이학석사 학위논문

Rotationally Invariant Clustering with Implicit Neural Representations

음적 신경망 표현을 활용한 회전 불변 군집화

2022년 8월

서울대학교 대학원
수리과학부
권 세 현

Rotationally Invariant Clustering with Implicit Neural Representations

지도교수 Ernest K.Ryu

이 논문을 이학석사 학위논문으로 제출함

2022년 8월

서울대학교 대학원
수리과학부
권 세 현

권세현의 이학석사 학위논문을 인준함
2022년 8월

위 원 장	강명주	(인)
부위원장	Ernest K.Ryu	(인)
위 원	Otto van Koert	(인)

Abstract

Image clustering is important task in machine learning, deep learning and industry. Especially in deep learning, clustering is becoming increasingly important because it is used not only for clustering but also for other purposes such as pretext task in self-supervised learning. Many previous literatures have shown excellent performance in benchmark datasets, but these datasets are not rotated. However, in practically there is no guarantee that the dataset will always be placed right. We tackle that existing prior algorithms do not work well when images are randomly rotated. In this paper, by leveraging Implicit Neural Representations(INR), 1. We obtain a latent vector, where latent rotation angle and rotationally invariant latent vector are disentangled from each other. 2. We show that clustering by rotationally invariant latent vectors have superior performance on randomly rotated datasets than other methods. To the best of our knowledge, it is the first approach to cluster with Implicit Neural Representations.

Keywords: Rotationally invariant, Clustering, Deep learning, Implicit neural representations, Artificial intelligence.

Student ID: 2020-21133

Contents

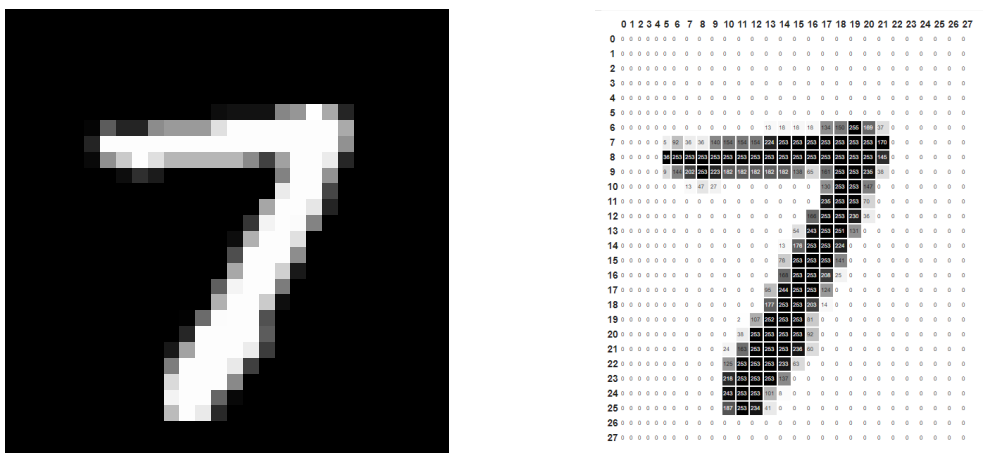
List of Figures

List of Tables

1 Review the Implicit Neural Representations

1.1 Introduction to INR

In machine learning, data is traditionally represented by discrete signal. For example, images are represented by their 2D array pixels(Fig1.1), audio is vectors of discretely sampled waveforms, and 3D shapes are usually parameterized as grids of voxels, point clouds, or meshes. In image, all we know is



(a) Image **(b)** Represented as 2D pixel arrays

Figure 1.1 Traditional representation of image.

RGB values of finite points in spatial domain of the image. For example, if spatial domain of the image is normalized to $[0,1]^2$ box and resolution of the image is $28 * 28$, we know RGB values at 784 points(e.g, $(0,0),(0,\frac{1}{28}),\ldots$) in $[0,1]^2$ box. However, the underlying real image not only has RGB values

at 784 points, but also has RGB values at all points in the box $[0,1]^2$(i.e, infinite resolution). Of course, it is possible to measure the RGB values at any point in the spatial domain by taking multiple pictures. Underlying signals of other domains are often continous as well as image. Therefore, it is natural to try to represent these signals as continuous manner. However, these continuous expressions of signals are not tractable in the sense that it is impossible write down to a mathematical formula. **Implicit Neural Representations(INR)** parameterize these signals by neural network that maps domain of the signal(e.g, coordinates of pixel in image) to some values of at that domain(e.g, RGB value of image).

In INR, general signals are parametrized by continuous function as following:

$$f : \Lambda \times \mathbb{R}^d \longrightarrow \mathbb{R}^k$$
$$\big(\lambda, (x_1, \ldots, x_d)\big) \longmapsto f\left(\lambda, (x_1, \ldots, x_d)\right)$$

where Λ is parameter space of neural network, d is domain dimension, and k is feature dimension of the signal. For each one $\lambda \in \Lambda$ represents one signal. For example, in 2-dimensional images, d(dimension of xy coordinate) and k(dimension of RGB value) is going to be 2 and 3 respectively. So 2-dimensional images can be represented by the following form:

$$f : \Lambda \times \mathbb{R}^2 \longrightarrow \mathbb{R}^3$$
$$\big(\lambda, (x, y)\big) \longmapsto \big(\lambda, (x, y)\big) = (R, G, B)$$

Note again, the one $\lambda \in \Lambda$ corresponds to one image, so if we fix the parameter $\lambda \in \Lambda$, then f_λ represents one image(Fig 1.2), where

$$
\begin{aligned}
f_\lambda : \mathbb{R}^2 &\longrightarrow \mathbb{R}^3 \\
(x, y) &\longmapsto f_\lambda(x, y) \\
&= f\left(\lambda, (x, y)\right) \\
&= (R, G, B)
\end{aligned}
$$

The goal of the INR is to find $\lambda \in \Lambda$. We obtain λ through training the neural

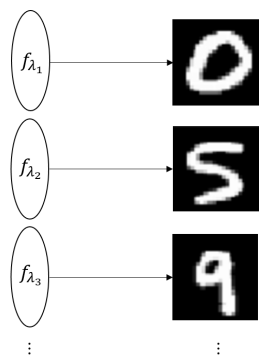
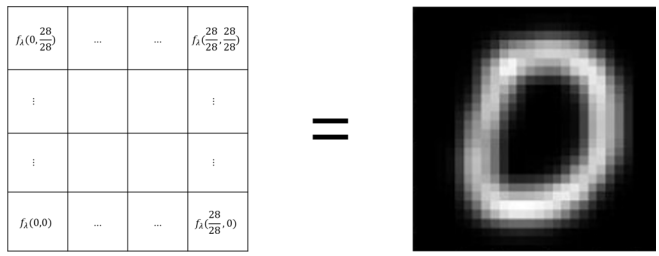


Figure 1.2 Each λ represents different images.

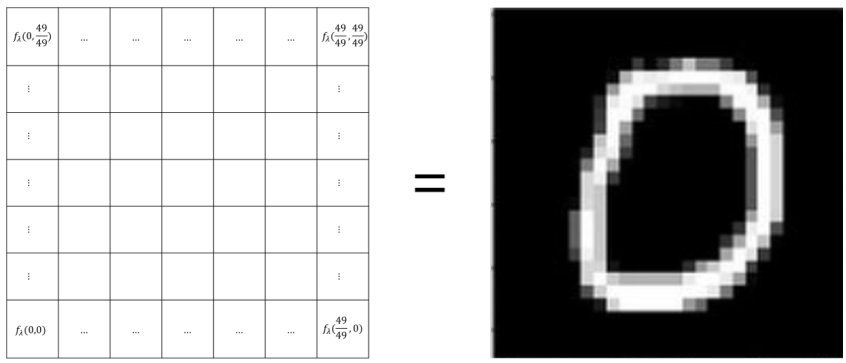
network. After obtaining the λ, then we can express a discrete signal again. For example, assume that the scale of coordinates system is normalized to $[0, 1]^2$ and we want to generate 2-dimensional discrete grey image with resolution $28 * 28$. All we have to do is input the 784 coordinates to the f_λ, i.e. input

$$
\begin{cases}
(0,0), (\frac{1}{28}, 0), (\frac{2}{28}, 0), \cdots, (\frac{28}{28}, 0), \\
(0, \frac{1}{28}), (\frac{1}{28}, \frac{1}{28}), (\frac{2}{28}, \frac{1}{28}), \cdots, (0, \frac{28}{28}), \\
\vdots \\
(0, \frac{28}{28}), (\frac{1}{28}, \frac{28}{28}), (\frac{2}{28}, \frac{28}{28}), \cdots (\frac{28}{28}, \frac{28}{28})
\end{cases}
$$

then collect 784 outputs, and finally reshape it to $28 * 28$ dimensional vector to obtain an image. It is possible to obtain any other resolutions(Fig 1.3). Not only represents these discrete signals, INR have some nice properties



(a) $28 * 28$ resolution



(b) $49 * 49$ resolution

Figure 1.3 Generate any resolution of an image.

and applications. First, *memory efficient*. Data can be stored and expressed regardless of spatial resolution(e.g, resolution of the 2-d image), depending only on the complexity of the underlying signal (i.e, complex signal needs many parameters and simple signal needs a few parameters of neural network). Second, *super resolution*. Since INR is a function defined on spatial domain(e.g, 2-D plane), it can represent any resolution. Thrid, *gradient of data*. Gradients of derivative of the data can be analytically calculated, which makes INR to a new method for solving inverse problems and differential equations. And

there exist many other applications, such as image inpainting. We review these properties in the following section.

1.2 Applications of INR

There are some reasons that Implicit Neural Reprsentations(INR) is interesting. First,*memory efficient*. The way of expressing pixel grid image or 3D-scene data depends on spatial resolution. For example, spatial dimension of the image is 2, so if the image becomes n times larger, the required memory becomes n^2 more. And spatial resolution of volumetric scene is 5(spatial location (x, y, z) with (θ, ϕ) view direction, so if the 3d-scene becomes n times larger, the required memory becomes n^5 more. However, in INR, data doesn't depend on their spatial resolution, but depends only on the complexity of the underlying signal (i.e, complex signal needs many parameters and simple signal needs a few parameters of neural network). This memory efficient property becomes huge advantage in 3D computer vision. NeRF[22] parametrizes volumetric scene using MLP, whose input is 5d coordinate (3-dimensional spatial location and viewing direction (θ, ϕ)) and whose output is the volume density and view-dependent RGB value. (Fig 1.4)

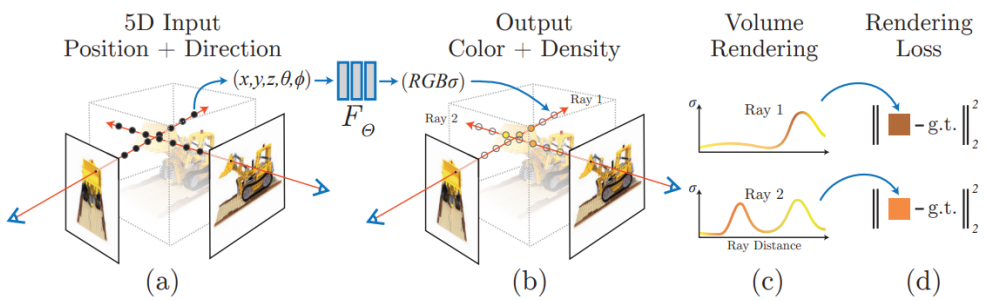


Figure 1.4 Diagram of NeRF[22]

and succesfully and memory efficiently representing the 3d scene (Fig 1.5).

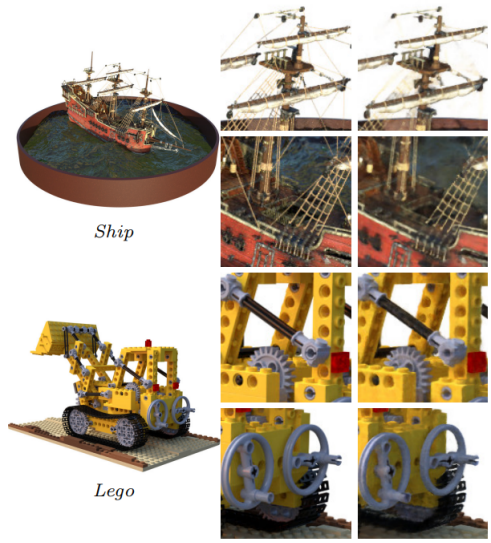


Figure 1.5 reconstruct 3-dimensional scene by NeRF[22]

Second, since INR is a function defined on spatial domain, it can do super resolution very easily: just input the coordinates corresponding to the resolution to the neural network.

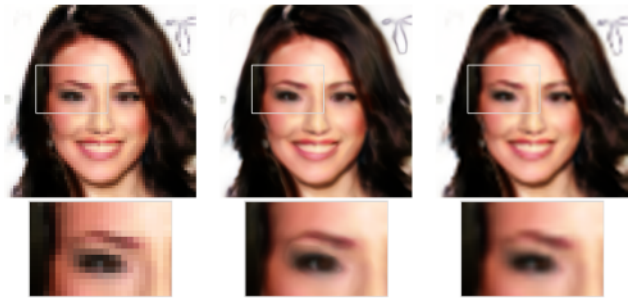


Figure 1.6 Superresolution. The first column is original resolution, the second column is 4 times resolution than the original, and the third column is bicubic upsampling[11]

.

Thrid, INR provides analytically calculable gradient of data. Using this fact, there are some applications of solving inverse problems, differential equations and others. For example, SIREN solved Helmholtz equations using INR.

Helmholtz equation is formulated as

$$\left(\nabla^2 + \frac{w^2}{c(\mathbf{x})^2}\right) p(\mathbf{x}) = -q(\mathbf{x})$$

where $p(\mathbf{x})$ is the unknown wavefield, $q(\mathbf{x})$ is known source, and $c(\mathbf{x})$ is a function of the wave velocity. SIREN solved this equation by parameterizing unkwon wavefield $p(\mathbf{x}$ by implicit neural representations. Domain is $\Omega = \left\{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{x}\|_\infty < 1\right\}$. So input randomly sampled $\mathbf{x} \in \Omega$ to p to minimize the loss function: $\mathcal{L}_{\text{Helmholtz}} = \int_\Omega \|(\nabla^2 + \frac{w^2}{c(\mathbf{x})^2})p(\mathbf{x}) + q(\mathbf{x})\|_1 d\mathbf{x}$.

