}}$ + $\mathrm{L}^{\mathcal{W}}$	82.1	53.7	49.3	66.3	28.6	40.8
$\mathrm{L}_{\mathrm{Ang\_NP}}$	80.7	49.5	46.3	64.2	24.6	35.7
$\mathrm{L}_{\mathrm{Ang\_NP}}$ + L^{L}	81.6	51.2	47.2	65.5	25.8	37.3
$\mathrm{L}_{\mathrm{Ang\_NP}}$ + $\mathrm{L}^{\mathcal{W}}$	82.1	52.7	48.7	67.1	28.1	38.5

Noisy Labels Dataset Image Classification:

In this section, we evaluate DCDL in the two noisy labels settings as described before. Table 6.4 and 6.5 show the results reported on the three datasets for various configurations of noise and $\mathrm{L}_{\mathrm{Local}}$ in the symmetric and asymmetric noisy labels settings respectively. We can easily observe that barring a few configuration settings; the accuracy obtained with symmetric noise is lower than the one obtained via asymmetric noise corruption. It is not surprising as the transitional categories for asymmetric noise are visually quite similar to each other, and so are the features extracted and learnt by our network; thereby leading to improved results over symmetric-noise settings. Moreover, it is also observed that adding L^{L} and $\mathrm{L}^{\mathcal{W}}$ to $\mathrm{L}_{\mathrm{Local}}$ outperform its corresponding conventional local loss functions (*i.e.* $\mathrm{L}_{\mathrm{Local}}$) by a considerable margin in terms of classification accuracy, thereby comprehensively

Table 6.5: Experimental results (in terms of test classification accuracy) demonstrating the importance of incorporating L^{L} and $\mathrm{L}^{\mathcal{W}}$ in Eqn. (6.6) across the three datasets for different values of δ in the **asymmetric** noisy labels settings.

	δ=0.1		δ=0.2	
Method \ Dataset	C-10	S-10	C-10	S-10
$\mathrm{L}_{\mathrm{Trip}}$	82.5	55.2	78.4	54.6
$\mathrm{L}_{\mathrm{Trip}}$ + L^{L}	84.1	57.5	79.7	55.8
$\mathrm{L}_{\mathrm{Trip}}$ + $\mathrm{L}^{\mathcal{W}}$	84.9	59.0	81.7	57.9
L_{NP}	64.2	43.7	53.9	42.1
L_{NP} + L^{L}	65.9	46.4	55.2	45.5
L_{NP} + $\mathrm{L}^{\mathcal{W}}$	67.2	49.8	56.9	47.3
$\mathrm{L}_{\mathrm{Ang}}$	82.0	53.5	78.6	50.6
$\mathrm{L}_{\mathrm{Ang}}$ + L^{L}	83.3	55.1	79.7	51.3
$\mathrm{L}_{\mathrm{Ang}}$ + $\mathrm{L}^{\mathcal{W}}$	85.1	56.2	82.0	53.6
$\mathrm{L}_{\mathrm{Ang\_NP}}$	81.7	52.1	78.2	50.9
$\mathrm{L}_{\mathrm{Ang\_NP}}$ + L^{L}	82.5	52.7	79.6	50.3
$\mathrm{L}_{\mathrm{Ang\_NP}}$ + $\mathrm{L}^{\mathcal{W}}$	83.2	54.6	81.7	53.9

demonstrating the need of enforcing such class-wise discrepancy constraints in learning a superior embedding in the noisy labels settings. Similar to the results obtained in § 6.4.2, our empirical evaluations again show that $\mathrm{L}^{\mathcal{W}}$ leads to superior results over L^{L} across the various experimental setups.

Computational complexity: The computational complexity of the regularized Sinkhorn Divergence (SD) is $\mathcal{O}(\mathrm{N}_{\mathrm{B}}^2)$, where N_{B} is the number of samples in a mini-batch. However, a big portion of the computational training time is devoted to computing the pair-wise distances between samples. Such pairwise distances are calculated in $\mathrm{L}_{\mathrm{Local}}$ which can be used explicitly in the SD module; thereby reducing its overall computational overhead. As observed, the average computational time per training epoch is 0.53, 0.57 and 0.61 seconds for $\mathrm{L}_{\mathrm{Trip}}$ and its MMD (*i.e.* $\mathrm{L}_{\mathrm{Trip}}+\mathrm{L}^{\mathrm{L}}$) and Wasserstein Distance (*i.e.* $\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathcal{W}}$) counterparts respectively when trained on the STL-10 dataset with a 64 dimensional embedding space in the clean labels settings. This clearly shows that the additional computational overhead of SD is very small due to the sharing of pairwise distance calculations among $\mathrm{L}_{\mathrm{Local}}$ and L^{φ}, while the inference time remains the same. We acknowledge that our method incurs higher computational cost compared to basic noisy-label learning approaches; however, this additional cost leads to substantial performance gains. Moreover, when compared to methods achieving similar performance—such as JC [150]—our approach demonstrates shorter training time under identical experimental settings. We also observed that batch size plays a crucial role in performance: larger batch sizes enhance accuracy (as shown in Section 5.4.2) but inevitably increase training

time. For fairness, we used the same batch size when comparing our method with baselines. Overall, the computational overhead of our approach is reasonable given the improved effectiveness.

Experiments with Cross Entropy Loss:

In this section, we examine the compatibility of adding the conventional cross entropy loss to the training loss $\mathrm{L}_{\mathrm{Train}}$ (Eqn. (6.6)) used for DCDL. Thus, our new loss is defined as follows:

$$\mathrm{L}^{\mathrm{xent}}_{\mathrm{Train}} = \mathrm{L}_{\mathrm{Train}} + \lambda_{\mathrm{xent}}\, \mathrm{L}_{\mathrm{xent}}. \tag{6.7}$$

Table 6.6: Experimental results while training with $\mathrm{L}^{\mathrm{xent}}_{\mathrm{Train}}$ in the **clean** labels setting with three different datasets..

Method \ Dataset	C-10	C-100	S-10
$\mathrm{L}_{\mathrm{xent}}$	86.3	70.9	65.7
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}_{\mathrm{xent}}$	87.2	71.2	67.9
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathrm{L}} + \mathrm{L}_{\mathrm{xent}}$	87.8	72.4	69.2
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathcal{W}} + \mathrm{L}_{\mathrm{xent}}$	87.9	72.1	68.8
$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}_{\mathrm{xent}}$	87.3	72.1	67.7
$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathrm{L}} + \mathrm{L}_{\mathrm{xent}}$	87.2	71.9	68.2
$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathcal{W}} + \mathrm{L}_{\mathrm{xent}}$	89.5	72.7	67.9
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}_{\mathrm{xent}}$	87.1	71.5	68.2
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathrm{L}} + \mathrm{L}_{\mathrm{xent}}$	87.6	72.0	68.9
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathcal{W}} + \mathrm{L}_{\mathrm{xent}}$	87.9	71.8	68.5

We have fixed the value of λ_{xent} to 1.0 in all our experiments. The results are shown in Table 6.6, 6.7 and 6.8 in the clean, symmetric-noisy and asymmetric-noisy labels settings respectively. As observed, and to no surprise, training our models with $\mathrm{L}^{\mathrm{xent}}_{\mathrm{Train}}$ leads to a significant overall improvement in terms of classification accuracy across all the datasets and experimental settings. Having said that, $\mathrm{L}^{\mathcal{W}}$ still outperforms L^{L} in most of the settings except a few, in the presence/absence of noise. To some degree, $\mathrm{L}_{\mathrm{xent}}$ is a partial global loss as it aims to find class-separating sub-spaces in the output (or embedding) space in order to increase the inter-class distances between the embedded points. However, it fails to guarantee that the intra-class distances are reduced within these hyper-planes. With the addition of $\mathrm{L}_{\mathrm{Local}}$ such intra-class distances are reduced to some extent; but this too fails to ensure that the class-separating sub-spaces are also distant from one other. Further addition of the discrepancy loss in terms of L^{φ} leads to an increase in the separation between the class-separating sub-spaces due to the disparity induced between the distributions of class-wise embeddings for every class. This indeed demonstrates

the need and the ability of L^{φ} to learn discriminant embeddings even in the presence of a partially global cross entropy loss.

