이학박사 학위논문

Evenly Angle Dispersing Methods for Convolutional Kernel Regularization

(합성곱 커널 정규화를 위한 고른 각도분산방법)

2022년 8월

서울대학교 대학원

수 리 과 학 부

배 정 우

Evenly Angle Dispersing Methods for Convolutional Kernel Regularization

(합성곱 커널 정규화를 위한 고른 각도분산방법)

지도교수 강 명 주

이 논문을 이학박사 학위논문으로 제출함

2022년 4월

서울대학교 대학원

수 리 과 학 부

배 정 우

배 정 우의 이학박사 학위논문을 인준함

2022년 6월

위 원 장 _____ (인)

부 위 원 장 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

Evenly Angle Dispersing Methods for Convolutional Kernel Regularization

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Jeongwoo Bae

Dissertation Director : Professor Myungjoo Kang

Department of Mathematical Sciences
Seoul National University

August 2022

Abstract

In this thesis, we propose new convolutional kernel regularization methods. Along with the development of deep learning, there have been attempts to effectively regularize a convolutional layer, which is an important basic module of deep neural networks. Convolutional neural networks (CNN) are excellent at abstracting input data, but deepening causes gradient vanishing or explosion issues and produces redundant features. An approach to solve these issues is to directly regularize convolutional kernel weights of CNN. Its basic idea is to convert a convolutional kernel weight into a matrix and make the row or column vectors of the matrix orthogonal. However, this approach has some shortcomings. Firstly, it requires appropriate manipulation because overcomlete issue occurs when the number of vectors is larger than the dimension of vectors. As a method to deal with this issue, we define the concept of evenly dispersed state and propose **PH0** and **MST** regularizations using this. Secondly, prior regularizations which enforce the Gram matrix of a matrix to be an identity matrix might not be an optimal approach for orthogonality of the matrix. We point out that these rather reduces the update of angles between some two vectors when two vectors are adjacent. Therefore, to complement for this issue, we propose **EADK** and **EADC** regularizations which update directly the angle. Through various experiments, we demonstrate that **EADK** and **EADC** regularizations outperform prior methods in some neural network architectures and, in particular, **EADK** has fast learning time.

Key words: Deep Learning, Convolution, Kernel, Regularization, Orthogonality, Evenly Dispersed State

Student Number: 2012-23024

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Deep learning is a tool for dealing with problems in various fields such as image classification [**26**], image inpainting [**30**] and anomaly detection [**23**]. CNN (Convolutional Neural Network) [**12**] plays one of the important roles in this deep learning popularity. This network has a structure made with the motif of the eyes of life. CNN uses a kernel to capture local information of input features and to create an output features by sliding the kernel on input features. This process abstracts complex spatial information into various features. The deeper convolutional layers in CNN, the more abstract information cab be extracted. Abstract features obtained in this way are processed to solve various image related problems because local and global information of input data are properly combined.

However, learning a deep convolutional neural network could have gradient vanishing or expolosion issues. As a solution to this, ResNet [**13**] and Batch Normalization [**16**] have been studied. It is also known that CNN tends to create many redundant features, which make the model heavier than necessary and degrade performance. To deal with these issues, convolutional kernel regularizations have been

devised. The main approach used in convolutioinal kernel regularizations is to properly transform kernel weights of a convolutional layer into a matrix and make the row or column vectors of it orthonormal. This orthogonality method leads CNN to create various features as possible. To have these matrix orthonormal, prior regularizations mainly use the definition of orthogonality, i.e., to make the Gram matrix of it identity. However, if the number of vectors we want to orthonormalize is greater than the dimension of vectors, the overcomplete issue occurs. Therefore prior methods regularize its transpose matrix (subsection **2.2.2**) or use concepts generalizing the orthogonality (subsection **2.4.2**).

In this thesis, we introduce new approaches to regularize convolutional kernel weights. We argue the concept of dispersing vectors that generalizes orthogonality up to overcomplete case, and introduce **PH0** and **MST** in subsection **3.2.3** and **3.3.2** regularizations respectively that use the dispersing concept. Another problem with prior regularizations using inner product for orthogonality is that these have phenomena in which an angle-spreading update occurs relatively small when the angle between two vectors is close to 0. This means that when a model is finely optimized by lowering the learning rate, angles of two adjacent vectors are no longer effectively distant. Thus we introduce evenly angle dispersing regularizations **EADK** and **EADC** in subsection **4.2.1** and **4.2.2** respectively that regularize directly angles of two row vectors rather than inner products. These methods complement less spreading problem and make target matrices be stably orthogonal.

Our contributions are summarized as the followings:

- We introduce the concept of evenly dispersed state and show that this extends

the concept of orthogonality.

- We show that the concept of evenly dispersed state can be used for kernel regularization by utilizing topological techniques (**PH0**, **MST**) or angle dispersing techniques (**EADK**, **EADC**).
- We point out a deficiency of some prior convolutional kernel regularizations and propose our evenly angle dispersing regularizaitons as alternatives.
- We show that our proposed methods disperse vectors on $\mathbb{S}^2$ more effectively and allow us to conversely predict the ideal adjacent angle of evenly dispersed state of arbitrary points on $\mathbb{S}^2$.
- We evaluate prior and our proposed convolutional kernel regularizations on benchmark image datasets and demonstrate that our method achieves state-of-the-art (SOTA) performance in our experiments.

Chapter 2

Preliminaries

Before arguing our methods, in Chapter **2**, we examines the various convolutional kernel regularizations that currently exist. As a preliminary work on that, we first introduce two matrices, kernel matrix and convolution matrix in subsection **2.1.1** and **2.1.2** respectively, which are made form kernel weights of a convolutional layer in CNN. Kernel matrix is a matrix that is mainly used in the actual calculation of CNN operation simply by reshaping kernel weights. Convolution matrix is the matrix representation of the linear operation corresponding CNN operation, i.e., convolution operation. This representation retains the shift invariant property, which is the innate property of convolution.

In section **2.2**, we introduce **SO** and **DSO** regularizations as the most basic methods [**1, 28**]. **SO** is the regularization that makes the Gram matrix of the kernel matrix identity matrix. This leads to approximate orthogonality of the kernel matrix. However, depending on whether the shape of the kernel matrix is fat or thin, it can be an overcomplete problem. As an alternative way to compensate for this, there is **DSO** method of giving **SO** regularization to the kernel matrix and

its transpose matrix simultaneously.

In section **2.3**, we introduce the Mutual Coherence regularization [**8**]. The method is to find the two vectors with the smallest angle between them and to make them orthogonal. The effect of this method, however, is insignificant because the number of regularized weights is just two.

In section **2.4**, we deal with Restricted Isometry Property [**5**] and introduce **SRIP** regularization that is based on the property [**2**]. Restricted Isometry Property is a property that appears in the decoding problem of linear programming, plays a role of a condition for recovering original information from encoded data. This condition is considered to be a generalized orthogonality that allows us to talk about orthogonality for overcomplete problems. **SRIP** uses it to regularize a kernel matrix.

In section **2.5**, we introduce **OCNN** regularization [**27**]. Unlike the above regularization, **OCNN** is different in that it uses a convolution matrix instead of a kernel matrix. The authors show that orthogonalizing a convolution matrix is a stronger condition than orthogonalizing the corresponding kernel matrix. In the practical calculation, orthogonalizing operation is performed through convolutional operation. They also show that their method can ignore overcomplete issues.

2.1 Two Ways of Understanding CNN Layers as Matrix Operations

Convolution in Neural Networks is an operation in which, as show in **Figure 2.1**, a kernel moves by some stride value to produce an output pixel value for overlapping area. Since this operation is based on the shared-weight, it is also

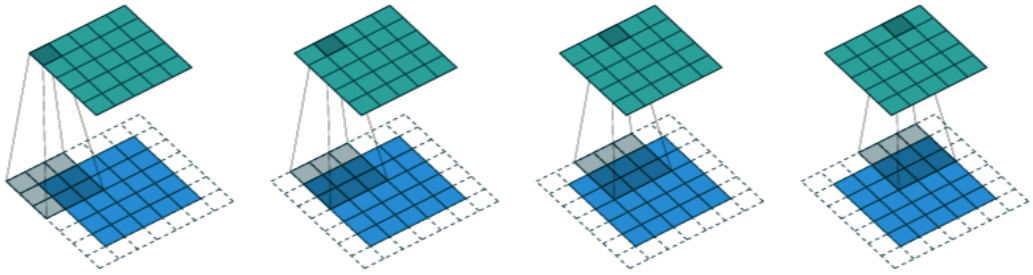

Figure 2.1: Convolving a 3 × 3 kernel over a 5 × 5 input using 1 padding and 1 stride [**9**].

know as Shift Invariant Operation [**32**]. In this section, we introduce two methods to formulate convolutional layer as matrix multiplication.

2.1.1 Kernel Matrix

Im2col [**14,29**] is a method of transforming the operation into linear transform. Since it enables efficient GPU computation, Im2col is fast and has been widely used in deep neural networks.

The specific method of performing im2col in the case of going from one channel input to multi-channel output is as follows (see **Figure 2.2**).

1. Remake the input into the matrix I_c with columns corresponding to all receptive fields that appear while sliding a kernel.

2. Each kernel is reshaped to a row vector of a matrix K so that the inner product are performed with the the receptive field vectorized into column.

3. The output of the matrix multiplication of two matrices K and I_c is reshaped into the final output features.

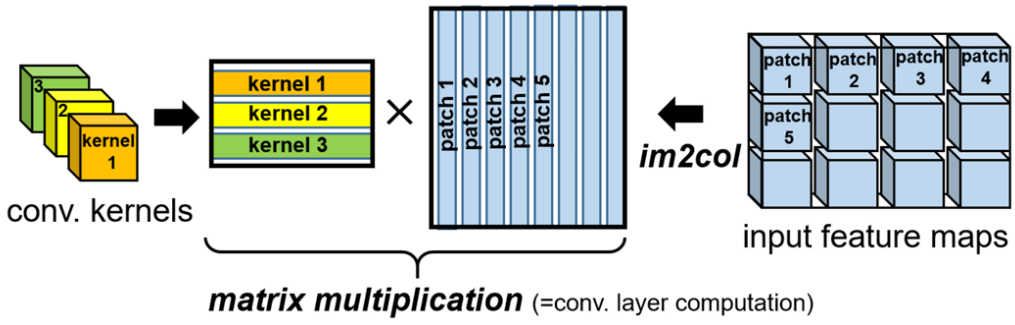

Figure 2.2: Illustration of im2col method [**29**].

The reshaped matrix K of convolutional kernel weight in 2 is called **Kernel Matrix**. If an input feature has multiple channels, columnized input matrices are connected vertically and kernel matrices are horizontally attached. In this case the kernel matrix K is of the shape $M \times Ck_hk_w$, where M, C, k_h and k_w are the number of output channels, the number of input channels, kernel width and kernel height respectively.

$$K \in \mathbb{R}^{M \times Ck_hk_w} \tag{2.1}$$

2.1.2 Convolution Matrix

The matrix I_c in the Im2col method have duplicated information of input features, but the kernel matrix K retains its original size. Conversely, it is also possible to retain the size of input features and duplicate the information of the convolutional kernel weights. Since convolution operation is linear, this approach is naturally defined and reasonable in theoretical perspective.

Let y be the output obtained by applying circular convolution with a kernel **a** and an input **x** [**24, 25**]. Since circular convolution is linear with respect to **x**, we

can find an appropriate matrix $\mathbf{C_a}$ that satisfies **Equation 2.2**, in which $\mathbf{x}$ is used also to denote the flattened vector of the input feature.

$$\mathbf{y} = \mathbf{a} * \mathbf{x} = \mathbf{C_a x} \tag{2.2}$$

Let $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{a} = [a_0, a_1, \ldots, a_{n-1}]$ have the same size by doing zero padding if necessary. Then $\mathbf{C_a}$ is of the form in (2.3) and this matrix are called **circulant matrix** [**7**].

$$\mathbf{C_a} = \begin{bmatrix} a_0 & a_1 & \cdots & a_{n-1} \\ a_{n-1} & a_0 & \cdots & a_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & \cdots & a_0 \end{bmatrix} \tag{2.3}$$

In convolutional neural networks, kernels and input features are 2D shaped and thus we need to reshape these into 1D shape. Suppose, for simplicity, that each number of input and output channel is one. For the matrix operation in (2.2) fit our 2D convolutional operation, the convolutional kernel weight should be reformed to a sparse matrix like in (2.5).

For example, let $\mathbf{a}$ in (2.4) be a convolutional kernel weight and $\mathbf{x}$ an input feature with 3×3 size.

$$\mathbf{a} = \begin{bmatrix} a_0 & a_1 \\ a_2 & a_3 \end{bmatrix} \tag{2.4}$$

Then the kernel is reformed into $\mathbf{C_a}$ in (2.5).

$$\mathbf{C_a} = \begin{bmatrix} a_0 & a_1 & 0 & a_2 & a_3 & 0 & 0 & 0 & 0 \\ 0 & a_0 & a_1 & 0 & a_2 & a_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_0 & a_1 & 0 & a_2 & a_3 & 0 \\ 0 & 0 & 0 & 0 & a_0 & a_1 & 0 & a_2 & a_3 \end{bmatrix} \tag{2.5}$$

It should be noted that, strictly speaking, the matrix (2.5) is not a circulant matrix but a **doubly block-Toepliz matrix** [**4**]. This is because the convolution in convolutional neural networks does not work in a way that circulates at the boundary of the input feature. However, if the kernel size is significantly smaller than the input feature size, we may think that the convolution of convolutional neural networks is also a circular convolution and develop a theory.

For the convolutional layer with C_I input channels, C_O output channels and $\mathbf{x} \in \mathbb{R}^{H \times W \times C_I}$, the output $\mathbf{y} \in \mathbb{R}^{H' \times W' \times C_O}$ is evaluated by matrix multiplication with the matrix $\mathcal{K}$ in (2.6) and $\mathbf{x}$ [**20, 27**]. The elements $\mathbf{C}_{\mathbf{a}ij}$ of $\mathcal{K}$ in (2.6) are circulant matrix(or precisely doubly block-Toepliz matrix) like in (2.5). This method is illustrated in **Figure 2.3**.

$$\mathcal{K} = \begin{bmatrix} \mathbf{C}_{\mathbf{a}00} & \mathbf{C}_{\mathbf{a}01} & \cdots & \mathbf{C}_{\mathbf{a}0C_I} \\ \mathbf{C}_{\mathbf{a}10} & \mathbf{C}_{\mathbf{a}11} & \cdots & \mathbf{C}_{\mathbf{a}1C_I} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_{\mathbf{a}C_O0} & \mathbf{C}_{\mathbf{a}C_O1} & \cdots & \mathbf{C}_{\mathbf{a}C_OC_I} \end{bmatrix} \tag{2.6}$$

The matrix $\mathcal{K}$ in (2.6) is called **Convolution Matrix**.

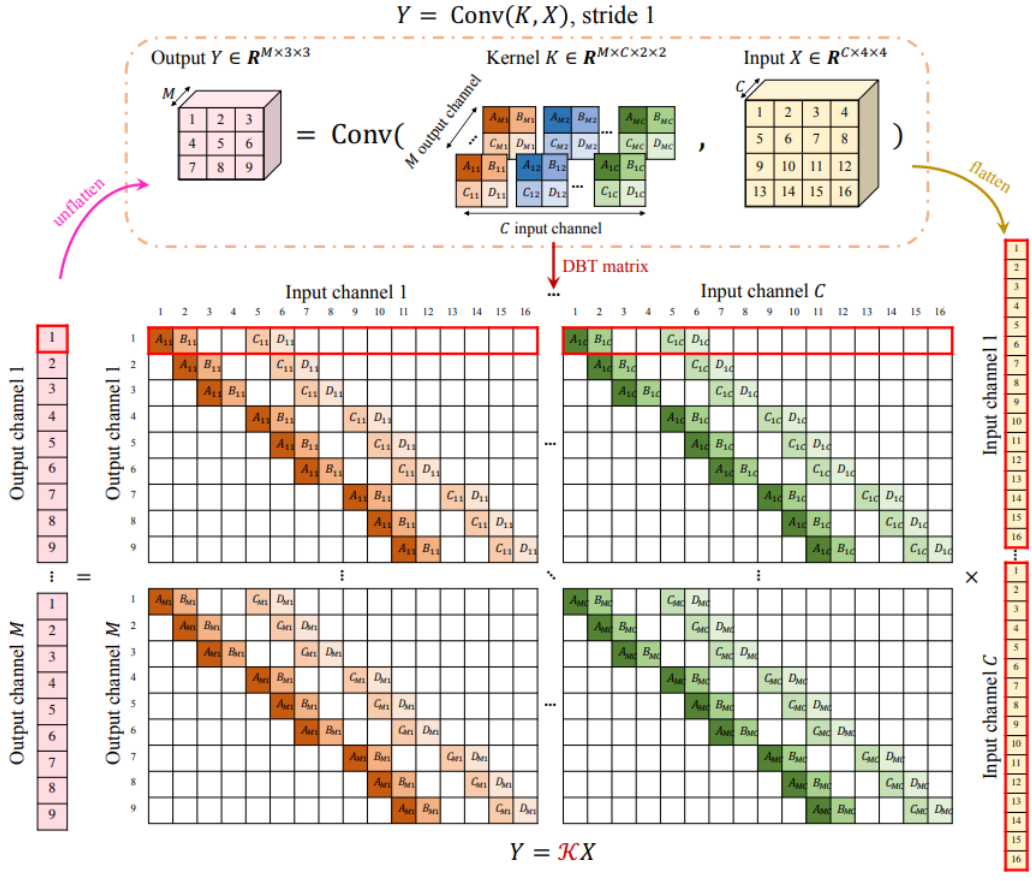

Figure 2.3: Convolution matrix [**27**]

2.2 Soft Orthogonality

2.2.1 SO Regularization

The most basic method to regularize convolutional kernel weights in CNN is to give kernel matrices an orthogonal condition. Let a kernel matrix W be of the shape $M \times Ck_hk_w$, where M is output channel number, C is input channel number and k_h, k_w are height, width of the convolutional kernel respectively. The most naive method to give W orthogonality condition is to enforce the Gram matrix of the kernel matrix W to be close to identity matrix I under Frobenius norm **[28]**, which is termed as **Soft Orthogonality (SO)** regularization:

$$\lambda||WW^T - I||_F^2, \tag{2.7}$$

where λ is a regularization coefficient **[2]**.

