Robust Image Classification and Clustering via Distribution learning

by

Zijie Tan

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Distribution learning has long been a key area of research in computer vision. However, the potential of combining distribution learning with deep learning remains underexplored. To bridge this gap, this thesis discusses two proposed methods. The first, Differentiable Arithmetic Distribution Module (DADM), introduces differentiability to the construction of histograms, enabling deep learning models to leverage distributional information more effectively. By employing Kernel Density Estimate (KDE) within a deep learning framework, DADM captures distribution information that is nearly invariant to affine transformations, significantly enhancing the robustness of image classification models against such variations. The second method, Deep Clustering via Distribution Learning (DCDL), extends the application of distribution learning to clustering tasks, particularly in high-dimensional data spaces. DCDL integrates distribution learning into deep clustering frameworks through the introduction of Monte-Carlo Marginalization for Clustering (MCMarg-C), an algorithm that optimizes cluster formation by directly learning the underlying data distribution. This method improves clustering performance by maintaining data structure through dimensionality reduction and manifold approximation. Overall, this thesis aims to leverage the integration of distribution learning and deep learning to address the limitations of traditional deep learning methods, thereby developing more robust and scalable models for computer vision tasks such as image classification and clustering.

Preface

The main chapters in this thesis are based on papers that either have been published or currently under review. Chapter 2 is based on the article published as "Affine-Transformation-Invariant Image Classification by Differentiable Arithmetic Distribution Module" in 4th International Conference on Smart Multimedia, 2024. I was responsible for the idea formulation, implementation, and manuscript writing. Chapter 3 is based on the article "Deep Clustering via Distribution Learning" which is submitted to *Applied Intelligence*. I am a co-first author and was responsible for the idea formulation, theoretical analysis, and manuscript writing.

The work in which I was involved during my MSc study are listed below in chronological order:

- Zhao, C., Dong, G., Zhang, S., **Tan, Z.**, & Basu, A. (2023). Frequency Regularization: Reducing Information Redundancy in Convolutional Neural Networks. In *IEEE Access*.
- **Tan, Z.**, Dong, G., Zhao, C., & Basu, A. (2024). Affine-Transformation-Invariant Image Classification by Differentiable Arithmetic Distribution Module. In *International Conference on Smart Multimedia*
- Dong, G., **Tan, Z.**, Zhao, C., & Basu, A. (2024). Deep Clustering via Distribution Learning.

To the Count

For teaching me everything I need to know about math.

If people do not believe that mathematics is simple, it is only because they do not realize how complicated life is.

– John von Neumann

Acknowledgements

First, I would like to express my sincere gratitude to my supervisor, Prof. Anup Basu, for his valuable guidance and support across my MSc study. I have learned a lot while studying in the Multimedia Research Center in the last two years, which I believe will certainly benefit me for the rest of life.

I would like to thank my colleagues, Dr. Chenqiu Zhao and Guanfang Dong, for all the great suggestions and feedback they gave me. They helped me learn how to do research from the very beginning.

Also, I would like to thank Dr. Charles Chan for offering me the Chan Pang-Kuen Memorial Scholarship. Without him, I wouldn't be able to pursue my studies in Canada.

Finally, I wish to express my deep gratitude to my family for their love throughout this journey. Thank you to my parents for all their encouragement, and thank you to Shuying for all her support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Distribution learning refers to a category of machine learning techniques which focus on understanding and modeling an underlying probability distribution from which a set of independent data samples are drawn. It has been a classic and popular research topic in computer vision [12], [20], [35]. The commonly-used distribution learning methods are often divided into parametric and non-parametric approaches based on whether they assume that the data follows a known distribution with a fixed number of parameters [6], [30], [51]. For example, Gaussian Mixture Model (GMM), one of the most popular parametric distribution learning method, is based on the assumption that the data is generated from a mixture of a certain amount of Gaussian distributions [20]. On the other hand, Kernel Density Estimate (KDE) does not make such assumptions, making it more flexible and, therefore, applied in both Chapter 2 and Chapter 3 in this thesis.

Recently, researchers also develop various deep learning techniques that involves distribution information, covering a wide range of applications such as image generation, style transfer, text classification, and etc [14], [44], [86], [90]. Despite these advancements, the connection between distribution learning and deep learning is still not fully explored, especially in terms of how these two techniques can be integrated to improve performance in computer vision tasks. This thesis hence aims to bridge this gap by investigating the effective integration of distribution learning into deep learning models. The two proposed

methods discussed in this thesis **dong2024deep**, [69] are designed to explore and leverage the integration between distribution learning and deep learning techniques in the context of computer vision tasks.

The first work [69], discussed in Chapter 2, presents a distribution learning-based approach for the task of image classification, named Differentiable Arithmetic Distribution Module (DADM). It contributes to this topic by introducing differentiability to the construction of histogram , since the histograms of pixel values are commonly used to capture the distribution information in images. Traditionally, the construction of histograms is not differentiable, limiting their integration with deep learning frameworks that rely on gradient-based optimization. To the best of our knowledge, this is the first work to incorporate differentiable histograms within the context of distribution learning. By making histograms differentiable, this approach allows models to efficiently leverage distributional information while fully utilizing gradient-based optimization techniques. This is significant because conventional deep learning models often focus on spatial information, making them sensitive to variations such as affine transformations, which can distort the spatial structure of images. In contrast, DADM employs Kernel Density Estimation (KDE) to build differentiable histograms within a deep learning framework, capturing distribution information that is nearly invariant to affine transformations. We demonstrate through the experimental results that the model becomes more robust to such variations, enhancing its overall performance in image classification tasks.

Chapter 3 discusses the second study, Deep Clustering via Distribution Learning (DCDL) **dong2024deep**. It extends the application of distribution learning to clustering tasks, particularly in high-dimensional data spaces where traditional algorithms like k-means often suffer from the curse of dimensionality. DCDL integrates distribution learning into deep clustering frameworks to improve clustering accuracy and robustness. The method introduces Monte-Carlo Marginalization for Clustering (MCMarg-C), a novel algorithm that directly learns the underlying data distribution and optimizes cluster formation. Additionally, DCDL employs an autoencoder for dimensionality reduction and uses Uniform Manifold Approximation and Projection (UMAP) to maintain

the data's structure in a lower-dimensional space. This integration creates a more principled approach to clustering, supported by a theoretical foundation that bridges distribution learning and clustering. Experimental results show that DCDL outperforms state-of-the-art deep clustering methods, particularly in high-dimensional contexts, making it a significant contribution to both the theory and practice of clustering in computer vision.

1.2 Thesis Statement

This thesis proposes novel methodologies for integrating distribution learning and deep learning in the tasks of image classification and clustering to enhance the robustness and scalability of models in computer vision. By leveraging the statistical properties and probability models of the data, the proposed approaches aim to alleviate the limitations of traditional deep learning techniques, particularly their sensitivity to affine transformations and the curse of dimensionality.

1.3 Contributions

In this thesis, we present the following contributions:

- We propose the Differentiable Arithmetic Distribution Module (DADM) for image classification, which employs kernel density estimation to extract differentiable histograms from images. This method enables the model to learn distributional information that is invariant to affine transformations, thereby enhancing robustness.

- We introduce the Deep Clustering via Distribution Learning (DCDL) method, which integrates distribution learning into a deep clustering framework. This approach incorporates manifold learning and Monte Carlo marginalization techniques to improve clustering performance on high-dimensional data. We also conduct a theoretical analysis to explore the connection between distribution learning and clustering.

- Through extensive experimental evaluations, we demonstrate the effectiveness and robustness of the proposed methods compared to traditional deep learning techniques in the scenarios of image classification and clustering.

1.4 Thesis Layout

The rest of this thesis is organized as follows: In Chapter 2, we introduce the background, design, and evaluation of DADM. In Chapter 3, we cover the details of DCDL and discuss how we connect clustering and distribution learning. Chapter 4 gives the conclusion of the thesis and discusses about the potential future work.

Chapter 2

Affine-Transformation-Invariant Image Classification by Differentiable Arithmetic Distribution Module

2.1 Introduction

Convolutional Neural Network (CNN) has been a powerful and popular tool for extracting features from images, enabling state-of-the-art performance in various computer vision tasks, such as object detection, image segmentation, and image classification [16], [39], [60], [74]. However, CNNs are inherently weak against some simple affine transformations such as rotation, since they rely strongly on the spatial patterns in data [91]. Compared to the pattern information, distribution information has the potential to provide affine-transform invariance. Because it concentrates on the overall statistical properties and probability distribution of pixels regardless of their exact spatial arrangement. Therefore, to alleviate the aforementioned issue, we propose to learn the distribution information for classification task, in which the distribution learning techniques are incorporated.

Distribution learning techniques are a group of methods wherein the focus is shifted from learning explicit patterns to understanding the underlying statistical distributions and characteristics of the data. These approaches often seek to capture the broader, holistic properties of datasets rather than

narrowly focusing on specific, local patterns. For example, DIDL network [15] utilizes the temporal distribution of pixel values across video frames, and learns the underlying statistical features of background and foreground in different scenes. In this work, we will instead focus on learning the spatial distribution information of pixels in images, which can enable our model to capture the affine-transformation-invariant features in input data, and thus make it more robust against transformations such as rotation.

A common approach to describe distribution in computer vision is the histogram of pixel values. However, the construction of histograms is not differentiable, which makes it hard to efficiently integrate histograms and neural networks. To address this problem, we utilize the Kernel Density Estimation (KDE) to approximate the histogram. The key idea is that rather than counting pixels and assigning them to discrete bins, we represent the data distribution by overlaying a kernel function at each data point and summing the contributions of these kernels across all data points. In this way, the bins of histograms are transformed into smoothed probability density instead of discrete counts, while preserving the statistical information.

In this work, a KDE-based method is formulated for constructing differentiable histograms from images. Based on this, we propose a novel differentiable arithmetic distribution module, which is explicitly crafted to learn the underlying probability distribution of the input space. This global feature fortifies the model's robustness against specific affine distortions, notably rotations. At the same time, the differentiability enables spatial feature extraction for the distribution learning techniques. The main contributions of this work are summarized below:

- We propose the Differentiable Arithmetic Distribution Module (DADM) that is adept at extracting inherent distribution information from images while also offering resilience to certain affine transformations, such as rotations.

- We utilize a KDE-based approach of constructing smoothed and differentiable histograms, enabling a seamless integration of histograms and

neural network.

- We conduct experiments to evaluate and demonstrate the effectiveness and robustness of the formulated method, including comparison with the famous CNN-variant LeNet and an ablation study of the proposed DADM network.

2.2 Related Work

Distribution learning is a technique that focuses on understanding the underlying probability distributions of data from the observed samples, rather than solely identifying explicit patterns or features [35]. Modelling the entire data distribution can provide a holistic view, which enhances invariance to common transformations such as translations, scalings, and rotations. In addition, the distribution information has been shown to be instrumental in many tasks in computer vision, including image generation, background subtraction, segmentation [89].

Although a histogram is an appropriate way to describe the probability density function and thus to convey the distribution information, traditional histogram is discrete and non-differentiable, making it challenging to be directly integrated into modern neural network frameworks that rely on gradient-based optimization and backpropagation [57]. To address this issue, multiple methods have been proposed to approximate the soft histogram in a continuous and differentiable manner. One notable approach is the HOG [11] which utilizes linear filtering operations and convolutions to approximate a piecewise differentiable histogram for pose estimation. Wang et al. proposed the first learnable histogram layer for neural networks by formulating HOG with a series of convolutional modules [75]. Furthermore, Sedighi et al. presented a globally differentiable histogram layer by utilizing radial basis functions as step functions in the backpropagation [64]. Peeples et al. further extended their work to have adaptive number and width of bins [56].

While most of the aforementioned methods assume a predefined distribution of the data, KDE offers the flexibility to estimate the underlying distribu-

tion directly from the samples without prior assumptions [17]. For example, though not directly applying KDE, HistoGAN [2] uses an inverse-quadratic kernel function to compute a weighted contribution of pixels to the bins of output histogram. Another related method is DeepHist [4], which is also based on KDE. It uses a sigmoid-based kernel function to estimate the histogram of pixels to separate the edge and color features in images. Nevertheless, this method does not thoroughly explore the application of probability and distribution information, but primarily use the histogram as a way to represent color information. Our method, on the other hand, utilizes the Gaussian kernel and considers the histogram in a probabilistic viewpoint, which adds values to the robustness and effectiveness in image classification.

Based on these explorations, researchers seek to optimize the integration of histograms and neural networks. For example, ImHistNet [32] is capable of learning complex and subtle task-specific textural features and global statistical features directly from the image intensity, which is defined by a set of convolution operations. Similarly, PTFEM [94] is a texture enhancement network module that uses an adaptive histogram equalization mechanism that pays specific attention to texture details and propagates the distribution information across pyramid layers. While these traditional neural networks only use convolution operations to handle distribution information, the DIDL network [15] presents the arithmetic distribution layers that directly consider histograms as probability density functions. However, the construction of histogram in a DIDL network is not differentiable, limiting its feature extraction ability and computation speed. In this work, we enhance the arithmetic distribution modules in the DIDL network with the proposed KDE-based differentiable histogram module.