6.4.3 Analysis of Fine Grained Image Classification

Implementation Details:

We have used Inception-V1 [136] with Batch Normalization [60] pretrained on Imagenet [120] as the backbone network, with a randomly initialized [39] fully-connected embedding layer at the end[4]. Similar to [79], we have fixed the dimensionality of the embedding to 100 and 98 respectively for CUBS-200-2011 and CARS196. All the images are resized to 256×256 initially. The training images are then randomly cropped to 227×227, followed by random horizontal flipping; whereas the test images are cropped to 227×227 from the center. A Dropout layer with dropout rate of 0.1 is added before the embedding layer. The size of the mini-batch is set to 64 for both datasets. In our experiments with $\mathrm{L}_{\mathrm{Local}} = \mathrm{L}_{\mathrm{Trip}}$, we ensure that there are at-least 4 samples per class within the mini-batch B; while for the rest we use 2 samples per class in order to create the mini-batch. The entire network is fined tuned using the Stochastic Gradient Descent (SGD) optimizer. The initial learning rate is set to 0.01 for all datasets, and is decreased by a factor of 0.1 after every 25 epochs. However, empirically we have found that λ=0.5 gives superior results for experiments with L^{L} and $\mathrm{L}^{\mathcal{W}}$. Moreover, the value of ϵ is also set to 2.5×10^{-3}. We set the value of **(a)** τ in $\mathrm{L}_{\mathrm{Trip}}$ to 0.5, **(b)** α in $\mathrm{L}_{\mathrm{Ang}}$ and $\mathrm{L}_{\mathrm{Ang\_NP}}$ to $45°$. We report the best results obtained for all the hyper-parameter settings after training the models for 40 epochs.

In the evaluation phase, we apply the standard KMeans algorithm on the unit-

[4]Like [79], we also pre-train the entire network with a classification loss before fine-tuning

Table 6.7: Experimental results while training with $\mathrm{L}^{\mathrm{xent}}_{\mathrm{Train}}$ in the **symmetric** noisy labels setting on three different datasets.

	δ=0.1			δ=0.3		
Method \ Dataset	C-10	C-100	S-10	C-10	C-100	S-10
$\mathrm{L}_{\mathrm{xent}}$	81.4	64.3	60.8	76.1	55.1	51.1
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}_{\mathrm{xent}}$	83.1	66.5	62.8	77.1	58.2	52.1
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathrm{L}} + \mathrm{L}_{\mathrm{xent}}$	83.0	66.3	63.1	78.2	58.7	51.9
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathcal{W}} + \mathrm{L}_{\mathrm{xent}}$	83.5	67.1	63.9	78.9	59.0	52.9
$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}_{\mathrm{xent}}$	83.2	65.8	63.6	77.5	58.8	53.4
$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathrm{L}} + \mathrm{L}_{\mathrm{xent}}$	83.7	66.2	63.9	78.9	60.1	54.1
$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathcal{W}} + \mathrm{L}_{\mathrm{xent}}$	83.5	66.9	64.7	79.8	59.8	55.2
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}_{\mathrm{xent}}$	83.2	66.3	63.4	77.9	56.6	52.8
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathrm{L}} + \mathrm{L}_{\mathrm{xent}}$	83.6	66.1	64.5	78.6	57.7	53.2
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathcal{W}} + \mathrm{L}_{\mathrm{xent}}$	83.7	64.7	64.6	78.9	57.5	54.1

Table 6.8: Experimental results while training with $\mathrm{L}_{\mathrm{Train}}^{\mathrm{xent}}$ in the **asymmetric** noisy labels setting on two different datasets.

	δ=0.1		δ=0.2	
Method \ Dataset	C-10	S-10	C-10	S-10
$\mathrm{L}_{\mathrm{xent}}$	83.8	63.9	80.7	61.4
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}_{\mathrm{xent}}$	84.1	65.1	80.9	62.7
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathrm{L}} + \mathrm{L}_{\mathrm{xent}}$	84.6	65.8	81.5	64.1
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathcal{W}} + \mathrm{L}_{\mathrm{xent}}$	84.3	66.4	82.3	63.8
$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}_{\mathrm{xent}}$	84.4	64.9	81.9	63.1
$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathrm{L}} + \mathrm{L}_{\mathrm{xent}}$	84.8	66.1	82.6	63.7
$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathcal{W}} + \mathrm{L}_{\mathrm{xent}}$	84.2	65.8	84.1	64.0
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}_{\mathrm{xent}}$	83.9	65.1	82.4	62.4
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathrm{L}} + \mathrm{L}_{\mathrm{xent}}$	84.4	65.9	82.2	64.2
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathcal{W}} + \mathrm{L}_{\mathrm{xent}}$	84.8	66.4	82.6	64.1

Table 6.9: Experimental results on CUBS-200-2011 in the **clean** and **symmetric**-noisy label settings.

	Clean Labels								
Method	NMI	R@1	R@2	R@4	Method	NMI	R@1	R@2	R@4
$\mathrm{L}_{\mathrm{Trip}}$	55.4	42.6	55.1	66.4	L_{NP}	57.2	45.4	58.4	69.5
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathrm{L}}$	56.2	43.3	55.2	66.8	$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathrm{L}}$	57.4	45.3	58.3	70.6
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathcal{W}}$	56.9	44.1	56.1	67.5	$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathcal{W}}$	57.9	45.9	58.5	70.5
$\mathrm{L}_{\mathrm{Ang}}$	57.7	45.7	58.1	70.3	$\mathrm{L}_{\mathrm{Ang\_NP}}$	57.9	45.9	58.3	70.4
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathrm{L}}$	58.1	46.4	58.7	70.8	$\mathrm{L}_{\mathrm{Ang\_NP}} + \mathrm{L}^{\mathrm{L}}$	58.1	46.4	58.7	70.5
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathcal{W}}$	58.5	46.3	58.7	70.7	$\mathrm{L}_{\mathrm{Ang\_NP}} + \mathrm{L}^{\mathcal{W}}$	58.3	46.2	58.5	70.4
	Symmetric Noise with δ=0.1								
$\mathrm{L}_{\mathrm{Trip}}$	53.9	41.7	53.8	64.5	L_{NP}	55.2	43.7	56.3	68.3
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathrm{L}}$	54.3	42.3	54.1	65.0	$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathrm{L}}$	56.9	45.2	57.5	68.5
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathcal{W}}$	54.7	42.5	54.4	65.4	$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathcal{W}}$	57.3	44.8	57.9	69.8
$\mathrm{L}_{\mathrm{Ang}}$	56.9	45.0	57.1	69.5	$\mathrm{L}_{\mathrm{Ang\_NP}} + \mathrm{L}^{\mathrm{L}}$	57.2	45.7	58.7	70.1
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathrm{L}}$	57.4	45.4	58.0	69.8	$\mathrm{L}_{\mathrm{Ang\_NP}} + \mathrm{L}^{\mathrm{L}}$	57.9	45.6	57.8	69.8
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathcal{W}}$	58.0	45.5	58.1	70.4	$\mathrm{L}_{\mathrm{Ang\_NP}} + \mathrm{L}^{\mathcal{W}}$	57.8	45.7	56.9	69.6
	Symmetric Noise with δ=0.3								
$\mathrm{L}_{\mathrm{Trip}}$	52.1	40.8	52.6	62.8	L_{NP}	54.7	42.0	54.6	66.6
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathrm{L}}$	52.9	41.4	53.1	62.6	$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathrm{L}}$	55.9	43.1	55.3	67.2
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathcal{W}}$	53.7	42.2	54.0	63.9	$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathcal{W}}$	56.1	43.8	56.3	67.8
$\mathrm{L}_{\mathrm{Ang}}$	55.4	42.1	55.1	67.3	$\mathrm{L}_{\mathrm{Ang\_NP}}$	56.5	42.9	55.9	68.0
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathrm{L}}$	56.2	42.8	56.0	68.0	$\mathrm{L}_{\mathrm{Ang\_NP}} + \mathrm{L}^{\mathrm{L}}$	56.1	43.5	56.1	67.9
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathcal{W}}$	56.5	43.0	55.9	68.2	$\mathrm{L}_{\mathrm{Ang\_NP}} + \mathrm{L}^{\mathcal{W}}$	56.2	43.7	56.3	68.6

norm output representations and calculate the conventional NMI and R@K metrics similar to [107, 129, 130]. Like before, we evaluate DCDL in two different label settings: **(a) clean** labels and **(b) noisy** labels; with symmetric noise (δ=$\{0.1, 0.3\}$).

Table 6.10: Experimental results on CARS196 dataset in the **clean** and **symmetric**-noisy label settings.