The effect of the orthogonality regularization is firstly the diversification of output features. Orthogonality condition makes the frame composed of row vectors of W be close to a Stiefel manifold **[17]**, so the dissimilarity of the kernels increases and the output features are diversified. Another effect of orthogonality is that since orthogonal matrix is isometry, it makes features preserve energy through convolutional layers. The other effect is that orthogonality regularization normalize its kernel weights. j-th channel of an output feature is obtained by convolving C distinct kernels with corresponding channels of an input features and applying an activation function to the channel-wise sum of convolving results. If we call the kernels used here $k_{j1}, \ldots, k_{jC}$, this corresponds to a row of the kernel matrix W, the norm of these are normalized to one as in (2.8).

$$1 = \|(k_{j1}, \ldots, k_{jC})\|_F^2 = \|k_{j1}\|_F^2 + \ldots \|k_{jC}\|_F^2 \tag{2.8}$$

Regardless of the number of input channels, the sum of the squares of norm of kernels which correspond to a channel of output feature is one. The orthogonality regularization normalize the kernel matrix W in this sense and the problems of vanishing and exploding gradients are reduced.

2.2.2 DSO Regularization

Soft orthogonality regularization is the simplest orthogonalizing method, but when kernel matrix is fat, i.e., $M < Ck_hk_w$, it is overcomplete and cannot satisfy orthogonality. Therefore better ways should be suggested to solve this problem. A method is to use a regularization term as in (2.7) if the matrix is tall, i.e., $M \leq Ck_hk_w$, else as in (2.9).

$$\lambda\|W^TW - I\|_F^2 \tag{2.9}$$

Another method is to use the sum of terms of (2.7) and (2.9) without considering the shape of kernel matrix as in (2.10), and is called **Double Soft Orthogonality (DSO)** regularization. DSO regularization is similar to the previous selective method, but the authors of **[2]** argue that DSO always outperforms that in their experiments.

$$\lambda(\|W^TW - I\|_F^2 + \|WW^T - I\|_F^2) \tag{2.10}$$

2.3 Mutual Coherence

2.3.1 MC Regularization

The mutual coherence [**8**] is a method of reducing the degree of correlation between the overall row vectors of kernel matrix W by choosing the two most correlated vectors and making them decorrelated. The degree of correlation can be expressed as the absolute value of the cosine between some two vectors as shown in (2.11), where w_i denotes i-th row of kernel matrix W, and the largest absolute value of cosine for any distinct two vectors is used as a regularization.

$$\mu_W = \max_{i \neq j} \frac{|< w_i, w_j >|}{\|w_i\| \cdot \|w_j\|} \tag{2.11}$$

Authors of [**2**] suggest an alternative expression of the mutual coherence, also is called **Mutual Coherence (MC)**, as shown in (2.12).

$$\lambda \|WW^T - I\|_\infty \tag{2.12}$$

In the MC regularizer (2.12), there is no direct normalization of the row of W by dividing it by its norm, but it becomes the same expression as (2.11) because the norm is enforced to be one.

2.4 Spectral Restricted Isometry Property

2.4.1 Restricted Isometry Property

Authors of Decoding by Linear Programming [**5**] concentrate on the problem of recovering an input vector $f \in \mathbb{R}^n$ from corrupted measurements $y = Af + e$

[5]. Here, A is an m by n matrix and e is an arbitrary and unknown vector of errors. Restricted isometry property is used as a condition to solve the problem. This condition is related to the degree of orthogonality of the columns of matrix with any shape. Since convolutional kernel regularization is mainly interested in orthogonality, we focus, in this section, on dealing with the restricted isometry property rather than solving the recovering problem mentioned above.

Consider the question of whether $f \in \mathbb{R}^n$ can be uniquely recovered when a vector $y \in \mathbb{R}^m$ and an $m \times n$ matrix A are known such that $y = Af$. For a square matrix A which is orthogonal, since it has full rank, f is uniquely recovered, i.e., $f = A^{-1}y$. However, for a general matrix, the answer is not obvious. Then, under what conditions can we recover f when y and A are given?

We denote by $(a^j)_{j\in J}$ the columns of the matrix A and by H the Hilbert space spanned by these vectors. For any $T \subseteq J$, we let A_T be the submatrix with column indices $j \in T$ so that

$$A_T c = \sum_{j\in T} c_j a^j \in H, \tag{2.13}$$

where $(c_j)_{j\in T} \in \mathbb{R}^{|T|}$ [5].

Definition 1. *[5] Let A be the matrix with the finite collection of $(a^j \in \mathbb{R}^m)_{j\in J}$ as columns. For every integer $1 \leq S \leq |J|$, we define the* ***S-restricted isometry constants*** *δ_S to be the smallest quantity such that A_T obeys*

$$(1 - \delta_S)\|c\|^2 \leq \|A_T c\|^2 \leq (1 + \delta_S)\|c\|^2 \tag{2.14}$$

for all subsets $T \subseteq J$ of cardinality at most S, and all real coefficients $(c_j)_{j\in T}$.

Theorem 1. *[5] Given A, suppose that $S \geq 1$ such that $\delta_{2S} < 1$, and let $T \subseteq J$*

such that $|T| \leq S$. *Let* $f := A_T c$ *for some arbitrary* $|T|$*-dimensional vector* c. *Then the set* T *and the coefficients* $(c_j)_{j \in T}$ *can be reconstructed uniquely from knowledge of the vector* f *and the* a^j*'s.*

Proof. Suppose for contradiction that f has tow distinct sparse representation $f = A_T c = A_{T'} c'$ where $|T|, |T'| \leq S$.Then $A_T c - A_{T'} c' = 0$ and this is reformulated as

$$A_{T \cup T'} \tilde{c} := \begin{cases} c_i & \text{if } i \in T \backslash T' \\ c_i - c'_i & \text{if } i \in T \cap T' \\ -c'_i & \text{if } i \in T' \backslash T. \end{cases}$$

Since $|T \cup T'| \leq 2S$ and $\delta_{2S} \leq 1$,

$$0 \leq (1 - \delta_{2S}) \|\tilde{c}\| \leq \|A_{T \cup T'} \tilde{c}\| = 0,$$

and thus $\tilde{c} = 0$ which is contradiction to the hypothesis that two representations were distinct. □

2.4.2 SRIP Regularization

In subsection **2.4.1**, the concept of isometry is extended regardless of the shape of matrices and is used to solve the recovering problem. Since, in the case of square matrix, isometry and orthogonality are equivalent, this can also be seen as an extension of orthogonality to general shape of matrices. Hence, this concept can be used as a regularizer that enforces the convolutional kernel matrix W to be as orthogonal as possible without worrying about overcomplete problem.

We call $z \in \mathbb{R}^m$ k-sparse if the number of nonzero elements of z is less than or

equal to k. From the restricted isometry property condition in (2.14), we assume that for all vectors $z \in \mathbb{R}^m$ that is k-sparse, there exists a small $\delta_W \in (0,1)$ such that

$$1 - \delta_W \leq \frac{\|W^T z\|^2}{\|z\|^2} \leq 1 + \delta_W. \tag{2.15}$$

The above condition (2.15) essentially requires that every set with cardinality no larger than k of row vectors of W shall behave like an orthogonal system. Taking an extreme case with $k = m$, this condition turns into a criterion that enforces the W to be close to orthogonal.

We rewrite the condition (2.15) with $k = m$ in the form

$$\left| \frac{\|W^T z\|^2}{\|z\|^2} - 1 \right| \leq \delta_W, \forall z \in \mathbb{R}^m. \tag{2.16}$$

Since the spectral norm of W^T is equivalent to the largest singular value of W^T, i.e.,

$$\sigma(W^T) = \sup_{z \in \mathbb{R}^m, z \neq 0} \frac{\|W^T z\|}{\|z\|}, \tag{2.17}$$

we reach the following proposition.

Proposition 1. $\sigma(WW^T - I) = \sup_{z \in \mathbb{R}^m, z \neq 0} \left| \frac{\|W^T z\|^2}{\|z\|^2} - 1 \right|$.

Proof. By the relation (2.17),

$$\sigma(WW^T - I) = \sup_{z \in \mathbb{R}^m, z \neq 0} \frac{\|WW^T z - z\|}{\|z\|}.$$

We may restrict the domain of z to the vectors with unit norm. Hence the equation

of the proposition is equivalent to

$$\sup_{z\in\mathbb{R}^m,\|z\|=1} \|WW^Tz - z\| = \sup_{z\in\mathbb{R}^m,\|z\|=1} \left|\|W^Tz\|^2 - 1\right|.$$

Therefore it suffices to show that for all z with $\|z\| = 1$,

$$\|WW^Tz - z\| = \left|\|W^Tz\|^2 - 1\right|. \tag{2.18}$$

We complete the proof by the following argument.

$$\begin{aligned}
\|WW^Tz - z\|^2 &= (z^TWW^T - z^T)(WW^Tz - z) \\
&= z^TWW^TWW^Tz - 2z^TWW^Tz + z^Tz \\
&= z^TWW^Tzz^TWW^Tz - 2z^TWW^Tz + z^Tz \\
&= \|W^Tz\|^4 - 2\|W^Tz\|^2 + 1 \\
&= \left|\|W^Tz\|^2 - 1\right|^2
\end{aligned}$$

□

Hence enforcing W^T to be as orthogonal as possible is equivalent to minimizing $\sigma(WW^T - I)$. This method is termed as the Spectral Restricted Isometry Property (SRIP) regularization.

$$\lambda \cdot \sigma(WW^T - I) \tag{2.19}$$

However computing the objective value of (2.19) is costly. To avoid that, we approximate the computation of spectral norm by using the power iteration method **[21]**. Starting with a randomly initialized $v \in \mathbb{R}^m$, we iteratively perform the fol-

lowing procedure a small number of times (2 times by default).

$$u \leftarrow (WW^T - I)v$$
$$v \leftarrow (WW^T - I)u$$
$$\sigma(WW^T - I) \leftarrow \frac{\|v\|}{\|u\|}$$

2.5 Orthogonal Convolutional Neural Networks

2.5.1 OCNN Regularizaiton

In the paper of Orthogoanl Convolutional Neural Networks [27], convolving computations in CNN layers are interpreted as a convolutional structure based manner as described in subsection **2.1.2** and **Figure 2.4 (b)**. Since convolution is linear, if an input is flattened, the matrix corresponding the convolution exists naturally and is called convolution matrix $\mathcal{K}$ with the form like (2.6). Orthogonal convolutional neural networks aims to give this convolution matrix orthogonality.

In the case of multi-channels, $\mathcal{K}$ is of the form in the **Figure 2.3**. Each row of $\mathcal{K}$ implies a flattened form of a kernel calculated at a particular spatial location. Therefore, in order for these rows to be orthogonal, it is sufficient to make sure that the inner product with itself is one and are zero when calculating with rows which is overlapping with the row in the 2D sense. This is illustrated in **Figure 2.5**. When kernel is square of size k and stride is S, the padding size to check orthogonality is $P = \lfloor \frac{k-1}{S} \rfloor \cdot S$. The row orthogonality condition can be easily represented by

$$\text{Conv}(K, K, padding = P, stride = S) = I_{r0}, \tag{2.20}$$

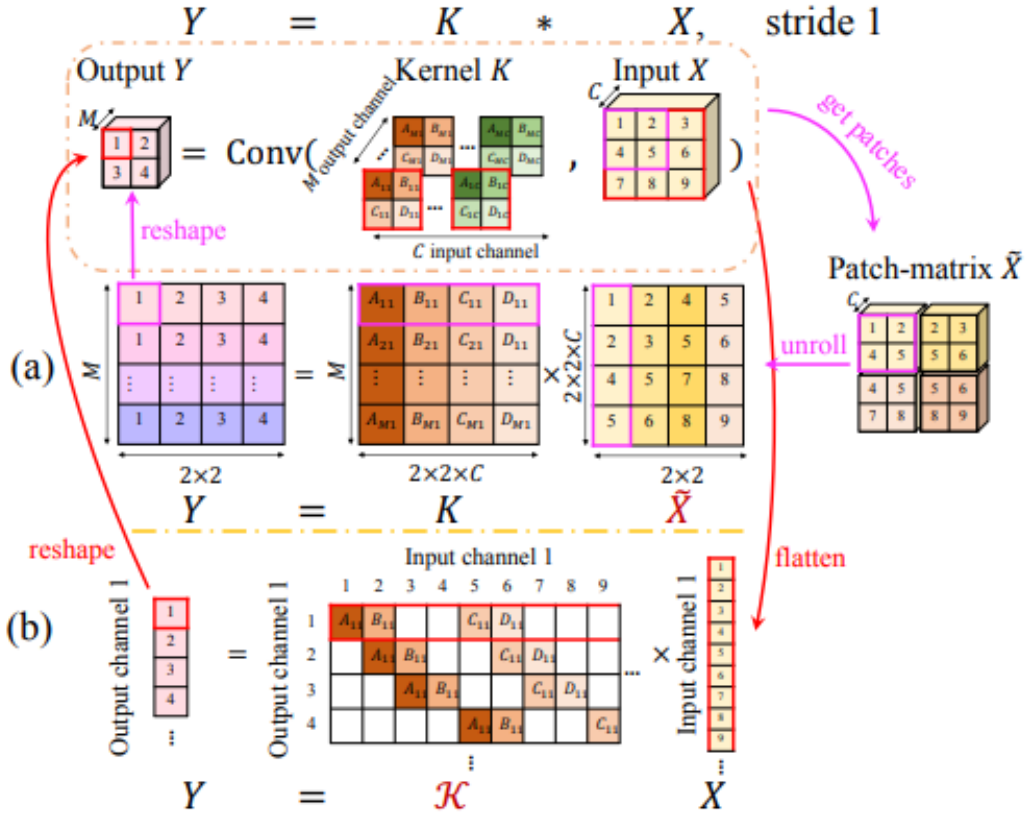

Figure 2.4: A convolutional layer $Y = \mathrm{Conv}(K, X)$ can be formulated as matrix multiplications in two ways: a) im2col methods (kernel matrix K). b) convolutional structure based methods (convolution matrix $\mathcal{K}$) [27].

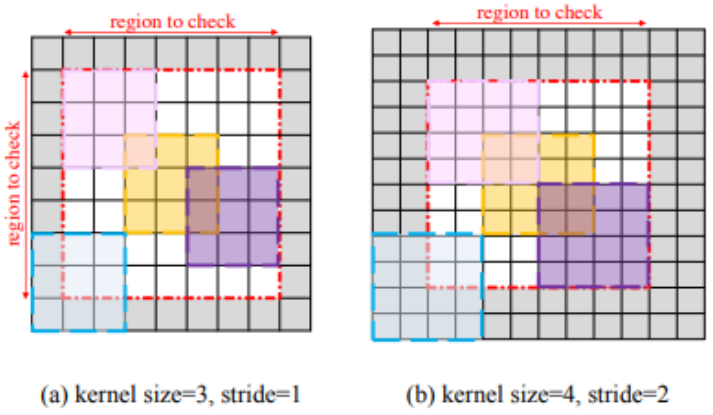

Figure 2.5: The spatial region to check for row orthogonality. It is only necessary to check overlapping filter patches [27].

where $I_{r0} \in \mathbb{R}^{M\times M\times(2P/S+1)\times(2P/S+1)}$ and $K \in \mathbb{R}^{M\times C\times k\times k}$ is the convolutional kernel weight. I_{r0} is a tensor which has zero entries except for the center $M \times M$ matrix which is equal to an identity matrix.