2.3 Method

In this section, we will discuss the mathematical details and implementations of the proposed differentiable histogram module and the corresponding DADM network. Since we focus on the task of image classification, we can first assume

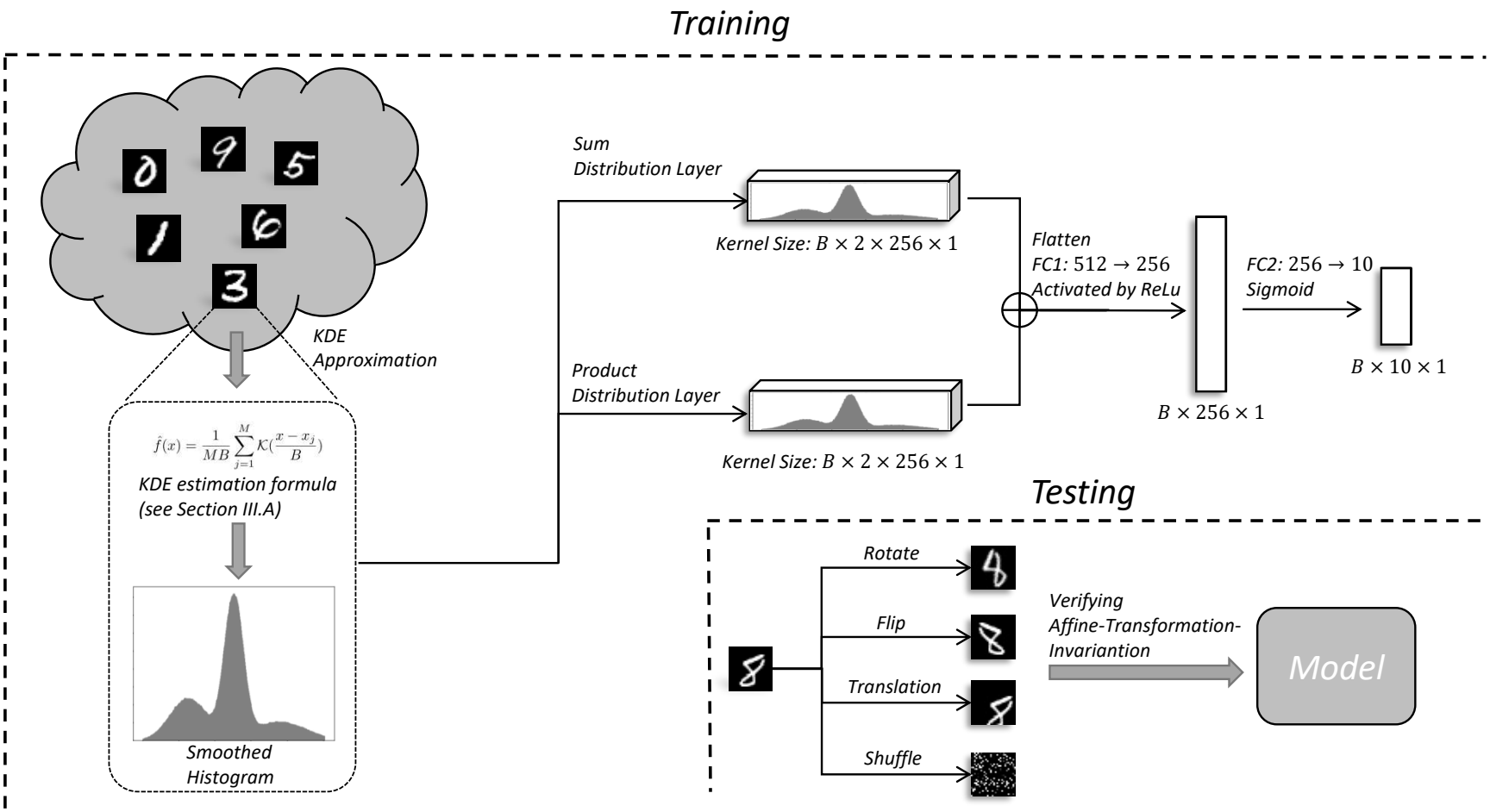


Figure 2.1: Network architecture and pipeline for our proposed method. First, an image is transformed into a smoothed histogram via Kernel Density Estimation (KDE). Then, the smoothed histogram is processed by the Sum Distribution Layer and Product Distribution Layer to learn distributional information. Each layer incorporates two 256×1 histogram kernels. Finally, the outputs from both layers are classified through a Fully Connected Layer. During testing, to verify affine transformation invariance, the input images are rotated, flipped, translated, and shuffled. These affine transformations are applied only during the testing phase.

that the input space is the set of gray-scale images with one single channel ranging from 0 to 255. This will not compromise the generalizability of our approach because each channel of multi-channel images can also be seen as a gray-scale image. In our implementation, the pixel values are narrowed into the range of $[-1, 1]$.

2.3.1 Differentiable Histogram

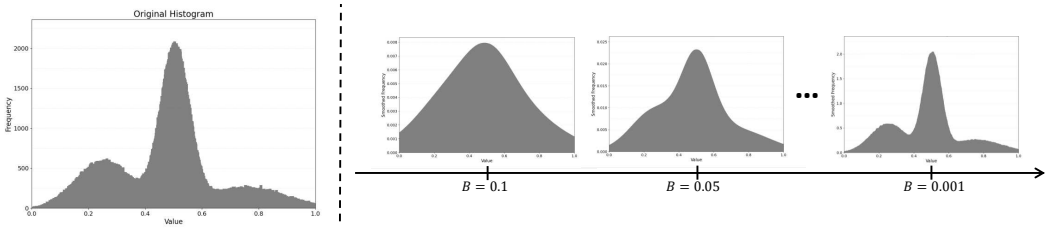


Figure 2.2: An example of KDE-based approximations of histogram with different bandwidth B. Note that the smoothed histogram gets closer to the original histogram as B grows. We choose $B = 0.001$ in our implementation.

We represent the distribution of pixels in image with histogram since it is simple and straightforward to interpret. And more importantly, it does not assume a specific distribution for the data. This can be advantageous when dealing with data of an unknown or complex distribution [37]. To integrate histogram into the arithmetic distribution model, we approximate it using differentiable functions, as illustrated in Fig 2.2.

To transform the discrete histogram into a smooth, differentiable approximated representation, we will first need to partition the range of pixel values. The details of partitioning and the notations for deriving the differentiable histogram are given below.

Notation: Let N be the number of bins in the desired representation of histogram, where each bin will have the width $W = \frac{1-(-1)}{N} = \frac{2}{N}$ and span $\Delta = \frac{W}{2}$. For the i-th bin, it has left bound $L_i = -1 + (i-1)W$ and right bound $R_i = -1 + iW$. Therefore, the i-th bin is $X_i = [-1 + (i-1)W, -1 + iW]$ centering at $\mu_i = -1 + (i - \frac{1}{2})W$, where $i = 1, 2, 3, ..., N$.

Inspired by DeepHist [4], we also utilize the KDE to compute the differen-

tiable approximation of the histogram, which operates by placing a kernel on each data point (or pixel value in our case) and summing up the contributions from all these kernels to obtain a smooth probability density function. Given a set of pixel values $x_1, x_2, ..., x_M$ from image, the KDE estimate at point x in the input space is given by:

$$\hat{f}(x) = \frac{1}{MB} \sum_{j=1}^{M} \mathcal{K}(\frac{x - x_j}{B}), \tag{2.1}$$

where $\mathcal{K}(\cdot)$ is the selected non-negative kernel function, and $B > 0$ is a free parameter called bandwidth which controls the smoothness of the estimated function. And then by definition, the probability of a bin X_i is given by the integral of the KDE function over the bin's range:

$$P(X_i = \mu_i) = \int_{L_i}^{R_i} \hat{f}(x)dx = \int_{\mu_i - \Delta}^{\mu_i + \Delta} \hat{f}(x)dx \tag{2.2}$$

Combining the Equations 2.1 and 2.2, we derive a unified expression for computing the probability of bins with respect to the kernel function:

$$\begin{aligned} P(X_i = \mu_i) &= \int_{\mu_i - \Delta}^{\mu_i + \Delta} \hat{f}(x)dx \\ &= \int_{\mu_i - \Delta}^{\mu_i + \Delta} \frac{1}{MB} \sum_{j=1}^{M} \mathcal{K}(\frac{x - x_j}{B})dx \\ &= \frac{1}{MB} \sum_{j=1}^{M} \left[\int_{\mu_i - \Delta}^{\mu_i + \Delta} \mathcal{K}(\frac{x - x_j}{B})dx \right] \end{aligned} \tag{2.3}$$

The actual value of the above equation will depend on the selection of kernel function $\mathcal{K}(\cdot)$. Some popular kernel functions in KDE include Gaussian Kernel, Epanechnikov kernel, Uniform Kernel, etc. In this paper, we choose the Gaussian distribution $\mathcal{K}(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ as the kernel function. The final

equations for computing our differentiable histogram are given by:

$$\begin{aligned} P(X_i = \mu_i) &= \frac{1}{MB} \sum_{j=1}^{M} \left[\int_{\mu_i - \Delta}^{\mu_i + \Delta} \mathcal{K}(\frac{x - x_j}{B}) dx \right] \\ &= \frac{1}{MB} \sum_{j=1}^{M} \left[erf(\frac{x - x_j}{B}) \Big|_{\mu_i - \Delta}^{\mu_i + \Delta} \right] \\ &= \frac{1}{MB} \sum_{j=1}^{M} \mathcal{G}_i(x_j), \\ \text{where } \mathcal{G}_i(x) &= erf(\frac{\mu_i - x + \Delta}{B}) - erf(\frac{\mu_i - x - \Delta}{B}) \\ \text{and } erf(x) &= \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt. \end{aligned} \tag{2.4}$$

2.3.2 Differentiable Arithmetic Distribution Learning

Together with the obtained differentiable function above, we adapt the product distribution layer and sum distribution layer proposed in our previous work [15] to design the differentiable arithmetic distribution module. For understandability, we will also introduce these two layers below.

In contrast to the convolution layer, which view input histograms merely as vectors, the product distribution layer and sum distribution layer interpret them as distributions to better describe the probability information and the correlation between histogram entries. To accomplish this, these two layers represent their learning kernels as histograms. That is, given the input and output distributions denoted by random variables X and Z, the distribution layers learn the distributions of learning kernels denoted by random variables W and B such that $Z = WX + B$. Note that all distributions are described by histograms in our work. The expressions of forward pass and backpropagation are formulated as follow:

Product distribution layer:

$$\begin{aligned} f_Z(z) &= \int_{-\infty}^{\infty} f_W(w) f_X(\frac{z}{w}) \frac{1}{|w|} dw, \quad \text{forward} \\ \nabla w_i &= \sum_{j=-\infty}^{\infty} \nabla z_j f_X(\frac{z_j}{i}) \frac{1}{|i|}, \quad \text{backward} \end{aligned} \tag{2.5}$$

where $f_Z(z)$, $f_W(w)$, and $f_X(x)$ are probability density functions (PDF) that represent the distributions of Z, W, X, respectively. And z, w, and x are the entries of corresponding histogram.

Sum distribution layer:

$$\begin{aligned} f_Z(z) &= \int_{-\infty}^{\infty} f_B(b) f_X(z-b) db, \quad \text{forward} \\ \nabla b_k &= \sum_{j=-\infty}^{\infty} \nabla z_j f_X(z_j - k), \quad \text{backward} \end{aligned} \tag{2.6}$$

where, likewise, $f_B(b)$ and b are the PDF and histogram entries of the distribution represented by B, respectively.

The proposed neural network module consists of a differentiable histogram layer, a product distribution layer, and a sum distribution layer. Specifically, the input images of size $B \times 1 \times H \times W$ will be first fed into the differentiable histogram layer to generate the smoothed histograms of size $B \times N \times 1$, where B is the batch size and N is a parameter representing the number of bins in the desired histogram. The distribution layer then employs the learning kernel of size $N \times 1$ on this histogram and generate the output of the same shape. This output will be further fed into the classifier module for image classification result. More details of network architecture and the pipeline are show in Fig. 2.1. The parameter configuration of network architecture is given in Tab 2.1.

2.4 Experimental Results

In this section, we will discuss the experiments we conducted to evaluate the performance of method and examine the affine transformation invariance. All training and testing are performed on a NVIDIA RTX A4000. During training, we applied the Adam optimizer with a learning rate of 0.001 and the Negative Log Likelihood function as the loss function. An overall illustration of the evaluation results is shown in Figure 2.3.

Table 2.1: Details of architecture of the proposed model and the networks selected for experiments.

LeNet		**Base Classifier**		**CNN**		**DADM**	
type	size	type	size	type	size	type	size
Conv	$(1, 5, 5) \times 6$						
Relu							
MaxPool	$(2, 2)$						
Conv	$(6, 5, 5) \times 16$						
Relu				Conv	$(1, 3, 3) \times 4$	DiffDis	
MaxPool	$(2, 2)$			Relu		ProDis	$1 \times 256 \times 1$
Linear	120×256	Linear	256×784	Linear	256×784	SumDis	$1 \times 256 \times 1$
Relu		Relu		Relu		Relu	
Linear	84×120	Linear	512×256	Linear	512×256	Linear	512×256
Relu		Relu		Relu		Relu	
Linear	10×84	Linear	10×512	Linear	10×512	Linear	10×512
Softmax		Softmax		Softmax		Softmax	

Note: Conv - Convolution layer, Relu - Rectified Linear Unit, ProDis - Product Distribution Layer, SumDis - Sum Distribution Layer, DiffDis - Differentiable Histogram Layer.