	Clean Labels								
	NMI	R@1	R@2	R@4		NMI	R@1	R@2	R@4
$\mathrm{L}_{\mathrm{Trip}}$	55.4	58.1	69.8	79.5	L_{NP}	56.9	60.8	72.5	80.9
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathrm{L}}$	55.9	58.9	70.7	80.5	$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathrm{L}}$	57.3	62.1	73.2	81.7
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathcal{W}}$	56.3	58.6	70.8	80.9	$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathcal{W}}$	57.2	62.0	73.3	81.6
$\mathrm{L}_{\mathrm{Ang}}$	57.1	60.9	71.7	81.0	$\mathrm{L}_{\mathrm{Ang\_NP}}$	57.1	61.2	72.4	81.2
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathrm{L}}$	57.5	61.5	72.9	81.5	$\mathrm{L}_{\mathrm{Ang\_NP}} + \mathrm{L}^{\mathrm{L}}$	57.4	61.9	72.9	81.7
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathcal{W}}$	57.9	61.8	72.8	81.2	$\mathrm{L}_{\mathrm{Ang\_NP}} + \mathrm{L}^{\mathcal{W}}$	57.3	61.8	72.8	81.6
	Symmetric Noise with δ=0.1								
$\mathrm{L}_{\mathrm{Trip}}$	54.5	58.5	69.6	79.4	L_{NP}	54.7	58.6	70.5	79.9
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathrm{L}}$	54.8	58.7	70.4	79.7	$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathrm{L}}$	55.5	59.1	70.6	79.7
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathcal{W}}$	55.3	58.9	70.4	80.2	$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathcal{W}}$	55.4	59.2	71.1	80.1
$\mathrm{L}_{\mathrm{Ang}}$	54.2	58.3	70.3	79.2	$\mathrm{L}_{\mathrm{Ang\_NP}}$	54.7	58.2	70.1	79.4
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathrm{L}}$	55.0	58.3	70.0	79.3	$\mathrm{L}_{\mathrm{Ang\_NP}} + \mathrm{L}^{\mathrm{L}}$	55.2	59.3	71.4	80.5
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathcal{W}}$	54.7	59.5	70.8	80.0	$\mathrm{L}_{\mathrm{Ang\_NP}} + \mathrm{L}^{\mathcal{W}}$	55.6	59.6	70.7	80.2
	Symmetric Noise with δ=0.3								
$\mathrm{L}_{\mathrm{Trip}}$	49.3	51.1	62.8	73.8	L_{NP}	49.2	50.8	62.6	73.6
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathrm{L}}$	49.6	52.1	64.1	75.4	$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathrm{L}}$	49.7	51.1	63.4	74.3
$\mathrm{L}_{\mathrm{Trip}} + \mathrm{L}^{\mathcal{W}}$	49.3	52.1	64.0	74.9	$\mathrm{L}_{\mathrm{NP}} + \mathrm{L}^{\mathcal{W}}$	49.9	51.9	64.3	74.9
$\mathrm{L}_{\mathrm{Ang}}$	50.1	51.4	63.2	73.9	$\mathrm{L}_{\mathrm{Ang\_NP}}$	49.5	50.8	63.2	73.9
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathrm{L}}$	50.2	51.3	63.7	74.7	$\mathrm{L}_{\mathrm{Ang\_NP}} + \mathrm{L}^{\mathrm{L}}$	50.1	51.6	63.7	74.6
$\mathrm{L}_{\mathrm{Ang}} + \mathrm{L}^{\mathcal{W}}$	50.3	52.2	64.5	74.9	$\mathrm{L}_{\mathrm{Ang\_NP}} + \mathrm{L}^{\mathcal{W}}$	49.9	51.4	64.2	74.4

Table 6.9 and 6.10 shows the results with $\mathrm{L}_{\mathrm{Local}} = \{\mathrm{L}_{\mathrm{Trip}},\ \mathrm{L}_{\mathrm{NP}},\ \mathrm{L}_{\mathrm{Ang}},\ \mathrm{L}_{\mathrm{Ang\_NP}}\}$ for the clean and symmetric-noisy labels settings on the CUBS-200-2011 and CARS196 datasets respectively. It is again evident that adding both L^{L} and $\mathrm{L}^{\mathcal{W}}$ (so as to introduce class-wise discrepancy constraints) leads to superior NMI and R@K metrics for both the datasets across most of the settings (*i.e.* clean and noisy labels). It further reinforces our hypothesis of the need to incorporate and enforce such class-wise discrepancy constraints in order to learn a discriminative embedding space. Furthermore, DSC [79] designs a complex structured loss that takes into consideration the spectral clustering constraints to separate the class-wise embeddings and achieves $56.1\%/57.1\%$ (NMI/R@1) on the CARS196 dataset. On the other hand, simply by precisely enforcing the discrepancy constraints with either a simpler L_{NP}, $\mathrm{L}_{\mathrm{Ang}}$, or $\mathrm{L}_{\mathrm{Ang\_NP}}$ outperforms DSC in terms of NMI/R@1 respectively for both L^{L} and $\mathrm{L}^{\mathcal{W}}$. These results clearly indicate that modeling comprehensive class-wise discrepancy constraints using L^{L} and $\mathrm{L}^{\mathcal{W}}$ to **push-away** the class-wise probability distributions are indeed important to learn a discriminative embedding space in the presence/absence of noisy labels.

Comparison to State of the Art Deep Metric Learning methods:

In this section, we study the effects of equipping the loss function of the current state-of-the-art **MIC** [118] algorithm with L^{L} and $\mathrm{L}^{\mathcal{W}}$. MIC proposes to use a separate encoder trained with a novel surrogate task to incorporate structure inter-class

Table 6.11: Comparison against several baselines and state-of-the-art algorithms. The best results are shown in red.

Dataset	CUBS-200-2011				CARS196			
Method	NMI	R@1	R@2	R@4	NMI	R@1	R@2	R@4
Trip	55.4	42.6	54.9	66.4	53.4	51.5	63.8	73.5
DSC	58.1	49.8	62.6	73.6	58.0	59.4	71.3	80.6
Lifetd-Struct	56.5	43.6	56.6	68.6	56.9	53.0	65.7	76.0
NPairs	57.2	45.4	58.4	69.5	57.8	53.9	66.8	77.8
NMI-based	59.2	48.2	61.4	71.8	59.0	58.1	70.6	80.3
Stiefel	62.3	52.3	64.5	75.3	64.2	73.2	82.2	88.6
Hist	-	52.8	64.4	74.7	-	66.2	77.2	85.0
Ang	61.0	53.6	65.0	75.3	62.7	68.9	78.9	85.8
DAML	61.3	52.7	65.4	75.5	66.0	75.1	83.8	89.7
HADML	62.6	53.7	65.7	76.7	69.7	79.1	87.1	92.1
DVML	61.4	52.7	65.1	75.5	67.6	82.0	88.4	93.3
BIER	-	55.3	67.2	76.9	-	78.0	85.8	91.1
HTL	-	57.1	68.8	78.7	-	81.4	88.0	92.7
HTG	-	59.5	71.8	81.3	-	76.5	84.7	90.4
A-BIER	-	57.5	68.7	78.3	-	82.0	89.0	93.2
DWS	66.3	62.9	74.1	82.9	66.3	80.0	87.7	92.3
ProxyNCA	62.5	57.4	69.2	79.1	59.5	73.0	81.3	87.9
MIC	69.7	66.1	76.8	85.6	68.4	82.6	89.1	93.2
MIC*	69.1	64.9	75.8	84.7	68.1	80.7	88.0	92.9
MIC+L^L	70.1	65.8	77.1	84.9	68.7	81.2	88.4	92.7
MIC+$L^{\mathcal{W}}$	70.4	66.2	77.4	85.2	68.3	82.2	88.9	92.9

characteristics while learning a discriminative embedding. The encoder is trained to learn shared characteristics along using standard metric learning loss functions. We have followed the same training protocol as proposed in MIC. We have used ResNet50 [55] in our implementation of MIC (indicated using $*$) and its MMD and Wasserstein counterparts. The various baseline algorithms considered for this study are **(i)** Trip [124], **(ii)** DSC [79], **(iii)** Lifetd-Struct [107], **(iv)** NPairs [129], **(v)** NMI-based [130], **(vi)** Stiefel [119], **(vii)** Hist [142], **(viii)** Ang [148], **(ix)** DAML [27], **(x)** HADML [169], **(xi)** DVML [88], **(xii)** BIER [108], **(xiii)** HTL [36], **(xiv)** HTG [167], **(xv)** A-BIER [109], **(xvi)** DWS [152], **(xvii)** ProxyNCA [102] and **(xviii)** MIC [118]. Table 6.11 also shows the results on CUBS-200-2011 [145] and CARS196 [71] where the class-wise information is incorporated into MIC via L^L and $L^{\mathcal{W}}$ [5]. As observed, equipping the loss function of MIC with L^L and $L^{\mathcal{W}}$ leads to an increase in terms of NMI and R@K for both datasets. This clearly indicates that MIC also benefits when such comprehensive class-wise constraints are taken into account and a discrepancy loss is enforced between the class-wise embeddings using L^L and $L^{\mathcal{W}}$.