Similar to the row orthogonality, column orthogonality condition can be formulated as

$$\text{Conv}(K^T, K^T, padding = k-1, stride = 1) = I_{c0}, \tag{2.21}$$

where $I_{r0} \in \mathbb{R}^{C\times C\times(2k-1)\times(2k-1)}$ has all zeros except for the center $C \times C$ matrix which has entries like an identity matrix.

The convolutional orthogonality is stronger condition than kernel orthogonality because kernel row- and column-orthogonality condition ((2.7) and (2.9)) can be represented by

$$\begin{cases} \text{Conv}(K, K, padding = 0) = I_{r0} \\ \text{Conv}(K^T, K^T, padding = 0) = I_{c0}, \end{cases} \tag{2.22}$$

where $I_{r0} \in \mathbb{R}^{M\times M\times 1\times 1}, I_{c0} \in \mathbb{R}^{C\times C\times 1\times 1}$ are both equivalent to identity matrices.

One of the interesting points of orthogonal convolutional neural networks is that row- and column-orthogonality are actually equivalent in the MSE sense. Therefore, it is sufficient to consider row orthogonality regardless of whether $\mathcal{K}$ is fat or tall. This method is called **OCNN** regularization.

Lemma 1. *The row orthogonality and column orthogonality are equivalent in the MSE sense, i.e.* $||\mathcal{K}\mathcal{K}^T - I||_F^2 = ||\mathcal{K}^T\mathcal{K} - I'||_F^2 + U$ *for some constant* U*.*

Chapter 3

Topological Dispersing Regularizations

In this chapter, we introduce the concept of dispersing to generalize orthogonality and to circumvent overcomplete issue and also introduce new regularizations **PH0** and **MST** based on this concept of dispersing.

In section **3.1.1**, we illustrate the fact that our dispersing concept on a sphere can generalize orthogonality. In subsection **3.1.2**, we mention potential energy of points in a compact metric space using physics. Then we define the concept of evenly dispersed state of points in real projective space as the state with the lowest potential energy of given points. With this concept, we prove that orthogonality of points on a sphere is equivalent to an evenly dispersed state of points mapped in real projective space if the number of points is less than or equal to the dimension of the ambient space of the sphere. We also define adjacent distance and find some lower and upper bounds for their expected values. In section **3.2**, we introduce a method using persistent homology and the concept of dispersing. To this end, we

first briefly review Čech and Vietoris-Rips complexes and persistent homology and we define α-weighted lifetime sum. The lifetime sum obtained on the 0-th persistent homology is used to construct **PH0** regularization, which disperse row vectors of a kernel matrix.

In section **3.3**, we review the minimum spanning tree and introduce **MST** regularization using minimum spanning tree. To use **MST**, we first construct an edge weighted graph from a kernel matrix, by considering row vectors as points and distances between any two row vectors as edge weights. Then we disperse the edges of the minimum spanning tree by making its edge weight sum larger. An interest thing is that this edge weight sum is just a constant multiple of the 1-weighted life time in **PH0**. So **PH0** and **MST** are actually equivalent.

3.1 Evenly Dispersed State

3.1.1 Dispersing Vectors on Sphere

Previous works aim to enforce convolutional kernel matrices as orthogonal as possible. This condition is closely related to feature diversity and energy conservation. Suppose that a kernel matrix W as in (2.1) is orthogonal matrix. Then this means that row vectors of W are quite different each others and makes output features diverse. Also W can be seen as an isometry transform, this implies that $\|Wx\| = \|x\|$ and thus W converse the energy of the input.

In practice, orthogonality cannot occur in an overcomplete case where there are more row vectors in an $m \times n$ matrix W than the dimension of the ambient space of row vectors, i.e., $m > n$. Therefore, with simple methods such as soft orthogonality in subsection **2.2.1**, the matrix cannot converge. To avoid this overcompleteness

problem, various methods have been devised. DSO in subsection **2.2.2** detour the problem by using with the transpose of kernel matrix, SRIP in subsection **2.4.2** uses a generalized isometry concept for applying to general form of kernel matrix and OCNN in subsection **2.5.1** proves that their regularization does the same backpropogation process regardless of whether the matrix is transposed or not.

To avoid the overcompleteness problem, we introduce dispsersed state which is similar to the uniform distribution. The most basic way to perform dispersing is to widen a distance between two distinct row vectors which have the nearest distance. This method is used in MC as in subsection **2.3.1**. However, if a weight is regularized one by one in this way, the impact is too weak. So it would be much more efficient to spread out appropriately many vectors at once.

If we just make the vectors uniformly distributed, there may be antipodal or almost antipodal row vectors in W. These two antipodal row vectors induces the same output features with only opposite sign. These results oppose the diversification of output features, one of the reasons of convolutional kernel regularization. To solve this problem, we consider the union of the set of row vectors of W and its all antipodal vectors and make vectors of this set to be uniformly distributed on $\mathbb{S}^n$. We call this operation **Dispersing on** $\mathbb{P}^n$.

Dispersing on $\mathbb{P}^n$ can also be seen as a generalization of orthogonality on $\mathbb{S}^n$. For a brief explanation, making two vectors on $\mathbb{S}^1$ orthogonal is equivalent to adding their two antipodal vectors and dispersing these vectors as far as possible. A detailed discussion on dispsersed state will be addressed in subsection **3.1.2**.

When dispersing is applied to a kernel matrix W, the property of energy conserving of orthogonality is lost since the norm of row of W is one but these row vectors may not be orthogonal. For overcomplete case, output energy is larger than

input energy and for undercomplete case vice versa. However, in the case for high dimensional ambient space and overcomplete matrix, for a fixed row vector w_i of W, dispersing makes other row vectors lie in near a hyperplane orthogonal to w_i. This observation suggests that the extent to increasing energy is not much so this regularization is almost energy perserving.

3.1.2 Evenly Dispersed State in the Real Projective Spaces

The state in which finite points are uniformly distributed in a compact metric space can be interpretable in terms of physics. To be specifically, consider two protons with positive charge q_1 and q_2 on a compact metric space $\mathbf{S}$ with a metric d. The force F acting between these two protons is of the form

$$F = k\frac{q_1 q_2}{r_{12}^2}, \tag{3.1}$$

where k is a constant and r_{12} is a distance between two protons **[15]**.

If a finite set of protons $\{p_i\}_{i=1}^m$ of $\mathbf{S}$ exists, the potential energy of these can be obtained as follows. For simplicity, suppose $k = 1$ and all charges of protons are 1.

$$\begin{aligned}
\text{potential energy of } \{p_i\}_{i=1}^m &= \sum_{j\neq i} \int_{\epsilon}^{d(p_i,p_j)} F(r)dr \\
&= \sum_{j\neq i} \int_{\epsilon}^{d(p_i,p_j)} \frac{-1}{r^2}dr \\
&= \sum_{j\neq i} \left(\frac{1}{d(p_i,p_j)} - \frac{1}{\epsilon}\right)
\end{aligned}$$

Since we only care about the relative variation in potential energy, we may redefine

the potential energy as

$$\mathrm{PE}(\{p_i\}_{i=1}^m) = \sum_{j \neq i} \frac{1}{d(p_i, p_j)}.$$

The state with the lowest potential energy is meaningful because this is an equilibrium state.

Definition 2. *Let* $\mathbf{S}$ *be a compact metric space and* d *its metric. We call* m *distinct points in* $\mathbf{S}$ *are* ***evenly dispersed state*** *if, when we look at the points as protons with charge 1, these have the lowest potential energy, or equivalently are on the state of equilibrium.*

The space we are interested in in this subsection is the real projective space $\mathbb{P}^n$. In order to talk about the evenly dispersed state in $\mathbb{P}^n$, a metric must be defined first. $\mathbb{P}^n$ is the topological space of lines passing through the origin in $\mathbb{R}^{n+1}$. Equivalently $\mathbb{P}^n$ can also be formed by identifying antipodal points of the unit n-sphere, $\mathbb{S}^n$. We define a metric $\Theta : \mathbb{P}^n \times \mathbb{P}^n \to [0, \frac{\pi}{2}]$ in $\mathbb{P}^n$. For $\mathrm{v}, \mathrm{w} \in \mathbb{P}^n$, $\Theta(\mathrm{v}, \mathrm{w})$ is the angle in $[0, \frac{\pi}{2}]$ between v and w when these are viewed as lines passing through the origin in $\mathbb{R}^{n+1}$.

Proposition 2. $\Theta : \mathbb{P}^n \times \mathbb{P}^n \to [0, \frac{\pi}{2}]$ *is indeed a metric in* $\mathbb{P}^n$.

Proof. Let $\pi : \mathbb{S}^n \to \mathbb{P}^n$ be a natural quotient map, $\theta : \mathbb{S}^n \times \mathbb{S}^n \to \mathbb{R}$ the round metric on $\mathbb{S}^n$ and $\pi^{-1}(\mathrm{v}) = \{\mathrm{v}_1, \mathrm{v}_2\}, \pi^{-1}(\mathrm{w}) = \{\mathrm{w}_1, \mathrm{w}_2\}$ for $\mathrm{v}, \mathrm{w} \in \mathbb{P}^n$ respectively. Then Θ can be equivalently redefined as follows.

$$\Theta(\mathrm{v}, \mathrm{w}) = \min\{\theta(\mathrm{v}_1, \mathrm{w}_1), \theta(\mathrm{v}_1, \mathrm{w}_2)\} \tag{3.2}$$

Axioms of metric are satisfied obviously except triangle inequality. Let's show

that

$$\Theta(\mathrm{v}, \mathrm{w}) \leq \Theta(\mathrm{v}, \mathrm{u}) + \Theta(\mathrm{u}, \mathrm{w}). \tag{3.3}$$

Since the range of Θ is $[0, \frac{\pi}{2}]$ by the definition, if the right hand sum in (3.3) is greater than or equal to $\pi/2$, the inequality holds trivially. Otherwise, there are $\mathrm{v}_1 \in \pi^{-1}(\mathrm{v})$, $\mathrm{u}_1 \in \pi^{-1}(\mathrm{u})$ and $\mathrm{w}_1 \in \pi^{-1}(\mathrm{w})$ such that $\theta(\mathrm{v}_1, \mathrm{u}_1) < \frac{\pi}{2}$, $\theta(\mathrm{u}_1, \mathrm{w}_1) < \frac{\pi}{2}$ and $\theta(\mathrm{v}_1, \mathrm{w}_1) < \frac{\pi}{2}$. By the triangle equality of the round metric d,

$$\theta(\mathrm{v}_1, \mathrm{w}_1) \leq \theta(\mathrm{v}_1, \mathrm{u}_1) + \theta(\mathrm{u}_1, \mathrm{w}_1) \tag{3.4}$$

is hold. From this we can derive the inequality (3.3). □

Thus the potential energy of a finite set of points $\{\mathrm{v}_i\}$ of $\mathbb{P}^n$ is

$$\sum_{j \neq i} \frac{1}{\Theta(\mathrm{v}_i, \mathrm{v}_j)}. \tag{3.5}$$

An interesting fact about evenly dispersed state in $\mathbb{P}^n$ is that it is a concept of extending orthogonality in $\mathbb{S}^n$. A set of points of $\mathbb{S}^n$ is called orthogonal if, when we view these points as vectors in $\mathbb{R}^{n+1}$, the set of these vectors is orthogonal.

Theorem 2. *Let $m \leq n+1$ and $\{\mathrm{v}_1, \ldots \mathrm{v}_m\}$ be a set of $\mathbb{S}^n$. Then $\mathrm{v}_1, \ldots \mathrm{v}_m$ are orthogonal in $\mathbb{S}^n$ if and only if $\pi(\mathrm{v}_1), \ldots, \pi(\mathrm{v}_m)$ are evenly dispersed in $\mathbb{P}^n$.*

Proof. The potential energy of $\pi(\mathrm{v}_1), \ldots, \pi(\mathrm{v}_m)$ is

$$\mathrm{PE}(\{\pi(\mathrm{V}_i)\}) = \sum_{j \neq i} \frac{1}{\Theta(\pi(\mathrm{v}_i), \pi(\mathrm{v}_j))}.$$

Since the value of Θ is less or equal to $\pi/2$,

$$\begin{aligned}\mathrm{PE} &\geq \sum_{j\neq i} \frac{1}{\pi/2} \\ &= \binom{m}{2}\frac{2}{\pi} \\ &= \frac{m(m-1)}{\pi}.\end{aligned}$$

The condition $m \leq n+1$ implies that we can find m lines through the origin in $\mathbb{R}^{n+1}$ such that all of the distances between distinct two lines are $\pi/2$, for example by choosing lines from axis lines. Thus the lowest potential energy is exactly $m(m-1)/\pi$.

$$\begin{aligned}&\pi(\mathrm{v}_1), \ldots, \pi(\mathrm{v}_m) \text{ are evenly dispersed in } \mathbb{P}^n \\ \iff\ &\mathrm{PE}(\{\pi(\mathrm{V}_i)\}) = \frac{m(m-1)}{\pi} \\ \iff\ &\Theta(\pi(\mathrm{v}_i), \pi(\mathrm{v}_j)) = \frac{\pi}{2} \text{ for all } i \neq j \\ \iff\ &\theta(\mathrm{v}_i, \mathrm{v}_j) = \frac{\pi}{2} \text{ for all } i \neq j \\ \iff\ &\mathrm{v}_1, \ldots \mathrm{v}_m \text{ are orthogonal in } \mathbb{S}^n\end{aligned}$$

This completes the proof. □

In general, it is not easy to find an evenly dispersed state of points in $\mathbb{P}^n$ or to demonstrate that a set of points is dispersed or not because evenly dispersed state of points may have an irregular distribution for more than $n+1$ points in $\mathbb{P}^n$. Instead, with randomly distributed points in $\mathbb{P}^n$, we can construct an approximately dispersed set of points with adjacent distances (**Definition 3**). A method to obtain

an approximately evenly dispersed state of points is to increase the minimum of all adjacent distances of given points.

Definition 3. *Let* $\mathrm{v}_1, \ldots, \mathrm{v}_m$ *be some points in a metric space* $\mathbf{S}$*. Then, for any* v_i*, the* ***adjacent distance*** *of* v_i *means the minimum of a set of distances from* v_i *to* v_j*,* $j \neq i$*.*

For a finite set of points in $\mathbb{P}^n$, we can find upper bounds and lower bounds for the expected value of adjacent distances of it. An obvious upper bounds is $\pi/2$ and another upper bound is obtained by geometrical analyses.

Theorem 3. *Let* $\mathrm{v}_1, \ldots, \mathrm{v}_m$ *be a set of points in* $\mathbb{P}^n$*. If* $ad(\mathrm{v}_i)$ *indicates the adjacent distance of* v_i*, then*

$$\mathbb{E}\left[\sin\left(\frac{ad(\mathrm{v}_i)}{2}\right)\right] \leq \sqrt[n]{\frac{S_n}{2mV_n}}, \tag{3.6}$$

where S_n *is the surface area of* n*-sphere* $\mathbb{S}^n$ *and* V_n *is the volume of unit* n*-ball* $\mathbb{B}^n$*.*

Proof. Define

$$\mathrm{B}(\mathrm{v_i}, r) := \{\mathrm{w} \in \mathbb{P}^n | \Theta(\mathrm{v_i}, \mathrm{w}) < r\}.$$

Since $\mathbb{P}^n$ is the quotient space of $\mathbb{S}^n$ by identifying antipodal points and Θ is induced by the metric θ of $\mathbb{S}^n$, the area of $\mathbb{P}^n$ is half of that of $\mathbb{S}^n$, i.e., $S_n/2$. The collection of sets $\mathrm{B}(\mathrm{v}_i, \mathrm{ad}(\mathrm{v}_i)/2)$ are disjoint. Hence the sum of areas $\mathrm{Area}(\mathrm{B}(\mathrm{v}_i, \mathrm{ad}(\mathrm{v}_i)/2))$ of $\mathrm{B}(\mathrm{v}_i, \mathrm{ad}(\mathrm{v}_i)/2)$ is less than $S_n/2$.