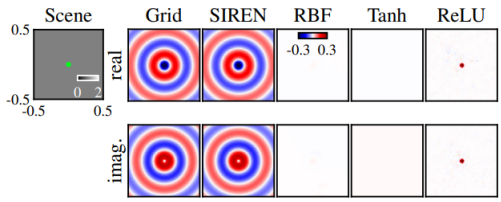


Figure 1.7 Single point source(green dot) is located in the center of $[-0.5, 0.5]^2$ box with uniform wave propagation(top left). SIREN (second column of right image) obtained a similar solution of the helmholtz equation compared to principled grid solver(first column of right image). while other activation function based methods failed to solve.(thrid, fourth and fifth column of right image)[35]

Also, there are interesting observations with implicit neural representations. For example, *extrapolating*. [11] build generative models that generate image as a parametrized continuous function. They used a hypernetwork to generate the parameters of the image function, and a discriminator that takes coordinates (e.g. pixel locations) and features (e.g. RGB values) as an input, and trained with an adversarial approach. (Fig1.8)

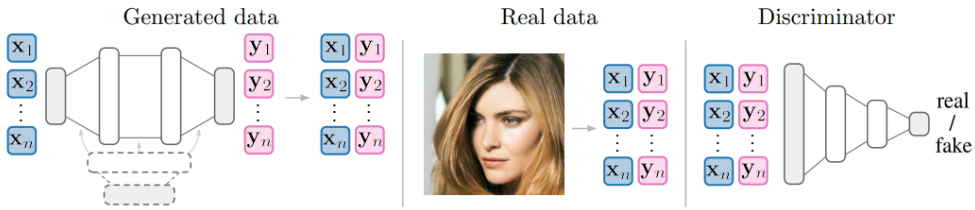


Figure 1.8 Generating image function by hypernetwork[11]

In contrast to existing generative models are based on discrete signal(e.g, pixel grid), it learns distributions of continuous signals, and hence to agnostic to discretization. Not only they succeeded to generate a continuous signal well, they observed one interesting thing. During training, the coordinates used for training were normalized to $[0, 1]$, but after training, it was observed that realistic images are generated even for coordinates that are out of this range(e.g, $[-0.3, 1.3]$). (Fig 1.9)

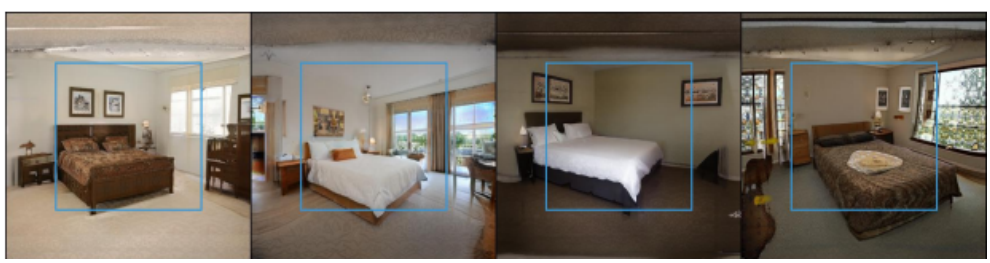


Figure 1.9 Extrapolating. During training, only $[0, 1]^2$ squared coordinates was used, and evaluate it on coordinates from $[-0.3, 1.3]^2$ square.[11]

This would be an interesting example to observe in a generative model using implicit neural representations. In the next section, we will discuss the important techniques of the implicit neural representations that improves the quality and hence make these applications possible.

1.3 Techniques of INR

We reviewed some applications in the previous section. However, these applications are not achieved by naively apply to idea of implicit neural representations. In this section, we will review the techniques from two perspectives of INR. The first is "*how to catch the high frequency of the data*", and the second is "*how to construct the weight of the function representation*". We will first review at the first perspective. In previous works, ReLU based multilayer perceptron

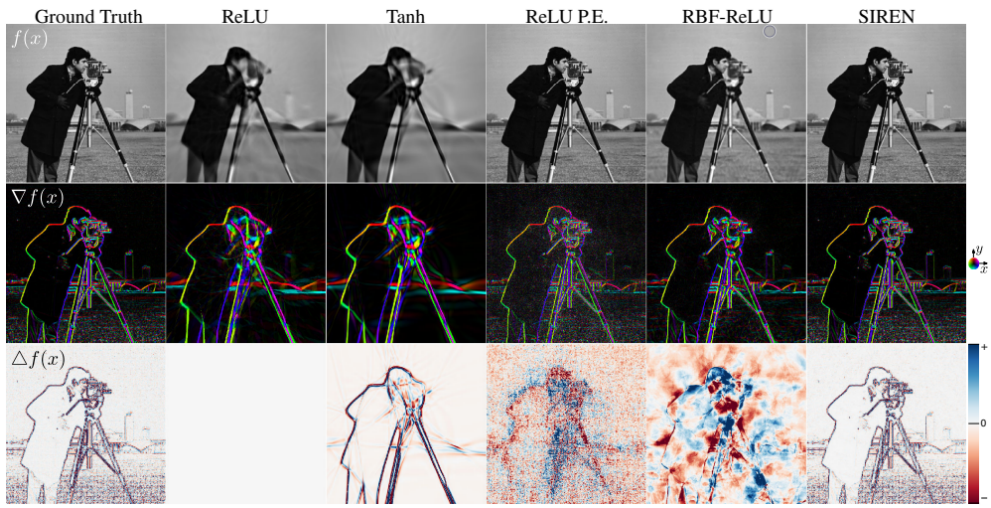


Figure 1.10 SIREN cathes the high frequency, and first and second order derivative.[35]

was used for parametrizing continuous signal, but it lacks in accurately representing the fine detail of the underlying signal[35]. For solving this issue, SIREN assumes that this problem occurs due to the second order derivative of ReLU is zero for almost everywhere. And SIREN shows that implicit neural representations using periodic activation functions(which has non vanishing higher order derivative) can catch the complicate signals and their derivatives robustly(Fig 1.10). SIREN used sine function for activation function, which

can be formulated by:

$$f(\mathbf{x}) = \mathbf{W}_n \left(f_{n-1} \circ f_{n-2} \circ \ldots \circ f_0\right)(\mathbf{x}) + \mathbf{b}_n, \quad \mathbf{x}_i \mapsto f_i(\mathbf{x}_i) = \sin\left(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i\right)$$

where $f_i : \mathbb{R}^{N_i} \mapsto \mathbb{R}^{N_{i+1}}$ is the i^{it} layer of the neural network, weight matrix $\mathbf{W}_i \in \mathbb{R}^{N_{i+1} \times N_i}$, bias $\mathbf{b}_i \in \mathbb{R}^{N_{i+1}}$ applied on the input $\mathbf{x}_i \in \mathbb{R}^{N_i}$.

For representing the high frequency in INR, different from SIREN[35], NeRF[22] proposed the positional encoding. And [38, 11] developed the idea of positional encoding, and they proposed random fourier feature($\mathbf{F}$). $\mathbf{F}$ proposed not to naively input the coordinate to the neural network, but to transform the coordinate and then input to the neural network. Given a coordinate $\mathbf{x} \in \mathbb{R}^s$, $\mathbf{F} : \mathbb{R}^s \to \mathbb{R}^{2m}$ is defined as

$$\mathbf{F}(\mathbf{x}) = \begin{pmatrix} \cos(2\pi \mathbf{B}\mathbf{x}) \\ \sin(2\pi \mathbf{B}\mathbf{x}) \end{pmatrix}$$

where $\mathbf{B} \in \mathbb{R}^{m \times s}$ is a random matrix whose entries are sampled from $\mathcal{N}\left(0, \sigma^2 I\right)$ and can be learnable. The number of frequencies m and the variance σ^2 are hyperparameters.

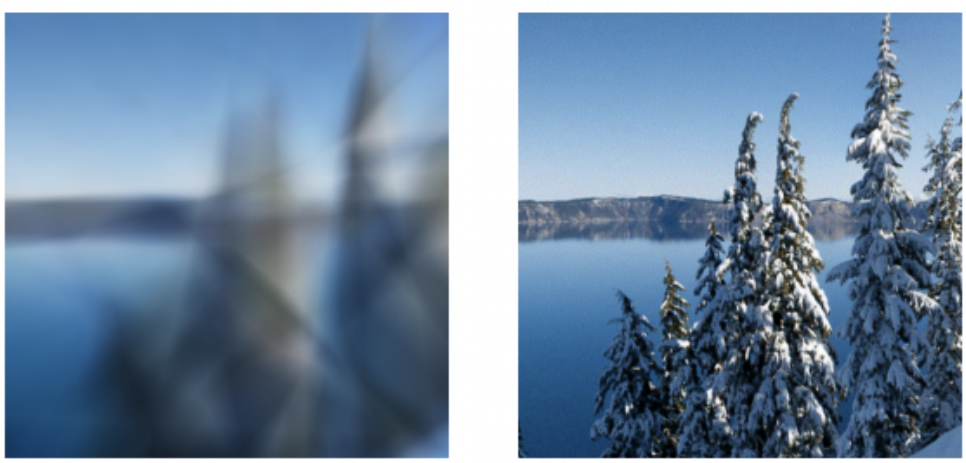


Figure 1.11 Using random fourier features(right) and not(left)[11]

[38] further researched $\mathbf{F}$ in an NTK framework, and showed that mapping input vectors to simple fourier feature before input to the neural network enables to learn a good representation of the high-frequency components, and [1] observed the relationship between SIREN and Fourier mapping before passing the MLP. Fourier mapping is structurally equal to one hidden layer SIREN.

And then, we will review at the second perspective. The purpose of INR is to obtain weights $\lambda_I \in \Lambda$ of the function representation corresponding to data I. Let $f_{\lambda_I} : \mathbb{R}^d \to \mathbb{R}^k$ be a function representation where $\mathbb{R}^d$ be a spatial coordinates space. Then "*how to construct(obtain) the weight of the function representation?*" In the context of learning function representation, there are two types: 1. embedding based method[37, 10] and 2.hypernetworks based method[11, 36, 35].

Embedding based method learns a fixed function that takes the form $f_{\lambda_I} = q(x, e(I))$ where $e(I)$ is an encoded as a conditioning vector[13]. For example, modulation[31] is one of the embedding based methods. Following [31], INR-GAN[37] proposes factorized multiplicative modulation to build continuous image GAN. Using $e(I)$ encoded with input I, INR-GAN[37] obtains a function representation corresponding to I by modulating the fixed function:

$$\mathbf{W} = \mathbf{W}_s \odot \sigma(\mathbf{W}_h)$$

where $\mathbf{W}$ is desired weight of function representation, $\mathbf{W}_s$ is weight of fixed function, $\mathbf{W}_h$ is obtained through $e(I)$ and σ is non-linear function(weights of fixed function are shared to every data).

Hypernetworks based method parametrizes a function representation using a hypernetworks[11, 35, 36, 15]. $e(I)$ is fed to the Hypernetwork h and

then h generate $\lambda_I = h(e(I))$ which is the set of parameters of the function representation(Fig 1.12). In this case, function representation takes the form $f_{\lambda_I} = g(x; \lambda_I) = g(x; h(e(I))$. For example, [11] builds generative models over implicit neural representations by adversarial approach, and they set generator using the hypernetwork approach.

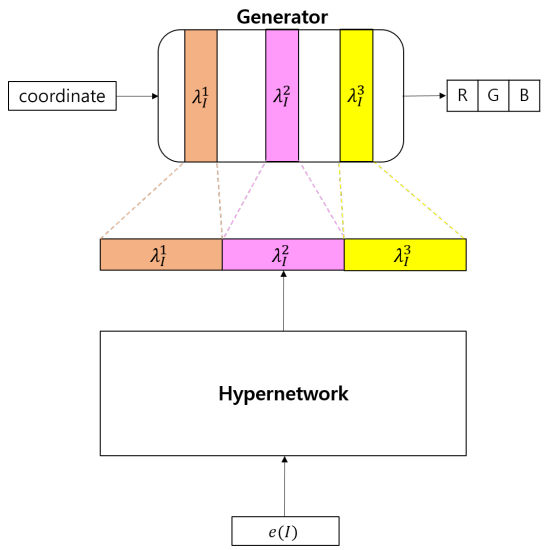


Figure 1.12 Diagram of hypernetwork

There are some studies such as [13, 21] on which is better, the embedding based method or the hypernetwork based method. However, since it is established under the limited assumptions, hence, it is necessary to study the theory established in more practical assumptions.

2 Review the Clustering

Clustering is the unsupervised task of grouping data points for the same group to be contain more similar features or properties to each other than other groups. Clustering is important task in machine learning, data analysis, statistical analysis, pattern recognition, computer vision and has a various application in practical. There are various clustering algorithms depending on the dataset and specific the goal of the clustering. In this chapter, firstly, we review some clustering algorithms which is widely used in machine learning(section 2.1) and second, we review some clustering algorithms based on deep learning(section 2.2).

2.1 Classic Algorithms

2.1.1 Centroid Based Clustering

Basic intuition of centroid based clustering is that the data point is assigned to closest center. For this reason, the result of the centroid based clustering makes the shape of the each cluster to be circular. Hence, it may be suitable for if the actual clustering results are circular shape. The most popular method of the centroid based clustering is **K-Means Clustering**. The standard k-means clustering algorithm is as below:

Algorithm 1 K-means clustering

Input: Given dataset $\{x_1, \cdots, x_N\}$, $K \coloneqq$ the number of clusters

1: Randomly choose the centers $m_1, m_2, \ldots, m_K \in$ dataset
2: **while** Assignment converges **do**
3: $\quad S_i = \left\{ x : \|x - m_i\|^2 \leq \left\|x - m_j\right\|^2 \forall j, 1 \leq j \leq K \right\}$ ▷ Assignment step
4: $\quad m_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$ ▷ Update center
5: **return** $S_1, S_2, \ldots S_K$

K-means clustering algorithm can be proved to always terminated in finite iterations. But it is not global optimum but local optimum and is not garunteed to converge same result because of the randomly initialized centers.

2.1.2 Hierarchical Clustering

We might consider that data can be described by simple partitions. For example, if data consists of dogs, cats and dolphins. we might cluster it by some hierarchies. $A : \{(\text{dogs}), (\text{cats}), (\text{dolphins})\}$, $B : \{(\text{dogs}, \text{cats}), (\text{dolphins})\}$, $C : \{(\text{dogs}, \text{cats}, \text{dolphins})\}$, where A is classified according to species, B is classified according to class, and C is classified according to kingdom. Hierarchical clustering is a method of cluster analysis that finds A, B and C in this case. Also, unlike k-means clustering, hierarchical clustering method does not need to predetermine the number of clusters. Hierarchical clustering builds a tree over the data. Individual data make up the leaves, but the root is a single cluster that consists of all of the data. Intermediate clusters exist between the root and the leaves, including subsets of the data. The primary principle behind hierarchical clustering is to build a tree by creating 'clusters of clusters' that travel upwards. To create such a tree, there are two basic methods. 1. Each datum is placed in its own singleton cluster at the bottom of the hierarchy before groupings are combined using hierarchical agglomerative clustering (HAC). 2. Divisive

clustering begins with all the data in a single, large group and slices it up until each piece of data gets its own singleton group. In this section, we only review the hierarchical agglomerative clustering(HAC) method which is the most popular algorithm in hierarchical clustering.

Algorithm 2 Hierarchical agglomerative clustering

Input: Given data $\{x_1, \cdots, x_N\}$ and groupwise distance $Dist(\mathrm{G}_1, \mathrm{G}_2)$

$\mathcal{A} \leftarrow \emptyset$ ▷ Initialize active set to empty
for $n \leftarrow 1 \cdots N$ **do** ▷ Loop over the data
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{\{x_n\}\}$ ▷ Add one data to one cluster.
$\mathcal{T} \leftarrow \mathcal{A}$ ▷ Store the tree as a sequence of merges
while $|\mathcal{A}| > 1$ **do** ▷ Loop until the active set has one item
    $G_1^\star, G_2^\star \leftarrow \underset{G_1, G_2 \in \mathcal{A}}{\operatorname{argmin}} \, Dist(G_1, G_2)$ ▷ Choose pair in $\mathcal{A}$ with best distance
    $\mathcal{A} \leftarrow \left(\mathcal{A} \setminus \{G_1^\star\}\right) \setminus \{G_2^\star\}$ ▷ Remove each from active set
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{G_1^\star, G_2^\star\}$ ▷ Add union to active set
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{G_1^\star, G_2^\star\}$ ▷ Add union to tree
return $\mathcal{T}$

Algorithm 2 displays the general algorithm for hierarchical agglomerative clustering. Active set $\mathcal{A}$ contains clusters which is merged in each stage. In the first stage, one data constitutes one cluster, and after that, clusters with the best distance are merged to one cluster. Finally, we get a tree that records this series of processes. There are various methods, such as defining the distance between cluster groups as the distance between centroids or the average of the distances between each point. Hierarchical agglometrative clustering can be visualized through a dendrogram.