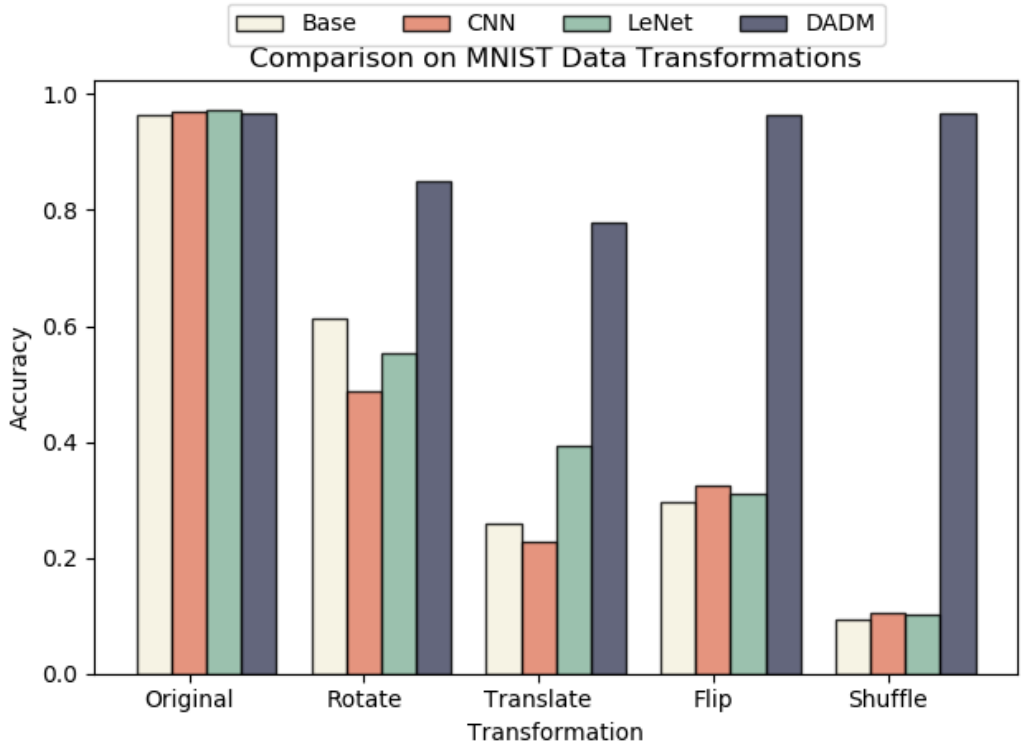


Figure 2.3: Histogram showing the Top-1 accuracy (%) of baseline methods and DADM on MNIST dataset under various transformations.

2.4.1 Dataset

To validate the robustness of our proposed model, we employ the MNIST dataset, which is a widely-recognized benchmark in computer vision research. The dataset encompasses 60,000 training images and an additional 10,000 testing images, each featuring handwritten digits ranging from 0 to 9. All images are in grayscale and have dimensions of 28×28 pixels. We subject the MNIST images to various transformations in order to evaluate and compare the models' performance under varying conditions.

2.4.2 Comparison with Convolutional Neural Network

To highlight the advantages of our proposed methodology over traditional Convolutional Neural Networks, we conduct a comprehensive comparative analysis against the well-known LeNet [38] architecture. The details of architecture and parameter are given in Table 2.1. Specifically, we evaluate the model's robust-

Table 2.2: Results of experiments comparing LeNet and our method on the original and transformed data in MNIST, reported in Top-1 accuracy (%). Numbers in parentheses are the performance decrease compared to tests on original images, where lower values represent better performance.

	Original	Rotate	Translate	Flip	Shuffle
LeNet	97.39	55.20 (42.19)	39.30 (58.09)	31.09 (66.3)	10.33 (87.06)
DADM	96.57	**85.02** (**11.65**)	**77.93** (**18.74**)	**96.34** (**0.33**)	**96.56** (**0.11**)

ness and performance under various affine transformations. The transformations in experiments include rotation, translation, and flipping. In addition, we also test the model's performance on randomly shuffled images. Training is performed on the training set of the original MNIST dataset, while the testing are performed on the testing set of the original and transformed dataset.

The details of selected transformation in experiment are listed as follow: for the rotation, input images are rotated by a random degree ranging from 0 to 90; for the translation, input images are shifted to a arbitrary direction for a random number of pixels with a maximum offset of 8; for the flipping, input images are randomly flipped horizontally or vertically.

The results of experiments are listed in Table 2.2. Through the result of comparison between LeNet and our method, we can see that though both method perform closely on the original MNIST dataset with our method being only slightly less favorable, our method significantly outperforms LeNet across all other categories of experiments where specific transformations are applied to the input images. This demonstrates that our method is much more robust against affine transformations than the classic CNN architectures such as LeNet, while maintaining a comparable power to them in terms of effectiveness.

2.4.3 Ablation Study

We perform an ablation study to further demonstrate the contributions of individual components in our proposed neural network architecture. Specifically, we investigate the role of DADM. As shown in Table 2.1, we derive two baseline models for comparative analysis by removing the differentiable arithmetic distribution module and replacing it with a convolution module. They are tested under a similar experiment configuration to the one in Section 2.4.2, where various transformations are applied to input data. Likewise, the training only use the training images of the original MNIST dataset, and the testing use the original and transformed testing images.

The results are summarized in Table 2.3. Several observations can be made from the results. For the original data, the extremely narrow gap between our

Table 2.3: Results of experiments comparing the base classifier, CNN, and our method on the original and transformed data in MNIST, reported in Top-1 accuracy (%). Numbers in parentheses are the performance decrease compared to tests on original images, where lower values represent better performance.

	Original	Rotate	Translate	Flip	Shuffle
Base	96.29	61.23 (35.06)	25.97 (70.32)	29.69 (66.60)	9.39 (86.90)
CNN	97.03	48.82 (48.21)	22.65 (74.38)	32.37 (64.66)	10.43 (86.60)
DADM	96.57	**85.02** (**11.65**)	**77.93** (**18.74**)	**96.34** (**0.33**)	**96.56** (**0.11**)

method and the two baselines suggests that the proposed approach maintains a similar power as the CNN in terms of effectiveness. While for the transformed data, our method again demonstrates its robustness against the affine transformations since its accuracy are significantly higher than that of the baselines.

2.4.4 Case Study

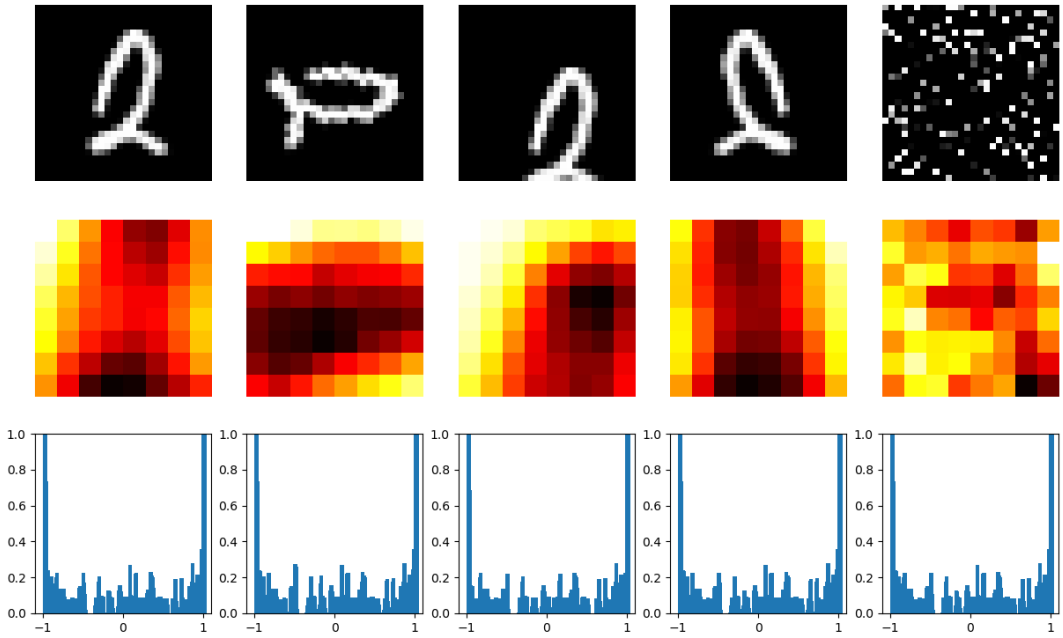


Figure 2.4: An example of features extracted by CNN and DADM. From top to bottom, the rows are: input images, feature maps extracted from LeNet, and the histograms computed by DADM. From left to right, the columns represent: original, rotated, translated, flipped, and shuffled images.

To further reveal the reason behind such robustness of our model, we also conduct a case study on the class-wise performance of DADM. The results are detailed in Table 2.4, where the class-wise accuracy of LeNet is also listed as the baseline. While both methods demonstrate high accuracy on the unaltered images, LeNet shows relatively high sensitivity to various transformations of images across all categories, especially for digits with complex shapes or less distinct features such as "3" or "6". This sensitivity in LeNet can be attributed to its reliance on spatial features, which can be significantly altered by transformations like rotation or flipping. DADM, on the other hand, shows higher

Table 2.4: The class-wise classification performance of LeNet and DADM on MNIST, reported in Top-1 accuracy (%).

	Original		Rotate		Translate		Flip		Shuffle	
	LeNet	DADM	LeNet	DADM	LeNet	DADM	LeNet	DADM	LeNet	DADM
Class 0	98.44	96.07	87.77	91.90	26.21	69.09	59.55	95.84	3.14	96.07
Class 1	98.40	99.43	63.38	96.22	35.34	98.41	81.49	99.20	6.33	99.43
Class 2	96.73	95.05	55.12	80.37	44.07	70.28	6.28	94.82	20.15	95.05
Class 3	97.44	96.59	47.99	79.36	48.34	72.50	38.72	96.35	21.05	96.59
Class 4	97.54	96.74	53.79	85.84	38.44	79.75	39.22	96.51	6.88	96.74
Class 5	98.11	95.58	56.10	82.16	51.4	71.81	0.48	95.35	22.04	95.58
Class 6	96.31	95.25	36.74	77.51	45.41	76.94	1.06	95.02	1.32	95.25
Class 7	97.75	97.28	42.49	91.68	39.26	84.36	2.51	95.05	12.96	97.28
Class 8	97.16	96.62	66.40	82.69	34.41	74.20	71.08	96.39	9.81	96.62
Class 9	95.89	96.56	42.32	80.49	31.28	78.49	3.23	96.33	0.59	96.56

stability under these transformations across different categories. Such stability is largely due to DADM's ability to learn and utilize the global distributional information, which to some extent is invariant to spatial change.

Figure 2.4 shows an example of the feature extraction in LeNet and DADM when classifying the digit "2", providing further insights. While the image is changed by various transformations, the histograms computed by DADM maintain a consistent pattern, which is in contrast to the feature maps from LeNet where each transformation results in a visibly different feature representation. This highlights that DADM can recognize the underlying distribution despite the change of spatial arrangement of pixels, and thus can robustly perform image classification against affine transformation or even more challenging transformations such as shuffling pixels.

2.5 Discussion

One significant advantage of this work is that it explores a new direction that more closely resembles human visual capabilities, particularly regarding handling affine transformations such as rotations. The presented method is indeed more robust under these conditions, thereby making the model more applicable to real-world scenarios where data can be in various orientations and positions.

Furthermore, the proposed approach provides a differentiable histogram construction method for the distribution layers. This differentiability allows the distribution layers to learn not only the probability information in the raw input images, but also the distributions of features extracted by other neural network layers such as CNNs. This can further add value to distribution learning and enhance the explainability of the input feature. We will further our study to explore the abilities of our model in terms of these potential directions in future work.

2.6 Conclusion

In this work, we proposed differentiable arithmetic distribution learning to tackle the inherent limitations of CNNs in handling affine transformations. A cornerstone of our approach was the application of a KDE-based differentiable histogram as a replacement for traditional histograms. This provides us with a differentiable approach to model data distributions, thereby paving the way for seamless integration between distribution and neural networks. Accordingly, we formulate a novel neural network module called DADM that effectively captures the inherent distributional attributes of the input data. This results in a model that is not only robust to affine transformations, but also retains the advantages of conventional CNNs in local spatial feature extraction. Our experiments, which includes an ablation study and a comparison with LeNet, demonstrates the effectiveness and robustness of our approach. The results show that, while the performance is comparable on original datasets, the resilience of our model against affine transformations is notably superior.

Chapter 3

Deep Clustering via Distribution Learning

3.1 Introduction

Clustering is a fundamental task in the fields of data mining and computer vision [93]. It involves grouping data points from a dataset into clusters, where data points within the same cluster exhibit high similarity. While the optimization target seems straightforward, the design of an end-to-end clustering optimization method is not easy, especially considering high dimensional data. Thus, deep clustering is proposed by leveraging the fitting ability of deep neural networks to reduce the dimensionality of data, which achieves better results [61]. With this motivation, we embed the concept of deep clustering in our proposed algorithm.

Given the dimension-reduced data, we still need a clustering algorithm to form clusters in an unsupervised manner. In contrast to traditional clustering algorithms like k-means, distribution learning aims to learn the probability density functions from a set of data samples. Although some existing methods embed distribution learning models such as Gaussian Mixture Model (GMM) in deep clustering, there is still a lack of theoretical analysis to support their relationship. Besides, most distribution learning methods are not optimized for deep clustering. This leads to a constrained search space for distribution learning algorithms, where dimensionality of the data cannot be high. Also, unoptimized algorithms may form imbalanced or meaningless clusters.