$\mathrm{Area}(\mathrm{B}(\mathrm{v}_i, \mathrm{ad}(\mathrm{v}_i)/2))$ is hard to be explicitly formulated. Thus we use an explicitly represented lower bound to replace $\mathrm{Area}(\mathrm{B}(\mathrm{v}_i, \mathrm{ad}(\mathrm{v}_i)/2))$. Since $\mathrm{ad}(\mathrm{v}_i)/2$ is less than or equal to $\pi/4$, $\mathrm{B}(\mathrm{v}_i, \mathrm{ad}(\mathrm{v}_i)/2)$ can be embedded in $\mathbb{S}^n$ and v_i can be viewed as a vector in $\mathbb{R}^{n+1}$. Projecting $\mathrm{B}(\mathrm{v}_i, \mathrm{ad}(\mathrm{v}_i)/2)$ onto a hyperplane orthogonal to v_i, we obtain an n-ball with radius $\sin(\mathrm{ad}(\mathrm{v}_i)/2)$. Therefore we get the

following inequalities.

$$\begin{aligned} S_n/2 &\geq \sum_{i=1}^{m} \mathrm{Area}(\mathrm{B}(\mathrm{v}_i, \mathrm{ad}(\mathrm{v}_i)/2)) \\ &\geq \sum_{i=1}^{m} \sin^n(\mathrm{ad}(\mathrm{v}_i)/2) V_n \\ &\geq m V_n \mathbb{E}\left[\sin^n(\mathrm{ad}(\mathrm{v}_i)/2)\right] \end{aligned}$$

Using Jensen's inequality,

$$\mathbb{E}\left[\sin(\mathrm{ad}(\mathrm{v}_i)/2)\right]^n \leq \mathbb{E}\left[\sin^n(\mathrm{ad}(\mathrm{v}_i)/2)\right] \leq \frac{S_n}{2mV_n}$$

Finally, we obtain the inequality (3.6). □

If the number m of points of $\mathbb{P}^n$ is sufficiently larger than the n, we also have rough upper and lower bounds in another way.

Theorem 4. *Conditions are the same as the previous theorem and additionally we assume that* $\mathrm{v}_1, \ldots, \mathrm{v}_m$ *are evenly dispersed and* m *is sufficiently large so that, for a domain of which diameter is as twice much as the mean of adjacent distances in an evenly dispersed state, the curvature of the domain is almost zero. Then*

$$\sqrt[n]{\frac{(n+1)S_n}{2mV_n}} \leq \mathbb{E}\left[ad(\mathrm{v}_i)\right] \leq \frac{1}{h_n}\sqrt[n]{\frac{(n+1)S_n}{2mV_n}}, \tag{3.7}$$

where h_n *is the height from a vertex of a unit regular* n*-simplex to the opposite facet.*

Proof. Since the volume of $\mathbb{P}^n$ is $S_n/2$, the volume of the domain occupied by each

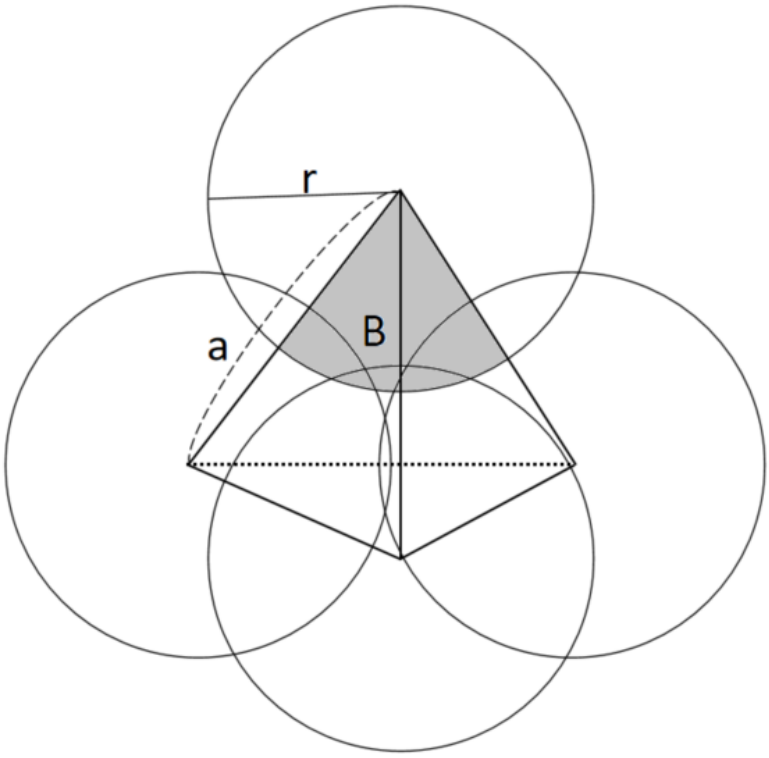

Figure 3.1: Average shape of the simplices.

of m points is $S_n/(2m)$ on average and there is a radius r such that

$$\frac{S_n}{2m} = r^n V_n. \tag{3.8}$$

On the other hand, fixing the m points as vertices, we perform triangulation to the $\mathbb{P}^n$ with n-simplices as regular as possible. The average shape of the simplices will approximately be a regular n-simplex as show in **Figure 3.1**. The edge length of this average regular n-simplex is the average adjacent distance. To find out this distance, we draw balls with a radius r around vertices of the average n-simplex. In **Figure 3.1**, the shaded domain is the area that a sphere passes through the simplex and B is its volume. Then we have a relational expression

$$(n+1)B = \mathbb{E}\left[\mathrm{ad}(\mathrm{v}_i)\right]^n X_n \tag{3.9}$$

, where $\mathbb{E}\left[\mathrm{ad}(\mathrm{v}_i)\right]$ is the expected adjacent distance, i.e. the edge length of the average n-simplex, and X_n is the volume of the unit regular n-simplex.

The volume B is hard for us to calculate, so instead of B we consider lower and upper bounds of B like in **Figure 3.2**. A is the volume of regular n-simplex with edge length r and C is the volume of regular n-simplex with height r. Therefore we have

$$A = r^n X_n, \quad C = \left(\frac{r}{h_n}\right)^n X_n. \tag{3.10}$$

Combining (3.9) with (3.10),

$$
\begin{aligned}
(n+1)\frac{A}{X_n} \leq \mathbb{E}\left[\mathrm{ad}(\mathrm{v}_i)\right]^n = (n+1)\frac{B}{X_n} \leq (n+1)\frac{C}{X_n} \\
(n+1)\frac{r^n X_n}{X_n} \leq \mathbb{E}\left[\mathrm{ad}(\mathrm{v}_i)\right]^n \leq (n+1)\frac{(r/h_n)^n X_n}{X_n} \\
(n+1)r^n \leq \mathbb{E}\left[\mathrm{ad}(\mathrm{v}_i)\right]^n \leq \frac{(n+1)r^n}{h_n^n}
\end{aligned}
$$

From the (3.8), we have

$$
r^n = \frac{S_n}{2mV_n}, \tag{3.11}
$$

and we result in the inequality (3.7).

$$
\begin{aligned}
(n+1)\frac{S_n}{2mV_n} \leq \mathbb{E}\left[\mathrm{ad}(\mathrm{v}_i)\right]^n \leq \frac{(n+1)}{h_n^n}\frac{S_n}{2mV_n} \\
\sqrt[n]{\frac{(n+1)S_n}{2mV_n}} \leq \mathbb{E}\left[\mathrm{ad}(\mathrm{v}_i)\right] \leq \frac{1}{h_n}\sqrt[n]{\frac{(n+1)S_n}{2mV_n}}
\end{aligned}
$$

□

3.2 Persistent Homology Regularization

Persistent homology is a method for extracting meaningful topological features by analysing various spacial resolutions of a topological space **[6]**. Specifically, a filtration that emerges along t (usally t is considered as time while in Čech and Vietoris-Rips complexes radius r is used) in space is constructed, and uses the life of homological basis obtained by observing the birth and collapse of the basis of homology as t goes by as an analysis tool of the space. Typically, persistent homologies using Čech complex or Vietoris-Rips complex for point cloud is widely

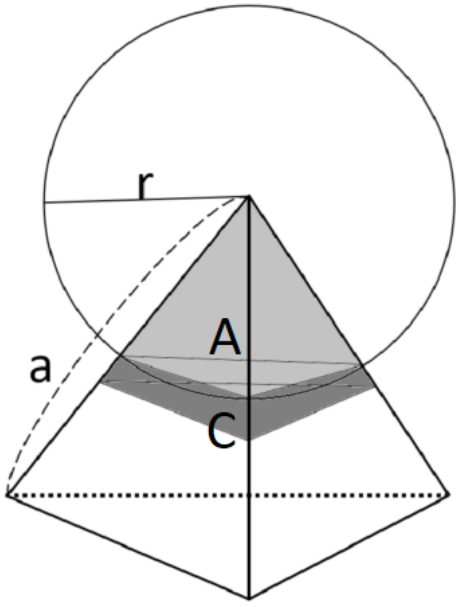

Figure 3.2: The lower bound area (A) and the uppder bound area (C) of the traversed volume (B).

used.

3.2.1 Čech and Vietoris-Rips Complexes

Nerve

The nerve of the covering $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ is the simplicial complex $N(\mathcal{U})$ which is constructed as the following.

- the indexing set A which is also called vertex set
- $\{\alpha_0, \alpha_1, ..., \alpha_k\}$ is a k-simplex in $N(\mathcal{U})$ if and only if $U_{\alpha_0} \cap U_{\alpha_1} \cap ... \cap U_{\alpha_k} \neq \emptyset$

Čech complexes

Let S be a finite set of points in $\mathbb{R}^d$ and write $B_x(r) = x + r\mathbb{B}^d$ for the closed ball with center x and radius r. The Čech complex of S and r is the nerve of these balls replacing balls with the center of each ball, that is,

$$\check{\text{C}}\text{ech}(r) = \{\sigma \subseteq S | \bigcap_{x \in \sigma} B_x(r) \neq \emptyset\}. \tag{3.12}$$

As the radius increases, more overlapping relationship are created retaining the previous ones. Hence $\check{\text{C}}\text{ech}(r_0) \subseteq \check{\text{C}}\text{ech}(r_1)$ if $r_0 \leq r_1$. As we continuously increase the radius, from 0 to ∞, we get a discrete family of nested Čech complexes **[10]**.

Vietoris-Rips complexes

Checking that a subcollection of S is in a Čech complex requires a lot of computation costs if the cardinality of the subcollection is large. To alleviate this shortcomings of Čech complex, Vietoris-Rips complexes appear. In Vietoris-Risp complexe,

to judge that a subcollection belongs the complex, we need to only check whether all pairs of a subcollection have distances less than or equal to $2r$. This condition is equivalent to checking whether the diameter of a subcollection is less than or equal to $2r$.

$$\mathrm{VR}(r) = \{\sigma \subseteq S | \mathrm{diam}\sigma \leq 2r\} \tag{3.13}$$

3.2.2 Persistent Homology

Enumerating Čech or Vietoris-Rips complexes K_r as radius r increases yields a continuous sequence of complexes. When $r_0 < r_1$, a simplex of r_0 complex is also a simplex of r_1 complex because overlapping balls still overlap even if their radius increases. Thus these sequences are filtrations.

$$K_{r_0} \subseteq K_{r_1} \text{ if } r_0 \leq r_1 \tag{3.14}$$

In addition, the emergence of a new simplex appears at discrete radius, so continuous filtration can be seen as a finite filtration.

$$S = K_0 \subset K_1 \subset \cdots \subset K_n \tag{3.15}$$

For every $i \leq j$, we have an inclusion map from K_i to K_j, and therefore an induced homomorphism between homologies, i.e., $f_p^{i,j} : \mathrm{H}_p(K_i) \to \mathrm{H}_p(K_j)$, for each dimension p. The filtration (3.15) therefore corresponds to a sequence of homology groups connected by homomorphisms for each dimension p **[10]**.

$$\mathrm{H}_p(S) = \mathrm{H}_p(K_0) \to \mathrm{H}_p(K_1) \to \cdots \to \mathrm{H}_p(K_n) \tag{3.16}$$

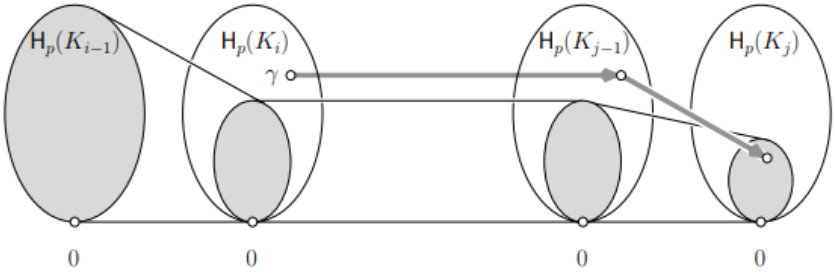

Figure 3.3: γ is born at i and dies at j. The life time of γ is $j - i$ **[10]**.

With this induced homomorphisms, we define persistent homology groups and persistent Betti numbers as follows.

Definition 4. *The* ***p-th persistent homology groups*** *are the images of the homomorphisms induced by inclusion,* $H_p^{i,j} = imf_p^{i,j}$. *The corresponding* ***p-th persistent Betti numbers*** *are the ranks of these groups,* $\beta_p^{i,j} = rank\ H_p^{i,j}$

Let γ be a class in $\mathrm{H}_p(K_i)$. We say it is **born**(**birth**) at K_i if $\gamma \notin \mathrm{H}_p^{i-1,i}$. If γ is born at K_i(or briefly i) and $f_p^{i,j-1}(\gamma) \notin \mathrm{H}_p^{i-1,j-1}$ but $f_p^{i,j}(\gamma) \in \mathrm{H}_p^{i-1,j}$, then we say it is **dies**(**death**) at K_j(or briefly j). If γ is born at K_i and dies at K_j, the difference of indexes $j - i$ is called the **persistence** or **life time** of γ. See **Figure 3.3**. In sequence (3.16), the birth and death analysis can be restricted to generators. The set of generators in the sequence (3.16) is call p-th persistent homology of K_* and denoted by $PH_p(K_*)$. For the next subsection, we define lifetime sum as follows **[3]**.

Definition 5 (α-Weighted Lifetime Sum)**.** *For a finite set* S*, the weighted* p*-th homology lifetime sum is defined as follows.*

$$E_\alpha^p(S) = \sum_{\gamma \in PH_p(K_*)} I(\gamma)^\alpha, \tag{3.17}$$

where $PH_p(K_*)$ is the p-th persistent homology of a filtration K_* and $I(\gamma)$ is the lifetime of γ.

3.2.3 PH_0 Regularization

In this subsection, we introduce a method of using persistent homology to disperse the kernel matrix W on a convolutional layer on $\mathbb{P}^n$ as mentioned in 3.1. To consider the dispersed state of points in a projective space, we add their antipodal points. Let $\tilde{W}$ be the concatenation of W and $-W$ along with rows. Consider the row vectors of $\tilde{W}$ as a point cloud data in the ambient space $\mathbb{R}^{n+1}$. With this point cloud, we have Vietoris-Rips complexes $\mathrm{VR}(r)$ and a filtration K_* with respect to radius r. Therefore we obtain the 0-th persistent homolgy $\mathrm{PH}_0(K_*)$ of the set of row vectors of $\tilde{W}$.

Generators of 0-th persistent homology $\mathrm{PH}_0(K_*)$ correspond to row vectors of $\tilde{W}$ one by one and the lifetime of a generator tells us the information about the shortest distance to other row vectors. Hence, if we regularize the lifetime sum of generators of $\mathrm{PH}_0(K_*)$ to increase, row vectors of $\tilde{W}$ will be dispersed.

$$-\lambda_2 \sum_{\gamma \in \mathrm{PH}_p(K_*)} I(\gamma) \tag{3.18}$$

However, with the regularizer (3.18) alone, row vectors of $\tilde{W}$ may be infinitely distant. Thus we need a regularizer that binds these vectors on $\mathbb{S}^n$.

$$\lambda_1 \sum_{w:\text{row vectors of } \tilde{W}} \left| \|w\|^2 - 1 \right| \tag{3.19}$$

Finally, combining (3.18) and (3.19), we obtain the following regularization.

$$\lambda_1 \sum_{w:\text{row vectors of } \tilde{W}} \left| \|w\|^2 - 1 \right| - \lambda_2 \sum_{\gamma \in \mathrm{PH}_0(K_*)} I(\gamma) \tag{3.20}$$

We call the regularizing method with (3.20) $\mathrm{PH}_\mathbf{0}$ **regularization**.

The advantages of PH_0 regularization are that first all row vectors are involved in the dispersing update and second the number of these dispersing updates is minimal. However, there is a limit to applying in deep structured nets because calculating $\mathrm{PH}_0(K_*)$ takes a lot of time.

3.3 Minimum Spanning Tree Regularization

The minimum spanning tree is a spanning tree in which the sum of the edges weights of it is minimum. Minimum spanning tree is related to the minimum cost of connecting all points.

3.3.1 Minimum Spanning Tree

An edge weighted graph is a graph in which each edge is given a numerical weight. A tree is an undirected graph in which any two vertices are connected by exactly one path. A spanning tree is a tree which connects all the vertices of the graph. The minimum spanning tree (MST) of an edge weighted graph is a spanning tree whose sum of the edge weights is minimum [**11**].

For specific example, we set vertices as all points in a point cloud and edges as all possible edges connecting two distinct vertices. If we put a weight on each edge as the shortest distance between the vertices of edge we have an edge weighted graph. Here, if we think of vertices, edges and weights as cities, roads and the shortest distances between two corresponding cities respectively, the minimum spanning tree

of this edge weighted graph suggests the method of constructing roads by which all cities are connected. Also the sum of the edge weights of minimum spanning tree tells us the minimum cost connecting all the vertices.