2.1.3 Density Based Clustering

Since the k-means algorithm's result is depend on 'distance function'. The most common choice is euclidean distance, and in this distance metric, k-means algorithm forms only circular clusters. Hence, it may not be suitable for

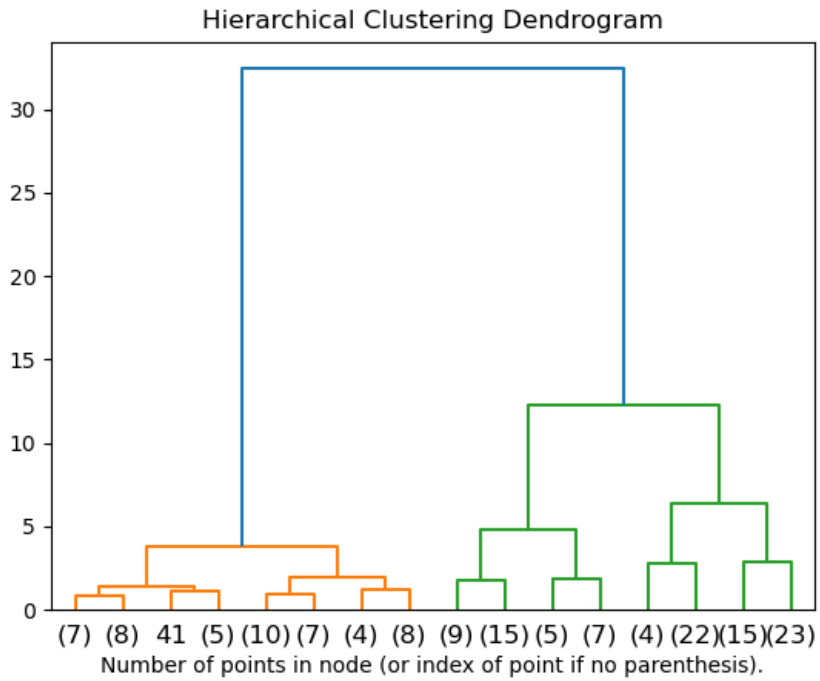


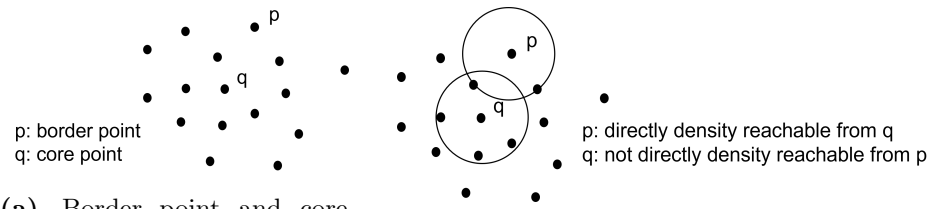
Figure 2.1 Visualization of HAC through dendrogram.

clustering non-circular shape. Also, k-means clustering is sensitive to outliers, the cluster results may become strange. Density based clustering algorithm may solve these issues. The fundamental premise of density-based clustering is that each point in the cluster must have a minimum number of points within a predetermined radius. i.e, the density must be exceeded at least a certain level. we briefly review the density-based spatial clustering of applications with noise (DBSCAN) which is the most popular algorithm in density based method.

We define some definitions for describing the algorithm.

Definition 2.1 *(ε-neighborhood of a point)* The ε-neighborhood of a point p, denoted by $N_\varepsilon(p)$, is defined by $N_\varepsilon(p) = \{q \in D \mid dist(p, q) \leq \varepsilon\}$.

Definition 2.2 *(directly density-reachable)* A point p is directly density-reachable from a point q with respect to ε, MinPts if



(a) Border point and core point

(b) Directly density reachable point

Figure 2.2 Definition 3.2

(1) $\mathrm{p} \in N_{\varepsilon}(\mathrm{q})$

(2) $\left|N_{\varepsilon}(\mathrm{q})\right| \geq$ MinPts (core point condition)

i.e, a point p is directly density reachable from a point q means that q is not a border point and contains minimum points in the ε-neighborhood.

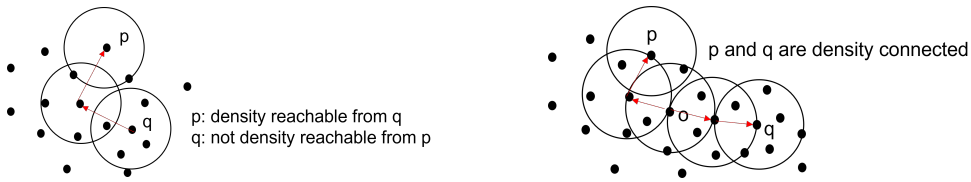
Definition 2.3 *(density-reachable)* A point p is density reachable from a point q with respect to ε and MinPts if there is a chain of points $\mathrm{p}_1, \ldots, \mathrm{p}_n, \mathrm{p}_1 = \mathrm{q}, \mathrm{p}_\mathrm{n} = \mathrm{p}$ such that $\mathrm{p}_{\mathrm{i}+1}$ is directly density-reachable from p_i.

Even if two points are in the same cluster, if both points are the border point, then they are not *density-reachable*. Therefore, a new definition which is called *density-connected* is needed for this problem.

Definition 2.4 *(density-connected)* A point p is density connected to a point q with respect to ε and MinPts if there is a point o such that both, p and q are density-reachable from o with respect to ε and MinPts.

Based on what we defined so far, finally, we can define cluster and noise.

Definition 2.5 *(cluster)* Let D be a database of points. A cluster C with respect to ε and MinPts is a non-empty subset of D satisfying the following conditions:



(a) Density reachable point **(b)** Density connected points

Figure 2.3 Definition 3.3 and 3.4

(1) $\forall$p, q : if p $\in$ C and q is density-reachable from p with respect to ε and MinPts, then q $\in$ C. (Maximality)

(2) $\forall$p, q $\in$ C : p is density-connected to q with respect to ε and MinPts. (Connectivity)

Definition 2.6 *(noise)* Let $\mathrm{C}_1, \ldots, \mathrm{C}_k$ be the clusters of the database D with respect to parameters ε_i and $\mathrm{MinPts}_i, i = 1, \ldots, k$. Then we define the noise as the set of points in the database D not belonging to any cluster C_i, i.e. noise $= \{\mathrm{p} \in D \mid \forall \mathrm{i} : \mathrm{p} \notin \mathrm{C}_i\}$.

DBSCAN requires predefined ε, minimum points(MinPts) and distance measure function(*DistFunc*). And then classifies the points as core point, border point and noise. And finally DBSCAN groups the reachable points as single one cluster. The algorithm is shown below.

Algorithm 3 DBSCAN

Input: Dataset, ε, MinPts and *DistFunc*

$C \coloneqq 0$ ▷ Label of cluster
for point P in Dataset **do**
    if label(P) $\neq$ undefined **then**
        continue
    Neighbors $N \coloneqq$ FindNeighbor(Dataset, *DistFunc*, P, ε)
    if $|N| <$ MinPts **then**
        label(P) $\coloneqq$ Noise
        continue
    $C \coloneqq C + 1$
    label(P) $\coloneqq C$
    SeedSet $\coloneqq N \setminus \{P\}$
    for point Q in S **do**
        if label(Q) = Noise **then**
            label(Q) $:==$ C
        if label(Q) $\neq$ undefined **then**
            continue
        label(Q) $\coloneqq$ C
        Neighbors N $\coloneqq$ FinedNeighbor(Dataset, *DistFunc*, Q, ε)
        if $|N| \geq$ MinPts **then**
            S $\coloneqq S \cup N$

function FINDNEIGHBOR((Dataset, *DistFunc*, Q, ε)
    Neighbors N $\coloneqq$ empty list
    for point P in dataset **do**
        if *DistFunc*(Q,P) $\leq \varepsilon$ **then**
            $N \coloneqq N \cup \{P\}$
    return N

2.2 Deep Learning based Clustering

Many fields of machine learning are being replaced by deep learning-based methods. Clustering is no exception. Recently, deep learning is showing superior performance in image clustering. SOTA on the benchmark datasets are also established by deep learning-based methods. In this chapter, we review the two

types of the clustering methods which we called 1.*two-stage learning method* and 2.*end-to-end learning method*.

The first one, *two-stage learning method*(such as DEC[41],DAC[6],Deep Cluster[5]) obtains semantic features of the unlabeled dataset through pretext tasks on the first stage. The first stage can be regarded as self-supervised learning, and there are various tasks for pretext task. For example, predicting the patch context [9, 25], inpainting patches[30], solving jigsaw puzzles [26, 27], colorizing images [45, 20], predicting noise[4], predicting rotations[14], spotting artifacts[17], generating images[32], and so on. And then, in the second stage, fine tunning step is performed to solve clustering task.

The second one, end-to-end learning method(such as IIC[18], IMSAT[16], SeLa[42]) simultaneously learns both the feature representation and clustering assignment without explicitly optimizing the clustering task. However, this method is prone to result degenerate solution which is predicting the all data into one cluster[18]. To prevent this problem, many literatures maximizes the mutual information between the class assignments of the paired data to learn similar representations, and then establish the high accuracy of clustering. However, there is a question that whether these learned representations are really meaningful and similar between positive pairs and different from negative pairs. Hence other methods have been introduced to prevent this problem.

Also very recently, contrastive learning has achieved good performance in clustering task. The basic concept of contrastive learning is to map feature vectors of positive pair data close and map feature vectors of negative pair data far away. There are clustering methods based on contrastive learning([46, 7, 34, 8, 39, 43]). For example, [39] proposed two-stage clustering method with

contrastive pre-training and then fine-tuning for clustering task. And [43] proposed to use infoNCE loss[29] with clustering by end to end manner.

The most representative of each method will be reviewed in the upcoming sections.

2.2.1 Two-Stage Learning Method

DEC(Deep Embedded Clustering)[41] is one of the first work to apply deep learning to clustering. Instead of directly clustering on the data space, DEC maps data through nonlinear mapping $f_\theta : X \to Z$, where θ are learnable parameters and Z is the latent space. DEC has a two stages:

1. feature space is obtained by the encoder of the autoencoder, (i.e, encoder is f_θ).
2. simultaneously optimizing for cluster assignment and θ for optimizing underlying feature space.

However, the second stage is challenging, because optimizing cluster assignment needs true label of the data. However, in unsupervised setting, the label is unknown. Hence, DEC proposed to optimizing clusters with an auxilary target distribution which is obtained by the current soft cluster assignments. Definitions are as follows:

Definition 2.7 *(Soft assignment)*

$$q_{ij} = \frac{\left(1 + \left\|z_i - \mu_j\right\|^2 / \alpha\right)^{-\frac{\alpha+1}{2}}}{\sum_{j'} \left(1 + \left\|z_i - \mu_{j'}\right\|^2 / \alpha\right)^{-\frac{\alpha+1}{2}}}$$

where $z_i = f_\theta(x_i) \in Z$ corresponds to $x_i \in X$ after embedding, α are the degrees of freedom of the Student's t distribution and q_{ij} can be interpreted

as the probability of assigning sample i to cluster j (i.e., a soft assignment). The authors set $\alpha = 1$ for all experiments.

Definition 2.8 *(Auxiliary target distribution)*

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j'} q_{ij'}^2 / f_{j'}}$$

where $f_j = \sum_i q_{ij}$

DEC updates the feature representation (i.e, θ) and cluster centroid for every iteration by minimizing KL-divergence between auxiliary target distribution p_i and soft assignments q_i. Hence the loss function is $L = \mathrm{KL}(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$. Overall algorithm for DEC is as follows:

Algorithm 4 DEC

1: pre-training the f_θ by autoencoder
Input: f_θ, initial centroid $\mu_j, (j = 1, \ldots, k)$
2: **while** Cluster assignments converge **do**
3: calculate soft assignment Q and auxiliary target distribution P
4: updates θ and μ_j by the gradient of the loss ▷ loss is KL-divergence between P and Q.

Another two-stage learning approch is SCAN(Semantic Clustering by Adopting Nearest neighbors)[40]. In a first stage, SCAN pretrains neural network Φ_θ to learn a feature representations through a pretext task τ. However, the feature representation of data belonging to the same cluster may not be invariant. To solve this problem, SCAN includes minimization of the distance between data X_i and their augmentations $T[X_i]$ in pretext task τ, which can be expressed as:

$$\min_\theta d\left(\Phi_\theta\left(X_i\right), \Phi_\theta\left(T\left[X_i\right]\right)\right)$$

And naively applying K-means clustering on the obtained features can lead to degenerate solution which means all data belongs to one cluster. To overcome

this problem, SCAN proposed to mining nearest neighbors. i.e, for every sample X_i, SCAN mines its K nearest neighbors in the feature space Φ_θ. SCAN defines the set $\mathcal{N}_{X_i}$ as the neighborhood of X_i in the dataset.

In a second stage, SCAN classifies each data and its mined neighbors together by using a following loss function:

$$\Lambda = -\frac{1}{|\mathcal{D}|}\sum_{X\in\mathcal{D}}\sum_{k\in\mathcal{N}_X}\log\left\langle\Phi_\eta(X),\Phi_\eta(k)\right\rangle + \lambda\sum_{c\in\mathcal{C}}\Phi_\eta'^c\log\Phi_\eta'^c,$$
$$\text{with } \Phi_\eta'^c = \frac{1}{|\mathcal{D}|}\sum_{X\in\mathcal{D}}\Phi_\eta^c(X)$$

where Φ_η is clustering function parameterized by η and final layer of Φ_η is softmax function to obtain probability of assignment over the each cluster label $\mathcal{C} = \{1,\ldots,C\}$,with $\Phi_\eta(X_i) \in [0,1]^C$, $\mathcal{N}_{X_i}$ is mined neighborhood of X_i. The probability of X_i being assigned to cluster c is denoted by ${\Phi_\eta}^c(X_i)$. Algorithm is as below:

Algorithm 5 SCAN

Input: Dataset $\mathcal{D}$, Clusters $\mathcal{C}$, Task τ, Φ_θ, Φ_η, Neighbors $\mathcal{N}_\mathcal{D} = \{\}$.

```
Pretraining Φ_θ with task τ.
for X_i ∈ D do
    N_D ← N_D ∪ N_{X_i}, with N_{X_i} = K neighboring samples of Φ_θ(X_i)
while SCAN Loss converges do
    Update Φ_η with SCAN-loss, i.e. Λ(Φ_η(D), N_D, C)
while Len(Y) increases do
    Y ← (Φ_η(D) > threshold )
    Update Φ_η with cross-entropy loss, i.e. H(Φ_η(D), Y)
return Φ_η(D)
```

2.2.2 End-to-End Learning Method

Invariant information clustering(IIC)[18] is one of the end-to-end learning method. Let $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ be a paired data. For example, $\mathbf{x}$ and $\mathbf{x}'$ could be different images but containing same semantic information. The goal of IIC(Information