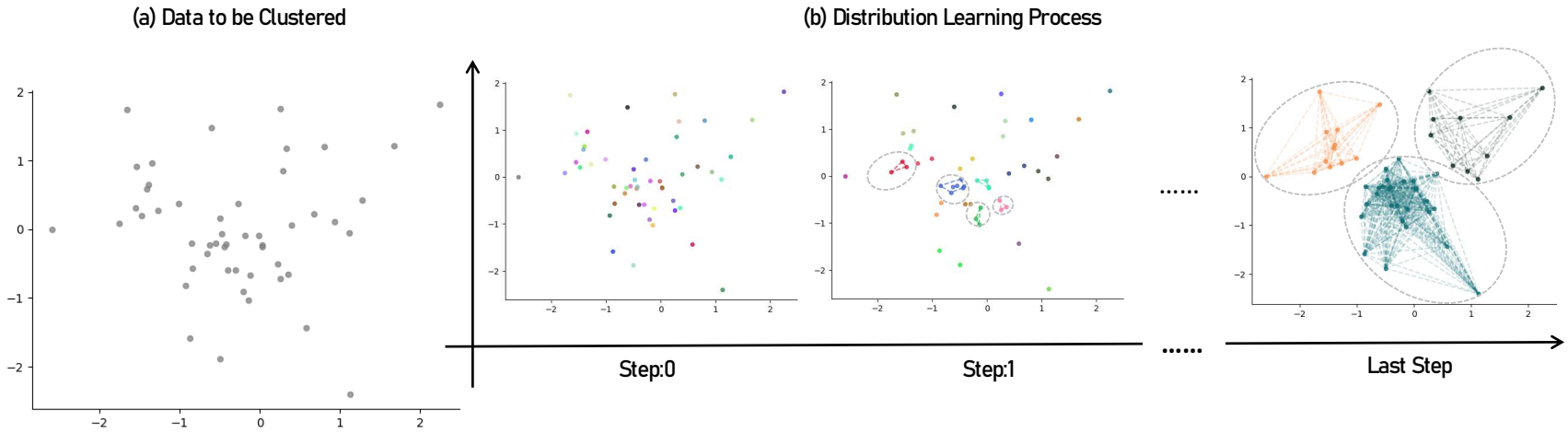


Figure 3.1: **The Relationship between Clustering and Distribution Learning.** (a): Gray points represent the data to be clustered. (b): The process of distribution learning. We consider each data point is sampled from an underlying distribution, shown as Step 0 with each point possessing a distinct color. Then, to formulate an explicit expression of the distribution with cluster information, we redistribute the model components and align with the underlying prior distribution iteratively, as shown from Step 1 to the last step. This optimization objective aligns with clustering.

The aforementioned limitations and the lack of theoretical foundations motivated us to propose Deep Clustering via Distribution Learning (DCDL). In DCDL, we first theoretically analyze the relationship between the clustering task and distribution learning. As Figure 3.1 shows, by treating each data point as a sample from an underlying distribution and considering the entire dataset as a mixture model, we can consider clustering as a process of simplifying a prior distribution. Subfigure (a) demonstrates the data to be clustered. Each point can represent a distribution component to form the initial mixture distribution, as shown in Subfigure (b) at Step 0. However, the explicit expression from Step 0 is meaningless since it does not convey the clustering information. We need to redistribute to learn a meaningful distribution for the clustering task. That is similar to compressing the prior distribution, as shown in the last step.

Following this concept, distribution learning can have the same optimization objective as clustering. The clustering results achieved by distribution learning can now be supported by theory rather than relying solely on empirical observation. Besides this, we propose a clustering-optimized distribution learning method called Monte-Carlo Marginalization for Clustering (MCMarg-C). In MCMarg-C, we penalize excessively large or small clusters and initialize centers of clusters by prior guidance. In addition, MCMarg-C can also directly learn distributions from very high dimensions (784 dimensions). These features, along with the remarkable experimental results on popular datasets, suggest that our MCMarg-C may be one of the best distribution learning methods in clustering.

The contributions in this paper are:

1. We conduct a theoretical analysis of the relationship between distribution learning and clustering. The analysis provides a novel perspective by viewing each data point as a distribution component. Thus, the distribution learning process can be seen as a redistribution of these Gaussian kernels. This aligns with the optimization objectives of clustering, providing theoretical support for using distribution learning in clustering

problems.

2. We introduce Deep Clustering via Distribution Learning (DCDL). In DCDL, we integrate distribution learning into the deep clustering framework. We employ an auto-encoder for dimensionality reduction and embed the latent vectors into a manifold space through manifold approximation. Finally, we use the proposed Monte-Carlo Marginalization for Clustering (MCMarg-C) algorithm for distribution learning to obtain cluster labels. Experimental results demonstrate that our DCDL outperforms state-of-the-art deep clustering methods.

3. We propose a distribution learning method that is specifically designed for the clustering task, named Monte Carlo Marginalization for Clustering (MCMarg-C). In MCMarg-C, we introduce prior guidance for means of cluster centers and penalize excessively large or small clusters. Considering that Monte Carlo Marginalization can also be used for extremely high dimensional data, MCMarg-C may be one of the best distribution learning methods in clustering.

3.2 Related Work

3.2.1 Deep Clustering

Due to the suboptimal performance of traditional clustering methods when applied directly to high-dimensional data, deep clustering methods have been proposed to map high-dimensional data into a feature space that is more suitable for clustering. In general, algorithms for deep clustering can be categorized into four main types: Deep Autoencoders (DAE) based algorithms [1], [10], [24], [26], [31], [40], [47], [58], [62], [65], [66], [79], [84], [85], Variational Autoencoders (VAE) based algorithms [8], [13], [21], [34], [41], [59], [77], Generative Adversarial Networks (GAN) based algorithms [23], [29], [49], [50], [68], [83], [92], and Graph Neural Networks (GNN) based algorithms [7], [70], [72], [87].

For DAE based deep clustering methods, Huang et al. introduced Deep

Embedding Network (DEN) [31] to enforce local constraints and group sparsity constraints on the learning objectives of DAE. Peng et al. [58] proposed deeP subspAce clusteRing with sparsiTY prior (PARTY) to enhance DAE with structural priors on the samples. Ren et al. introduced Deep Density-based Image Clustering (DDIC) [62] for density-based clustering on learned low-dimensional features. Xie et al. proposed Deep Embedding Clustering (DEC) [79] inspired by t-Distributed Stochastic Neighbor Embedding (t-SNE) [71], which jointly optimizes the learning of features and clustering objectives. Guo et al. improved DEC and introduced Improved Deep Embedded Clustering (IDEC) [26]. Their improvements ensure local structure preservation during the fine-tuning phase of DEC. Subsequently, Guo et al. also introduced Deep Embedded Clustering with Data Augmentation (DEC-DA) [27] to enhance the performance of DEC using data augmentation.

Recently, there has been a growing trend of DAE-based deep clustering methods with traditional machine learning algorithms. Affeldt et al. introduced Spectral Clustering via Ensemble Deep Autoencoder Learning (SC-EDAE) [1] to integrate spectral clustering into DAE. Chen et al. introduced Deep Manifold Clustering (DMC) [10]. They defined a locality-preserving objective to classify and parameterize unlabeled data points lying on multiple manifolds. Although Deep Clustering via Distribution Learning (DCDL) also embeds Uniform Manifold Approximation and Projection (UMAP), the only purpose is to map features to a suitable space for distribution learning. In fact, manifold space transformation is a common practice in both deep and non-deep clustering methods [10], [18], [47], [67].

3.2.2 Distribution Learning

Distribution Learning focuses on finding the explicit expression for a given distribution. Although distribution learning seems straightforward, there are a limited number of methods for approximating distributions. Kernel Density Estimation (KDE) [63] estimates the Probability Density Function (PDF) by selecting an appropriate kernel function and computing the density. Gaussian Mixture Model (GMM) [6] is a linear combination of multiple Gaussian distri-

butions, and each component distribution has its own mean, covariance, and mixture weights. For GMM, the Expectation-Maximization (EM) [12] algorithm is frequently used to update the GMM parameters. However, the EM algorithm suffers from challenges in handling high-dimensional data and non-differentiability, making it difficult to integrate with deep learning networks [14], [36]. Thus, based on these needs, we proposed Monte-Carlo Marginalization (MCMarg) [89] earlier, which is differentiable and can directly approximate high-dimensional data. Also, some deep-learning distribution learning methods have been proposed. Arithmetic Distribution Neural Network (ADNN) [90] converts distributions to histograms and histogram distribution kernels are updated. Based on ADNN, some applications have also been proposed, including moving object segmentation [15], vessel segmentation [88], and affine-transformation-invariant image classification [69].

3.3 Methodology

In this section, we first provide a theoretical analysis that explains the relationship between clustering and distribution learning. Subsequently, guided by the theoretical analysis, we demonstrate the pipeline and implementation detail of Deep Clustering via Distribution Learning (DCDL).

3.3.1 Problem Statement and Challenges

The purpose of clustering is to divide data points into k groups or clusters, such that points within the same cluster are similar to each other. However, due to the high-dimensionality of images (a 28×28 image has 784 dimensions), the complexity of searching for and optimizing cluster regions may increase exponentially with increase in the number of dimensions [28]. Thus, directly applying traditional clustering algorithms like K-means in a high-dimension yields suboptimal results [14].

Using Monte Carlo Marginalization (MCMarg) and direct observations in distribution learning, we significantly alleviated this issue, resulting in notable advantages over traditional clustering methods. Nonetheless, a formal analy-

sis of the relationship between distribution learning and clustering is missing, which leads to inefficiencies when directly applying distribution learning methods to clustering problems. In light of this, we will next provide a theoretical review about the relationship between distribution learning and clustering and extend our prior work accordingly.

3.3.2 Relationship between Distribution Learning and Clustering

Given a clustering task in a multi-dimensional space, we want to give a theoretical analysis that bridges the problems of distribution learning and clustering. Let us start from the perspective of distribution learning and consider that all data points are sampled from an intractable distribution. We first approximate the Probability Density Function (PDF) $\mathcal{F}(\cdot)$ of this underlying target distribution[1] using Kernel Density Estimation (KDE) with Gaussian kernels.

Here, KDE places a kernel over each data point, and computes the density estimate by adding these kernels. This process can be seen as computing a mixture model of distributions whose probability density functions are kernel functions. Mathematically, given a multi-dimensional dataset $\mathcal{X} = \{\mathbf{x}_i\} \subset \mathbb{R}^d$, the estimated probability density function $\mathcal{F}(\cdot)$ is:

$$\mathcal{F}(x) = \frac{1}{N \cdot |H|^{\frac{1}{2}}} \sum_{i=1}^{N} \mathcal{K}(H^{-\frac{1}{2}} \cdot (\mathbf{x} - \mathbf{x}_i)), \tag{3.1}$$

where $N = |\mathcal{X}|$, $\mathcal{K}(\cdot)$ is the multivariate kernel function, and H is a positive-definite $d \times d$ bandwidth matrix. Since we use the Gaussian kernel:

$$\mathcal{K}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d}} exp(-\frac{1}{2}\mathbf{x}^T\mathbf{x}), \tag{3.2}$$

we can rewrite $\mathcal{F}(\cdot)$ as:

$$\begin{aligned} \mathcal{F}(x) =& \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\sqrt{(2\pi)^d |H|}} exp(-\frac{1}{2}(\mathbf{x} - \mathbf{x}_i)^T H^{-1} (\mathbf{x} - \mathbf{x}_i)) \\ =& \sum_{i=1}^{N} \frac{1}{N} \mathcal{N}(\mathbf{x}; \mathbf{x}_i, H). \end{aligned} \tag{3.3}$$

[1]The target distribution is the distribution we want to learn. It means that this distribution has no explicit expression.

Compared to the PDF of a Gaussian Mixture Model (GMM):

$$p(\mathbf{x}) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x}; \mu_k, \mathbf{\Sigma}_k), \tag{3.4}$$

where w_k, μ_k and $\mathbf{\Sigma}_k$ are weight vector, mean vector and covariance matrix. We can conclude that the estimated target distribution in Equation 3.3 can be seen as another GMM that has N equal-weighted Gaussian components, each of which is centred at a unique data point x_i, and uses the bandwidth H as its covariance matrix, as Figure 3.1 (b), Step 0 shows.