One of the interesting things about minimum spanning tree is that the sum of edges of minimum spanning tree and the life time sum of 0-th persistent homology is closely related.

Theorem 5. *Let S be a finite point set contained in a bounded metric space and $PH_0(K_*)$ the 0-th persistent homology of the Čech or Vietoris-Rips complex of S. Then there is a bijection between the edges of the minimum spanning tree of the distance edge weighted graph with vertices S and generators with finite life time in $PH_0(K_*)$ **[18].***

A sketch of the proof. When we increase the radius to obtain 0-th persistent homology, a generator of PH_0 dies when two distinct components meet at some radius r. At this time, there are two vertices of centers of r-balls that touch. We pick these edges connecting these two vertices to construct a minimum spanning tree. If the cardinality of S is N, the number of fusions of components is $N-1$. Since the number of edges of a minimum spanning tree is also $N-1$, there is a bijection between finite life time generators of PH_0 and edges of the minimum spanning tree. □

The theorem **5** actually implies more. In other words, the births of all generators of PH_0 are zeros and when a generator dies, the death is r which is the radius of balls at that instance. At the same time, the weight of the corresponding edge of minimum spanning tree becomes $2r$. Therefore, we have the following corollary.

Corollary 1. *The sum of weights of minimum spanning tree is twice as much as 1-weighted life time sum of PH_0, i.e.,*

$$\sum_{e \in MST} weight(e) = 2 \sum_{\gamma \in PH_0} death(\gamma). \tag{3.21}$$

3.3.2 MST Regularization

In this subsection, we introduce a method of using minimum spanning tree to disperse the convolutional kernel matrix W on $\mathbb{P}^n$. As in subsection 3.2.3, we think the point cloud consisting of row vectors of $\tilde{W}$. We consider each pair of distinct row vectors of $\tilde{W}$ as an edge with weight of distance of them. This give us an edge weighted graph and we can obtain a minimum spanning tree of this graph.

If we regularize $\tilde{W}$ so that the sum of weights of a minimum spanning tree of $\tilde{W}$ become larger, row vectors of $\tilde{W}$ will be dispersing. Therefore, adding the regularizer (3.19) that enforcing it on $\mathbb{S}^n$, the final regularizer is as follows.

$$\lambda_1 \sum_{w:\text{row vectors of } \tilde{W}} \left| \|w\|^2 - 1 \right| - \lambda_2 \sum_{e \in MST} weight(e) \tag{3.22}$$

We call this regularizing method **MST regularization**.

In fact, according to the theorem 5 and corollary 1, the two regularization PH_0 and MST are essentially the same. But the MST regularization is much more faster than PH_0 regularization.

Chapter 4

Evenly Angle Dispersing Regularizations

In this chapter, we first demonstrate that prior regularizations based on inner product have some deficiencies, and then introduce our proposed evenly angle dispersing regularizations. In subsection **4.1.1**, we explains that regularizing loss based on inner product is actually a method of regularizing cosine values of two row vectors of kernel or convolution matrix. These method, however, might not widen sufficiently the angles of two vectors if these angles are close to 0. In subsection **4.2.1**, as an alternative to prior regularizations with the above mentioned problem, we introduce **EADK** regularization, a method that regularize directly the angles of two vectors of kernel matrix. In subsection **4.2.2**, we introduce **EADC** regularization which deals with convoluton matrix rather than kernel matrix, which is motivated by **OCNN** regularization.

4.1 Analysis of Soft Orthogonality

4.1.1 Analysis of Soft Orthogonality

Existing regularizations for orthogonality of convolutional kernel matrix W, although the specific details are different, make the angle of any two distinct row or column vectors of W be as close to $\pi/2$ as possible. Most of them use methods enforcing the inner product of two vectors to be 0. Assuming the norms of these two vectors are one, the inner product of the two vectors is in fact cosine value of the angle between the two vectors. However, it is necessary to examine whether this approach is really appropriate as criterion that makes the objective angle to be orthogonal.

Considering the situation where protons are dispersing, it would be natural for us that the amount of change in the angle is greater as the angle of two vectors are close to 0, and smaller as it approaches $\pi/2$. Let's find out through an example using SO regularization whether the update of angle is going as we wish in the way reducing cosine value to 0.

If a convolutional kernel matrix W is given by

$$W = \begin{bmatrix} u \\ v \end{bmatrix}, \tag{4.1}$$

where $u, v \in \mathbb{R}^2$, then the regularization formula is

$$\lambda \cdot \|WW^T - I\|_F^2 = \lambda \cdot \left\| \begin{bmatrix} \|u\|^2 - 1 & < u, v > \\ < v, u > & \|v\|^2 - 1 \end{bmatrix} \right\|_F^2$$

$$= \lambda \cdot \{(\|u\|^2 - 1)^2 + (\|v\|^2 - 1)^2 + 2 < u, v >^2\}.$$

For simplicity, let $u = (0, 1)$ be constant and only $v = (a, b)$ is variable. Then our regularizer L is

$$L(v) := \lambda \cdot \{(a^2 + b^2 - 1)^2 + 2b^2\}. \tag{4.2}$$

In the expression (4.2), $(a^2 + b^2 - 1)^2$ is the term causes the norm of v to be 1, and $2b^2$ is the term that make the cosine value of the angle between u and v smaller. To find out the effect of the preceding term on the change of the angle, let's calculate the gradient of the term.

$$\nabla_v (a^2 + b^2 - 1)^2 = \begin{pmatrix} 4a(a^2 + b^2 - 1) \\ 4b(a^2 + b^2 - 1) \end{pmatrix}$$

$$= 4(a^2 + b^2 - 1) \begin{pmatrix} a \\ b \end{pmatrix}$$

The gradient above is a constant multiple of v and so the update of v using this gradient does not affect the angle between u and v. Therefore, it is sufficient to only look at the second term b^2 to analyze the amount of change in the angle by gradient update.

Suppose that the current value of v is (a_0, b_0), $a_0^2 + b_0^2 = 1$ and $b_0 = \cos\theta_0$ for

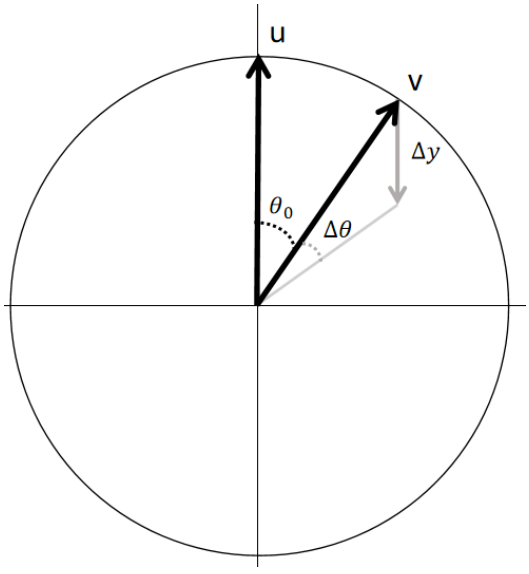

Figure 4.1: Angle update by soft orthogonality method.

some $\theta_0 \in (0, \pi/2)$. Since the gradient of b^2 is $(0, 2b)$, the update only occur in the direction of the second coordinate, see **Figure 4.1**. This update variation in second coordinate Δy is $2\lambda b$, which is a constant multiple of b.

$$\|\Delta y\| = 2\lambda b_0 = 2\lambda \cos\theta_0 \tag{4.3}$$

For these gradient updates to be stable, $\|\Delta y\|$ should be less than b_0. Thus λ should be in the open interval $(0, 1/2)$. With these λ, the parameters $\theta_0, \Delta\theta$ and Δy have the following relation.

$$\sin\theta_0 = (\cos\theta_0 - \|\Delta y\|)\tan(\theta_0 + \Delta\theta) \tag{4.4}$$

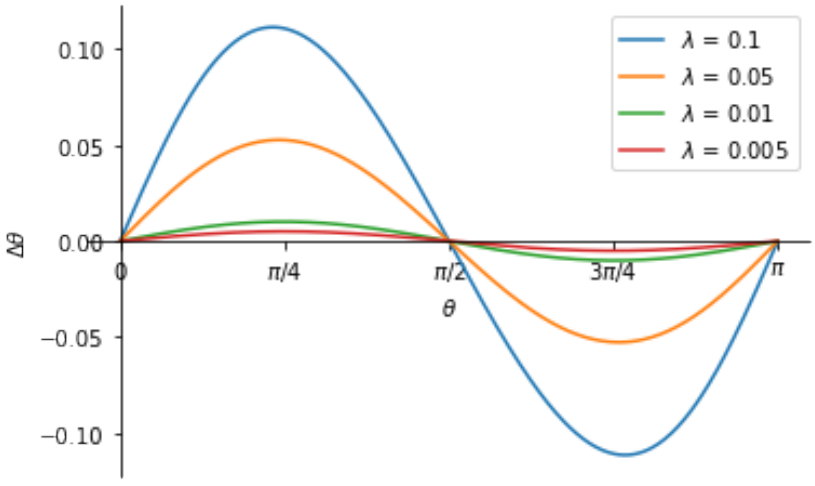

Figure 4.2: $\Delta\theta$ with respect to $\theta \in [0, \pi]$ obtained by the relation (4.8) for various λ.

Rewriting the equation (4.4) with respect to $\Delta\theta$, we have, if $\theta_0 \in (0, \pi/2]$,

$$\Delta\theta = \arctan\left(\frac{\sin\theta_0}{\cos\theta_0 - \|\Delta y\|}\right) - \theta_0 \tag{4.5}$$

$$= \arctan\left(\frac{\sin\theta_0}{\cos\theta_0 - 2\lambda\cos\theta_0}\right) - \theta_0 \tag{4.6}$$

$$= \arctan\left(\frac{1}{1-2\lambda}\tan\theta_0\right) - \theta_0 \tag{4.7}$$

and if $\theta_0 \in (\pi/2, \pi)$

$$\Delta\theta = \arctan\left(\frac{1}{1-2\lambda}\tan\theta_0\right) - \theta_0 + \pi. \tag{4.8}$$

Figure 4.2 is graphs that represents $\Delta\theta$ functions of θ for various λ. As seen in the **Figure 4.2**, if λ is small, the angular change $\Delta\theta$ is relatively insignificant for small θ compared to medium size θ, so two vectors with small intermediate

angle might not be far enough. Larger λ shows larger angular changes for small intermediate angle as we desire. However, the overall scale may increase, which may interfere with learning models. To solve this problem, it is necessary to first regularize with large λ and gradually reduce the λ.

Still, it is not satisfactory that the skewness of the graphs changes according to the λ because it confines the search for the optimal λ. Hence we propose new regularizers that complement these problems in the next section.

4.2 Evenly Angle Dispersing Regularizations

4.2.1 Evenly Angle Dispersing Regularization with Kernel Matrix

In subsection **4.1.1**, we find out that methods based on soft orthogonality tend to widen the intermediate angle θ of two vectors relatively less when θ is very small. However, we hope that the closer θ is to 0, the greater the angular update $\Delta\theta$ because this leads to a faster and more stable orthogonality of vectors. A regularizer which updates as we wish is in the form of a function with θ itself as a parameter. We analyze the simplest form of such function,

$$L(\theta) = \lambda \left(\theta - \frac{\pi}{2}\right)^2. \tag{4.9}$$

To make this analysis easier to deal with, we assume that the circumstance is like in subsection **4.1.1** and $a > 0$. First, differentiate L with respect to v.

$$\nabla_{\mathrm{v}} L = 2\lambda(\theta - \frac{\pi}{2})\nabla_{\mathrm{v}}\theta \tag{4.10}$$

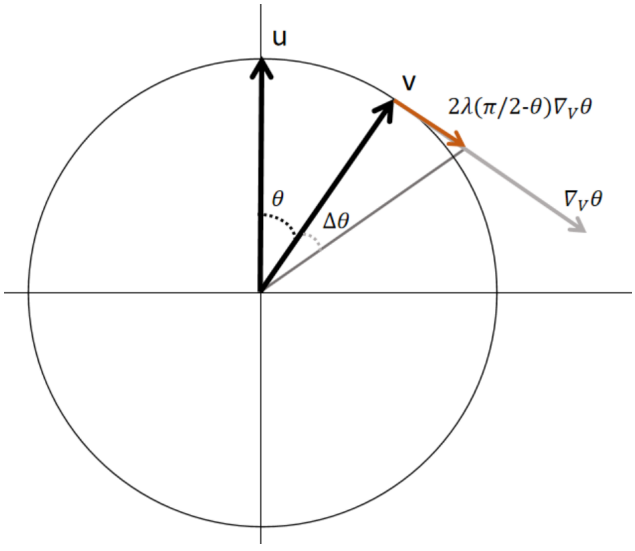

Figure 4.3: Update by equation (4.10).

Since $\cos\theta$ is expressed with u and v, i.e.,

$$\cos\theta = \frac{< \mathrm{u}, \mathrm{v} >}{\|\mathrm{u}\|\|\mathrm{v}\|}, \tag{4.11}$$

we get the expression of θ via v, i.e.,

$$\theta = \arccos\left(\frac{b}{r}\right), \tag{4.12}$$

where $r = \sqrt{a^2 + b^2}$. Hence

$$\begin{aligned}
\nabla_{\mathrm{v}}\theta &= \left(\frac{\partial\theta}{\partial a}, \frac{\partial\theta}{\partial b}\right) \\
&= \left(\frac{-1}{\sqrt{1-\frac{b^2}{r^2}}}\frac{-b\frac{a}{r}}{r^2}, \frac{-1}{\sqrt{1-\frac{b^2}{r^2}}}\frac{r-b\frac{b}{r}}{r^2}\right) \\
&= \left(\frac{ab}{\sqrt{a^2r^2}}, \frac{b^2-r^2}{\sqrt{a^2}r^2}\right) \\
&= \left(\frac{b}{r^2}, \frac{-a}{r^2}\right) \\
&= (b, -a)\,,
\end{aligned}$$

and thus $\nabla_{\mathrm{v}}L$ is orthogonal to vector v and the scale of it is proportional to $\pi/2-\theta$, see **Figure 4.3**. This observation leads to the relation with respect to $\Delta\theta$.

$$\tan\Delta\theta = 2\lambda\left(\frac{\pi}{2}-\theta\right)\|\nabla_{\mathrm{v}}\theta\| = 2\lambda\left(\frac{\pi}{2}-\theta\right) \tag{4.13}$$

Rearranging the equation (4.13), we have

$$\Delta\theta = \arctan\left(\lambda\left(\pi - 2\theta\right)\right). \tag{4.14}$$

In order for orthogonality by $\nabla_{\mathrm{v}}L$ to be achieved stably, it is suggested that the sign of second coordinate value of $\mathrm{v} + 2\lambda\left(\pi/2-\theta\right)\nabla_{\mathrm{v}}\theta$ is the same as that of v. This leads to the inequality

$$\left|\tan\left(\frac{\pi}{2}-\theta\right)\right| \geq \left|2\lambda\left(\frac{\pi}{2}-\theta\right)\right|, \tag{4.15}$$

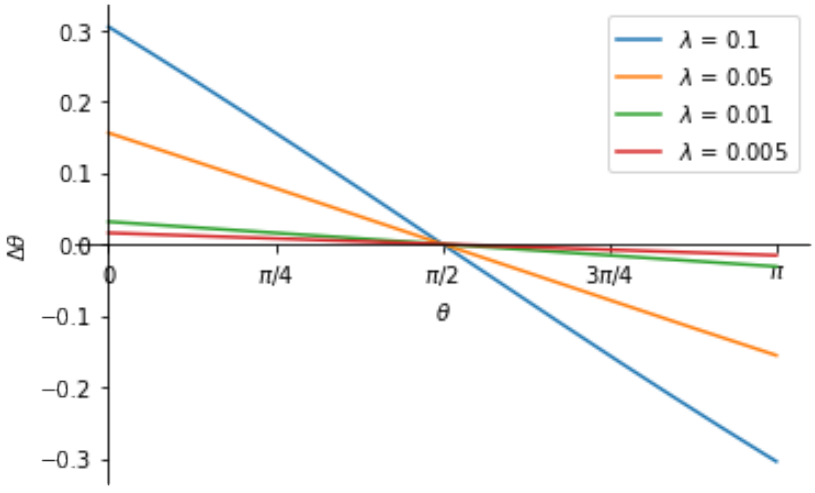

Figure 4.4: $\Delta\theta$ with respect to $\theta \in (0, \pi]$ obtained by the relation (4.14) for various λ.

see **Figure 4.3**. Since for $0 < \theta < \pi$

$$\frac{\tan\left(\frac{\pi}{2} - \theta\right)}{\frac{\pi}{2} - \theta} \geq 1, \tag{4.16}$$

λ is in the interval $(0, 1/2]$.

The graphs of equation (4.14) for several λ are shown in the **Figure 4.4**. We can see in the figure that the farther the angle θ is from $\pi/2$, the greater the change in the angle. This is consistent with the natural idea that the farther from the optimal, the larger the update. In addition, the shape of graphs along λ does not change much unlike the soft orthogonality regularizer in **Figure 4.2**. This makes finding the best λ in learning a model stable.