Invariant Clustering) is to learn $\Phi : \mathcal{X} \to \mathcal{Y} = [0,1]^k$, where $\Phi(\mathbf{x})$ can be interpreted as the distribution of a discrete random variable z over k classes, formally given by $P(z = c|\mathbf{x}) = \Phi_c(\mathbf{x})$.($z$ can be considered as assignment random variable) Φ will learn what is in common information between $\mathbf{x}$ and $\mathbf{x}'$ while removing instance level details, and induces high probability to be assigned to the same cluster label between $\mathbf{x}$ and $\mathbf{x}'$. It achieved by maximizing the mutual information between z and z' where z, z' are assignment random variable. However, in unsupervised setting, we don't know what $\mathbf{x}$ and $\mathbf{x}'$ are paried. Therefore, $\mathbf{x}'$ can be generated by augmentation function g, i.e, $\mathbf{x}' = g(\mathbf{x})$. Hence, The goal of IIC is maximizing the mutual information:

$$\max_{\Phi} MI\left(z, z'\right) = \max_{\Phi} \sum_{c=1}^{k} \sum_{c'=1}^{k} \mathbf{P}_{cc'} \cdot \ln \frac{\mathbf{P}_{cc'}}{\mathbf{P}_c \cdot \mathbf{P}_{c'}},$$

The authors assume that $P(z = c, z = c'|\mathbf{x}, \mathbf{x}') = P(z = c|\mathbf{x}) \cdot P(c'|\mathbf{x}')$. This means that z and z' are independent when specific $\mathbf{x}$ and $\mathbf{x}'$ are given. Then the joint probability over z and z' is calculated by the output of the neural network Φ:

$$\begin{aligned} P(z = c, z' = c') =& \sum_{\mathbf{x},\mathbf{x}'} P(z = c, z' = c'|\mathbf{x}, \mathbf{x}') \cdot P(\mathbf{x}, \mathbf{x}') \\ =& \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{x}'} \left[P(z = c, z' = c'|\mathbf{x}, \mathbf{x}')\right] \\ \approx& \frac{1}{n} \sum_{i=1}^{n} P(z = c, z' = c'|\mathbf{x}_i, \mathbf{x'}_i) \end{aligned}$$

Then the joint probability distribution over z and z' can be expressed as matrix $\mathbf{P}$ where each element at cth row and c'th column denoted as $\mathbf{P}_{cc'} = P(z = c, z' = c')$:

$$\mathbf{P} = \frac{1}{n} \sum_{i=1}^{n} \Phi\left(\mathbf{x}_i\right) \cdot \Phi\left(\mathbf{x}'_i\right)^{\top}.$$

And the lastly, $P(z = c)(P(z' = c'))$ are obtained by summing the row(column) of the $\mathbf{P}$.

Since $MI(z, z') = H(z) - H(z|z')$, maximizing mutual information has a trade-off between maximizing individual cluster assignments entropy $H(z)$ and minimizing the conditional cluster assignment entropy $H(z|z')$. The first one $H(z)$ is maximzed when all cluster assignment probability be same which prevents a degenerate solution. However, if $H(z)$ is fully maximzed, then $P(z)$ will be uniform distribution which means no clustering. The authors claims that the second term $H(z|z')$ can prevents this problem, and it is minimized when the cluster assignments are exactly predictable from each other.

3 chapter3

3.1 Related Works

Our works are related to transformation invariant feature extraction methods[14, 23, 12, 19, 28, 33, 44, 26] and deep learning based clustering methods[24, 41, 18, 40]. Transformation invariant feature extractions are mainly studied on self-supervised learning for pretext tasks for learning semantic informations of the data. Especially, [23, 12] proposed to learning rotation invariant feature for pretext task. However, these studies have limitations in that the rotation degrees are finite($0, \frac{\pi}{2}, \pi, \frac{3}{2}\pi$), and only show that there is a performance gain in the downstream task, but do not show how rotation invariant features are learned in the pretext task stage. Also, there are methods that the ultimate goal is to learn rotation invariant features and rotation degrees are infinite(i.e, $0 \sim 2\pi$) [2, 3]. Spatial-VAE[2] disentangles image rotation from other unstructured latent factors in a variational autoencoder (VAE) framework. Spatial-VAE formulates the generative model as a function of the spatial coordinate(i.e, implicit neural representations perspective). By leveraging this perspective, predicted rotation degree which is obtained by front part of output of an encoder rotates input spatial coordinates, Spatial-VAE makes the reconstruction error be able to differentiate with respect to predicted rotation degree. Also, by minimizing the KL-divergence term so that the distribution of inferenced

rotation degree by the front part of the encoder output becomes the prior distribution of rotation degree, the rear part of the encoder becomes a rotation invariant feature. However, Spatial-VAE assumed that the rotation degrees of the dataset are sampled from gaussian distribution such as $\mathcal{N}\left(0, \frac{\pi^2}{4^2}\right)$ instead of uniform distribution over $[0, 2\pi]$. This makes that many data are rotated by close to zero degree. And, although they measured the correlation between the latent variable inferred by the encoder and the ground truth rotation degree for the performance of learning rotation invariant feature, but NOT directly verified how the rotation invariant features were learned well. It is different from that our work learned rotation invariant features and showed good performance in a downstream task(i.e, clustering).

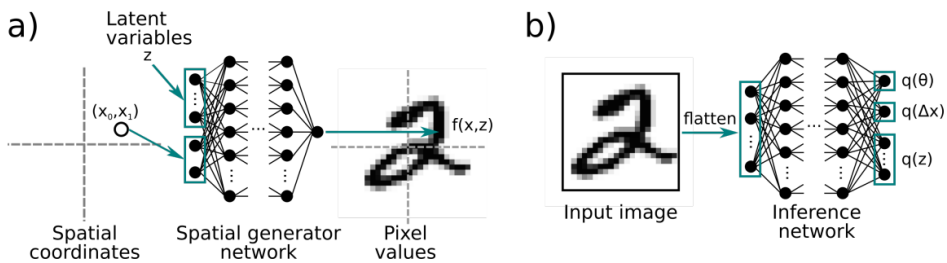


Figure 3.1 Diagram of Spatial-VAE. **a)** Modelling generative model as mapping coordinates and latent variables to the pixel intensity(RGB) at that coordinate. **b)** Inference network(encoder) inferences the rotation and unstructured latent variables.

3.2 Aim of The Proposed Method

Most of the literature focuses on achieving high performance on benchmark datasets. However, in the case of image domain, datasets such as MNIST, FashionMNIST, CIFAR10, STL10, ImageNet and etc contain human's prior knowledge. For example, the number digits in MNIST are not rotated and are

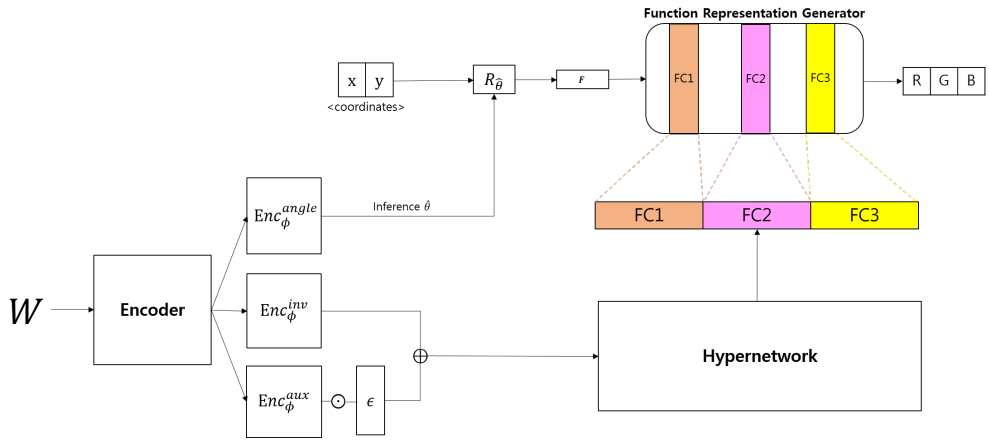


Figure 3.2 Diagram of the propsed method.

placed at the 'zero degree' which is considered as a natrual and normal image to human sense. However, in practically, dataset of the number digits may be randomly rotated by 0 to 2π degree. In generally, there are no reason to neural network to recognize 2 and rotated 2 are same digit. However, in many situations, we want to our neural network recognize these images are same. To make this possible, the feature output by the neural network should be equal to 2 and rotated 2. That is, the neural network needs to extract feature of the data in rotation invariant manner. This problem can be formally defined in the more general context not only rotation. Let $\mathcal{X}$ is a dataset, and G is a Group, and $\alpha : G \times \mathcal{X} \to \mathcal{X}$ a group action and $f : \mathcal{X} \to \mathcal{Z}$ is a neural network. Let $\mathbf{x} \in \mathcal{X}$, then we want to train the neural network satisfying:

$$f(\mathbf{x}) = f(\alpha(g, \mathbf{x})) = f(g\mathbf{x})), \text{ for all } g \in G. \tag{3.1}$$

Also, we define 'f **is invariant under action of** G' and 'f **is** G **invariant**' if f satisfies equation 3.1.

In this paper, when $G = SO(2)$, i.e, G is rotation group, our goal is to train the neural network be rotation invariant, and cluster the randomly rotated images by its rotation invariant feature. It can be established by two stage:

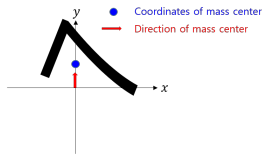
1. Training the neural network to explicitly extract rotationally invariant latent vector(we call it *latent*) and latent rotation angle(we call it *latent angle*). And by formulating the image as INR, reconstruction loss can be differentialbe with respect to *latent angle*, and it is the easy way to obtain good *latent angle* and *latent*. In addition, since INR catch the high frequency of the image, our method are robustly worked on semiconductor dataset(non-INR methods failed to reconstruct the semiconductor).
2. Using the (pre-trained) encoder and *latent*, we cluster the randomly rotated images by DEC[41] manner, i.e, rotationally invariant clustering.

Our contributions can be summuraized as:

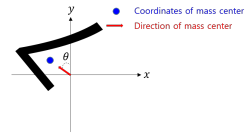
- We propose novel method to learn the rotaion invariant feature by leveraging implicit neural representations. We explicitly disentangle the latent rotation angle(*latent angle*) and rotationally invariant latent vector(*latent*) from the output of the encoder.
- We show that our method is high performance on rotationally invariant clustering task than other methods. To the best of our knowledge, it is first attempt to cluster with implicit neural representations.

3.3 Breaking the Symmetry

When we inference the rotation angle θ of the image, we need to know what is the unrotated image, which is called by *reference image*. Until the *reference image* is given, θ is not well defined due to the symmetry. We tried to find



(a) Reference image $\mathbf{X} \in \mathcal{X}$ whose direction of the center mass is parallel to the positive y direction.



(b) Our actual training data image $\mathbf{W} \in \mathcal{W}$ where θ is rotation degree from the reference image $\mathbf{X}$.

Figure 3.3 Reference image and our actual training data image. Red arrow is direction of the center mass.

a reference image by the neural network itself without any supervision, but this problem may too hard to neural network, and hence it failed. Hence we propose the rule to define *reference image* for breaking the symmetry. For breaking the symmetry, we define the *reference image* is to be the image whose the center of mass is parallel to positive y-axis direction. Note that this rule works even in unsupervised setting. Let's denote $\{\mathbf{X}_i\}_{i=1}^N \in \mathcal{X}$ is *reference image*. Now our dataset can be considered as a set of images rotated from the reference dataset, denoted by $\{\mathbf{W}_i\}_{i=1}^N \in \mathcal{W}$. Note that $\mathbf{W}_i = \mathcal{R}_{\theta_i}\mathbf{X}_i$, where $\mathcal{R}$ is rotation operator and θ_i is *rotation angle*.

3.4 Encoder

Let $\mathbf{Enc}_\phi$ be an encoder network where ϕ is learnable parameters. Our goal is to obtain a latent vector of $\mathbf{W} \in \mathcal{W}$ from the encoder, where *rotation angle* of the image and rotationally invariant latent vector(*latent*) are disentangled. We denote that former is $\mathbf{Enc}_\phi^{angle}(\mathbf{W})$, and later is $\mathbf{Enc}_\phi^{inv}(\mathbf{W})$. There are some details about the encoder.

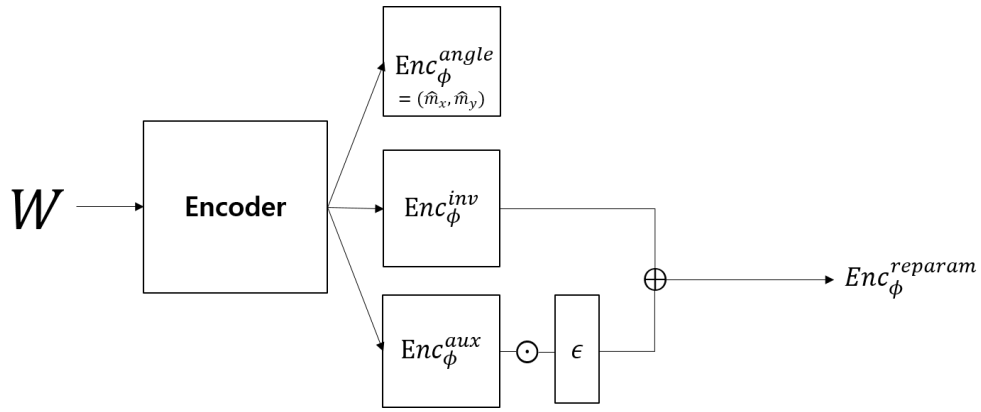


Figure 3.4 Encoder.

First, in *rotation angle*. We found that it is not appropriate to naively representing *rotation angle* as $\theta \in \mathbb{R}^1$. Instead, we represent *rotation angle* as coordinates of center of mass $\mathbf{m_{center}}(\mathbf{W}) = \mathbf{m_{center}}(\mathcal{R}_\theta \mathbf{X}) = (m_x, m_y)$. Using inner product, it can be easily shown that there is one-to-one correspondence between rotation degree and the center of mass.

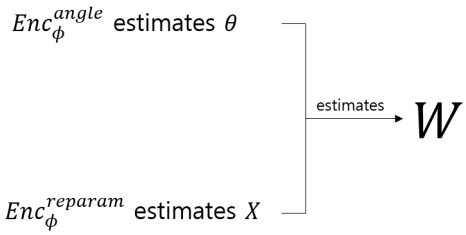


Figure 3.5 Encoder estimates *latent angle* and *latent*.

Second, We found that reparametrizing $\mathbf{Enc}_\phi^{inv}$ before input to hypernetwork performs better. So encoder not only output $\mathbf{Enc}_\phi^{inv}$, but also output auxilary latent vector $\mathbf{Enc}_\phi^{aux}$. Proposed reparametrization is simple as following:

$$\mathbf{Enc}_\phi^{\text{reparam}}(\mathbf{W}) = \mathbf{Enc}_\phi^{inv}(\mathbf{W}) + \varepsilon \odot \mathbf{Enc}_\phi^{aux}(\mathbf{W})$$

where $\varepsilon \sim \mathcal{N}(0, I)$. This reparametrization makes latent vectors around near the $\mathbf{Enc}_{\phi}^{inv}(\mathbf{W})$ to have the similar feature as $\mathbf{W}$.

Hence, our encoder output three latent vectors as following:

$$\mathbf{Enc}_{\phi}(\mathbf{W}) = \left(\mathbf{Enc}_{\phi}^{angle}(\mathbf{W}), \mathbf{Enc}_{\phi}^{inv}(\mathbf{W}), \mathbf{Enc}_{\phi}^{aux}(\mathbf{W})\right)$$

where $\mathbf{Enc}_{\phi}^{angle} : \mathbb{R}^r \to \mathbb{R}^2, \mathbf{Enc}_{\phi}^{inv} : \mathbb{R}^r \to \mathbb{R}^d, \mathbf{Enc}_{\phi}^{aux} : \mathbb{R}^r \to \mathbb{R}^d$, and r is resolution of the image, d is dimension of the *latent*.