In order to validate the performance gain due to L^L and $L^{\mathcal{W}}$ over MIC, we have performed significance analysis using the Welch t-statistic for unpaired two-samples

[5] We have repeated the experiments 6 times with random weights for each trial, and reported the best results according to NMI for MIC + L^L and MIC+$L^{\mathcal{W}}$.

t-test. The formula for calculating the p-values is as follows

$$\mathrm{p} = \frac{\mu_{\mathbf{A}} - \mu_{\mathbf{B}}}{\sqrt{\frac{\sigma_{\mathbf{A}}^2}{N_{\mathbf{A}}} + \frac{\sigma_{\mathbf{B}}^2}{N_{\mathbf{B}}}}} \quad , \tag{6.8}$$

where **A** and **B** denote the two sets of experiments taken into account. $\mu_{\mathbf{A}}$, $\sigma_{\mathbf{A}}^2$ and $N_{\mathbf{A}}$ denote the sample mean, standard deviation and sample size of the experiment set **A**. Similar notations for **B** are represented as $\mu_{\mathbf{B}}$, $\sigma_{\mathbf{B}}^2$ and $N_{\mathbf{B}}$. The sample size (*i.e.* $N_{\mathbf{A}}$ and $N_{\mathbf{B}}$) is fixed to 6. **A** denotes MIC* where as **B** denotes either MIC$+\mathrm{L}^{\mathrm{L}}$ and MIC$+\mathrm{L}^{\mathcal{W}}$. The "p-values" between MIC* and MIC$+\mathrm{L}^{\mathrm{L}}$ are 7×10^{-5} and 85×10^{-6} for CUBS-200-2011 and CARS196 datasets respectively. Similarly the p-values between MIC* and MIC$+\mathrm{L}^{\mathcal{W}}$ are 85×10^{-6} and 51×10^{-3} respectively for CUBS-200-2011 and CARS196 datasets. These values clearly indicate that the results obtained are statistically significant over MIC across both the datasets. In the future, one can study the effect of L^{L} and $\mathrm{L}^{\mathcal{W}}$ when integrated with other loss functions as mentioned in Table 6.11.

Note: It is to be noted that our proposed algorithm has not been explicitly designed to alleviate the effects of noise from the input data and bridge the performance gap between the noisy and noiseless baseline methods. Indeed, through our detailed empirical results we have demonstrated that our proposed optimal transport based loss function is able to separate the class-wise distributions even in the noisy input data, thus being more robust in handling the side effects of noise compared to the baseline methods.

6.5 Summary

In this chapter, we proposed a novel Deep Class-wise Discrepancy Loss (DCDL) and addressed the problem of learning an efficient and discriminative embedding space. To this end, we have incorporated and increased *class-wise* dissimilarity constraints so as to decrease the intra-class variances, and increase the inter-class variances thereby leading to the formation of tight but well separated clusters of embedded points. Towards achieving this objective, we employed Maximum Mean Discrepancy and Wasserstein Distance, with roots in Optimal Transport, in order to separate the probability distributions between the embeddings belonging to each and every class. We have evaluated our proposed methodology across **clean** and **noisy** label settings over five well known datasets. Our empirical evaluations support and substantiate the need of enforcing such global dissimilarity constraints in learning a superior embedding over the baseline algorithms. A similar trend was also observed while training with an additional partially-global cross entropy loss; thereby further reinforcing the utility of utilizing DCDL in learning a tightly knit embedding space. We have also demonstrated superior performance of DCDL in the two fine-grained image classification datasets for both clean and noisy labels settings. Our empirical evaluations showed that global class-wise divergences are needed to learn an efficient and effective embedding space. In the future, we aim to

design a mutually integrated loss function to further improve the learning ability of DCDL.

Chapter 7

Conclusion

In this section, we first conclude the entire thesis, and then elaborate the possible trends for future research.

7.1 Thesis Summary

Image classification is a popular research area in computer vision. However, training a neural network to classify image dataset requires large and clean (corrected labeled) images. Noise is ubiquitous in the world around us. Difficulty in estimating the noise within a dataset makes learning from such a dataset a difficult and challenging task. In this thesis, we have investigated how to learn from noisy labeled image datasets with robust models.

Despite their significant learning capacity, convolutional neural networks (CNNs) have a tendency to overfit in the presence of samples with noisy labels. Alleviating this issue, we use a Co-Training framework as a fundamental basis. A Co-Training framework consists of two parallel models which is supposed to learn from each other and corrected wrong predictions for each other. So the key point here comes to how to improve that ability. In this thesis, we introduced our work on how to improve the ability of Co-Training framework to learn from noisy labels.

One of the key challenges in collaborative training frameworks is the tendency of two parallel models to mimic and copy each other, leading to redundancy and limited diversity in learned features. To address this issue, we modified the traditional collaborative training approach by introducing discrepancy constraints between the feature extractors of the parallel models. This adjustment encourages the models to learn distinct yet complementary features, thereby enhancing overall discriminative capability. This solution, which forms the basis of our work in Chapter 3, effectively mitigates the problem of model redundancy and improves feature diversity.

However, while introducing discrepancy constraints helped in learning distinct features, we observed that maintaining sufficient diversity between the models was still challenging. To further enhance model diversity, we incorporated Sinkhorn (S_ϵ) divergence, which offers a broader and more flexible range of discrepancy

metrics compared to traditional methods. By leveraging the geometric properties of Sinkhorn divergence, we were able to better encourage diversity between the models, resulting in improved performance and robustness. This approach is the primary focus of Chapter 4.

During our exploration of Co-Training methods, another challenge became apparent: traditional Co-Training frameworks often discard potentially corrupted labeled images to maintain data integrity. However, this approach leads to a significant waste of valuable training data. To address this issue, we developed a Contrastive Co-Training framework that effectively utilizes these noisy images instead of discarding them. This method is not only simple and fast but also significantly boosted performance compared to state-of-the-art approaches. As we continued to enhance the Co-Training framework, we identified a further limitation in the use of Convolutional Neural Networks (CNNs) as the parallel models. CNNs primarily capture global features, which may not be sufficient for learning robust representations from noisy labels. To overcome this, we introduced a novel Transformer-based Co-Training structure that leverages the local attention capabilities of transformers.

Chapters 3–5 build upon established methods, particularly parallel networks, which are widely recognized as an effective approach for noisy-label learning. However, by introducing our divergence modules into this framework, we position the divergence component—and its universal applicability—as the core novelty of our method. We further demonstrate its versatility by applying it to a metric-learning task, which is fundamentally different from noisy-label learning. While we acknowledge that our overall framework is not an entirely new paradigm, the advantages and novelty introduced by the divergence modules are clear and significant.

Building on previous research, we conducted an in-depth exploration of learning from noisy labels, utilizing the advantages of Sinkhorn Divergence to improve model robustness and accuracy. Leveraging these insights, we extended our investigation to the field of metric learning, examining the effectiveness of Sinkhorn Divergence on both noisy and clean datasets. Our approach enhanced representation learning and similarity measurement while preserving robustness against label noise. This work is comprehensively discussed in Chapter 6.

Extensive experimental evaluations on corrupted data from six standard benchmark datasets, including the real-world noisy dataset Clothing1M, consistently demonstrate that our proposed methods outperform existing state-of-the-art techniques.

Regarding the universal applicability of our method, we are confident that it can be used beyond noisy-label learning, extending naturally to semi-supervised and other weakly supervised settings. First, as detailed in Chapters 3–5, our modules—such as the Sinkhorn Loss—can be easily integrated into existing architectures for noisy-label learning. We also provide guidance for hyper-parameter tuning, and when extensive experimentation is not possible, Chapters 3 and 4 offer general default values that still achieve strong performance. Second, we show that our method applies to semi-supervised learning as well. In Chapter 5, we incorporate contrastive learning, demonstrating that our modules can also be used in unsuper-

vised or weakly supervised scenarios. Additionally, our application of the Sinkhorn Divergence to a metric-learning task further supports the broad adaptability of our approach. Together, these results confirm the universal applicability and robustness of our method across different learning paradigms.