Thus, the optimization in a clustering problem can be reformulated from the viewpoint of a distribution learning problem. That is, given the dataset $\mathcal{X}$, we consider it as a mixture model of N underlying distributions, where each data point is sampled from the corresponding distribution component. This underlying mixture model can be approximated by an equal-weighted GMM mirroring the entire dataset $q(\mathbf{x}) = \sum_{i=1}^{N} \frac{1}{N} \mathcal{N}(\mathbf{x}; \mathbf{x}_i, H)$. We then need to optimize a set of parameters $\Theta = \{(\mu_k, \mathbf{\Sigma}_k) | k = 1, \ldots, K\}$ for our GMM $p(\mathbf{x}; \Theta) = \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x}; \mu_k, \mathbf{\Sigma}_k)$, where K corresponds to the number of clusters. The optimization is guided by minimizing its divergence $\mathcal{D}(\cdot)$ from the underlying distribution $q(\mathbf{x})$. Mathematically,

$$\Theta^* = \arg\min_{\Theta} \mathcal{D}(p(\mathbf{x}; \Theta) || q(\mathbf{x})). \tag{3.5}$$

Now, each data point $\mathbf{x}$ must be assigned to exactly one cluster, which is represented by a Gaussian component in our context. Hence, we can define a latent variable z_k such that $z_k = 1$ if the k-th cluster is selected; and $z_k = 0$ otherwise. Thus, $\sum_k P(z_k = 1) = 1$. Now, we can formulate the process of assigning data points to clusters as finding the k that maximizes the probability of $z_k = 1$. That is, given the GMM parameter Θ, the cluster of data point $\mathbf{x}$ is:

$$\mathcal{C}(\mathbf{x}) = \arg\max_{k} P(z_k = 1 | \mathbf{x}, \Theta). \tag{3.6}$$

From Bayes' theorem, we have $P(z_k = 1 | \mathbf{x}, \Theta) = \frac{P(\mathbf{x} | z_k = 1, \Theta) P(z_k = 1)}{P(\mathbf{x} | \Theta)}$. Now, since we have $P(\mathbf{x}|\Theta) = \sum_k P(z_k = 1) P(\mathbf{x} | z_k = 1, \Theta) = \sum_k w_k \mathcal{N}(\mathbf{x}; \Theta_k)$, it can be inferred that the a prior probability of $P(z_k = 1)$ can be represented by w_k in

this scenario. Thus, Equation 3.6 can be rewritten as:

$$\mathcal{C}(\mathbf{x}) = \arg\max_k \frac{w_k \mathcal{N}(\mathbf{x}; \Theta_k)}{\sum_i w_i \mathcal{N}(\mathbf{x}; \Theta_i)}. \tag{3.7}$$

Additionally, to prevent any Gaussian component from having a significantly larger weight, we introduce a loss penalty term to minimize the variance of the weights of each Gaussian component. Specifically, we introduce GMM Weight Standard Deviation Loss ($L_{\text{GMM-WSD}}$):

$$\mathcal{L}_{\text{GMM-WSD}} = \sqrt{\frac{1}{K}\sum_{k=1}^{K}(w_k - \bar{w})^2} \tag{3.8}$$

Where $\bar{w}$ is the mean of weights in the learned GMM. Since standard deviation of the weights is penalized, the weights are stabilized. We can use the following Equation to determine the clusters:

$$\begin{aligned} \mathcal{C}(\mathbf{x}) &= \arg\max_k \frac{w_k \mathcal{N}(\mathbf{x}; \Theta_k)}{\sum_i w_i \mathcal{N}(\mathbf{x}; \Theta_i)} \\ &\simeq \arg\max_k \mathcal{N}(\mathbf{x}; \Theta_k). \end{aligned} \tag{3.9}$$

This way, the clustering problem is reformulated as a distribution learning problem.

In practice, we use the monte-carlo marginalization method to approximate the Kullback-Leibler (KL) divergence and guide the optimization process. It first selects random unit vectors $\vec{u}$, and marginalizes both the estimated target distribution $q(\mathbf{x})$ and our GMM $p(\mathbf{x}; \Theta)$ along $\vec{u}$ to obtain their respective marginal distributions, which are then compared in a lower-dimensional space. The optimization of the GMM parameters Θ is guided by minimizing the KL divergence between these marginal distributions. Mathematically, this is formulated as:

$$\begin{aligned} \mathcal{L}_{\text{KL}}(q(\mathbf{x}), p(\mathbf{x}; \Theta)) &= \int_{\vec{u}\in\mathbb{U}} \mathcal{D}_{KL}(q_{\vec{u}}(\mathbf{x}\cdot\vec{u}) || p_{\vec{u}}(\mathbf{x}\cdot\vec{u}; \Theta)) d\vec{u} \\ &\simeq \sum_{\vec{u}\in\mathbb{U}} \mathcal{D}_{KL}(q_{\vec{u}}(\mathbf{x}\cdot\vec{u}) || p_{\vec{u}}(\mathbf{x}\cdot\vec{u}; \Theta)), \end{aligned} \tag{3.10}$$

where $\mathcal{D}_{KL}(\cdot)$ is the KL divergence, and $\mathbb{U} = \{\vec{u} \in \mathcal{M} \mid \|\vec{u}\| = 1\}$.

Combining Equations 3.8 and 3.10, we can now derive a complete objective function $\mathcal{L}(\cdot)$. In our implementation, we also vectorize the computation of this objective function for efficiency as follows:

$$\mathcal{L}(\mathbf{x}, \mathbf{w}, \Theta) = \mathcal{L}_{\text{KL}}(q(\mathbf{x}), \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}; \Theta)) + c \times \mathcal{L}_{\text{GMM-WSD}}, \tag{3.11}$$

where $\mathbf{w} = [w_1 \; w_2 \; \cdots \; w_K]^T$ is a vector consisting of weights in the GMM, c is a parameter that controls the impact of $\mathcal{L}_{\text{GMM-WSD}}$, and $\mathbf{\Psi}(\mathbf{x}; \Theta) = [\mathcal{N}(\mathbf{x}; \Theta_1) \;\; \mathcal{N}(\mathbf{x}; \Theta_2) \;\; \cdots \;\; \mathcal{N}(\mathbf{x}; \Theta_K)]^T$ is a vector-valued function, with each entry representing a Gaussian component of the GMM.

Therefore, the learning and clustering can be expressed as:

$$\mathcal{C}(\mathbf{x}) = \underset{(k, \mathbf{w}, \Theta)}{\arg\max} \; \mathcal{L}(\mathbf{x}, \mathbf{w}, \Theta)^{-1}. \tag{3.12}$$

3.3.3 Implementation Details of Deep Clustering via Distribution Learning (DCDL)

As depicted in Figure 3.2, the proposed Deep Clustering via Distribution Learning (DCDL) demonstrates significant improvements and modifications compared to previous approaches. Consider a high-dimensional dataset $\mathcal{X} = \{x_i\}_{i=1}^N$, where $x_i \in \mathbb{R}^n$ represents a high-dimensional data point. Initially, dimensionality reduction is performed through a non-linear transformation by an Autoencoder, i.e., $f_{\text{enc}} : \mathbb{R}^n \to \mathbb{R}^m$ where $m \ll n$, to find a compact representation $z_i = f_{\text{enc}}(x_i)$ for each high-dimensional data point x_i. As this process involves deep neural networks, DCDL can also be categorized as a Deep Clustering algorithm.

Subsequently, all the low-dimensional encoded data $\mathcal{Z} = \{z_i\}_{i=1}^N$ undergoes manifold approximation through Uniform Manifold Approximation and Projection (UMAP), denoted as $f_{\text{umap}} : \mathbb{R}^m \to \mathbb{R}^{m'}$. This maps the data to a manifold space $\mathcal{M} = \{m'_i\}_{i=1}^N$, where $m'_i = f_{\text{umap}}(z_i)$, to maintain the local and global structure of the data in the new space.

Following this theoretical analysis in Section 3.3.2, we introduce an improved version of the MCMarg distribution learning algorithm, named MCMarg-C, specifically optimized for clustering tasks. The main enhancements in

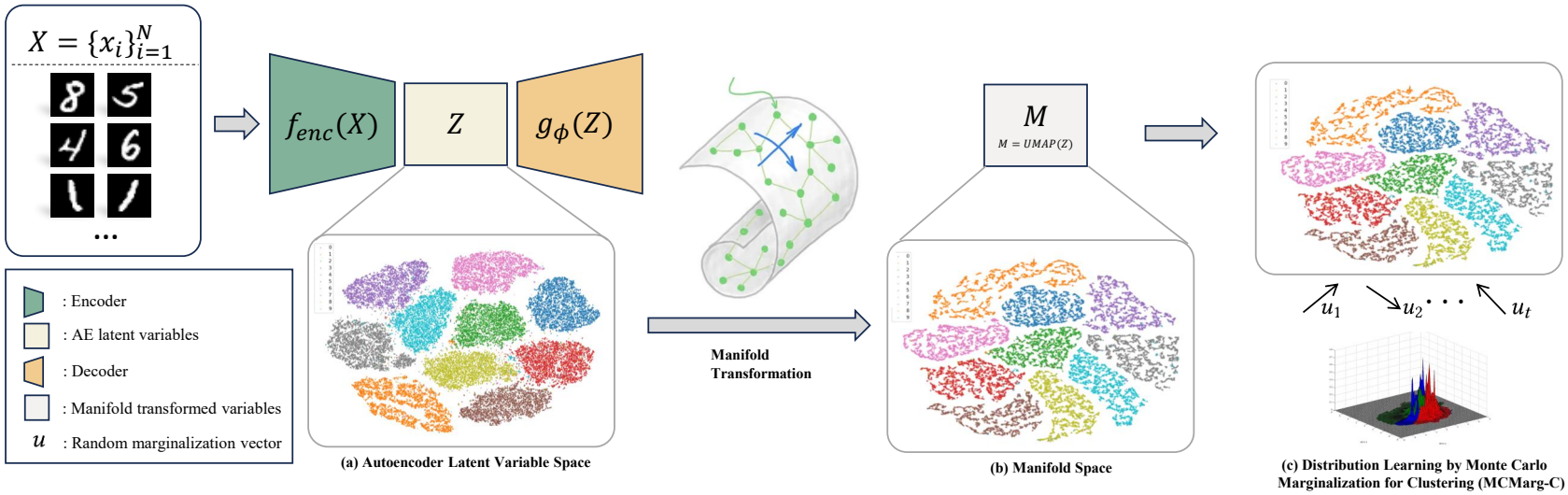


Figure 3.2: **Pipeline of Deep Clustering via Distribution Learning (DCDL).** The symbols depicted in the figure can be found with corresponding explanations in Algorithm 1. Different colors in subfigures (a), (b), and (c) represent different labels in the MNIST dataset. The arrows in (c) represent the direction of marginalization in Monte Carlo Marginalization for Clustering (MCMarg-C).

MCMarg-C include: (i) estimation of the Gaussian Mixture Model (GMM) means μ_k through the k-means algorithm, i.e., $\mu_k = \text{k-means}(\mathcal{M})$, to enhance the independence between model components; (ii) the introduction of a Gaussian Mixture Model Weight Standard Deviation Loss ($L_{\text{GMM-WSD}}$), to achieve a balanced distribution of weights during the model learning process. This enhances the stability of the clustering results and avoids the bias of an overly dominant Gaussian component.

3.3.4 Image Encoding

The autoencoder (AE) is a commonly used method in deep clustering to reduce data dimension. AE consists of an encoder $f_{\text{enc}}(x)$ and a decoder $g_\phi(z)$; the encoder maps a high-dimensional data point into a latent vector like $f_{\text{enc}} : \mathbb{R}^n \to \mathbb{R}^m$. The decoder performs the opposite operation, mapping from the latent space of dimension m back to the original high-dimensional space, as $g_\phi(z) : \mathbb{R}^m \to \mathbb{R}^n$. Suppose we have N input data points in our high-dimensional dataset $\mathcal{X}$; by applying the encoding function, we can obtain N points in the latent space, collected into a matrix $\mathcal{Z} = \{z_i | i \in [1, N]\} \in \mathbb{R}^{N \times m}$. We approximate the distribution of $\mathcal{Z}$ in the next step.

We use a simple autoencoder to perform feature embedding for high-dimensional data. The first layer reduces the dimension from n to 500, followed by another layer maintaining the dimension at 500. Finally, we pass the data to a bottleneck layer that compresses the data into an m-dimensional latent space. The decoder part of the AE operates in the reverse direction. It takes the m-dimensional latent vector and gradually reconstructs the data back to its original dimensionality.

3.3.5 Uniform Manifold Approximation and Projection (UMAP)

In the previous section, our high-dimensional data $\mathcal{X}$ was mapped to $\mathcal{Z} = \{z_i | i \in [1, N]\} \in \mathbb{R}^{N \times m}$. A manifold is a hypothetical space where locally it resembles a Euclidean space, but globally it may have different shapes and structures. In our implementation, we chose to perform a Manifold Approximation

of the embedded data obtained from the autoencoder using Uniform Manifold Approximation and Projection (UMAP) [48]. Specifically, the UMAP algorithm consists of two steps. First, it reconstructs a neighborhood graph. In this step, for each point x_i in the dataset, UMAP determines its neighborhood size and calculates a distance metric, such as the Euclidean distance metric. Then, UMAP constructs a weighted graph w, where the weight of each point x_j in the local neighborhood of point x_i is given by:

$$w_{ij} = \exp\left(-\frac{d(x_i, x_j)^2}{\sigma_i}\right) \tag{3.13}$$

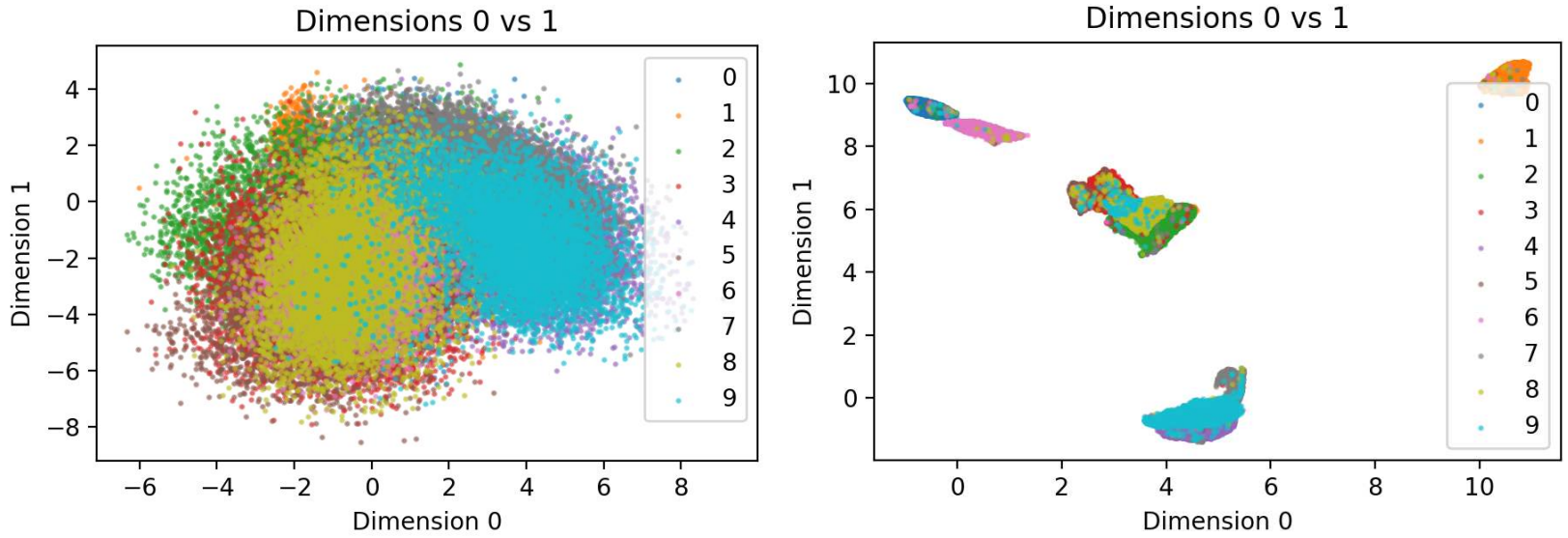
Here, $d(x_i, x_j)$ represents the distance between points x_i and x_j, and σ_i is a parameter that adjusts the density of the local neighborhood.