3.5 Hypernetwork

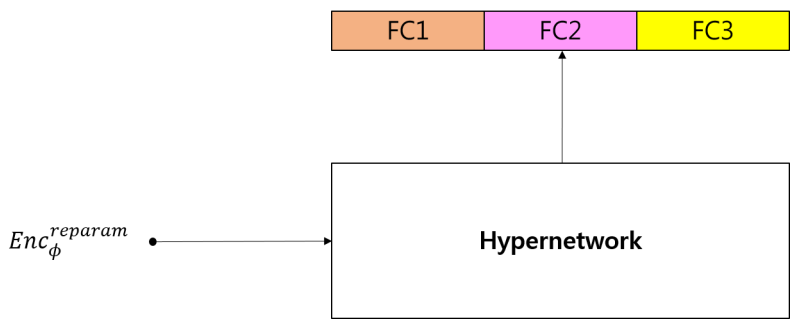


Figure 3.6 Hyper Network

Let $\mathbf{H}_{\psi}$ be a hypernetwork where ψ is learnable parameters. $\mathbf{H}_{\psi}$ receives $\mathbf{Enc}_{\phi}^{\text{reparam}}$ as an input. By doing this, we induce points near $\mathbf{Enc}_{\phi}^{\text{inv}}$ to ouptut the same function representation. Therefore, it helps to form a latent space be a cluster friendly. We experimentally confirmed that this reparametrization process improves clustering performance.

And $\mathbf{H}_{\psi}$ outputs the parameter of the function representation generator $\mathbf{G}$ that corresponding to implicit neural representations of the input image $\mathbf{W}$.

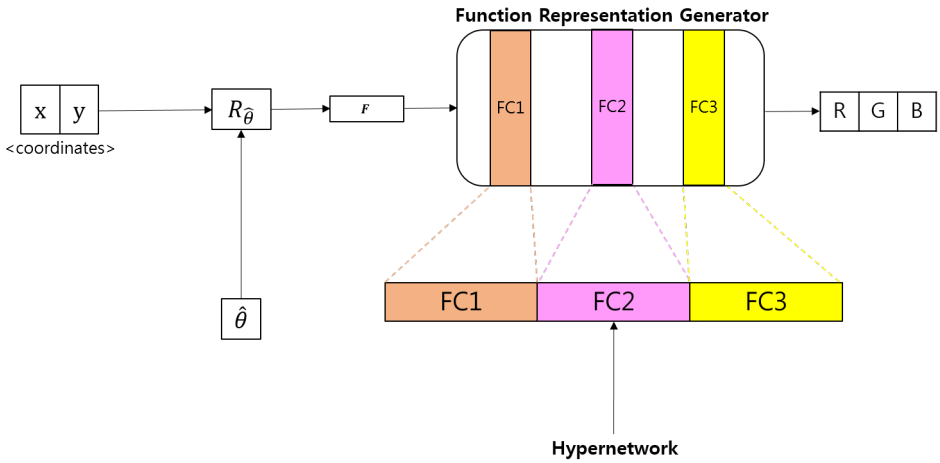


Figure 3.7 Function Representation Generator.

3.6 Function Representation Generator

Naive function representation generator $\mathbf{G}^{naive}$ recieves parameters of function representation and transformed coordinates as inputs, and then outputs RGB value. Hence, $\mathbf{G}^{naive}$ is function such that:

$$\mathbf{G}^{naive} : \Omega \times \mathbb{R}^s \longrightarrow \mathbb{R}^k$$

where Ω is space of function representations, s is dimension of spatial coordinates(e.g, $s = 2$ for 2-dimensional image), and k is dimension of feature value(e.g, $k = 3$ for RGB value). hence in our case,

$$\mathbf{G}^{naive}\left(\mathbf{H}_{\psi}(\mathbf{Enc}_{\phi}^{reparam}), (x, y)\right) = (R, G, B)$$

Note that there are no learnable parameter in $\mathbf{G}^{naive}$. However, we suggest different form of function representation generator. First, pixel coordinates are rotated by $\hat{\theta}$ which is obtained from *rotation angle* $\mathbf{Enc}_{\phi}^{angle}$. Second, following [11], coordinates are transformed by random fourier features(which is called

by $\mathbf{F}$). Random fourier feature $\mathbf{F} : \mathbb{R}^s \to \mathbb{R}^{2m}$ is defined as

$$\mathbf{F}(\mathbf{x}) = \begin{pmatrix} \cos(2\pi \mathbf{B}\mathbf{x}) \\ \sin(2\pi \mathbf{B}\mathbf{x}) \end{pmatrix}$$

where $\mathbf{x} \in \mathbb{R}^s$ is spatial coordinate, $\mathbf{B} \in \mathbb{R}^{m \times s}$ is a (potentially learnable) random matrix whose entries are sampled from $\mathcal{N}\left(0, \sigma^2\right)$. The number of frequencies m and the variance σ^2 are hyperparameters. Therefore, our proposed function representation generator $\mathbf{G}$ is as following:

$$\mathbf{G}\left(\mathbf{H}_\psi(\mathbf{Enc}_\phi^{reparam}), \mathbf{F}(\mathcal{R}_{\hat{\theta}}(x, y))\right) = (R, G, B)$$

3.7 Objective Function

Objective function is consisted of three terms:

$$\mathcal{L} = \lambda_{\text{angle}} \cdot \mathcal{L}_{\text{angle}} + \lambda_{\text{consis}} \cdot \mathcal{L}_{\text{consis}} + \lambda_{\text{recon}} \cdot \mathcal{L}_{\text{recon}}$$

Angle Loss $\mathcal{L}_{\text{angle}}$

In section 3.3, we break the symmetry problem by setting the rule so that the rotation angle θ becomes well defined. Our proposed reference image is that the center of mass direction is parallel to the positive y-axis. This rule works well even in unsupervised setting. Assume that $\mathbf{W} \in \mathcal{W}$ is our training data, hence, there exists the reference image $\mathbf{X}$ and rotation angle θ such that $\mathbf{W} = \mathcal{R}_\theta \mathbf{X}$. Our goal is to obtain a θ from the input image. We found that minimize euclidean distance between rotation angle and predicted rotation angle, (i.e, θ and $\hat{\theta}$) does not work well. Instead, we minimize euclidean distance between $\hat{\mathbf{m}}_{\mathbf{center}}(\mathbf{W})(= \mathbf{Enc}_\phi^{angle}(\mathbf{W}))$ and the real center of mass $\mathbf{m}_{\mathbf{center}}(\mathbf{W})$.

Proposed angle loss $\mathcal{L}_{\text{angle}}$ is as following:

$$\begin{aligned}\mathcal{L}_{\text{angle}} &= \mathop{\mathbb{E}}_{\mathbf{X}} \mathop{\mathbb{E}}_{\substack{\theta \\ \sim[0,2\pi)}} \left[\left\| \mathbf{m_{center}}(\mathcal{R}_\theta \mathbf{X}) - \mathbf{Enc}_\phi^{angle}(\mathcal{R}_\theta \mathbf{X}) \right\|^2 \right] \\ &= \mathop{\mathbb{E}}_{\mathbf{W}} \left[\left\| \mathbf{m_{center}}(\mathbf{W}) - \mathbf{Enc}_\phi^{angle}(\mathbf{W}) \right\|^2 \right]\end{aligned}$$

We follow the empirical risk minimization. So empirical angle loss $\hat{\mathcal{L}}_{\text{angle}}$ is as following:

$$\hat{\mathcal{L}}_{\text{angle}} = \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{m_{center}}(\mathbf{W}_i) - \mathbf{Enc}_\phi^{angle}(\mathbf{W}_i) \right\|^2$$

where N is the number of samples.

Consistency Loss $\mathcal{L}_{\text{consis}}$

One of our goal is $\mathbf{Enc}_\phi^{inv}$ to be rotationally invariant, i.e, for $\mathbf{W} \in \mathcal{W}$, and for all $\theta' \in [0, 2\pi)$,

$$\mathbf{Enc}_\phi^{inv}(\mathbf{W}) = \mathbf{Enc}_\phi^{inv}(\mathcal{R}_{\theta'} \mathbf{W})$$

To accomplish this property, we propose the consistency loss $\mathcal{L}_{\text{consis}}$ as following:

$$\begin{aligned}\mathcal{L}_{\text{consis}} &= \mathop{\mathbb{E}}_{\mathbf{X},\theta} \mathop{\mathbb{E}}_{\substack{\theta' \\ \sim[0,2\pi)}} \left[\left\| \mathbf{Enc}_\phi^{inv}(\mathcal{R}_\theta \mathbf{X}) - \mathbf{Enc}_\phi^{inv}(\mathcal{R}_{\theta'} \mathbf{X}) \right\|^2 \right] \\ &= \mathop{\mathbb{E}}_{\mathbf{W}} \mathop{\mathbb{E}}_{\substack{\theta' \\ \sim[0,2\pi)}} \left[\left\| \mathbf{Enc}_\phi^{inv}(\mathbf{W}) - \mathbf{Enc}_\phi^{inv}(\mathcal{R}_{\theta'} \mathbf{W}) \right\|^2 \right]\end{aligned}$$

Since θ, θ' are uniform, the second equality holds. Next, we follow the empirical risk minimization. So empirical consistency loss $\hat{\mathcal{L}}_{\text{inv}}$ is as following:

$$\hat{\mathcal{L}}_{\text{consis}} = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \left\| \mathbf{Enc}_\phi^{inv}(\mathbf{W}_i) - \mathbf{Enc}_\phi^{inv}(\mathcal{R}_{\theta_{ij}{}'} \mathbf{W}_i) \right\|^2$$

where $\theta_{ij}{}'$ are randomly sampled from $U[0, 2\pi]$ and M is the number of sampling.

Reconstruction Loss $\mathcal{L}_{\text{recon}}$

Minimizing reconstruction loss makes our latent space to be more meaningful. We propose reconstruction loss $\mathcal{L}_{\text{recon}}$ as following:

$$\mathcal{L}_{\text{recon}} = \frac{1}{Res} \sum_{r=1}^{Res} \mathop{\mathbb{E}}_{\mathbf{W}} \left[\left\| (\mathbf{W})_r - \mathbf{G}\left(\mathbf{H}_\psi(\mathbf{Enc}_\phi^{reparam}(\mathbf{W})), \mathbf{F}(\mathcal{R}_{\hat{\theta}}(x_r, y_r)) \right) \right\|^2 \right]$$

where $(\mathbf{W})_r$ denotes feature value at rth coordinates, (x_r, y_r) denotes rth coordinates, and Res denotes resolution of the training image. We follow the empirical risk minimization. So empirical reconstruction loss $\hat{\mathcal{L}}_{\text{recon}}$ is as following:

$$\hat{\mathcal{L}}_{\text{recon}} = \frac{1}{Res \cdot N} \sum_{r=1}^{Res} \sum_{i=1}^{N} \left\| (\mathbf{W}_i)_r - \mathbf{G}\left(\mathbf{H}_\psi(\mathbf{Enc}_\phi^{reparam}(\mathbf{W}_i)), \mathbf{F}(\mathcal{R}_{\hat{\theta}}(x_r, y_r)) \right) \right\|^2$$

where N is the number of samples.

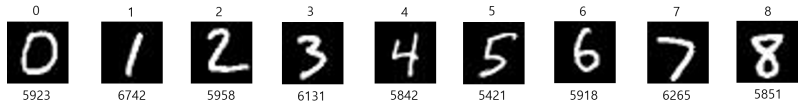
4 Experiments

Dataset

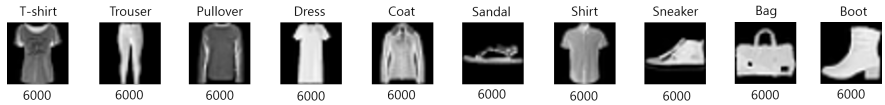
We use randomly rotated MNIST, randomly rotated FashionMNIST(rotate θ degree for all images, where $\theta \sim \text{Uniform}[0, 2\pi]$ and WM811k. MNIST is a dataset of handwritten digits with 10 labels, since rotated 6 and rotated 9 should be treated same, we removed both 9 from the training and testing phases. The reason why 6 and 9 are not merged as same label is to balance the number of each labels. FashionMNIST is a dataset of article images with 10 labels and WM811k is a dataset of wafer maps of semiconductor with 9 labels. Since original WM811k is very unbalenced between the number of each labels, we adjusted the number between labels (data augmentation was not applied because it could result in changing the label of the wafer map). In particular, in the case of WM811k, since each data is already randomly rotated, there is no need to create a rotated dataset separately. WM811k is a dataset that shows rotationally invariant clustering is important task in practically.

Architecture

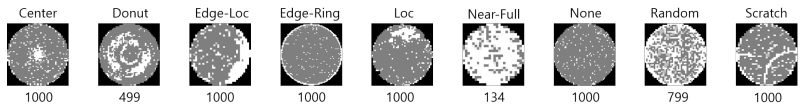
We use only MLP architecture for encoder, hypernetwork and function representation generator. Encoder is 5-layer MLP such that 32(input dim)-128-128-128-66(2+32+32). Hypernetwork is 3-layer MLP such that 32(latent dim)-128-



(a) MNIST(9 is removed)



(b) FashionMNIST



(c) WM811k

Figure 4.1 Sample of the training dataset. The number below the image indicates the number of that label.

128-256-66049. Function representation generator is 3-layer MLP such that 256-128-128-1(grey scale).

Hyperparameter

We set $\lambda_{\text{angle}} = 15, \lambda_{\text{consis}} = 1, \lambda_{\text{recon}} = 1$. Random matrix $\mathbf{B}$ of fourier random feature $\mathbf{F}$ is 128 by 2 matrix. The number of samples in consistency loss is $M = 3$. Learning rate is 10^{-4} and use 10^{-5} weight decay for all parameters.

Evaluation of output images

We visualize input images from randomly rotated training dataset, and output images obtained by input coordinates are rotated by $\hat{\theta}$ (Figure 4.2). These results are shown in Figure 4.4, 4.7 and 4.10.

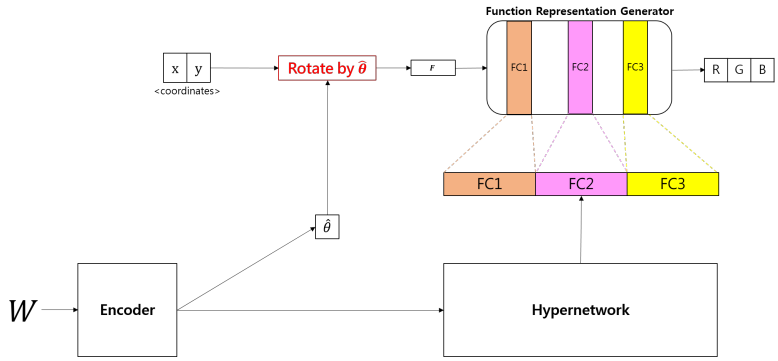


Figure 4.2 Input coordinates are rotated by $\hat{\theta}$.

In addition, we generate new images by rotating $0 \sim 2\pi$ degree from fixed image. And then input to the neural network while input coordinates are NOT rotated (Figure 4.3). These results are shown in Figure 4.5, 4.9, 4.8, 4.9 and 4.11. The reconstructed images are consistently almost same, i.e, it means that $\mathbf{Enc}_{\phi}^{inv}$ is rotationally invariant.