While our work primarily focuses on robustness to label noise, we emphasise that the proposed methods have been extensively evaluated on real-world noisy datasets, not only synthetic benchmarks. These datasets contain naturally occurring noise and bias patterns, enabling us to assess robustness under realistic conditions. Our results further show that the model remains reliable even under extremely high noise ratios, demonstrating strong generalisation and resilience to label corruption. Although robustness does not directly guarantee fairness or interpretability, the stable performance observed across challenging real-world settings indicates that the model maintains practicality and reliability in noisy environments. We acknowledge, however, that incorporating explicit interpretability and fairness considerations is an important direction for future work—particularly for sensitive applications such as medical imaging. A corresponding paragraph has been added to the Discussion section to highlight this limitation and outline future research directions.

To summarize, we systematically addressed the challenges encountered in collaborative training frameworks by introducing four key contributions: (1) discrepancy constraints for feature diversity, (2) Sinkhorn divergence for enhanced model differentiation, (3) Contrastive Co-Training to utilize noisy images effectively, and (4) a Transformer-based Co-Training structure for improved feature extraction. By progressively refining our methods to tackle emerging challenges, we not only addressed the limitations of existing approaches but also advanced the state of the art in learning from noisy labels.

7.2 Future Work

While this thesis focuses on learning from noisy labeled datasets using Co-Training frameworks, there are many opportunities for further exploration. One potential direction is extending these methods to multi-label classification, where each image belongs to multiple categories. This is common in real-world applications like medical diagnosis, environmental monitoring, and social media analysis, where labels are often interdependent and noisy. Enhancing Co-Training to handle complex label relationships while reducing noise would expand its applicability.

Our work primarily focuses on 2D noisy-label image classification, including real-world noisy datasets. However, we have also conducted extensive experiments on related tasks such as semi-supervised learning and metric learning. Extending our approach to low-resource and underrepresented domains represents a promising direction for future research.

Another key area is domain adaptation, where clean labeled datasets exist in a source domain, but target domains contain noisy or incomplete labels. Adapting Co-Training to leverage knowledge from source domains could greatly improve its

effectiveness. Techniques like adversarial training, feature alignment, and domain generalization could enhance transferability across different domains.

Self-supervised learning presents another promising direction. Pretraining models in a self-supervised manner can improve feature robustness and reduce sensitivity to noisy labels. Combining self-supervised learning with Sinkhorn divergence or contrastive learning could enhance diversity and collaboration between parallel models, leading to more efficient training paradigms in noisy environments.

This thesis also highlights transformers' robustness in handling noisy labels. Expanding their use to temporal sequences, 3D image data, or multi-modal datasets is a natural next step. Exploring hybrid architectures that combine CNNs and transformers could further enhance performance in challenging settings.

From an algorithmic perspective, incorporating ensemble learning could improve scalability and generalization. Training diverse models on different subsets of noisy data can reduce overfitting. Additionally, dynamic collaboration strategies based on real-time noise estimation could make the framework more adaptive, especially for large-scale datasets or real-time applications.

What's more, in our experiments (Tables 5.3–5.5), τ was assumed to be known for controlled comparison with prior work; however, we agree that understanding the sensitivity to τ is crucial for practical deployment. We outline them as potential future work toward making the method fully practical in settings where τ is unknown.

Finally, expanding this research to low-resource and underrepresented domains could have practical significance. Noisy labels are prevalent in such datasets due to annotation limitations. Optimizing these frameworks for minimal labeled data could benefit applications like wildlife conservation, disaster response, and humanitarian aid. Creating synthetic noisy datasets with domain experts could further validate the models' robustness.

By exploring these directions, the methodologies in this thesis could be refined and extended to tackle even more complex challenges, further advancing robust learning from noisy labeled data.

Bibliography

[1] ALVAREZ, M. A., ROSASCO, L., LAWRENCE, N. D., ET AL. Kernels for Vector-valued Functions: A Review. *Foundations and Trends® in Machine Learning 4*, 3 (2012), 195–266.

[2] ARGYRIOU, A., EVGENIOU, T., AND PONTIL, M. Multi-Task Feature Learning. In *Advances in neural information processing systems* (2007), pp. 41–48.

[3] ARORA, S., KHANDEPARKAR, H., KHODAK, M., PLEVRAKIS, O., AND SAUNSHI, N. A Theoretical Analysis of Contrastive Unsupervised Representation Learning. *arXiv preprint arXiv:1902.09229* (2019).

[4] BACHMAN, P., HJELM, R. D., AND BUCHWALTER, W. Learning Representations by Maximizing Mutual Information Across Views. In *Advances in Neural Information Processing Systems* (2019), pp. 15535–15545.

[5] BAKTASHMOTLAGH, M., HARANDI, M., AND SALZMANN, M. Distribution-Matching Embedding for Visual Domain Adaptation. *The Journal of Machine Learning Research 17*, 1 (2016), 3760–3789.

[6] BANFIELD, R. E., HALL, L. O., BOWYER, K. W., AND KEGELMEYER, W. P. Ensemble diversity measures and their application to thinning. *Information Fusion 6*, 1 (2005), 49–62.

[7] BELLET, A., HABRARD, A., AND SEBBAN, M. A Survey on Metric Learning for Feature Vectors and Structured Data. *arXiv preprint arXiv:1306.6709* (2013).

[8] BERTINETTO, L., VALMADRE, J., HENRIQUES, J. F., VEDALDI, A., AND TORR, P. H. Fully-Convolutional Siamese Networks for Object Tracking. In *Proc. European Conference on Computer Vision (ECCV)* (2016), Springer, pp. 850–865.

[9] BHATIA, R., JAIN, T., AND LIM, Y. On the Bures–Wasserstein Distance Between Positive Definite Matrices. *Expositiones Mathematicae 37*, 2 (2019), 165–191.

[10] BLUM, A., AND MITCHELL, T. Combining Labeled and Unlabeled Data with Co-Training. In *Proceedings of the eleventh annual conference on Computational learning theory* (1998), pp. 92–100.

[11] CARUANA, R. Multitask Learning. *Machine learning 28*, 1 (1997), 41–75.

[12] CHAI, Y., LEMPITSKY, V., AND ZISSERMAN, A. Bicos: A Bi-Level Co-Segmentation Method for Image Classification. In *2011 International Conference on Computer Vision* (2011), IEEE, pp. 2579–2586.

[13] CHEN, H., WANG, Y., GUO, T., XU, C., DENG, Y., LIU, Z., MA, S., XU, C., XU, C., AND GAO, W. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 12299–12310.

[14] CHEN, S., BORTSOVA, G., JUÁREZ, A. G.-U., VAN TULDER, G., AND DE BRUIJNE, M. Multi-task Attention-based Semi-supervised Learning for Medical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2019), Springer, pp. 457–465.

[15] CHEN, T., KORNBLITH, S., NOROUZI, M., AND HINTON, G. A Simple Framework for Contrastive Learning of Visual Representations. *arXiv preprint arXiv:2002.05709* (2020).

[16] CHEN, T., SUN, Y., SHI, Y., AND HONG, L. On Sampling Strategies for Neural Network-based Collaborative Filtering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2017), pp. 767–776.

[17] COATES, A., NG, A., AND LEE, H. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (2011), pp. 215–223.

[18] CONNEAU, A., AND LAMPLE, G. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems 32* (2019), 7059–7069.

[19] CRAVEN, P., AND WAHBA, G. Smoothing Noisy Data with Spline Functions. *Numerische mathematik 31*, 4 (1978), 377–403.

[20] CUTURI, M. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems* (2013), pp. 2292–2300.

[21] DENG, J., DONG, W., SOCHER, R., LI, L.-J., LI, K., AND FEI-FEI, L. Imagenet: A Large-scale Hierarchical Image Database. In *2009 IEEE conference on computer vision and pattern recognition* (2009), Ieee, pp. 248–255.

[22] DENG, J., DONG, W., SOCHER, R., LI, L.-J., LI, K., AND FEI-FEI, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09* (2009).

[23] DENG, J., RUSSAKOVSKY, O., KRAUSE, J., BERNSTEIN, M., BERG, A. C., AND FEI-FEI, L. Scalable multi-label annotation. In *ACM Conference on Human Factors in Computing Systems (CHI)* (2014).

[24] DOERSCH, C., AND ZISSERMAN, A. Multi-task Self-supervised Visual Learning. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 2051–2060.

[25] DOSOVITSKIY, A., BEYER, L., KOLESNIKOV, A., WEISSENBORN, D., ZHAI, X., UNTERTHINER, T., DEHGHANI, M., MINDERER, M., HEIGOLD, G., GELLY, S., ET AL. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[26] DUAN, Y., CHEN, L., LU, J., AND ZHOU, J. Deep Embedding Learning with Discriminative Sampling Policy. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 4964–4973.