Next, UMAP randomly selects some points in the manifold space for initialization. UMAP uses stochastic gradient descent to optimize the positions of points in the manifold space. The objective is to minimize the cross-entropy loss function, which quantifies the difference between the neighborhood graphs in the high-dimensional and manifold spaces. Specifically, the following loss function C is applied:

$$C = \sum_{i,j} w_{ij} \log\left(\frac{1}{1 + a||y_i - y_j||^{2b}}\right) + (1 - w_{ij}) \log\left(1 - \frac{1}{1 + a||y_i - y_j||^{2b}}\right) \tag{3.14}$$

Here, y_i and y_j represent points in the manifold space, and a and b are curve parameters learned from the data using a robust regression model.

We visualize the embeddings of these two different spaces in Figure 3.3 for the MNIST dataset. The data distribution without transformation appears more disordered. However, the two-dimensional visualization results obtained with UMAP are more cohesive.



(a) MNIST latent space without UMAP

(b) MNIST latent space with UMAP

Figure 3.3: **Visualizing latent space of the MNIST dataset using autoencoder with and without using UMAP.** We visualize the plane projections of 0- and 1-dimensional spaces. We observe that the latent space transformed by UMAP exhibits sparser distributions between different labels and denser concentrations of points within each label.

3.3.6 Algorithm for Deep Clustering via Distribution Learning (DCDL)

Algorithm 1 Deep Clustering via Distribution Learning (DCDL) with n Clusters

Require: High-dimensional data $\{x_i\}_{i=1}^N$, where $x_i \in \mathbb{R}^n$
Ensure: Transformed representation $\{m_i'\}_{i=1}^N$, GMM parameters $\{\theta_{\text{GMM},k}\}_{k=1}^n$

Initialize autoencoder parameters θ_{enc}
for each training iteration **do**
    Encode x_i to get $z_i = f_{\text{enc}}(x_i)$
    Decode z_i to reconstruct $\hat{x}_i = g_\phi(z_i)$
    Update θ_{enc} by minimizing $\sum_{i=1}^N \|x_i - \hat{x}_i\|^2$
end for
Apply UMAP on encoded data $\{z_i\}_{i=1}^N$ to obtain $\{m_i'\}_{i=1}^N$
Initialize GMM parameters $\{\theta_{\text{GMM},k}\}_{k=1}^n$ for n clusters
for each transformed sample m_i' **do**
    for each cluster $k = 1$ to n **do**
        Update parameters $\theta_{\text{GMM},k}$ using MCMarg-C with m_i'
        // The update can be represented as: $\theta_{\text{GMM},k}^{(t+1)} = \text{MCMarg-C}\left(\theta_{\text{GMM},k}^{(t)}, m_i'\right)$
    end for
end for

In order to better understand the proposed Deep Clustering via Distribution Learning (DCDL), we present the algorithm for DCDL in this section, namely Algorithm 1.

3.4 Experimental Results

3.4.1 Experimental Setting

Our experiments are conducted on a NVIDIA RTX A4000 GPU. For the Auto-Encoder (AE), we utilize the Adam optimizer with a learning rate of 0.001 and apply batch normalization before generating the encoded vector. For the Gaussian Mixture Model (GMM), we use the Adam optimizer with a learning rate of 0.0001, and the number of unit vectors sampled each time is 32. For more experimental details, please refer our publicly available code.

3.4.2 Experimental Design

We first conduct a qualitative and quantitative analysis of the clustering results for Deep Clustering by Distribution Learning (DCDL) on different datasets. Next, we compare DCDL with the traditional Gaussian Mixture Model updating algorithm, i.e., the Expectation-Maximization (EM) algorithm. Additionally, we present a qualitative and quantitative comparison between Monte Carlo Marginalization Clustering (MCMarg-C) and the original MCMarg. Finally, we discuss the limitations of applying distribution learning in clustering problems and potential improvement strategies.

Baseline Methods: Our initial motivation for designing DCDL is to achieve state-of-the-art results. With this motivation, we conduct comparisons with state-of-the-art deep clustering methods on popular datasets to demonstrate the effectiveness of DCDL. These methods include DeepCluster [9], DCN [80], IDEC [26], SR-k-mean [33], VaDE [34], ClusterGAN [50], JULE [81], DEPICT [24], and DBC [40]; they have top performance in the field of deep clustering.

Subsequently, a deep neural network encodes the high-dimensional image data into a low-dimensional space. This makes it possible to directly compare our distribution learning approach (MCMarg-C) with the Expectation-Maximization (EM) algorithm. Finally, we make a direct comparison with the original MCMarg method to demonstrate the superiority of MCMarg-C.

Evaluation Metric: Our clustering performance is evaluated by three metrics: Adjusted Rand Index (ARI) [76], Normalized Mutual Information (NMI) [19] and Top-1 Accuracy (ACC).

Adjusted Rand Index (ARI) is a corrected version of the Rand Index (RI) and considers the effect of chance, making it suitable for evaluating the similarity between true and predicted cluster assignments. The formula for calculating the Adjusted Rand Index takes into account the combinations of items within the clusters:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]/\binom{n}{2}}, \tag{3.15}$$

where n_{ij} is the number of objects in both cluster i of the true clustering and

cluster j of the predicted clustering. a_i is the sum of n_{ij} over all j for a fixed i. b_j is the sum of n_{ij} over all i for a fixed j.

Normalized Mutual Information (NMI) is a statistical tool used to measure the similarity of clustering effects between two datasets. NMI is developed based on the concept of Mutual Information (MI) and involves normalization to ensure the evaluation is not affected by the size of clusters. Here, Mutual Information is a measure of the mutual dependence between two random variables. The definition of Mutual Information is given by:

$$MI(U,V) = \sum_{u \in U} \sum_{v \in V} P(u,v) \log \frac{P(u,v)}{P(u)P(v)}, \tag{3.16}$$

where U and V are two random variables, $P(u,v)$ is their joint probability distribution, and $P(u)$ and $P(v)$ are their marginal probability distributions. To overcome the issue of MI increasing with the number of clusters, normalization is introduced. Normalized Mutual Information (NMI) is achieved by dividing the MI value by a form of normalization, typically expressed as:

$$NMI(U,V) = \frac{2 \times MI(U,V)}{H(U) + H(V)}, \tag{3.17}$$

where $H(U)$ and $H(V)$ are the entropies of the random variables U and V respectively. This normalization ensures that the NMI value lies between 0 and 1, where 0 indicates no correlation and 1 indicates perfect correlation.

Top-1 Accuracy (ACC) is defined as the ratio of the number of times the clustering algorithm correctly predicts the most likely category to the total number of predictions made. Mathematically, assume a dataset contains N samples. For each sample i, we can generate a predicted category $\hat{y}_i$ using the clustering algorithm and a true category y_i. Top-1 Accuracy can be represented by the following formula:

$$\text{Top-1 Accuracy} = \frac{1}{N} \sum_{i=1}^{N} 1(\hat{y}_i = y_i) \tag{3.18}$$

Here, $1(\hat{y}_i = y_i)$ is an indicator function that takes the value 1 when the clustered category $\hat{y}_i$ equals the true category y_i, and is 0 otherwise.

3.4.3 Experimental Results

Comparison with State-of-the-Art Methods

Table 3.1 compares the clustering results of DCDL with state-of-the-art methods on different datasets. Since these datasets come with labeled data, we are able to calculate accuracy (ACC) and Normalized Mutual Information (NMI). Quantitative comparisons based on ACC and NMI demonstrate the promising clustering performance of DCDL. As a deep clustering algorithm, DCDL achieves three first-place rankings in ACC and one second-place ranking. For NMI, DCDL secures first-place ranking across all datasets.

In particular, although both explicit and implicit distribution learning methods are indirectly used in deep clustering, they are both unable to deal with high-dimensional data and imbalanced clusters. For example, DeepGMM [73] learned the distribution explicitly via GMM. The EM algorithm they used to update GMM is not specifically designed for the clustering task, which produces unsatisfactory result. On the other hands, VaDE [34] and ClusterGAN [50], which learn implicit distribution formation, also achieves suboptimal results. VaDE utilizes Variational Autoencoder [36] to map the data into the hypothetical distribution space. However, the actual data distribution may not follow the distribution hypothesized by VAE, as shown by [3], [52], [78], [89]. Thus, due to the differences between actual and presumed distributions, the clustering task does not perform well. This drawbacks are also presented in GAN based methods, where GAN [25] adversarially updates generators to produce data that are close to the real data distribution. For deep clustering, GAN may learn better data distributions than VAE, as ClusterGAN is better than VaDE in Table 3.1. However, MCMarg-C directly learns the explicit distribution without the concerns of high-dimensionality, which helps MCMarg-C achieve better result than EM-based, VAE-based and GAN-based deep clustering methods.

Moreover, for other deep clustering methods, we observe that DGG [82] achieves higher accuracy on MNIST (0.9760) compared to DCDL (0.9722). However, DGG's NMI score is significantly lower than ours (0.8800, com-

Table 3.1: **Comparison of different methods on MNIST, FashionMNIST, USPS, and Pendigits datasets.** Black bold represents the leading values, while red bold represents the second-ranked values.

Method	**MNIST**		**FashionMNIST**		**USPS**		**Pendigits**	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
DeepGMM [73]	0.7250	0.6400	0.4540	0.4100	0.6540	0.5100	-	-
DeepCluster [9]	0.7970	0.6610	0.5420	0.5100	0.5620	0.5400	-	-
DCN [80]	0.8300	0.8100	0.5010	0.5580	0.6880	0.6830	0.7200	0.6900
DEC [79]	0.8630	0.8340	0.5180	0.5460	0.7620	0.7670	0.7010	0.6780
IDEC [26]	0.8810	0.8670	0.5290	0.5570	0.7610	0.7850	0.7840	0.7230
SC-EDAE [1]	0.9323	0.8793	-	-	0.8178	0.8317	**0.8731**	**0.8100**
SR-k-means [33]	0.9390	0.8660	0.5070	0.5480	0.9010	0.9120	-	-
VaDE [34]	0.9450	0.8760	0.5780	0.6300	0.5660	0.5120	-	-
ClusterGAN [50]	0.9640	**0.9210**	**0.6300**	**0.6400**	-	-	0.7700	0.7300
JULE [81]	0.9640	0.9130	0.5630	0.6080	**0.9500**	**0.9130**	-	-
DBC [40]	0.9640	0.9170	-	-	-	-	-	-
DEPICT [24]	0.9650	0.9170	0.5830	0.6200	0.8990	0.9060	-	-
DGG [82]	**0.9760**	0.8800	0.6060	0.6100	0.9040	0.8200	-	-
DCDL (Our)	**0.9722**	**0.9278**	**0.6331**	**0.6992**	**0.9687**	**0.9222**	**0.8940**	**0.8768**

pared to DCDL's 0.9278). This also validates that our algorithm can actually produce more balanced clustering results. Compared with non-distribution learning deep clustering methods, these methods usually do not consider the formation of data distribution but form clusters based on other evidences, such as distance. However, since we provide a theoretical analysis for distribution learning and design MCMarg-C that is more suitable for clustering, distribution learning may become a good choice for deep clustering. The superior performance compared to non-distribution methods also verifies this.

Comparison with Expectation-Maximization (EM) and Original Monte Carlo Marginalization (MCMarg)

The Expectation-Maximization (EM) algorithm is an iterative optimization strategy used for estimating parameters in probabilistic models. The EM algorithm was introduced in the 1970s [12]. Over the following 50 years, although there have been numerous variations to the EM algorithm [20], [22], [42], [43], [53], [54], most of them have been based on theoretical innovations built upon the EM framework. No algorithm has managed to surpass the prominence of EM.