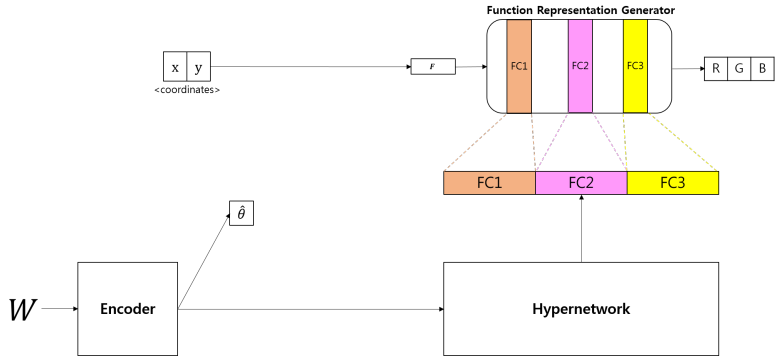
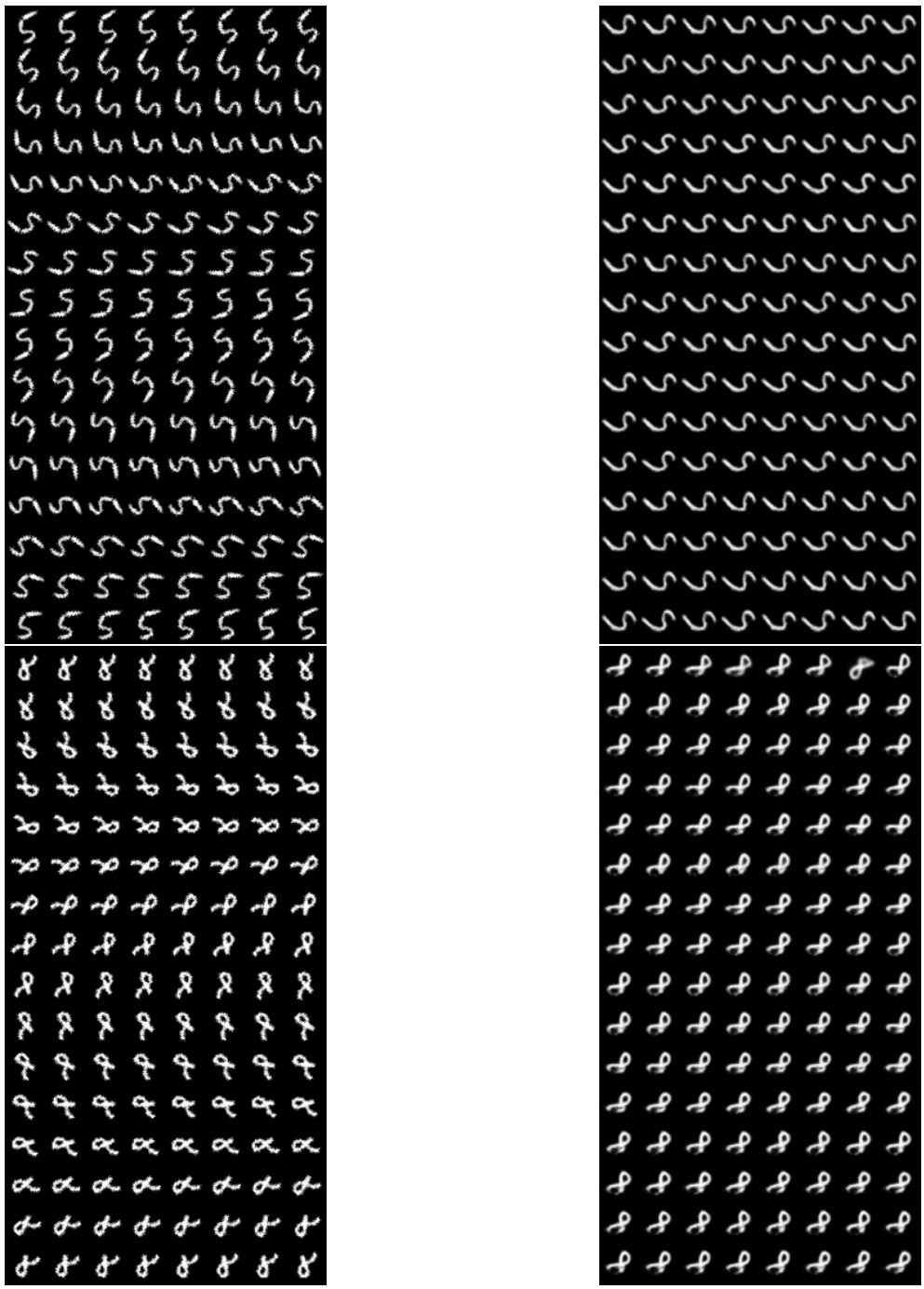


Figure 4.3 Input coordinates are NOT rotated.

(a) Input images

(b) Reconstructed images

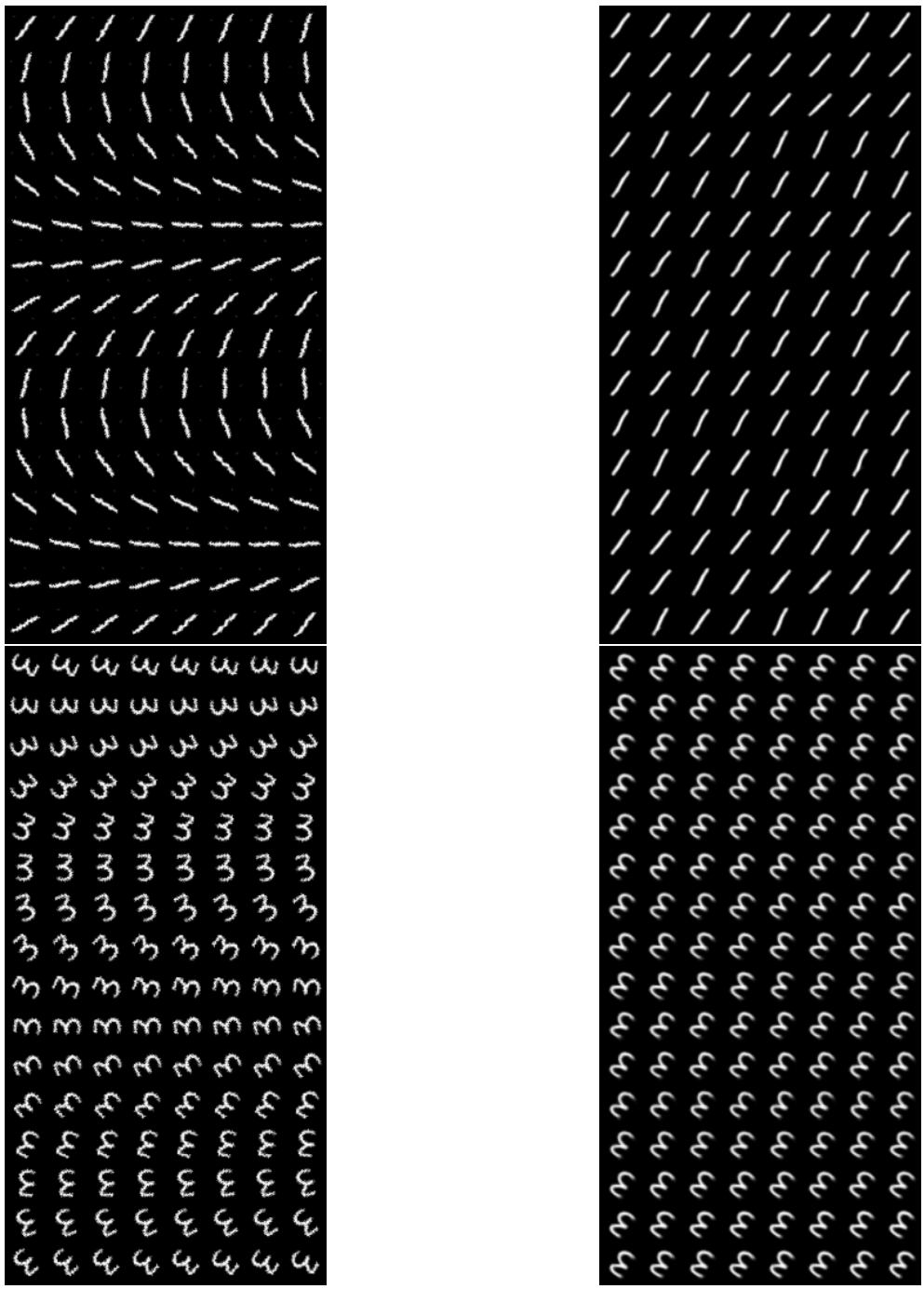
Figure 4.4 Reconstruction - MNIST. Input coordinates are rotated by $\hat{\theta}$.



(a) Rotated images

(b) Reconstructed images

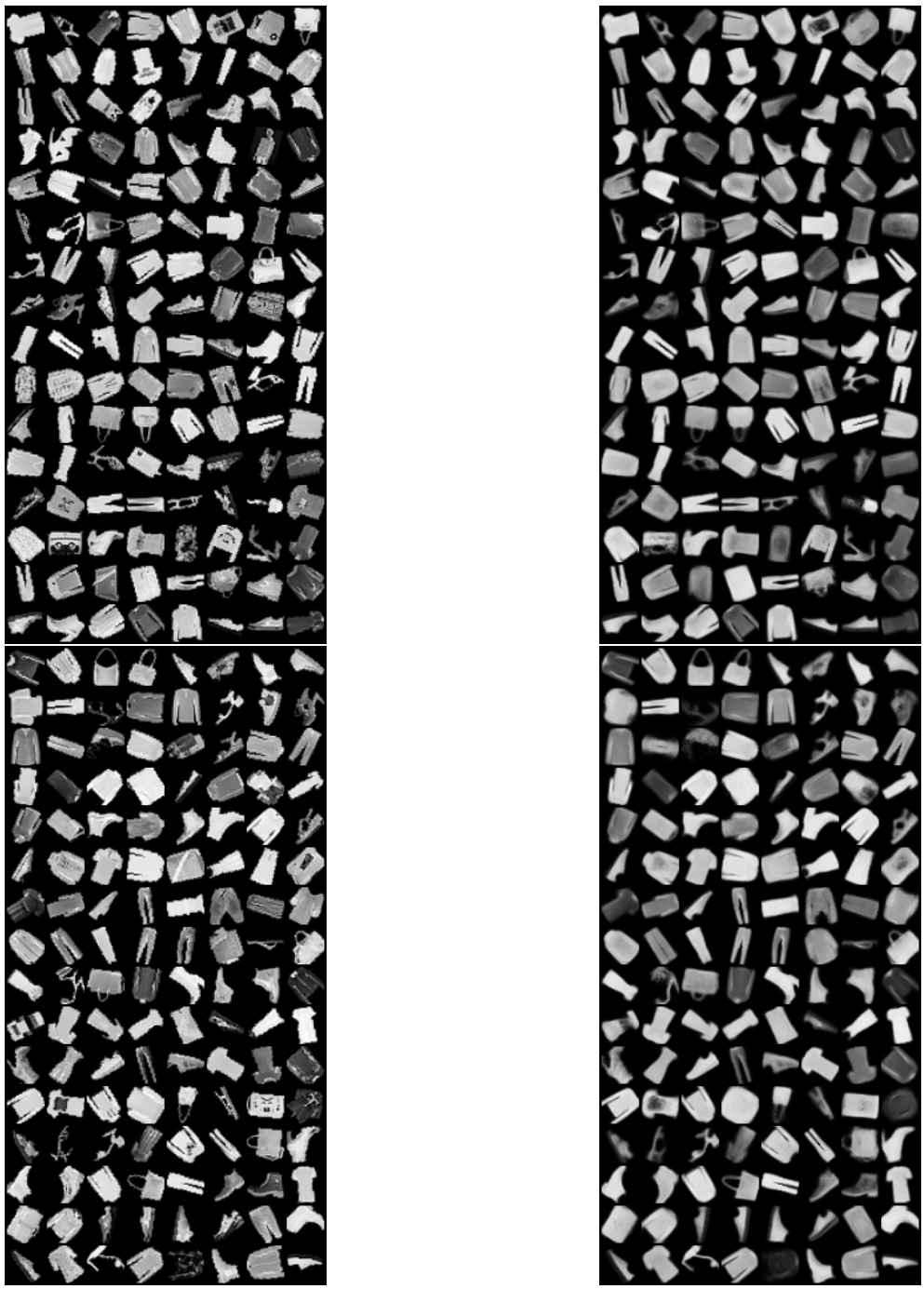
Figure 4.5 Rotation invariant reconstruction(1)- MNIST. Input coordinates are NOT rotated.



(a) Rotated images

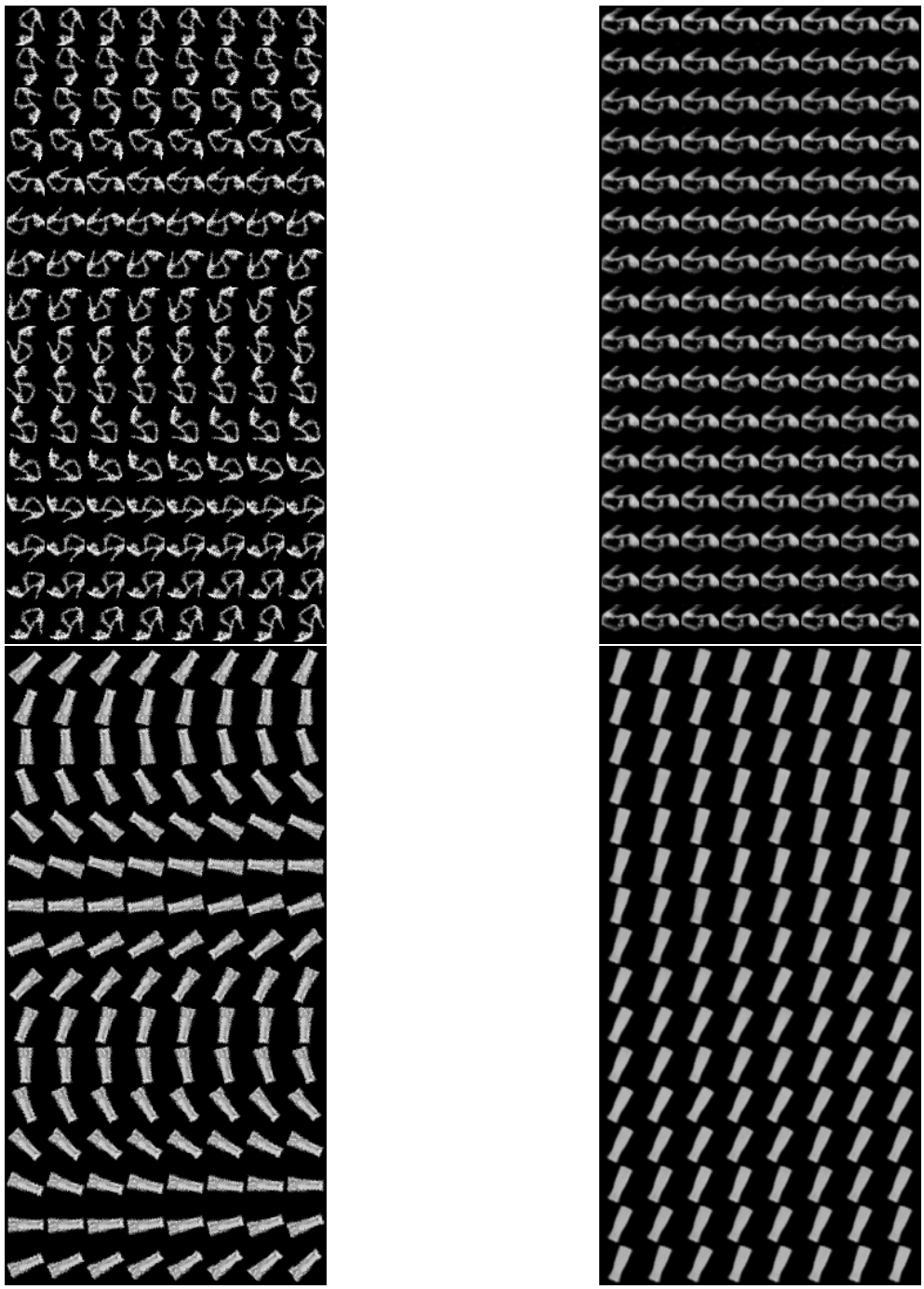
(b) Reconstructed images

Figure 4.6 Rotation invariant reconstruction(2) - MNIST. Input coordinates are NOT rotated.