In general, when convolutional kernel matrix W consists of row vectors $\mathrm{v}_1, \ldots, \mathrm{v}_m$,

our regularizer dispersing the vectors is of the form

$$\lambda \sum_{i \neq j} \left(\arccos \frac{< \mathrm{v}_i, \mathrm{v}_j >}{\|\mathrm{v}_i\| \|\mathrm{v}_j\|} - \frac{\pi}{2} \right)^2 . \tag{4.17}$$

Adding the normalizing terms which make row vectors have norm 1, our regularizer is as follows.

$$\lambda_1 \sum_i \left(1 - \|\mathrm{v}_i\|^2\right)^2 + \lambda_2 \sum_{i \neq j} \left(\arccos \frac{< \mathrm{v}_i, \mathrm{v}_j >}{\|\mathrm{v}_i\| \|\mathrm{v}_j\|} - \frac{\pi}{2} \right)^2 \tag{4.18}$$

The regularization using regularizer (4.18) is named **Evenly Angle Dispersing regularization for Kernel matrix (EADK)**.

The regularizer (4.18) enforces any two different row vectors to be orthogonal. However, as discussed in subsection **3.1.2**, if the number m of row vectors is greater than the dimension n of the ambient space, it is impossible that these vectors are orthogonal. To remedy this, we make all the adjacent distance of row vectors be the expected adjacent distance E in an evenly dispersed state of those. Therefore if two vectors is in the antipodal position each other, the redundancy of features increases, so we disperse these vectors on $\mathbb{P}^n$, not on $\mathbb{S}^n$. If the adjacent distance $\mathrm{ad}(v_i)$ of a row vector v_i in $\mathbb{S}^n$ is less than E or greater than $\pi - E$, then we regularize $\mathrm{ad}(v_i)$ to be E or $\pi - E$, respectively. To sum this up, our regularizer is

$$\begin{aligned} \lambda_1 \sum_i \left(1 - \|\mathrm{v}_i\|^2\right)^2 + \lambda_2 \sum_{\substack{i \neq j \\ \theta_{ij} < E}} \left(\arccos \frac{< \mathrm{v}_i, \mathrm{v}_j >}{\|\mathrm{v}_i\| \|\mathrm{v}_j\|} - E \right)^2 \\ + \lambda_2 \sum_{\substack{i \neq j \\ \pi - \theta_{ij} < E}} \left(\arccos \frac{< \mathrm{v}_i, \mathrm{v}_j >}{\|\mathrm{v}_i\| \|\mathrm{v}_j\|} - (\pi - E) \right)^2 . \end{aligned} \tag{4.19}$$

4.2.2 Evenly Angle Dispersing Regularization with Convolution Matrix

In OCNN regularization, convolution matrices as in (2.6) are enforced to be orthogonal to regularize convolutional kernel weights. This method can also be applied to our evenly angle dispersing regularization. As proposed by OCNN, convolution matrices are structural and sparse, so there exists a more efficient method than dealing directly with the entire matrices. Most of the inner products of two row vectors of convolution matrix is zero because of its sparsity. Thus it is sufficient to consider pairs of row vectors of convolution matrix $\mathcal{K}$ which are overlapping in the 2D sense.

Given convolution matrix $\mathcal{K} \in \mathbb{R}^{MH'W' \times CHW}$, as explained in the subsection **2.5.1**, we need only to regularize the following convolution.

$$C := \mathrm{Conv}(K, K, padding = P, stride = S) \tag{4.20}$$

The output C of (4.20) is of size $M \times M \times (2P/S+1) \times (2P/S+1)$. $K \in \mathbb{R}^{M \times C \times k \times k}$ is the corresponding convolutional kernel weight, k kernel size, S stride and P padding which is obtained by $P = \lfloor \frac{k-1}{S} \rfloor \cdot S$. To orthonormalize the row vectors of $\mathcal{K}$, the center $M \times M$ matrix of the output of (4.20) is an identity matrix and all other entries are zeros. We label the index set corresponding to self inner product as $\mathbb{1}$, in which the output should be ones.

To apply the arccosine method in the subsection **4.2.1**, we divide C elementwisely by its corresponding norms like in (4.11) and call this D. For each $i, j \in$

$\{0, 1, \ldots, m-1\}$, $D[i,j]$ is defined by

$$D[i,j] := \frac{C[i,j]}{\|K[i]\| \cdot \|K[j]\|}. \tag{4.21}$$

Then the ij-th element of D is the cosine value of angle between corresponding two tensors $K[i]$ and $K[j]$. Hence our regularizer for convolution matrix $\mathcal{K}$ is of the form

$$\lambda_1 \sum_{p \in \mathbb{1}} (C_p - 1)^2 + \lambda_2 \sum_{q \notin \mathbb{1}} \left(\arccos D_q - \frac{\pi}{2} \right)^2. \tag{4.22}$$

The regularization using regularizer (4.22) is named **Evenly Angle Dispersing regularization for Convolution matrix (EADC)**

As in (4.19), we can consider to use the concept of dispsersed state in the subsection **3.1.2**. To make all the adjacent distance of row vectors of $\mathcal{K}$ be the expected adjacent distance E in an evenly dispersed state of those, we could apply the following regularizer.

$$\begin{aligned} \lambda_1 \sum_{p \in \mathbb{1}} (C_p - 1)^2 + \lambda_2 \sum_{\substack{q \notin \mathbb{1} \\ \arccos D_q < E}} & (\arccos D_q - E)^2 \\ & + \lambda_2 \sum_{\substack{r \notin \mathbb{1} \\ \pi - \arccos D_r < E}} (\arccos D_r - (\pi - E))^2 . \end{aligned} \tag{4.23}$$

Chapter 5

Algorithms & Experiments

In this chapter, we experiment with our regularization methods. First, in section **5.1**, we write pseudo codes for **MST**, **EADK** and **EADC**. In subsection **5.2.1**, we experiment with how well our **MST** and **EADK** methods spread arbitrary points on $\mathbb{S}^2$ relative to **SO** and **SRIP** regularizations. In addition, we show that our methods can be used nor only for evenly dispersing points but also for the inference of ideal adjacent angle of an evenly dispersed state of points on $\mathbb{S}^2$. In subsection **5.2.2**, we tell the settings for classification experiments, and in subsection **5.2.3**, with the various depth ResNets and WideResnet, we compares the speed and performance of our methods with prior regularizations. Finally, in subsection **5.2.4**, we have ablation study for target regularized weights and the ratio of dispersing coefficient to normalizing coefficient.

5.1 Algorithms

5.1.1 PH0 and MST

PH0 and **MST** regularizations are equivalent as mentioned in section **3.3**. However when it comes to calculation speed, **MST** is faster than **PH0**. We compares the speed of calculating the regularizing losses of **PH0** and **MST** according to various kernel weight shapes, see **Table 5.1**. We uses convolutional kernel weights with various output and input channel numbers and fixed kernel size 3×3. Suppose that W is a convolutional kernel weight with shape $(M, C, 3, 3)$. To calculate with W the **PH0** or **MST** regularizing loss, we first transform W into kernel matrix K with shape $(M, C \times 3 \times 3)$. Here, $C \times 3 \times 3$ represents the dimension of the space to which row vectors of K belongs, and M corresponds to the number of row vectors. **Table 5.1** is the results of calculation times on CPU of regularizing loss using **PH0** and **MST** according to convolutional kernel weight shape. **PH0** generally does not have a large change in calculating speed according to weight shapes. On the other hand, in the case of **MST**, it can be seen that the speed decreases as the number of output channels relatively smaller than that of input channels. This is because the speed calculation of **PH0** depends on both the number of vectors and the dimension of an ambient space, whereas **MST** depends relatively only on the number of vectors. Therefore, we do not experiment with **PH0**, but only with **MST**.

out_chs/in_chs	128/2048	256/1024	512/512	1024/256	2048/128
PH0 (seconds)	3.77	3.74	3.44	4.13	6.51
MST (seconds)	0.32	0.59	1.12	2.33	5.26

Table 5.1: The calculation times of regularizing losses using **PH0** and **MST** according to convolutional kernel weight shapes. We experiment with various output and input channel numbers and fixed kernel size 3×3 on CPU.

Algorithm 1 is for **MST** regularization. A kernel matrix is transformed from a convolution kernel weight. Kernel matrix can even be fully connected layer weight.

Algorithm 1 MST Regularization

Require: model $\mathcal{M}$, regularizer rate λ_1, λ_2

Require: kernel matrices $\{K\}$ of $\mathcal{M}$ ▷ ndimension 2

for input X and label y **do**
  $L \leftarrow 0$
  output $\tilde{y} \leftarrow \mathcal{M}(X)$
  $L \leftarrow \text{CrossEntropyLoss}(y, \tilde{y})$
  for K in $\{K\}$ **do**
    $D_{i,j} \leftarrow \|K_i - K_j\|$ for all i, j row vectors
    $T \leftarrow$ minimum spanning tree of D
    $L \leftarrow L + \lambda_1 \left(\|K_i\|^2 - 1\right)^2$ for all i ▷ normalizing part
    $L \leftarrow L - \lambda_2 \|K_i - K_j\|$ for all $(i, j) \in T$ ▷ dispersing part
  end for
  backpropagate L
end for

5.1.2 EADK

Algorithm 2 is for **EADK** regularization. In dispersing part in **Algorithm 2**, $\pi/2$ could be replaced by the corresponding expected adjacent distance in an evenly dispersed state. However, because the difference of performance for target angle is not much, we use mainly $\pi/2$ in our experiments.

Algorithm 2 EADK Regularization

Require: model $\mathcal{M}$, regularizer rate λ_1, λ_2

Require: kernel matrices $\{K\}$ of $\mathcal{M}$ ▷ ndimension 2

for input X and label y **do**
  $L \leftarrow 0$
  output $\tilde{y} \leftarrow \mathcal{M}(X)$
  $L \leftarrow \text{CrossEntropyLoss}(y, \tilde{y})$
  for K in $\{K\}$ **do**
    $\Theta_{i,j} \leftarrow \arccos\left(\frac{< K_i, K_j >}{\|K_i\|\|K_j\|}\right)$ for all i, j row vectors
    $L \leftarrow L + \lambda_1 \left(\|K_i\|^2 - 1\right)^2$ for all i ▷ normalizing part
    $L \leftarrow L + \lambda_2 \left(\Theta_{i,j} - \frac{\pi}{2}\right)^2$ for all $i \neq j$ ▷ dispersing part
  end for
  backpropagate L
end for

5.1.3 EADC

Algorithm 3 is for **EADC** regularization. In **EADC** regularization, as in OCNN, to obtain regularizing loss, not only convolutional kernel weights but also strides of corresponding convolutional layers are required.

Algorithm 3 EADC Regularization

Require: model $\mathcal{M}$, regularizer rate λ_1, λ_2

Require: convolution kernel weights and strides $\{(W, s)\}$ of $\mathcal{M}$ ▷ ndimension 4

for input X and label y **do**
  $L \leftarrow 0$
  output $\tilde{y} \leftarrow \mathcal{M}(X)$
  $L \leftarrow \text{CrossEntropyLoss}(y, \tilde{y})$
  for W, s in $\{(W, s)\}$ **do**
    $k \leftarrow$ kernel size of W
    $p \leftarrow \lfloor \frac{k-1}{s} \rfloor \cdot s$
    $C \leftarrow \text{Conv}(W, W, padding = p, stride = s)$
    $D[i, j] \leftarrow \dfrac{C[i, j]}{\|W[i]\| \cdot \|W[j]\|}$ ▷ $\cos \theta_{ij}$
    $c \leftarrow \lfloor C.shape[-1]/2 \rfloor$
    $\mathbb{1} \leftarrow \{(i, i, c, c)\}_i$ ▷ self-inner product index set
    $L \leftarrow L + \lambda_1 \left(C_\ell - 1\right)^2$ for all $\ell \in \mathbb{1}$ ▷ normalizing part
    $L \leftarrow L + \lambda_2 \left(\arccos D_\ell - \dfrac{\pi}{2}\right)^2$ for all $\ell \notin \mathbb{1}$ ▷ dispersing part
  end for
  backpropagate L
end for

5.2 Experiments

5.2.1 Analysis for Angle Dispersing

In this subsection, we see how well our regularization **EADK** actually makes points on 2-dimension sphere $\mathbb{S}^2$ evenly dispersed. Before we move on, we define an adjacent angle. Suppose that there is a set $\{\mathrm{v}_i\}_{i=1}^N$ of points in a Euclidean space. If we see these points as vectors from the origin, **Adjacent Angle** of a vector v_i is the angle between v_i and the adjacent vector of v_i.

Table 5.2 is the result of experiment of how the regularized points are distributed by regularizing with **SO**, **SRIP**, **MST** and **EADK** for each 6, 12, 20 and 60 point sets which are initially randomly scattered on $\mathbb{S}^2$. We record average norms, average adjacent angles, minimum adjacent angles, maximum adjacent angles and differences between maximum and minimum of adjacent angles. In the regularizations of **SO** and **SRIP**, norms and angles of given vectors are regularized simultaneously in a regularizer term as in (2.7) and (2.19). These make neither norm nor angle update as we wish. Average norms of **SO** and **SRIP** tend to decrease as the number of points N increases. This is because the number of regularizing terms for norm increases as $O(N)$, while the number of regularizing terms for angle does as $O(N^2)$ On the other hand, since **MST** and **EADK** regularize norms and angles with different regularizer terms, by setting the learning coefficient for regularizing norm relatively larger than that for angle, we can make given points well dispersed as we desire. In particular, in the experiments with **EADK**, there are no significant differences between average adjacent angles and ideal adjacent angles, and likewise, the Max-Min are quite small.

Conversely, an ideal adjacent angle of a set of points could be inferred from an

experiment with **EADK**. For a set of 60 points, it is not easy to obtain the ideal adjacent angle for an evenly dispersed state. However, in our experiment in **Table 5.2**, since **EADK** has a small Max-Min adjacent angle value, it can be estimated that the ideal adjacent angle exists near 25°. The visualizations of the results of our experiments for 6, 12, 20 and 60 points are in the **Figure 5.1** and **5.2**. To give the possibility that **EADK** can be used to infer an ideal adjacent angle of a set of points, we experiment on various target angles with 12 points in **Table 5.3**. At target angle 63°, which is relatively close to the ideal adjacent angle 63.44°, the average norm is 1 and the Max-Min value is 0. Also, it can be observed that the Max-Min value increases when the target angle is less than the ideal adjacent angle. Conversely, when the target is greater than the ideal adjacent angle, Max-Min is still small, but the average norm becomes less than 1. From these observation, we can estimate that there is the ideal adjacent angle around 63°. The visualization of the result of this experiment is in the **Figure 5.3**.

For the experiment of **MST**, the performance is slightly worse than that of **EADK**, but **MST** does not require a target norm unlike **EADK**. Therefore **MST** can be used as the first experiment to find ideal adjacent angel of a set of points. Then for angles in the vicinity of approximated ideal adjacent angle obtained by **MST**, we repeat experiments with **EADK** to estimate ideal adjacent angle.

# of Points (Ideal AA)	Reg.	Average Norm	Average AA	MinAA	MaxAA	Max-Min	Error (Ideal-AA)
6 (90.00°)	**SO**	0.70	67.04°	48.13°	91.67°	43.54°	22.96°
	SRIP	0.77	55.00°	40.68°	68.18°	27.50°	35.00°
	MST	1.01	89.61°	89.32°	89.90°	0.58°	0.39°
	EADK	**1.00**	90.01°	90.01°	90.01°	**0.0°**	**0.01°**
12 (63.44°)	**SO**	0.50	30.94°	18.33°	69.33°	51.00°	32.50°
	SRIP	0.64	28.65°	12.61°	70.40°	57.79°	34.79°
	MST	1.01	62.83°	62.83°	63.15°	0.32°	0.61°
	EADK	**1.00**	63.40°	63.39°	63.40°	**0.01°**	**0.04°**
20 (41.81°)	**SO**	0.39	19.48°	7.45°	57.30°	49.85°	22.33°
	SRIP	0.50	18.33°	4.58°	56.72°	52.14°	23.48°
	MST	1.01	46.98°	46.98°	47.56°	0.58°	5.17°
	EADK	**1.00**	41.83°	41.83°	42.40°	**0.57°**	**0.02°**
60	**SO**	0.22	15.47°	4.58°	30.94°	26.36°	-
	SRIP	0.32	15.47°	5.16°	27.50°	22.34°	-
	MST	1.01	26.93°	25.21°	28.07°	2.86°	-
	EADK	**1.00**	24.64°	24.64°	26.36°	**1.72°**	-

Table 5.2: Angle analysis for various numbers of points on $\mathbb{S}^2$ and regularizations. AA denotes Adjacent Angle (Adjacent Distance).