[27] DUAN, Y., ZHENG, W., LIN, X., LU, J., AND ZHOU, J. Deep Adversarial Metric Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 2780–2789.

[28] DZIUGAITE, G. K., ROY, D. M., AND GHAHRAMANI, Z. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906* (2015).

[29] FANG, P., ZHOU, J., ROY, S. K., PETERSSON, L., AND HARANDI, M. Bilinear attention networks for person retrieval. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 8030–8039.

[30] FANG, P., ZHOU, J., ROY, S. K., PETERSSON, L., AND HARANDI, M. Bilinear Attention Networks for Person Retrieval. In *Proceedings of the IEEE International Conference on Computer Vision* (2019).

[31] FEYDY, J., SÉJOURNÉ, T., VIALARD, F.-X., AMARI, S.-I., TROUVÉ, A., AND PEYRÉ, G. Interpolating between Optimal Transport and MMD using Sinkhorn Divergences. *arXiv preprint arXiv:1810.08278* (2018).

[32] FEYDY, J., SÉJOURNÉ, T., VIALARD, F.-X., AMARI, S.-I., TROUVE, A., AND PEYRÉ, G. Interpolating between Optimal Transport and MMD using Sinkhorn Divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics* (2019), pp. 2681–2690.

[33] FISHER, R. A. On an Absolute Criterion for Fitting Frequency Curves. *Statistical science 12*, 1 (1997), 39–41.

[34] FREITAG, D. Information extraction from html: Application of a general machine learning approach. In *AAAI/IAAI* (1998), pp. 517–523.

[35] FU, J., LIU, J., TIAN, H., LI, Y., BAO, Y., FANG, Z., AND LU, H. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 3146–3154.

[36] GE, W. Deep Metric Learning with Hierarchical Triplet Loss. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 269–285.

[37] GENEVAY, A., CUTURI, M., PEYRÉ, G., AND BACH, F. Stochastic optimization for large-scale optimal transport. In *Advances in neural information processing systems* (2016), pp. 3440–3448.

[38] GENEVAY, A., PEYRÉ, G., AND CUTURI, M. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics* (2018), PMLR, pp. 1608–1617.

[39] GLOROT, X., AND BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), pp. 249–256.

[40] GOLMANT, N., VEMURI, N., YAO, Z., FEINBERG, V., GHOLAMI, A., ROTHAUGE, K., MAHONEY, M. W., AND GONZALEZ, J. On the Computational Inefficiency of Large Batch Sizes for Stochastic Gradient Descent. *arXiv preprint arXiv:1811.12941* (2018).

[41] GONG, Y., KE, Q., ISARD, M., AND LAZEBNIK, S. A Multi-View Embedding Space for Modeling Internet Images, Tags, and Their Semantics. *International journal of computer vision 106*, 2 (2014), 210–233.

[42] GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., XU, B., WARDE-FARLEY, D., OZAIR, S., COURVILLE, A., AND BENGIO, Y. Generative Adversarial Nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680.

[43] GOODFELLOW, I. J., SHLENS, J., AND SZEGEDY, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[44] GORDO, A., ALMAZAN, J., REVAUD, J., AND LARLUS, D. End-to-End Learning of Deep Visual Representations for Image Retrieval. *Int. Journal of Computer Vision 124*, 2 (2017), 237–254.

[45] GRETTON, A., BORGWARDT, K. M., RASCH, M. J., SCHÖLKOPF, B., AND SMOLA, A. A Kernel Two-sample Test. *Journal of Machine Learning Research 13*, Mar (2012), 723–773.

[46] HADSELL, R., CHOPRA, S., AND LECUN, Y. Dimensionality Reduction by Learning an Invariant Mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (2006), vol. 2, IEEE, pp. 1735–1742.

[47] HAN, B., TSANG, I. W., AND CHEN, L. On the convergence of a family of robust losses for stochastic gradient descent. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16* (2016), Springer, pp. 665–680.

[48] HAN, B., YAO, Q., YU, X., NIU, G., XU, M., HU, W., TSANG, I., AND SUGIYAMA, M. Co-teaching: Robust Training of Deep Neural Networks with Extremely Noisy Labels. In *Advances in neural information processing systems* (2018), pp. 8527–8537.

[49] HAN, J., CHOI, Y., AND LEE, J. Characteristics of the Partially Reflected Terahertz Wave: Truncated Beam Propagation. *JOSA B 36*, 6 (2019), 1551–1555.

[50] HAN, Y., ROY, S., PETERSSON, L., AND HARANDI, M. Learning from noisy labels via discrepant collaborative training. In *The IEEE Winter Conference on Applications of Computer Vision* (2020), pp. 3169–3178.

[51] HAN, Y., ROY, S., PETERSSON, L., AND HARANDI, M. Learning from Noisy Labels via Discrepant Collaborative Training. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (March 2020).

[52] HANE, C., ZACH, C., COHEN, A., ANGST, R., AND POLLEFEYS, M. Joint 3d scene reconstruction and class segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 97–104.

[53] HASSANI, A., WALTON, S., SHAH, N., ABUDUWEILI, A., LI, J., AND SHI, H. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704* (2021).

[54] HE, K., FAN, H., WU, Y., XIE, S., AND GIRSHICK, R. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 9729–9738.

[55] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[56] HINTON, G., VINYALS, O., AND DEAN, J. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531* (2015).

[57] HJELM, R. D., FEDOROV, A., LAVOIE-MARCHILDON, S., GREWAL, K., BACHMAN, P., TRISCHLER, A., AND BENGIO, Y. Learning Deep Representations by Mutual Information Estimation and Maximization. *arXiv preprint arXiv:1808.06670* (2018).

[58] HOFFER, E., AND AILON, N. Deep Metric Learning using Triplet Network. In *International Workshop on Similarity-Based Pattern Recognition* (2015), Springer, pp. 84–92.

[59] HORIGUCHI, S., IKAMI, D., AND AIZAWA, K. Significance of Softmax-based Features in Comparison to Distance Metric Learning-based features. *IEEE transactions on pattern analysis and machine intelligence* (2019).

[60] IOFFE, S., AND SZEGEDY, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning* (2015), pp. 448–456.

[61] ISOLA, P., ZORAN, D., KRISHNAN, D., AND ADELSON, E. H. Learning Visual Groups from Co-occurrences in Space and Time. *arXiv preprint arXiv:1511.06811* (2015).

[62] JIANG, L., HUANG, D., LIU, M., AND YANG, W. Beyond synthetic noise: Deep learning on controlled noisy labels. In *International Conference on Machine Learning* (2020), PMLR, pp. 4804–4815.

[63] JIANG, L., ZHOU, Z., LEUNG, T., LI, L.-J., AND FEI-FEI, L. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055* (2017).

[64] JIANG, L., ZHOU, Z., LEUNG, T., LI, L.-J., AND FEI-FEI, L. Mentornet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels. In *International Conference on Machine Learning* (2018), pp. 2304–2313.

[65] KANG, G., JIANG, L., YANG, Y., AND HAUPTMANN, A. G. Contrastive Adaptation Network for Unsupervised Domain Adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 4893–4902.

[66] KESKAR, N. S., MCCANN, B., VARSHNEY, L. R., XIONG, C., AND SOCHER, R. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858* (2019).

[67] KINGMA, D. P., AND BA, J. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).

[68] KINGMA, D. P., AND WELLING, M. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114* (2013).

[69] KIRITCHENKO, S., AND MATWIN, S. Email classification with co-training. In *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research* (2011), IBM Corp., pp. 301–312.

[70] KOCH, G., ZEMEL, R., AND SALAKHUTDINOV, R. Siamese Neural Networks for One-Shot Image Recognition. In *ICML Deep Learning Workshop* (2015).

[71] KRAUSE, J., STARK, M., DENG, J., AND FEI-FEI, L. 3D Object Representations for Fine-Grained Categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops* (2013), pp. 554–561.

[72] KRAUSE, J., STARK, M., DENG, J., AND FEI-FEI, L. 3D Object Representations for Fine-Grained Categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)* (Sydney, Australia, 2013).

[73] KRIZHEVSKY, A., AND HINTON, G. Learning Multiple Layers of Features from Tiny Images.

[74] KULIS, B., ET AL. Metric Learning: A Survey. *Foundations and Trends® in Machine Learning 5*, 4 (2013), 287–364.

[75] KULLBACK, S., AND LEIBLER, R. A. On Information and Sufficiency. *The annals of mathematical statistics 22*, 1 (1951), 79–86.