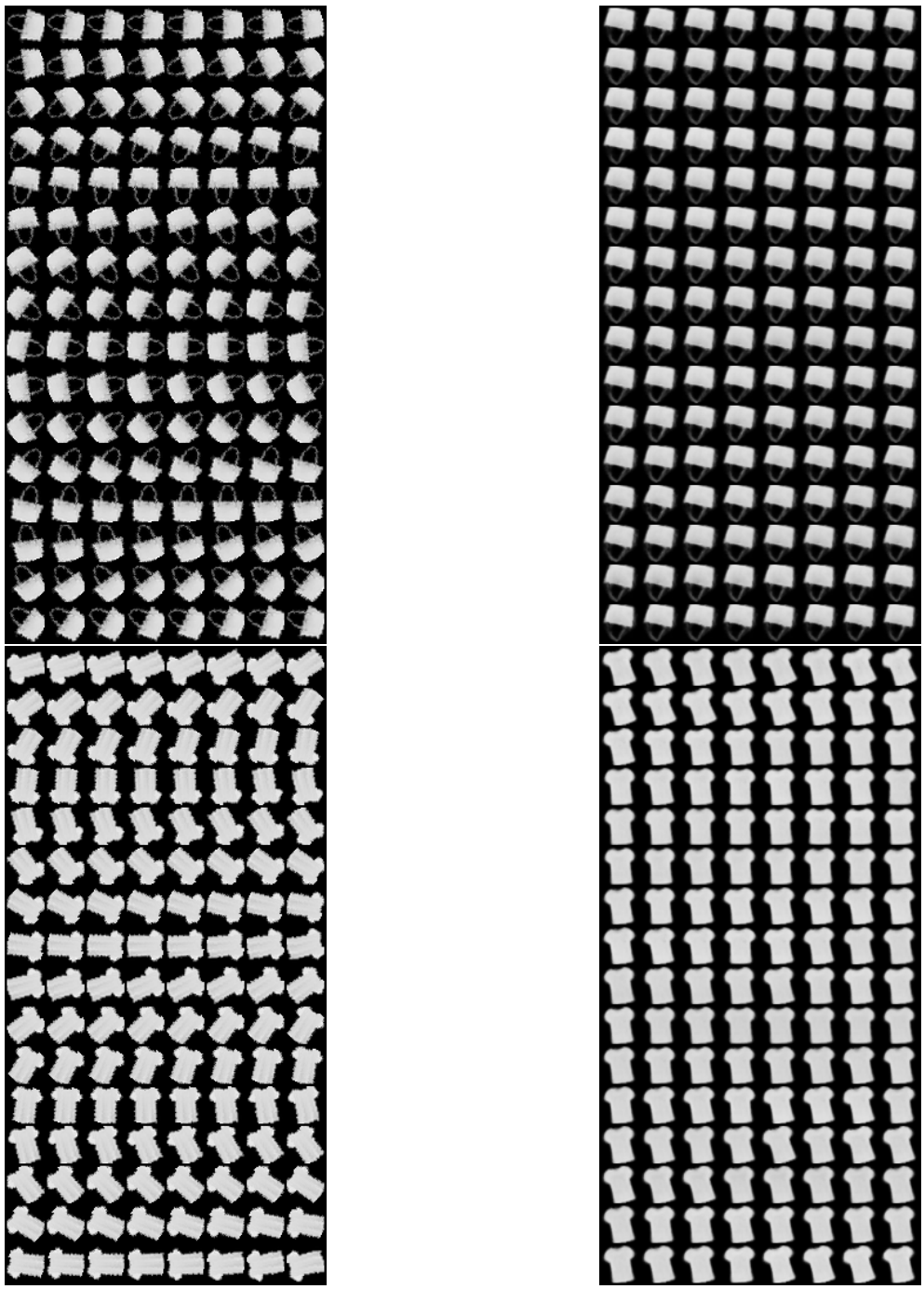
(a) Input images **(b)** Reconstructed images

Figure 4.7 Reconstruction - Fashion MNIST. Input coordinates are rotated by $\hat{\theta}$.



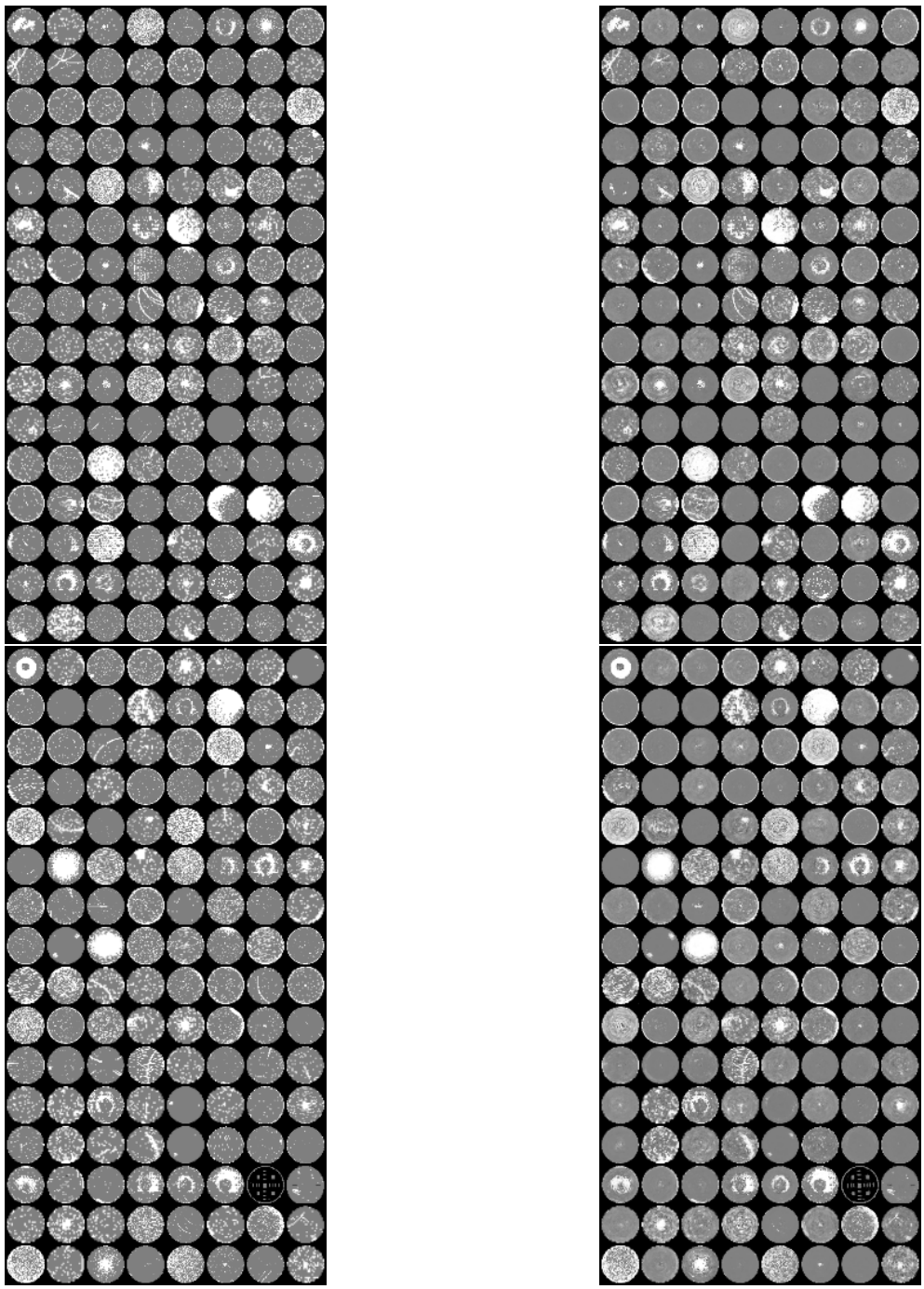
(a) Rotated images **(b)** Reconstructed images

Figure 4.8 Rotation invariant reconstruction(1) - Fashion MNIST. Input coordinates are NOT rotated.



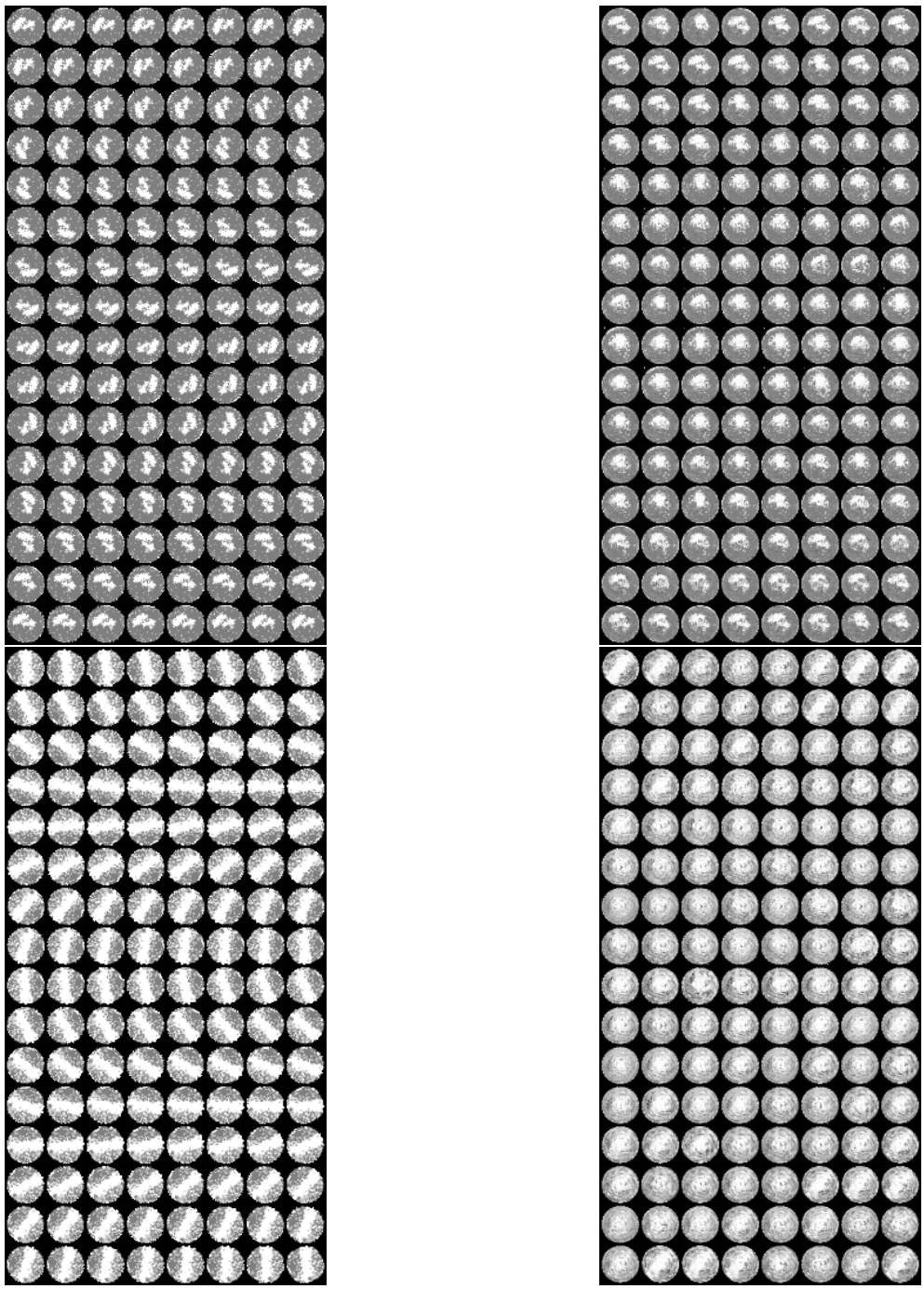
(a) Rotated images **(b)** Reconstructed images

Figure 4.9 Rotation invariant reconstruction(2) - Fashion MNIST. Input coordinates are NOT rotated.



(a) Input images **(b)** Reconstructed images

Figure 4.10 Reconstruction - WM811k. Input coordinates are rotated by $\hat{\theta}$.



(a) Rotated images **(b)** Reconstructed images

Figure 4.11 Rotation invariant reconstruction - WM811k. Input coordinates are NOT rotated.

Clustering Results

We show that our method has better performance than other clustering algorithms(k-means, DEC and Spatial VAE) on fully rotated MNIST, fully rotated FashionMNIST and WM811k. Explanation on clustering algorithms used in experiments is as below:

- **k-means**: Apply k-means directly to datasets.
- **DEC**: Apply DEC(section 2.2.1) to datasets. All hyper-parameters were used as suggested in the DEC paper.
- **Spatial VAE**: Pretraining the Spatial-VAE(section 3.1). And then apply k-means to the learned latent space(i.e, output space of the encoder). All hyper-parameters were used as suggested in the Spatial-VAE paper.
- **AE base-line**: Apply our algorithm to AutoEncoder, NOT based on Implicit Neural Representations(INR). The AE base-line shows that INR is an essential element in our methodology. In the AE base-line, the architecture used the same encoder structure as in our proposed method, but since the AE base-line is based on Auto-Encoder, rather than an INR based structure, a structure such as a hypernetwork or a function representation is not used. Instead, the AE base-line used the decoder with a reversed structure of the encoder. The objective function and learning algorithm of AE base-line are almost same as we proposed. The front part of the encoder output inferences the reference degree, and the rest part goes through reparameterization and input into the decoder. Finally, the decoder is trained to generate a reference image. Let $\mathbf{E}$ and $\mathbf{D}$ be the encoder and decoder respectively. The loss function $\mathcal{L}_{\text{AE base-line}}$

is as follows.

$$\mathcal{L}_{\text{AE base-line}} = \mathcal{L}_{\text{angle}} + \mathcal{L}_{\text{consis}} + \mathcal{L}_{\text{recon}}$$

$\mathcal{L}_{\text{angle}}$ and $\mathcal{L}_{\text{consis}}$ are same as we propsed in section 3.7, and

$$\mathcal{L}_{\text{recon}} = \frac{1}{Res} \sum_{r=1}^{Res} \mathop{\mathbb{E}}_{\mathbf{W}} \left[\left\| (\mathbf{W})_r - \mathcal{R}_{\hat{\theta}}(\mathbf{D}(\mathbf{E}(\mathbf{W})))_r \right\|^2 \right]$$

where $\mathcal{R}_{\hat{\theta}}$ is rotation function with rotation degree $\hat{\theta}$. The overall diagram is shown in Figure 4.12. And then apply k-means to the learned latent

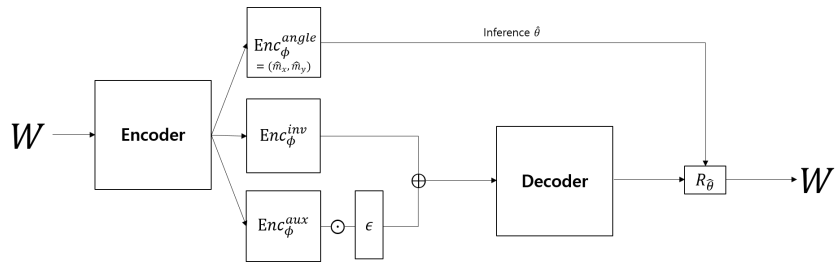


Figure 4.12 AE base-line

space(i.e, output space of the encoder).