Target Angle	Average Norm	AverageAA	MinAA	MaxAA	Max-Min
43°	1.00	44.69°	42.97°	60.73°	17.76°
53°	1.00	53.29°	52.71°	55.58°	2.87°
63°	1.00	63.03°	63.03°	63.03°	0°
73°	0.98	63.60°	63.60°	63.60°	0°
83°	0.95	63.60°	63.60°	63.60°	0°

Table 5.3: Angle analysis of **EADK** for various target angles with 12 points.

5.2.2 Experimental Setups

Datasets

To demonstrate the performance of our regularization methods, we experiment on various image classification datasets, i.e., CIFAR-10 [**19**], CIFAR-100 [**19**] and SVHN [**22**]. The CIFAR-10 consists of 60000 images in 32×32 sizes. There are 10 classes (airplane, automobile, bird, cat, etc)and each class consists of 6000 images. These images are divided into training and test data in 5 to 1 ratio. The CIFAR-100 is just like the CIFAR-10, except it has 100 classes (bed, chair, bee, bear, etc) containing 600 images each. Each class is divided in 5 to 1 ratio to make training and test data. Since the classes in the CIFAR-100 are grouped into 20 superclasses, there are ambiguous images so the classification tasks using CIFAR-100 are finer than CIFAR-10. SVHN is a dataset that has images of numbers from 0 to 9 that can be seen in the real world. SVHN is similar to MNIST, but consists of 600,000 images and it is much more difficult to distinguish numbers.

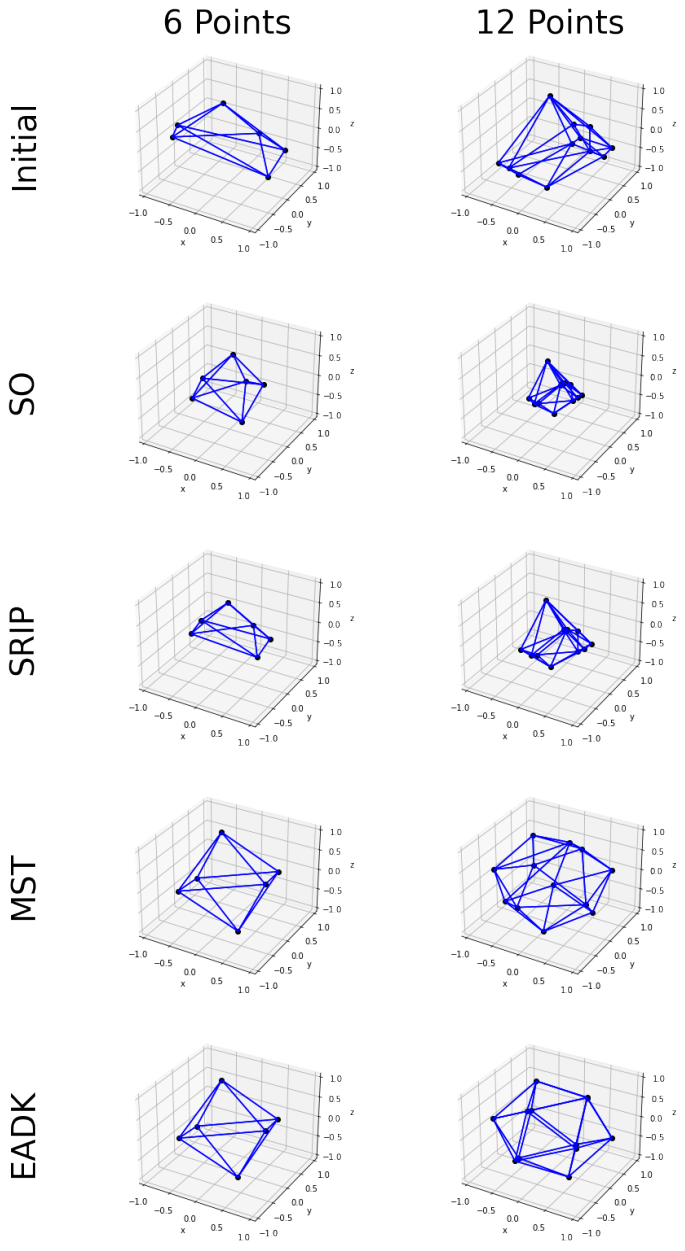

Figure 5.1: 3D Plotting of resultant 6 and 12 points regularized by various regularizations.

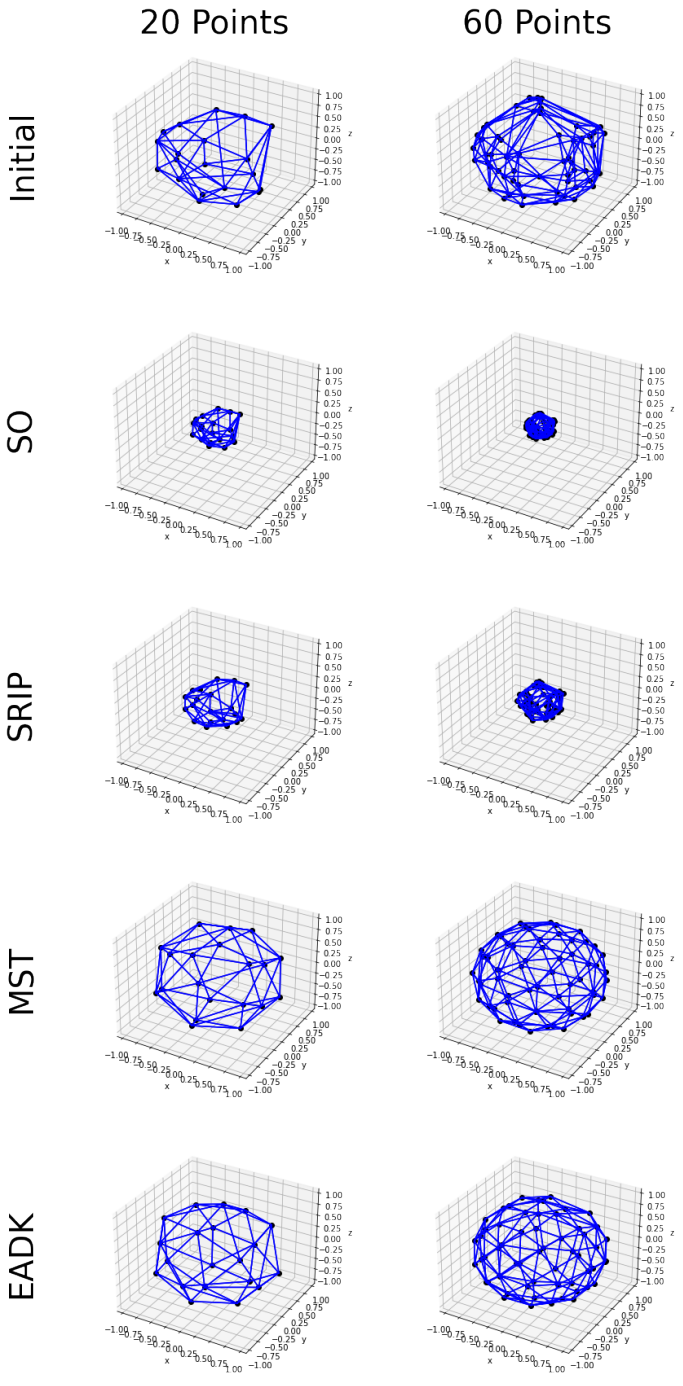

Figure 5.2: 3D Plotting of resultant 20 and 60 points regularized by various regularizations..

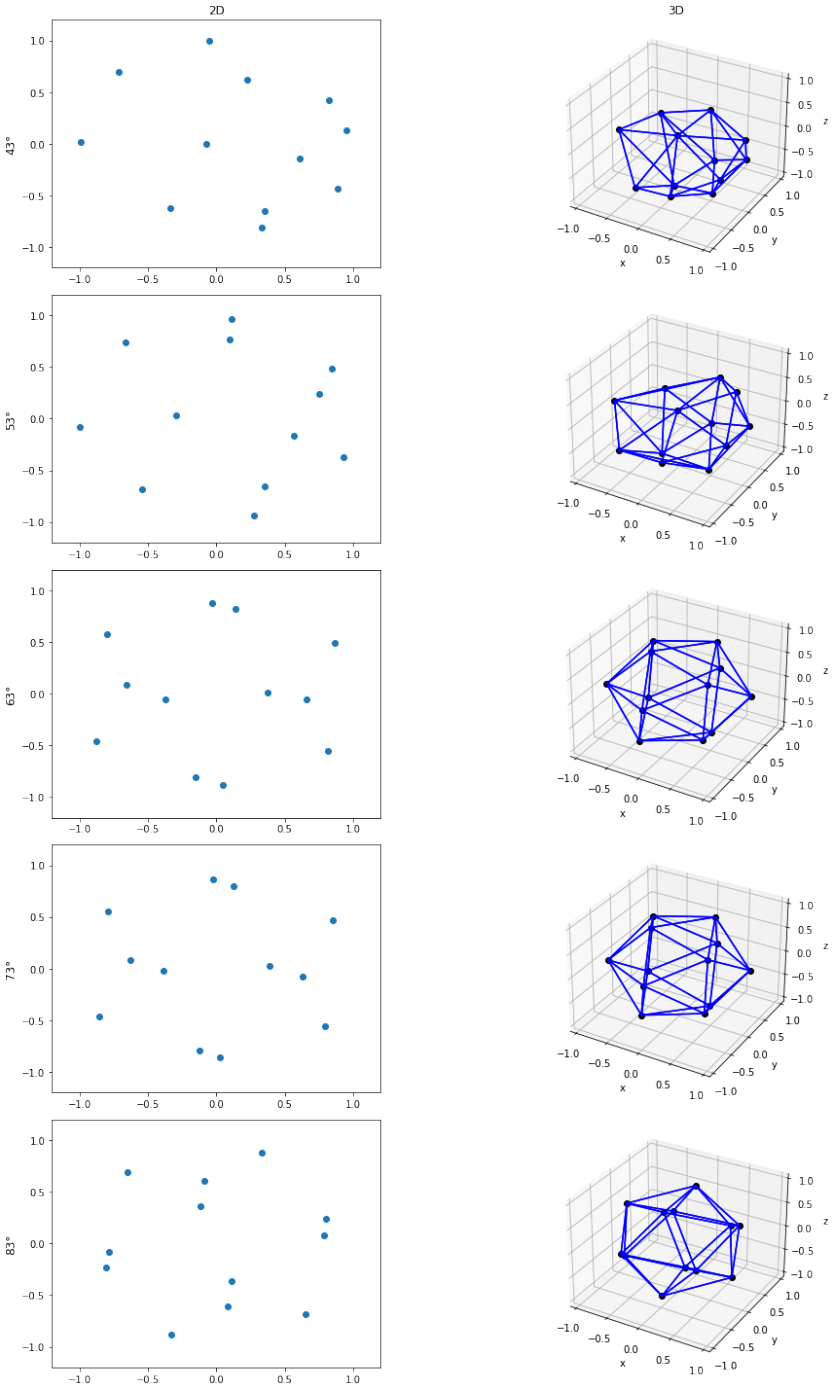

Figure 5.3: 2D and 3D plotting of resultant regularized 12 points regularized by **EADK** for various target angles.

Network Architecture

We consider ResNet [**13**] and WideResnet [**31**], which are mainly used in classification tasks. The ResNet Experiments are conducted with CIFAR-10 and CIFAR100. We experiment on ResNet18, ResNet34, ResNet50 and ResNet101 to compare the results according to the depth of the ResNet. To conduct on models with more up-to-date and good performance, we also experiment on WideResnet with CIFAR-10, CIFAR-100 and SVHN.

Baselines

We compare our methods with a variety of prior regularization methods mentioned in chapter **2** except **MC** regularization because its regularization effect is weak. Among our methods, we excluded **PH0** regularization in our experiments because it is equivalent to **MST** regularization but slower than that. We include **base** regularization in which we just use l2 regularization. As a result, we conduct comparative experiments on 8 cases: **base**, **SO**, **DSO**, **SRIP**, **OCNN**, **MST**, **EADK** and **EADC**. In WideResnet experiments, we exclude **MST** regularization because of GPU memory problem.

Target Regularized Weights

Among our methods, **MST** and **EADK** are applicable for all linear operational weights, but for comparison with other regularizations, we regularize only convolutional kernel weights of a model. In addition, layers with a kernel size of 1 are excluded to focus on the convolutonal kernel regularization. The first convolutional layer to receive input is excluded too because it causes gradient explosion in some regularization methods. For better performance, l2-regularization is applied to all weights for all methods.

Hyper Parameters

We train all networks in our experiments from scratch and optimize them by stochastic gradient descent with momentum 0.9. Initial learning rate is 0.1 and multiplied by 0.2 after epochs 60, 120 and 160, and total epochs are 200. Authors of prior works usually set batch size as 128, but in consideration of the recent improved GPU memories, we set batch size as 256 in ResNet experiments but, in WideResnet, we set 128. In DataLoader setting, the number of workers is set to 4. We use data augmentation by random-cropping in the state of padding 4 of images, giving horizontal flip and random-rotation in the range of 15 degrees.

For all networks and regularizations, regularization coefficients are commonly changed every 20, 50, 70 and 120 epochs. To determine values of regularization coefficients and weight decays, we find the best coefficient setting in about 10 different setting of experiments with ResNet18 and CIFAR-10 and call theses settings standard. If gradient explosion do not occur for the remaining experiments, the experiments are conducted with the standard coefficient settings as possible for each regularization. For example, the standard settings for **SO** regularization are [1e-1, 2e-2, 4e-3, 8e-4, 2e-4] for regularization coefficients and [1e-8, 1e-4, 1e-4, 1e-4, 1e-4] for weight decays. These elements correspond to initial value, after epochs 20, 50, 70 and 120 respectively. For other standard setting of regularizations in ResNet experiments, we put it on the **Table 5.4**. For **MST**, **EADK** and **EADC** experiments, we need to set another parameters, i.e., ratio of dispersing coefficient λ_2 to normalizing coefficient λ_1 and target angle in (3.22), (4.19) and (4.23). We set the ratio parameter λ_2/λ_1 as 0.1 and the target angle as $\pi/2$.

	Regularization Coefficients	Weight Decays
SO	[1e-1, 2e-2, 4e-3, 8e-4, 2e-4]	[1e-8, 1e-4, 1e-4, 1e-4, 1e-4]
DSO	[1e-1, 1e-3, 1e-4, 1e-6, 0]	[1e-8, 5e-4, 5e-4, 5e-4, 5e-4]
SRIP	[1e-4, 1e-5, 1e-6, 1e-7, 0]	[1e-8, 1e-4, 1e-4, 5e-4, 5e-4]
OCNN	[1e-2, 2e-3, 4e-4, 8e-5, 0]	[1e-8, 5e-4, 5e-4, 5e-4, 5e-4]
MST	[1e-3, 1e-4, 1e-5, 1e-6, 0]	[1e-8, 5e-4, 5e-4, 5e-4, 5e-4]
EADK	[2e-1, 1e-3, 1e-4, 1e-6, 0]	[1e-8, 5e-4, 5e-4, 5e-4, 5e-4]
EADC	[1e-2, 2e-3, 4e-4, 8e-5, 0.0]	[1e-8, 5e-4, 5e-4, 5e-4, 5e-4]

Table 5.4: Regularization coefficients and weight decays of standard settings, i.e., for Resnet18 and CIFAR-100 experiments.

Evaluation Metric

Top-1/5 accuracy rate: The top-k accuracy rate represents the fraction of test images, in which the correct answer label are included in the top k most probable by model inference, among all test images. Then the top-1 accuracy rate is the ratio of correct answers. In our experiments, we use top-1 and top-5 accuracy rates to evaluate the generalization performances.

5.2.3 Classification Accuracy

Comparison with Various Depth ResNet

We compare Top-1 accuracy with existing and our methods for ResNet of different depth to show that our proposed methods are valid. **Table 5.5** is the result of our Top-1 accuracy experiment and values in the table are calculated by averaging Top-1 accuracy of validation in the last 5 epochs. Our experiment is conducted based on the aforementioned settings. Our results are somewhat different from the

results stated in the papers of SRIP [**2**] and OCNN [**27**] because only convolutional weights with kernel size larger than one are regularized. The most superior accuracy value is written in bold for each experiment. The table shows that our **EADK** and **EADC** regularizations outperform at ResNet101 on CIFAR-10 and all ResNets on CIFAR-100. Since **MST** requires twice or more as much GPU memory as others, **MST** experiments with ResNet101 are excluded. In our **SRIP** experiments, it is necessary to adjust the regularization coefficents frequently to avoid gradient explosion according to the depth of ResNet. **Figure 5.5** shows validation curves during traing for various ResNet on CIFAR-10 and CIFAR-100.