[76] KUMAR, B., CARNEIRO, G., REID, I., ET AL. Learning Local Image Descriptors with Deep Siamese and Triplet Convolutional Networks by Minimising Global Loss Functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 5385–5394.

[77] LAINE, S., AND AILA, T. Temporal Ensembling for Semi-Supervised Learning. *arXiv preprint arXiv:1610.02242* (2016).

[78] LAMPLE, G., AND CONNEAU, A. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291* (2019).

[79] LAW, M. T., URTASUN, R., AND ZEMEL, R. S. Deep Spectral Clustering Learning. In *Proc. Int. Conference on Machine Learning (ICML)* (2017), pp. 1985–1994.

[80] LECUN, Y. The MNIST Database of Handwritten Digits. *http://yann. lecun. com/exdb/mnist/* (1998).

[81] LECUN, Y., BOTTOU, L., BENGIO, Y., AND HAFFNER, P. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[82] LI, H., XU, Z., TAYLOR, G., STUDER, C., AND GOLDSTEIN, T. Visualizing the Loss Landscape of Neural Nets. In *Advances in Neural Information Processing Systems* (2018), pp. 6389–6399.

[83] LI, J., WONG, Y., ZHAO, Q., AND KANKANHALLI, M. S. Learning to Learn from Noisy Labeled Data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 5051–5059.

[84] LI, Y., SWERSKY, K., AND ZEMEL, R. Generative moment matching networks. In *International Conference on Machine Learning* (2015), pp. 1718–1727.

[85] LI, Y., YANG, J., SONG, Y., CAO, L., LUO, J., AND LI, L.-J. Learning from Noisy Labels with Distillation. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 1910–1918.

[86] LI, Y., ZHANG, K., CAO, J., TIMOFTE, R., AND VAN GOOL, L. Localvit: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707* (2021).

[87] LIN, X., DUAN, Y., DONG, Q., LU, J., AND ZHOU, J. Deep Variational Metric Learning. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 689–704.

[88] LIN *et al.*, X. Deep Variational Metric Learning. In *ECCV* (2018).

[89] LIU, F., XU, W., LU, J., ZHANG, G., GRETTON, A., AND SUTHERLAND, D. J. Learning deep kernels for non-parametric two-sample tests. In *International conference on machine learning* (2020), PMLR, pp. 6316–6326.

[90] LIU, T., AND TAO, D. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence 38*, 3 (2015), 447–461.

[91] LIU, Z., LIN, Y., CAO, Y., HU, H., WEI, Y., ZHANG, Z., LIN, S., AND GUO, B. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030* (2021).

[92] LIU, Z., LUO, P., QIU, S., WANG, X., AND TANG, X. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016).

[93] LOPEZ-PAZ, D., BOTTOU, L., SCHÖLKOPF, B., AND VAPNIK, V. Unifying Distillation and Privileged Information. *arXiv preprint arXiv:1511.03643* (2015).

[94] LU, D., AND WENG, Q. A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing 28*, 5 (2007), 823–870.

[95] MADAAN, A., SETLUR, A., PAREKH, T., POCZOS, B., NEUBIG, G., YANG, Y., SALAKHUTDINOV, R., BLACK, A. W., AND PRABHUMOYE, S. Politeness transfer: A tag and generate approach. *arXiv preprint arXiv:2004.14257* (2020).

[96] MALACH, E., AND SHALEV-SHWARTZ, S. Decoupling "When to Update" from "How to Update". In *Advances in Neural Information Processing Systems* (2017), pp. 960–970.

[97] MANNING, C. D., MANNING, C. D., AND SCHÜTZE, H. *Foundations of Statistical Natural Language Processing*. MIT press, 1999.

[98] MASTERS, D., AND LUSCHI, C. Revisiting Small Batch Training for Deep Neural Networks. *arXiv preprint arXiv:1804.07612* (2018).

[99] MIAYTO, T., DAI, A. M., AND GOODFELLOW, I. Virtual Adversarial Training for Semi-Supervised Text Classification.

[100] MIYATO, T., DAI, A. M., AND GOODFELLOW, I. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725* (2016).

[101] MIYATO, T., MAEDA, S.-I., KOYAMA, M., AND ISHII, S. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence 41*, 8 (2018), 1979–1993.

[102] MOVSHOVITZ-ATTIAS, Y., TOSHEV, A., LEUNG, T. K., IOFFE, S., AND SINGH, S. No Fuss Distance Metric Learning using Proxies. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 360–368.

[103] NASEER, M., RANASINGHE, K., KHAN, S., HAYAT, M., KHAN, F. S., AND YANG, M.-H. Intriguing properties of vision transformers. *arXiv preprint arXiv:2105.10497* (2021).

[104] NATARAJAN, N., DHILLON, I. S., RAVIKUMAR, P. K., AND TEWARI, A. Learning with Noisy Labels. In *Advances in neural information processing systems* (2013), pp. 1196–1204.

[105] NETZER, Y., WANG, T., COATES, A., BISSACCO, A., WU, B., AND NG, A. Y. Reading Digits in Natural Images with Unsupervised Feature Learning.

[106] NIGAM, K., AND GHANI, R. Analyzing the Effectiveness and Applicability of Co-Training. In *Cikm* (2000), vol. 5, p. 3.

[107] OH SONG, H., XIANG, Y., JEGELKA, S., AND SAVARESE, S. Deep Metric Learning via Lifted Structured Feature Embedding. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 4004–4012.

[108] OPITZ, M., WALTNER, G., POSSEGGER, H., AND BISCHOF, H. Bier-boosting Independent Embeddings Robustly. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 5189–5198.

[109] OPITZ, M., WALTNER, G., POSSEGGER, H., AND BISCHOF, H. Deep Metric Learning with Bier: Boosting Independent Embeddings Robustly. *IEEE transactions on pattern analysis and machine intelligence* (2018).

[110] ORTEGO, D., ARAZO, E., ALBERT, P., O'CONNOR, N. E., AND MCGUINNESS, K. Multi-objective interpolation training for robustness to label noise. *arXiv preprint arXiv:2012.04462* (2020).

[111] PASZKE, A., GROSS, S., CHINTALA, S., CHANAN, G., YANG, E., DEVITO, Z., LIN, Z., DESMAISON, A., ANTIGA, L., AND LERER, A. Automatic Differentiation in Pytorch. In *NIPS-W* (2017).

[112] PATRINI, G., ROZZA, A., KRISHNA MENON, A., NOCK, R., AND QU, L. Making Deep Neural Networks Robust to Label Noise: A Loss Correction Approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 1944–1952.

[113] PEYRÉ, G., AND CUTURI, M. Computational Optimal Transport. *Foundations and Trends in Machine Learning 11*, 5-6 (2018), 355–206.

[114] QIAN, Q., SHANG, L., SUN, B., HU, J., LI, H., AND JIN, R. Softtriple Loss: Deep Metric Learning Without Triplet Sampling. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 6450–6458.

[115] QIAO, S., SHEN, W., ZHANG, Z., WANG, B., AND YUILLE, A. Deep Co-Training for Semi-Supervised Image Recognition. In *The European Conference on Computer Vision (ECCV)* (September 2018).

[116] REN, M., ZENG, W., YANG, B., AND URTASUN, R. Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050* (2018).

[117] REYNOLDS, D. A., QUATIERI, T. F., AND DUNN, R. B. Speaker Verification using Adapted Gaussian Mixture Models. *Digital signal processing 10*, 1-3 (2000), 19–41.

[118] ROTH, K., BRATTOLI, B., AND OMMER, B. Mic: Mining Interclass Characteristics for Improved Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 8000–8009.

[119] ROY, S. K., HARANDI, M., NOCK, R., AND HARTLEY, R. Siamese networks: The tale of two manifolds. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 3046–3055.

[120] RUSSAKOVSKY, O., DENG, J., SU, H., KRAUSE, J., SATHEESH, S., MA, S., HUANG, Z., KARPATHY, A., KHOSLA, A., BERNSTEIN, M., ET AL. Imagenet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision 115*, 3 (2015), 211–252.

[121] SAJJADI, M., JAVANMARDI, M., AND TASDIZEN, T. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in neural information processing systems* (2016), pp. 1163–1171.

[122] SALIMANS, T., GOODFELLOW, I., ZAREMBA, W., CHEUNG, V., RADFORD, A., AND CHEN, X. Improved techniques for training gans. In *Advances in neural information processing systems* (2016), pp. 2234–2242.

[123] SANAKOYEU, A., TSCHERNEZKI, V., BUCHLER, U., AND OMMER, B. Divide and Conquer the Embedding Space for Metric Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 471–480.