- **Ours**: Pretrain with INR which is proposed method in this paper. And then apply k-means to the learned latent space(i.e, output space of the encoder)
- **Ours+DEC**: With pretrained model which is obtained by **Ours**. And then apply **DEC** to the learned latent space. We set $\alpha = 0.0001$ for degree of the freedom of the student distribution in **DEC**.

Naively applying k-means clustering on learned latent space of the encoder already outperformed to other methods. Our method with DEC showed the best performance on rotated MNIST, rotated FashionMNIST and WM811k(Table 4.1).

Table 4.1 Clustering results

	Rotated MNIST	Rotated Fashion MNIST	WM811k
k-means	26.7%	25.64 %	44.1 %
DEC	28.6 %	26.1 %	36.3%
Spatial VAE	56.4%	41.2 %	27.4%
AE base-line	53.9%	46.8%	49.8 %
Ours	79.3%	49.3 %	55.5 %
Ours+DEC	**81.1%**	**50.2 %**	**56.1%**

Also, we show that the images belonging to each predicted label after clustering(Fig 4.13, 4.14, 4.15). We randomly sample the images, not cherry picking.

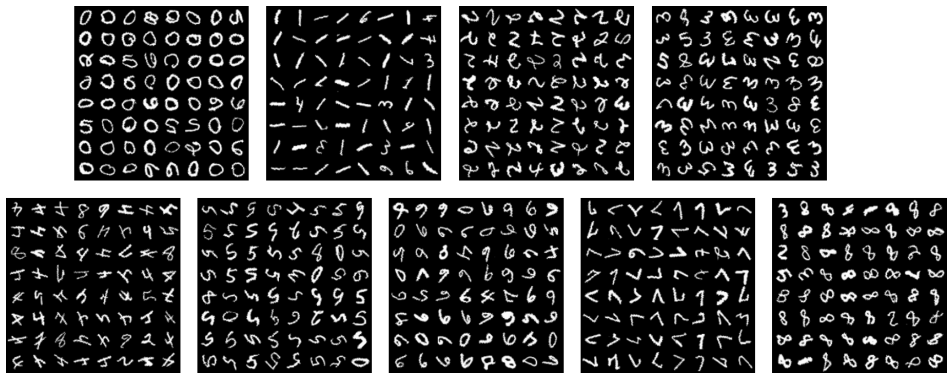


Figure 4.13 Clustered MNIST by Ours+DEC.

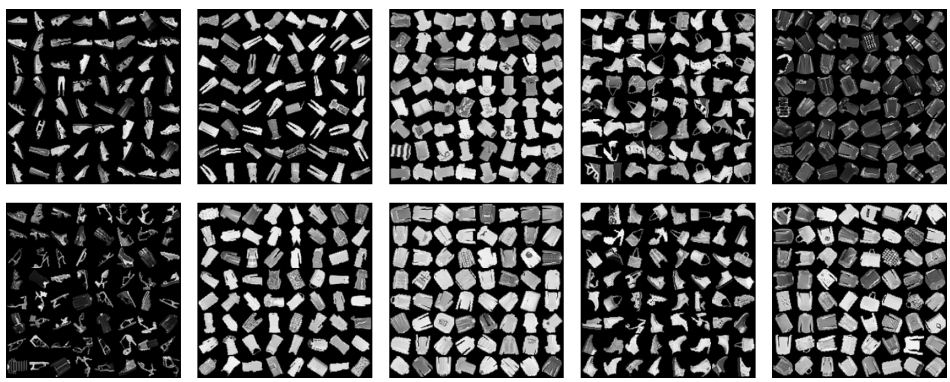


Figure 4.14 Clustered FashionMNIST by Ours+DEC.

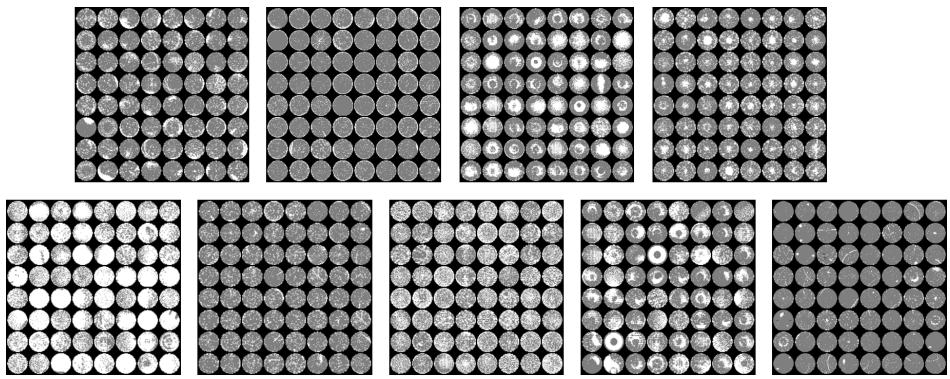


Figure 4.15 Clustered WM811k by Ours+DEC.

5 Conclusion

In this paper, we posed the necessity for breaking the symmetry in the task that predicting the rotation degree in unsupervised setting. We proposed to use the center of mass as a rule to break the symmetry. Using this rule and Implicit Neural Representations (INR), we proposed two-stage method to cluster randomly rotated image datasets. First stage, training the encoder to output the latent vector which is disentangled to latent rotation angle(*latent angle*) and rotationally invariant latent vector(*latent*). Second stage, clustering *latent* according to the DEC[41] has superior performance on randomly rotated datasets than other methods. It is first approach to cluster with implicit neural representations (INR) as far as we know.

Of course, various other methods can be applied to our method. For example, in the first stage, the consistency loss in the section 3.7 has the effect that collapsing the latent space, contrastive learning method can be applied instead of our consistency loss for similar features becomes closer and disimilar features becomes far. And also in the second stage, we IIC[18], SCAN[40] and other methods can be applied instead of DEC[41].

Bibliography

[1] Nuri Benbarka, Timon Höfer, Hamd ul Moqeet Riaz, and Andreas Zell: Seeing implicit neural representations as fourier series. *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (2022), 2283–2292.

[2] Tristan Bepler, Ellen Zhong, Kotaro Kelley, Edward Brignole, and Bonnie Berger: Explicitly disentangling image content from translation and rotation with spatial-vae. *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.

[3] Koby Bibas, Gili Weiss-Dicker, Dana Cohen, Noa Cahan, and Hayit Greenspan: Learning Rotation Invariant Features For Cryogenic Electron Microscopy Image Reconstruction. *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*. Apr. 2021, 563–566. DOI: 10.1109/ISBI48211.2021.9433789.

[4] Piotr Bojanowski, and Armand Joulin: Unsupervised learning by predicting noise. *ArXiv*, **abs/1704.05310** (2017).

[5] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze: Deep clustering for unsupervised learning of visual features. *ECCV*. 2018.

[6] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan: Deep adaptive image clustering. Oct. 2017, 5880–5888. DOI: 10.1109/ICCV.2017.626.

[7] Zhiyuan Dang, Cheng Deng, Xu Yang, Kun Wei, and Heng Huang: Nearest neighbor matching for deep clustering. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, 13693–13702.

[8] Kien Do, Truyen Tran, and Svetha Venkatesh: Clustering by maximizing mutual information across views. Oct. 2021, 9908–9918. DOI: 10.1109/ICCV48922.2021.00978.

[9] Carl Doersch, Abhinav Gupta, and Alexei Efros: Unsupervised visual representation learning by context prediction (May 2015). DOI: 10.1109/ICCV.2015.167.

[10] Emilien Dupont, Hyunjik Kim, Ali Eslami, Danilo Rezende, and Dan Rosenbaum: From data to functa: your data point is a function and you should treat it like one. *arXiv preprint arXiv: 2201.12204* (2022).

[11] Emilien Dupont, Yee Whye Teh, and Arnaud Doucet: Generative models as distributions of functions. *CoRR*, **abs/2102.04776** (2021).

[12] Zeyu Feng, Chang Xu, and Dacheng Tao: Self-supervised representation learning by rotation feature decoupling. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[13] Tomer Galanti, and Lior Wolf: On the modularity of hypernetworks. *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, 10409–10419.

[14] Spyros Gidaris, Praveer Singh, and Nikos Komodakis: Unsupervised representation learning by predicting image rotations. *International Conference on Learning Representations*. 2018.

[15] David Ha, Andrew M. Dai, and Quoc V. Le: Hypernetworks. *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[16] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama: Learning discrete representations via information maximizing self augmented training. Aug. 2017.

[17] S. Jenni, and Paolo Favaro: Self-supervised feature learning by learning to spot artifacts. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), 2733–2742.

[18] Xu Ji, Andrea Vedaldi, and Joao Henriques: Invariant Information Clustering for Unsupervised Image Classification and Segmentation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea (South): IEEE, Oct. 2019, 9864–9873. ISBN: 978-1-72814-803-8. DOI: 10.1109/ICCV.2019.00996.

[19] Angjoo Kanazawa, David W. Jacobs, and Manmohan Chandraker: Warpnet: weakly supervised matching for single-view reconstruction. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[20] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich: Colorization as a proxy task for visual understanding. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 840–849.

[21] Etai Littwin, Tomer Galanti, Lior Wolf, and Greg Yang: On infinite-width hypernetworks. *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, 13226–13237.

[22] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng: Nerf: representing scenes as neural radiance fields for view synthesis. *ECCV*. 2020.

[23] Ishan Misra, and Laurens van der Maaten: Self-supervised learning of pretext-invariant representations. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[24] Tom Monnier, Thibault Groueix, and Mathieu Aubry: Deep transformation-invariant clustering. *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, 7945–7955.

[25] T. Mundhenk, Daniel Ho, and Barry Chen: Improvements to context based self-supervised learning (Nov. 2017).

[26] Mehdi Noroozi, and Paolo Favaro: Unsupervised learning of visual representations by solving jigsaw puzzles. *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Cham: Springer International Publishing, 2016, 69–84. ISBN: 978-3-319-46466-4.

[27] Mehdi Noroozi, Ananth Vinjimoor, Paolo Favaro, and Hamed Pirsiavash: Boosting self-supervised learning via knowledge transfer. June 2018, 9359–9367. DOI: 10.1109/CVPR.2018.00975.

[28] David Novotny, Samuel Albanie, Diane Larlus, and Andrea Vedaldi: Self-supervised learning of geometrically stable features through probabilistic introspection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[29] Aaron van den Oord, Yazhe Li, and Oriol Vinyals: *Representation learning with contrastive predictive coding*. 2018. DOI: 10.48550/ARXIV.1807.03748.

[30] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros: Context encoders: feature learning by inpainting. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 2536–2544.

[31] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville: Film: visual reasoning with a general conditioning layer. *AAAI*. 2018.

[32] Zhongzheng Ren, and Yong Jae Lee: Cross-domain self-supervised multi-task feature learning using synthetic imagery. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), 762–771.

[33] Ignacio Rocco, Relja Arandjelovic, and Josef Sivic: Convolutional neural network architecture for geometric matching. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[34] Yuming Shen, Ziyi Shen, Menghan Wang, Jie Qin, Philip Torr, and Ling Shao: You never cluster alone. *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, 27734–27746.

[35] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein: Implicit neural representations with periodic activation functions. *Proc. NeurIPS*. 2020.

[36] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein: Scene representation networks: continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*. 2019.

[37] Ivan Skorokhodov, Savva Ignatyev, and Mohamed Elhoseiny: Adversarial generation of continuous images. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, 10753–10764.

[38] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng: Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS* (2020).

[39] Tsung Wei Tsai, Chongxuan Li, and Jun Zhu: Mi{ce}: mixture of contrastive experts for unsupervised image clustering. *International Conference on Learning Representations*. 2021.

[40] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool: SCAN: Learning to Classify Images Without Labels. *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Vol. 12355. Cham: Springer International Publishing, 2020, 268–285. ISBN: 978-3-030-58606-5 978-3-030-58607-2. DOI: 10.1007/978-3-030-58607-2_16.

[41] Junyuan Xie, Ross Girshick, and Ali Farhadi: Unsupervised Deep Embedding for Clustering Analysis (), 10.

[42] Asano YM., Rupprecht C., and Vedaldi A.: Self-labelling via simultaneous clustering and representation learning. *International Conference on Learning Representations*. 2020.

[43] Dejiao Zhang, Feng Nan, Xiaokai Wei, Shang-Wen Li, Henghui Zhu, Kathleen McKeown, Ramesh Nallapati, Andrew O. Arnold, and Bing Xiang: Supporting clustering with contrastive learning. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, June 2021, 5419–5430. DOI: 10.18653/v1/2021.naacl-main.427.

[44] Liheng Zhang, Guo-Jun Qi, Liqiang Wang, and Jiebo Luo: Aet vs. aed: unsupervised representation learning by auto-encoding transformations rather than data. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[45] Richard Zhang, Phillip Isola, and Alexei A. Efros: Colorful image colorization. *ECCV*. 2016.

[46] Zhun Zhong, Enrico Fini, Subhankar Roy, Zhiming Luo, Elisa Ricci, and Nicu Sebe: Neighborhood contrastive learning for novel class discovery. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, 10867–10875.

초 록

이미지 군집화는 머신 러닝, 딥 러닝 및 산업에서 중요한 과제이다. 특히, 딥러닝에서 이미지 군집화는 자기주도학습(self-supervised learning)에서 사전 과제(pretext task)로 사용되는 등 다양한 용도로 활용되기 시작했다. 많은 선행 연구들이 벤치마크 데이터셋에서 우수한 성능을 보여주었지만, 이러한 데이터셋의 이미지들은 회전되어 있지 않고 모두 올바른 방향으로 놓여 있다. 하지만, 실제 세계에서 얻은 데이터셋이 항상 이와 같이 올바로 놓여 있을 것이라는 보장은 없다. 우리는 이미지가 무작위로 회전되어 있을 때, 선행 연구들에서의 알고리즘이 잘 작동하지 않다는 것을 지적한다. 본 논문에서는 음적 신경망 표현(Implicit Neural Representations)을 활용하여 1. 이미지의 잠재 회전 각도(latent rotation angle)와 회전 불변인 잠재 벡터(rotationally invariant latent vector)가 분리되어 있는 잠재 벡터(latent vector)를 얻으며, 2. 회전 불변인 잠재 벡터를 이용한 군집화가 무작위로 회전된 데이터셋에서 우수한 성능을 가짐을 보인다. 우리가 아는 한, 이것은 음적 신경망 표현을 이용하여 군집화 하려는 첫 번째 시도이다.

주요어: 회전불변, 클러스터링, 딥러닝, 음적신경망표현, 인공지능

학 번: 2020-21133