In the E-step, based on the current parameter estimates, EM computes or estimates the expected values of hidden variables. Then, the M-step updates the parameter estimates to maximize the likelihood of the observed data. When applied to a high-dimensional space, the EM algorithm faces two primary challenges. First, the EM algorithm is very sensitive to initial values and may converge to a local optima in high-dimensional spaces. Second, the EM algorithm requires update of the mean and covariance matrices for each Gaussian component during the M-step. Since this process has polynomial time complexity, the efficiency of the EM algorithm is significantly influenced by dimensionality, making its convergence difficult and time-consuming in high-dimensional spaces. Note that this is also one of the reasons why Variational Autoencoders (VAE) [36] and Evidence Lower Bound (ELBO) were introduced. As mentioned in the original VAE paper (Section 2.1.1), *"the EM algorithm cannot be used"* if the distribution is intractable.

Since a deep neural network is involved in reducing the data dimensionality,

Table 3.2: **Comparison of DCDL (EM), DCDL (MCMarg), and DCDL (MCMarg-C) on MNIST, FashionMNIST, USPS, and Pendigits datasets.** Bold values signify the top performance metrics across the datasets.

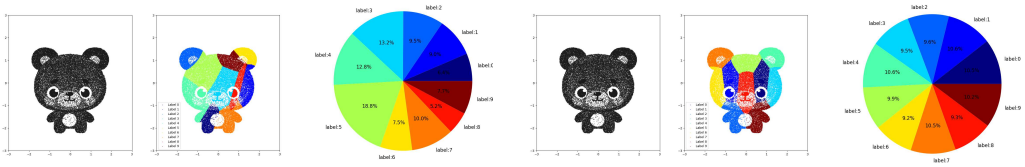
Method	**MNIST**			**FashionMNIST**		
	ACC	NMI	ARI	ACC	NMI	ARI
DCDL(EM)	0.9721	0.9276	0.9397	0.5899	0.6629	0.4668
DCDL(MCMarg)	0.8331	0.8882	0.8188	0.5332	0.6521	0.4543
DCDL(MCMarg-C)	**0.9722**	**0.9278**	**0.9399**	**0.6331**	**0.6992**	**0.5207**
Method	**USPS**			**Pendigits**		
	ACC	NMI	ARI	ACC	NMI	ARI
DCDL(EM)	0.9580	0.9012	**0.9404**	0.8928	0.8744	0.8070
DCDL(MCMarg)	0.8834	0.8976	0.8654	0.7307	0.8024	0.6359
DCDL(MCMarg-C)	**0.9687**	**0.9222**	0.9396	**0.8940**	**0.8768**	**0.8090**

we create a condition favorable to the EM algorithm. Thus, we also use the EM algorithm to update the GMM components and compared it with MCMarg-C. The comparison results are shown in Table 3.2.

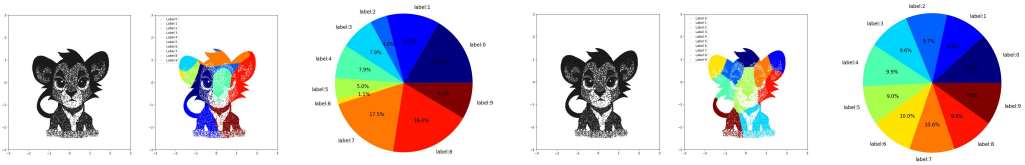
In the comparisons, we find in all dataset, MCMarg-C marginally outperforms others. In this experiment, we utilized the EM algorithm implemented in scikit-learn [55]. This experiment also confirms the excellent distribution learning capabilities of the EM algorithm in low-dimensional spaces, which is widely recognized in the academic community. Given a situation that is beneficial to EM, MCMarg-C achieves even better results. Considering that MCMarg-C is able to directly learn high-dimensional distributions and is differentiable, MCMarg-C may be the best distribution learning method in the field of clustering.

Table 3.2 presents a quantitative comparison between MCMarg-C and the original MCMarg [14]. We can see that the performance gap between MCMarg-C and MCMarg is even bigger than EM. The advantages of MCMarg-C come from two factors. First, MCMarg-C incorporates the GMM-Weight Standard Deviation Loss, which prevents clusters from dominating each other. Second, MCMarg-C utilizes k-means as a prior to initialize the GMM means, which accelerates the convergence of distribution learning. Thus, the improved MCMarg-C can achieve better performance compared to our original MCMarg.

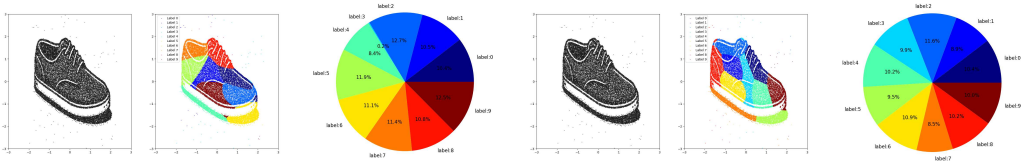
To further validate the above statement, we generate images using Dalle-3 [5] and obtain their two-dimensional data points based on the grayscale values. Then, we perform distribution learning using both the MCMarg-C and MCMarg methods. The results are shown in Figure 3.4. Each row in the figure represents a separate group. Different colors are used to represent each cluster with points. The pie chart illustrates the proportion of different points in the overall distribution. We can observe that MCMarg-C exhibits a more uniform clustering trend, while MCMarg tends to use a smaller number of Gaussians to describe the data distribution. For a clustering problem, MCMarg-C's performance is noticeably better.



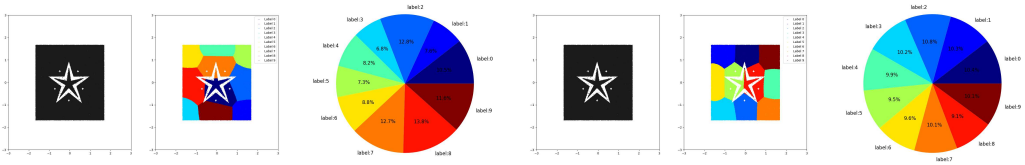
(a) Bear Clustering Result by MCMarg (b) Bear Clustering Result by MCMarg-C



(c) Lion Clustering Result by MCMarg (d) Lion Clustering Result by MCMarg-C



(e) Shoe Clustering Result by MCMarg (f) Shoe Clustering Result by MCMarg-C



(g) Star Clustering Result by MCMarg (h) Star Clustering Result by MCMarg-C

Figure 3.4: **Visual Comparison of MCMarg and MCMarg-C Clustering Result.** In each row, there is a separate control group. On the left side are the visual results of MCMarg, while on the right side are the visual results of MCMarg-C. Each cluster is represented by points of different colors. The pie chart illustrates the proportion of different points in the overall distribution. We can observe that MCMarg-C exhibits a more uniform clustering pattern, while MCMarg tends to use a smaller number of Gaussians to describe the data distribution.

Further Discussion

Figure 3.5 presents incorrect clustering examples by DCDL on the MNIST dataset. The Incorrect Cluster Visualization displays the misclassified examples. The numbers on each image represent DCDL's clustering labels. Additionally, we conduct a subjective study by three individuals with a background in computer vision to manually annotate these images. Before manual annotation, we shuffle the order of the images. Finally, we compute the accuracy of manual annotations compared to the ground truth labels. The results show that we do not achieve particularly high accuracy. The average accuracy among the three individuals was 65%.

Through the observation of misclassified images, we find that these images are indeed prone to confusion. For instance, digits belonging to category '3' achieve only 39% correct classification. For the digit '3', misclassified samples tend to share a similar appearance with other digits, which is one of the reasons for the ineffectiveness of DCDL. This finding shows the potential for DCDL in detecting mis-labelled data.

3.5 Conclusion

In this paper, we introduced a novel deep clustering algorithm called Deep Clustering via Distribution Learning (DCDL). This algorithm combines distribution learning with a deep clustering framework. With the theoretical analysis that supports distribution learning in clustering, we proposed clustering-optimized Monte Carlo Marginalization for clustering (MCMarg-C) to obtain clustering labels. Through the theoretical analysis and the improvements of MCMarg specifically designed for clustering, DCDL demonstrated superior performance compared to EM-based, VAE-based and GAN-based distribution learning deep clustering methods.

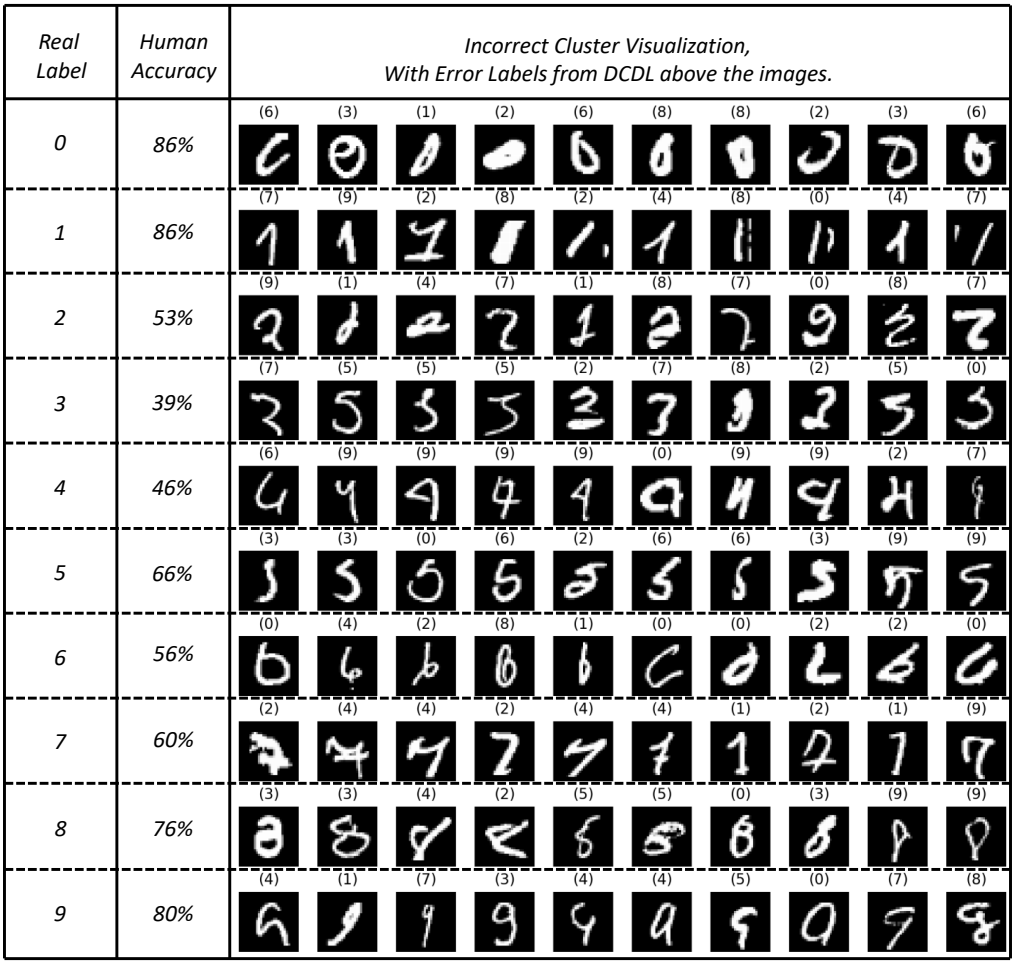


Figure 3.5: **DCDL Error Cluster Examples on the MNIST Dataset.** Real Label represents the true label of the images on the right. Incorrect Cluster Visualization shows the visual results of mis-clustered examples. The label results of DCDL are shown above each image. For Human Accuracy, we sought annotations from three individuals considering randomized image presentation. Accuracy reflects the agreement between human annotations and the ground truth labels in the dataset.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

In this thesis, we introduced two novel approaches for integrating distribution learning and deep learning into image classification and clustering tasks. The primary objective was to explore the connection between distribution learning and deep learning, and enhance the robustness and scalability of models accordingly, particularly in handling affine transformations and the curse of dimensionality.

In Chapter 2, the Differentiable Arithmetic Distribution Module (DADM) was proposed, which utilizes kernel density estimation to create differentiable histograms from images. This method enables the model to learn distributional information that is invariant to affine transformations, significantly enhancing the robustness of image classification models. Chapter 3 presented the Deep Clustering via Distribution Learning (DCDL), which integrates distribution learning into a deep clustering framework. By incorporating manifold learning and Monte Carlo marginalization techniques, DCDL can capture the underlying statistical distributions of embedded features, which improves the clustering performance on high-dimensional data.

Comprehensive experimental evaluations were also conducted to demonstrate the effectiveness and robustness of the proposed methods in comparison to the traditional and state-of-the-art approaches. Furthermore, this work provided theoretical insights into the relationships among distribution learning, deep learning, affine transformation invariance, and clustering. This can con-

tribute to a deeper understanding of the benefits and limitations of distribution learning techniques, offering insights for future research and development.

4.2 Future Work

The findings from this thesis have several implications for the field of computer vision. The integration of distribution learning into image classification and clustering frameworks offers a viable approach to enhancing the robustness of models against affine transformations and high-dimensional data challenges. Additionally, the proposed methods contribute to the development of more scalable deep learning models, capable of handling large and complex datasets more effectively and robustly.

Future research directions may include extending the principles and techniques developed in this thesis to other computer vision tasks such as object detection, semantic segmentation, and image generation. Further work could also focus on optimizing the computational efficiency of DADM and DCDL, making them more suitable for real-time applications. Investigating other distribution learning techniques and their integration with deep learning frameworks could yield additional improvements in model performance and robustness.

References

[1] S. Affeldt, L. Labiod, and M. Nadif, "Spectral clustering via ensemble deep autoencoder learning (sc-edae)," *Pattern Recognition*, vol. 108, p. 107 522, 2020.