[124] SCHROFF, F., KALENICHENKO, D., AND PHILBIN, J. Facenet: A Unified Embedding for Face Recognition and Clustering. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 815–823.

[125] SHARMA, V., GUPTA, M., KUMAR, A., AND MISHRA, D. Video processing using deep learning techniques: A systematic literature review. *IEEE Access 9* (2021), 139489–139507.

[126] SHEN, J., QU, Y., ZHANG, W., AND YU, Y. Wasserstein Distance Guided Representation Learning for Domain Adaptation. In *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).

[127] SIMON, C., KONIUSZ, P., AND HARANDI, M. Projective subspace networks for few-shot learning.

[128] SOHN, K. Improved Deep Metric Learning with Multi-Class N-Pair Loss Objective. In *Advances in neural information processing systems* (2016), pp. 1857–1865.

[129] SOHN, K. Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In *Proc. Advances in Neural Information Processing Systems (NIPS)* (2016), pp. 1857–1865.

[130] SONG, H. O., JEGELKA, S., RATHOD, V., AND MURPHY, K. Deep Metric Learning via Facility Location. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[131] SPALL, J. C. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, vol. 65. John Wiley & Sons, 2005.

[132] SRINIVAS, A., LASKIN, M., AND ABBEEL, P. Curl: Contrastive Unsupervised Representations for Reinforcement Learning. *arXiv preprint arXiv:2004.04136* (2020).

[133] SRIVASTAVA, A., VALKOV, L., RUSSELL, C., GUTMANN, M. U., AND SUTTON, C. VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning. In *Advances in Neural Information Processing Systems* (2017), pp. 3308–3318.

[134] SUKHBAATAR, S., BRUNA, J., PALURI, M., BOURDEV, L., AND FERGUS, R. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080* (2014).

[135] SUN, Y., LI, L., XIE, Z., XIE, Q., LI, X., AND XU, G. Co-Training an Improved Recurrent Neural Network with Probability Statistic Models for Named Entity Recognition. In *International Conference on Database Systems for Advanced Applications* (2017), Springer, pp. 545–555.

[136] SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V., AND RABINOVICH, A. Going Deeper with Convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 1–9.

[137] TAN, M., AND LE QV, E. rethinking model scaling for convolutional neural networks. 2019, 1905.

[138] TAO, R., GAVVES, E., AND SMEULDERS, A. W. Siamese Instance Search for Tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 1420–1429.

[139] TARVAINEN, A., AND VALPOLA, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems* (2017), pp. 1195–1204.

[140] TIAN, Y., SU, D., LAURIA, S., AND LIU, X. Recent advances on loss functions in deep learning for computer vision. *Neurocomputing 497* (2022), 129–158.

[141] TRIANTAFILLOU, E., ZEMEL, R., AND URTASUN, R. Few-Shot Learning Through an Information Retrieval Lens. In *Proc. Advances in Neural Information Processing Systems (NIPS)* (2017).

[142] USTINOVA, E., AND LEMPITSKY, V. Learning Deep Embeddings with Histogram Loss. In *Proc. Advances in Neural Information Processing Systems (NIPS)* (2016), pp. 4170–4178.

[143] VARIOR, R. R., HALOI, M., AND WANG, G. Gated Siamese Convolutional Neural Network Architecture for Human Re-Identification. In *Proc. European Conference on Computer Vision (ECCV)* (2016), pp. 791–808.

[144] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, Ł., AND POLOSUKHIN, I. Attention is all you need. In *Advances in neural information processing systems* (2017), pp. 5998–6008.

[145] WAH, C., BRANSON, S., WELINDER, P., PERONA, P., AND BELONGIE, S. The Caltech-USCD Birds-200-2011 Dataset.

[146] WAH, C., BRANSON, S., WELINDER, P., PERONA, P., AND BELONGIE, S. The Caltech-UCSD Birds-200-2011 Dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology, 2011.

[147] WANG, F., AND LIU, H. Understanding the behaviour of contrastive loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 2495–2504.

[148] WANG, J., ZHOU, F., WEN, S., LIU, X., AND LIN, Y. Deep Metric Learning with Angular Loss. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 2593–2601.

[149] WANG, W., XIE, E., LI, X., FAN, D.-P., SONG, K., LIANG, D., LU, T., LUO, P., AND SHAO, L. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122* (2021).

[150] WEI, H., FENG, L., CHEN, X., AND AN, B. Combating Noisy Labels by Agreement: A Joint Training Method with Co-Regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 13726–13735.

[151] WEINBERGER, K. Q., AND SAUL, L. K. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *Journal of Machine Learning Research 10*, Feb (2009), 207–244.

[152] WU, C.-Y., MANMATHA, R., SMOLA, A. J., AND KRÄHENBÜHL, P. Sampling Matters in Deep Embedding Learning. In *Proc. Int. Conference on Computer Vision (ICCV)* (2017).

[153] WU, X., HE, R., SUN, Z., AND TAN, T. A Light CNN for Deep Face Representation with Noisy Labels. *IEEE Transactions on Information Forensics and Security 13*, 11 (2018), 2884–2896.

[154] WU, Z., XIONG, Y., YU, S. X., AND LIN, D. Unsupervised Feature Learning via Non-Parametric Instance Discrimination. In *Proceedings of*

the IEEE Conference on Computer Vision and Pattern Recognition (2018), pp. 3733–3742.

[155] XIAO, T., XIA, T., YANG, Y., HUANG, C., AND WANG, X. Learning from massive noisy labeled data for image classification. In *CVPR* (2015).

[156] YAN, H., DING, Y., LI, P., WANG, Q., XU, Y., AND ZUO, W. Mind the Class Weight Bias: Weighted Maximum Mean Discrepancy for Unsupervised Domain Adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017).

[157] YAO, Y., SUN, Z., ZHANG, C., SHEN, F., WU, Q., ZHANG, J., AND TANG, Z. Jo-src: A contrastive approach for combating noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 5192–5201.

[158] YE, M., ZHANG, X., YUEN, P. C., AND CHANG, S.-F. Unsupervised Embedding Learning via Invariant and Spreading Instance Feature. In *Proceedings of the IEEE Conference on computer vision and pattern recognition* (2019), pp. 6210–6219.

[159] YU, X., HAN, B., YAO, J., NIU, G., TSANG, I. W., AND SUGIYAMA, M. How Does Disagreement Benefit Co-Teaching. *arXiv preprint arXiv:1901.04215 1*, 2 (2019), 5.

[160] YU, X., LIU, T., GONG, M., BATMANGHELICH, K., AND TAO, D. An efficient and provable approach for mixture proportion estimation using linear independence assumption. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 4480–4489.

[161] YUAN, L., CHEN, Y., WANG, T., YU, W., SHI, Y., JIANG, Z., TAY, F. E., FENG, J., AND YAN, S. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986* (2021).

[162] ZHANG, C., BENGIO, S., HARDT, M., RECHT, B., AND VINYALS, O. Understanding Deep Learning Requires Rethinking Generalization. *arXiv preprint arXiv:1611.03530* (2016).

[163] ZHANG, P., DAI, X., YANG, J., XIAO, B., YUAN, L., ZHANG, L., AND GAO, J. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. *arXiv preprint arXiv:2103.15358* (2021).

[164] ZHANG, Y., XIANG, T., HOSPEDALES, T. M., AND LU, H. Deep Mutual Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2018).

[165] ZHAO, N., WU, Z., LAU, R. W., AND LIN, S. Distilling Localization for Self-Supervised Representation Learning. *arXiv preprint arXiv:2004.06638* (2020).

[166] ZHAO, W., CHELLAPPA, R., PHILLIPS, P. J., AND ROSENFELD, A. Face recognition: A literature survey. *ACM computing surveys (CSUR) 35*, 4 (2003), 399–458.

[167] ZHAO, Y., JIN, Z., QI, G.-J., LU, H., AND HUA, X.-S. An Adversarial Approach to Hard Triplet Generation. In *Proceedings of the European conference on computer vision (ECCV)* (2018), pp. 501–517.

[168] ZHAO, Z.-Q., ZHENG, P., XU, S.-T., AND WU, X. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems 30*, 11 (2019), 3212–3232.

[169] ZHENG, W., CHEN, Z., LU, J., AND ZHOU, J. Hardness-Aware Deep Metric Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 72–81.

[170] ZOU, Z., CHEN, K., SHI, Z., GUO, Y., AND YE, J. Object detection in 20 years: A survey. *Proceedings of the IEEE 111*, 3 (2023), 257–276.