[2] M. Afifi, M. A. Brubaker, and M. S. Brown, "Histogan: Controlling colors of gan-generated and real images via color histograms," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 7941–7950.

[3] J. Aneja, A. Schwing, J. Kautz, and A. Vahdat, "A contrastive learning approach for training variational autoencoder priors," *Advances in neural information processing systems*, vol. 34, pp. 480–493, 2021.

[4] M. Avi-Aharon, A. Arbelle, and T. R. Raviv, "Deephist: Differentiable joint and color histogram layers for image-to-image translation," *arXiv preprint arXiv:2005.03995*, 2020.

[5] J. Betker, G. Goh, L. Jing, *et al.*, "Improving image generation with better captions," *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2023.

[6] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4.

[7] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, and P. Cui, "Structural deep clustering network," in *Proceedings of the web conference 2020*, 2020, pp. 1400–1410.

[8] L. Cao, S. Asadi, W. Zhu, C. Schmidli, and M. Sjöberg, "Simple, scalable, and stable variational deep clustering," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2020, pp. 108–124.

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

[10] D. Chen, J. Lv, and Y. Zhang, "Unsupervised multi-manifold clustering by learning deep representation," in *Workshops at the thirty-first AAAI conference on artificial intelligence*, 2017.

[11] W.-C. Chiu and M. Fritz, "See the difference: Direct pre-image reconstruction and pose estimation by differentiating hog," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 468–476.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society: series B (methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[13] N. Dilokthanakul, P. A. Mediano, M. Garnelo, *et al.*, "Deep unsupervised clustering with gaussian mixture variational autoencoders," *arXiv preprint arXiv:1611.02648*, 2016.

[14] G. Dong, C. Zhao, and A. Basu, "Bridging distribution learning and image clustering in high-dimensional space," *arXiv preprint arXiv:2308.15667*, 2023.

[15] G. Dong, C. Zhao, X. Pan, and A. Basu, "Learning temporal distribution and spatial correlation for universal moving object segmentation," *arXiv preprint arXiv:2304.09949*, 2023.

[16] G. Du, X. Cao, J. Liang, X. Chen, and Y. Zhan, "Medical image segmentation based on u-net: A review.," *Journal of Imaging Science & Technology*, vol. 64, no. 2, 2020.

[17] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proceedings of the IEEE*, vol. 90, no. 7, pp. 1151–1163, 2002.

[18] E. Elhamifar and R. Vidal, "Sparse manifold clustering and embedding," *Advances in neural information processing systems*, vol. 24, 2011.

[19] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Transactions on neural networks*, vol. 20, no. 2, pp. 189–201, 2009.

[20] J. A. Fessler and A. O. Hero, "Space-alternating generalized expectation-maximization algorithm," *IEEE Transactions on signal processing*, vol. 42, no. 10, pp. 2664–2677, 1994.

[21] J. A. Figueroa and A. R. Rivera, "Is simple better?: Revisiting simple generative models for unsupervised clustering," in *NIPS Workshop on Bayesian Deep Learning*, 2017.

[22] A. E. Gelfand and A. F. Smith, "Sampling-based approaches to calculating marginal densities," *Journal of the American statistical association*, vol. 85, no. 410, pp. 398–409, 1990.

[23] K. Ghasedi, X. Wang, C. Deng, and H. Huang, "Balanced self-paced learning for generative adversarial clustering network," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4391–4400.

[24] K. Ghasedi Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5736–5745.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[26] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation.," in *Ijcai*, vol. 17, 2017, pp. 1753–1759.

[27] X. Guo, E. Zhu, X. Liu, and J. Yin, "Deep embedded clustering with data augmentation," in *Asian conference on machine learning*, PMLR, 2018, pp. 550–565.

[28] P. Hammer, *Adaptive control processes: A guided tour (r. bellman)*, 1962.

[29] W. Harchaoui, P.-A. Mattei, and C. Bouveyron, "Deep adversarial gaussian mixture auto-encoder for clustering," 2017.

[30] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2.

[31] P. Huang, Y. Huang, W. Wang, and L. Wang, "Deep embedding network for clustering," in *2014 22nd International conference on pattern recognition*, IEEE, 2014, pp. 1532–1537.

[32] M. A. Hussain, G. Hamarneh, and R. Garbi, "Learnable image histograms-based deep radiomics for renal cell carcinoma grading and staging," *Computerized Medical Imaging and Graphics*, vol. 90, p. 101 924, 2021.

[33] M. Jabi, M. Pedersoli, A. Mitiche, and I. B. Ayed, "Deep clustering: On the link between discriminative models and k-means," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 6, pp. 1887–1896, 2019.

[34] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," *arXiv preprint arXiv:1611.05148*, 2016.

[35] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie, "On the learnability of discrete distributions," in *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, 1994, pp. 273–282.

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

[37] P. Kontkanen and P. Myllymäki, "Mdl histogram density estimation," in *Artificial intelligence and statistics*, PMLR, 2007, pp. 219–226.

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

[39] X. Lei, H. Pan, and X. Huang, "A dilated cnn model for image classification," *IEEE Access*, vol. 7, pp. 124 087–124 095, 2019.

[40] F. Li, H. Qiao, and B. Zhang, "Discriminatively boosted image clustering with fully convolutional auto-encoders," *Pattern Recognition*, vol. 83, pp. 161–173, 2018.

[41] X. Li, Z. Chen, L. K. Poon, and N. L. Zhang, "Learning latent superstructures in variational autoencoders for deep multidimensional clustering," *arXiv preprint arXiv:1803.05206*, 2018.

[42] C. Liu and D. B. Rubin, "The ecme algorithm: A simple extension of em and ecm with faster monotone convergence," *Biometrika*, vol. 81, no. 4, pp. 633–648, 1994.

[43] C. Liu, D. B. Rubin, and Y. N. Wu, "Parameter expansion to accelerate em: The px-em algorithm," *Biometrika*, vol. 85, no. 4, pp. 755–770, 1998.

[44] H. Liu, F. Zhang, X. Zhang, *et al.*, "Boosting few-shot text classification via distribution estimation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 13 219–13 227.

[45] L. Liu, Z. Pan, and B. Lei, "Learning a rotation invariant detector with rotatable bounding box," *arXiv preprint arXiv:1711.09405*, 2017.

[46] L. Liu, M. Saerbeck, and J. Dauwels, "Affine disentangled gan for interpretable and robust av perception," *arXiv preprint arXiv:1907.05274*, 2019.

[47] R. McConville, R. Santos-Rodriguez, R. J. Piechocki, and I. Craddock, "N2d:(not too) deep clustering via clustering the local manifold of an autoencoded embedding," in *2020 25th international conference on pattern recognition (ICPR)*, IEEE, 2021, pp. 5145–5152.

[48] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[49] N. Mrabah, M. Bouguessa, and R. Ksantini, "Adversarial deep embedded clustering: On a better trade-off between feature randomness and feature drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 4, pp. 1603–1617, 2020.

[50] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, "Clustergan: Latent space clustering in generative adversarial networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 4610–4617.

[51] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[52] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?" *arXiv preprint arXiv:1810.09136*, 2018.

[53] S. K. Ng, T. Krishnan, and G. J. McLachlan, "The em algorithm," *Handbook of computational statistics: concepts and methods*, pp. 139–172, 2012.

[54] S.-K. Ng and G. J. McLachlan, "On the choice of the number of blocks with the incremental em algorithm for the fitting of normal mixtures," *Statistics and Computing*, vol. 13, pp. 45–55, 2003.

[55] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[56] J. Peeples, W. Xu, and A. Zare, "Histogram layers for texture analysis," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 4, pp. 541–552, 2021.

[57] J. Peeples, A. Zare, J. Dale, and J. Keller, "Histogram layers for synthetic aperture sonar imagery," in *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2022, pp. 176–182.

[58] X. Peng, S. Xiao, J. Feng, W.-Y. Yau, and Z. Yi, "Deep subspace clustering with sparsity prior.," in *IJCAI*, 2016, pp. 1925–1931.

[59] V. Prasad, D. Das, and B. Bhowmick, "Variational clustering: Leveraging variational autoencoders for image clustering," in *2020 international joint conference on neural networks (IJCNN)*, IEEE, 2020, pp. 1–10.

[60] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[61] Y. Ren, J. Pu, Z. Yang, *et al.*, "Deep clustering: A comprehensive survey," *arXiv preprint arXiv:2210.04142*, 2022.

[62] Y. Ren, N. Wang, M. Li, and Z. Xu, "Deep density-based image clustering," *Knowledge-Based Systems*, vol. 197, p. 105 841, 2020.

[63] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *The annals of mathematical statistics*, pp. 832–837, 1956.

[64] V. Sedighi and J. Fridrich, "Histogram layer, moving convolutional neural networks towards feature-based steganalysis," *Electronic Imaging*, vol. 29, pp. 50–55, 2017.

[65] S. A. Shah and V. Koltun, "Robust continuous clustering," *Proceedings of the National Academy of Sciences*, vol. 114, no. 37, pp. 9814–9819, 2017.

[66] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "Auto-encoder based data clustering," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 18th Iberoamerican Congress, CIARP 2013, Havana, Cuba, November 20-23, 2013, Proceedings, Part I 18*, Springer, 2013, pp. 117–124.

[67] R. Souvenir and R. Pless, "Manifold clustering," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, IEEE, vol. 1, 2005, pp. 648–653.

[68] J. T. Springenberg, "Unsupervised and semi-supervised learning with categorical generative adversarial networks," *arXiv preprint arXiv:1511.06390*, 2015.

[69] Z. Tan, G. Dong, C. Zhao, and A. Basu, "Affine-transformation-invariant image classification by differentiable arithmetic distribution module," *arXiv preprint arXiv:2309.00752*, 2023.

[70] Z. Tao, H. Liu, J. Li, Z. Wang, and Y. Fu, "Adversarial graph embedding for ensemble clustering," in *International Joint Conferences on Artificial Intelligence Organization*, 2019.

[71] L. Van Der Maaten, "Learning a parametric embedding by preserving local structure," in *Artificial intelligence and statistics*, PMLR, 2009, pp. 384–391.

[72] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, "Attributed graph clustering: A deep attentional embedding approach," *arXiv preprint arXiv:1906.06532*, 2019.

[73] J. Wang and J. Jiang, "Unsupervised deep clustering via adaptive gmm modeling and optimization," *Neurocomputing*, vol. 433, pp. 199–211, 2021.

[74] W. Wang, Y. Yang, X. Wang, W. Wang, and J. Li, "Development of convolutional neural network and its application in image classification: A survey," *Optical Engineering*, vol. 58, no. 4, pp. 040 901–040 901, 2019.

[75] Z. Wang, H. Li, W. Ouyang, and X. Wang, "Learnable histogram: Statistical context features for deep neural networks," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, Springer, 2016, pp. 246–262.

[76] M. J. Warrens and H. van der Hoef, "Understanding the adjusted rand index and other partition comparison indices based on counting object pairs," *Journal of Classification*, vol. 39, no. 3, pp. 487–509, 2022.

[77] M. Willetts, S. Roberts, and C. Holmes, "Disentangling to cluster: Gaussian mixture variational ladder autoencoders," *arXiv preprint arXiv:1909.11501*, 2019.

[78] Z. Xiao, K. Kreis, J. Kautz, and A. Vahdat, "Vaebm: A symbiosis between variational autoencoders and energy-based models," *arXiv preprint arXiv:2010.00654*, 2020.

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

[80] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *international conference on machine learning*, PMLR, 2017, pp. 3861–3870.

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

[82] L. Yang, N.-M. Cheung, J. Li, and J. Fang, "Deep clustering by gaussian mixture variational autoencoders with graph embedding," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6440–6449.

[83] X. Yang, J. Yan, Y. Cheng, and Y. Zhang, "Learning deep generative clustering via mutual information maximization," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[84] X. Yang, C. Deng, K. Wei, J. Yan, and W. Liu, "Adversarial learning for robust deep clustering," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9098–9108, 2020.

[85] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu, "Deep spectral clustering using dual autoencoder network," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4066–4075.

[86] I. Yusuf, G. Igwegbe, and O. Azeez, "Differentiable histogram with hard-binning," *arXiv preprint arXiv:2012.06311*, 2020.

[87] X. Zhang, H. Liu, Q. Li, and X.-M. Wu, "Attributed graph clustering via adaptive graph convolution," *arXiv preprint arXiv:1906.01210*, 2019.

[88] C. Zhao and A. Basu, "Pixel distribution learning for vessel segmentation under multiple scales," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, IEEE, 2021, pp. 2717–2721.

[89] C. Zhao, G. Dong, and A. Basu, "Learning distributions via monte-carlo marginalization," *arXiv preprint arXiv:2308.06352*, 2023.

[90] C. Zhao, K. Hu, and A. Basu, "Universal background subtraction based on arithmetic distribution neural network," *IEEE Transactions on Image Processing*, vol. 31, pp. 2934–2949, 2022.

[91] J. Zhao12, J. Li, F. Zhao, S. Yan13, and J. Feng, "Marginalized cnn: Learning deep invariant representations," 2017.

[92] P. Zhou, Y. Hou, and J. Feng, "Deep adversarial subspace clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1596–1604.

[93] S. Zhou, H. Xu, Z. Zheng, *et al.*, "A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions," *arXiv preprint arXiv:2206.07579*, 2022.

[94] L. Zhu, D. Ji, S. Zhu, W. Gan, W. Wu, and J. Yan, "Learning statistical texture for semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 537–12 546.