Dataset	CIFAR-10				CIFAR-100			
ResNet	18	34	50	101	18	34	50	101
base	94.46	**95.06**	94.57	95.28	74.84	76.39	75.51	77.25
SO	94.19	94.43	93.93	94.02	74.94	75.48	74.75	75.98
DSO	94.67	94.87	**95.43**	95.04	75.86	76.72	77.26	78.75
SRIP	92.87	94.68	94.89	94.88	74.69	75.79	75.37	77.06
OCNN	**95.00**	94.89	94.76	94.96	75.86	77.12	76.91	78.27
MST	93.28	93.71	92.94	-	74.79	74.85	72.51	-
EADK	94.76	94.91	95.20	**95.35**	**76.51**	76.99	**77.79**	78.61
EADC	94.97	94.89	95.06	95.04	76.37	**77.54**	77.60	**78.78**

Table 5.5: Top-1 accuracy rates (%) of ResNet with various depth and regularizations on CIFAR-10 and CIFAR100. We evaluate accuracy rates by averaging the results of the last 5 epochs.

Table 5.6 shows one epoch learning times for combinations of various ResNets and regularizations. The base row of the table represents the average training time

per epoch when we use only l2 regularization. The rows below are the values obtained by dividing the average training time per epoch for each experiment by that of base. The last column of the table represents the mean of these values for each regularization. While **SO** is the fastest regularization, **DSO**, a similar regularization, takes the most mean time ratio. This is because transposing a kernel matrix in **DSO** often increases the size of its Gram matrix significantly. Of our proposed regularizations, **EADK** is the fastest one and not much different from the training speed of **SO**. Considering the validation accuracy of **EADK**, we find out that it has considerable advantages. **Figure 5.4** illustrates the relationship between relative training trime and accuracy based on the results in the **Table 5.5** and **Table 5.6**, indicationg that the more it is in the upper left derectioni, the better methods. In CIFAR-10, some prior regularizations show better performance than our methods, but the diagram shows that **EADK** is most in the upper left corner, so we can tell that **EADK** has the best performance. In the case of CIFAR-100, it is also confirmed that our methods have better performance than other methods even considering training time.

Dataset	CIFAR-10				CIFAR-100				
ResNet	18	34	50	101	18	34	50	101	
base	12.1s	22.7s	40.9s	68.7s	12.1s	22.6s	40.6s	68.7s	Mean
SO	1.14	1.13	1.03	1.04	1.14	1.14	1.04	1.03	1.09
DSO	8.43	9.34	3.64	3.85	8.86	9.39	3.67	3.08	6.28
SRIP	4.84	4.72	2.09	2.11	3.93	4.68	2.20	2.11	3.33
OCNN	1.90	1.92	1.23	1.26	1.89	1.95	1.23	1.26	1.58
MST	5.78	5.33	2.24	-	5.73	5.36	2.26	-	4.45
EADK	1.28	1.27	1.07	1.10	1.25	1.28	1.07	1.10	1.18
EADC	3.51	3.74	1.62	1.78	3.53	3.66	1.71	1.82	2.67

Table 5.6: The values in the base row of the table indicate elapsed time (seconds) of training one epoch for each experiment. The values in the below rows are the ratio of elapsed times of training one epoch for each experiment with a regularization to that of base.

Comparison with WideResnet

Table 5.7 shows Top-1 and Top-5 accuracy of experiments with WideResnet architecture for various regularizations. We use CIFAR-10, CIFAR-100 and SVHN as datasets. As in the ResNet experiments, only the set of convolutional kernel weights with kernel size greater than 1 is regularized by regularization methods. On CIFAR-10, **OCNN** and **SRIP** have high accuracy. On CIFAR-100 and SVHN, **DSO** outperform and our **EADK** and **EADC** are following. **Figure 5.6** shows validation curves during training WideResnet on the datasets. **Table 5.6** shows elapsed time of training one epoch for our experiments. Although **DSO** and **SRIP**

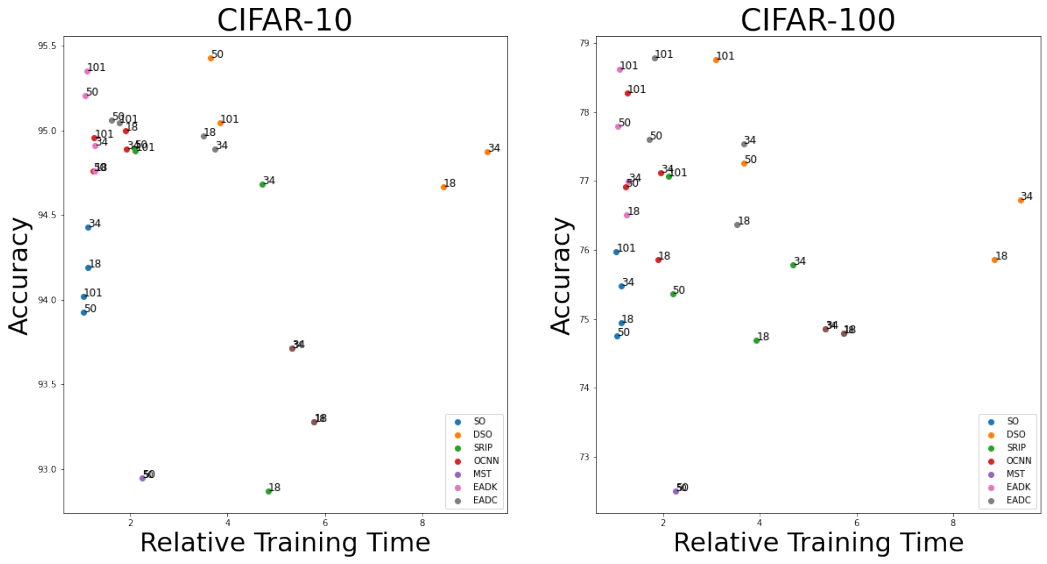

Figure 5.4: Diagrams of relationship between relative learning time and accuracy for various regularizations.

have achieved good results in the experiments, the learning time takes about 5 to 1- times more than **base** setting. On the other hand, our **EADK** only takes up to 13% more time than the base and thus **EADK** has fast learning speed while having excellent accuracy.

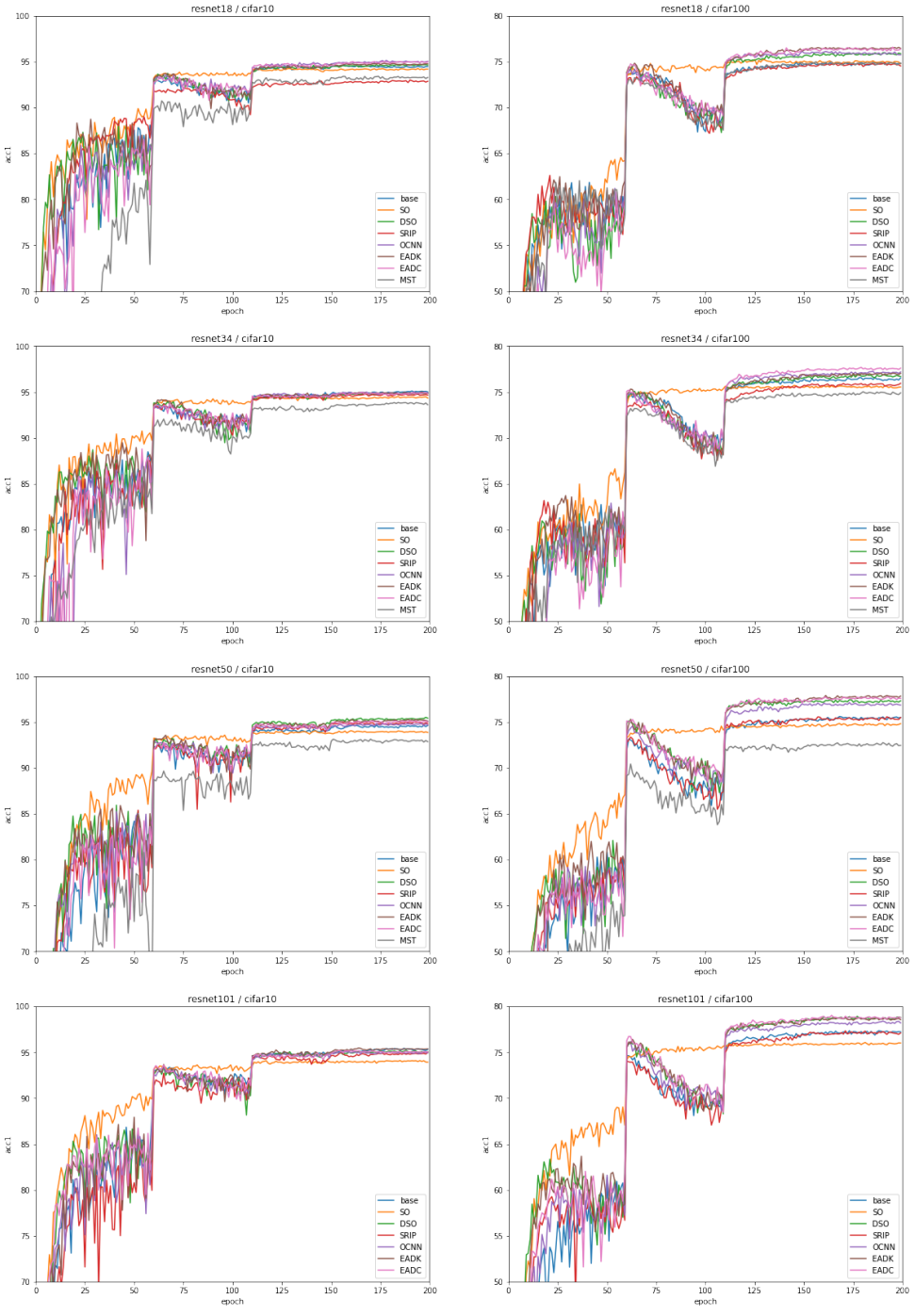

Figure 5.5: Validation curves during training various ResNet on CIFAR-10 and CIFAR-100.

	CIFAR-10	CIFAR-100	SVHN
base	95.83/99.85	79.53/94.70	96.86/99.62
SO	95.63/99.88	78.89/94.34	96.88/99.63
DSO	95.87/99.91	**79.87**/94.83	**97.08**/99.63
SRIP	**95.91**/99.90	79.49/94.60	96.98/99.67
OCNN	**95.93**/99.88	79.65/95.00	96.98/99.62
EADK	95.77/99.88	79.66/94.80	**97.02**/99.65
EADC	95.73/99.90	**79.67**/94.91	**97.02**/99.65

Table 5.7: Top-1/Top-5 accuracy rates (%) of WideResnet with various regularizations on CIFAR-10, CIFAR100 and SVHN. We evaluate accuracy rates by averaging the results of the last 5 epochs.

	CIFAR-10	CIFAR-100	SVHN
base	78.78s	78.16s	118.84s
SO	1.07	1.08	1.06
DSO	9.48	9.55	4.94
SRIP	4.85	4.89	2.84
OCNN	2.07	2.21	1.65
EADK	1.13	1.13	1.09
EADC	3.89	3.56	2.32

Table 5.8: The values in the base row of the table indicate elapsed time (seconds) of training one epoch for each experiment. The values in the below rows are the ratio of elapsed times of training one epoch for each experiment with a regularization to that of base.

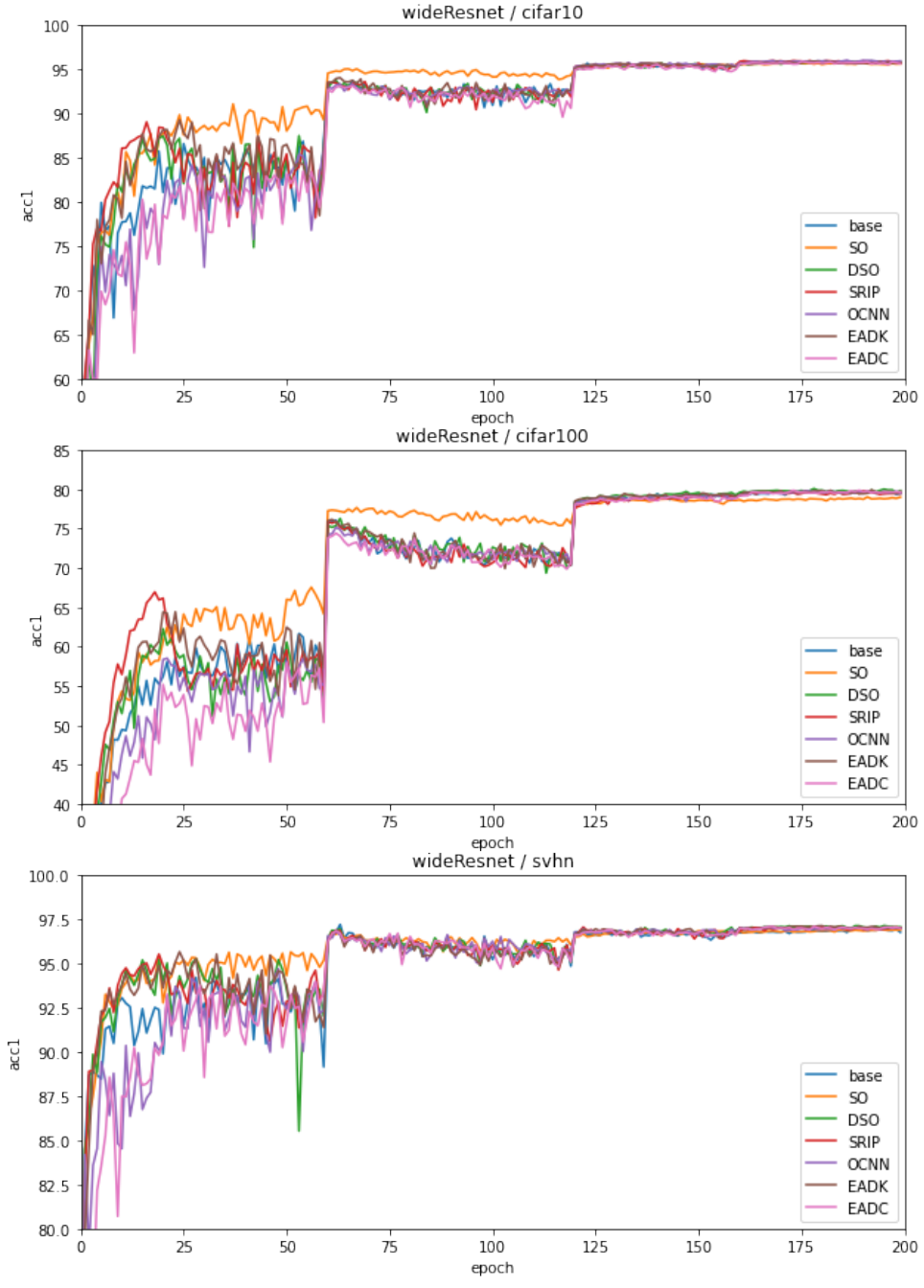

Figure 5.6: Validation curves during training WideResnet on CIFAR-10, CIFAR-100 and SVHN.

5.2.4 Additional Experiments

Table 5.9 shows the accuracy for various target regularized weights on our **EADK** and **EADC** regularizations. We experiment this with the model ResNet18 and dataset CIFAR-100. We divide target regularized weights into three categories: F: fully connected weights, C1: convolutional weights with kernel size 1 and C3: convolutional weights with kernel size greater than 1. We conduct this experiment on various combinations of these categories. As a result of the experiment, it could be confirmed that ResNet18 with **EADK** of **EADC** has generally high accuracy when C3 is regularized. This shows that for sufficiently complicated kernels, the effect of our regularizations is greater.

weights	F	C1	C3	F+C1	C1+C3	F+C1+C3
EADK	74.41	74.86	76.51	74.81	**76.57**	76.19
EADC	75.04	73.02	**76.37**	74.43	75.79	75.82

Table 5.9: Experiment for various target regularized weights. F, C1 and C3 are the set of fully connected weights, convolutional weights of kernel size 1 and convolutional weights of kernel size greater than 1, respectively. Values are Top-1 accuracy for various target regularized weights. We use ResNet18 and CIFAR-100 for this experiment.

Table 5.10 is the result for accuracy of **EADK** and **EADC** when the ratio λ_2/λ_1 of dispersing coefficient λ_2 to normalizing coefficient λ_1 varies in (4.19) and (4.23). We also do this experiment with ResNet18 and CIFAR-100. Large λ_2/λ_1 means that the update of changing angles to $\pi/2$ is relatively larger than the update of changing norms to one. In the case of **EADC** with λ_2/λ_1 1, gradient explosion occurs and so excluded from our experiment. The result shows the best performance

at λ_2/λ_1 0.1, and thus we use this value in subsection **5.2.3**.

λ_2/λ_1	0.01	0.02	0.05	0.1	0.2	0.5	1
EADK	75.86	76.03	75.81	**76.51**	76.25	76.16	76.23
EADC	76.17	76.25	75.90	**76.37**	75.98	75.70	-

Table 5.10: λ_2/λ_1 is the rate of dispersing part coefficient λ_2 to normalizing part coefficient λ_1 in our regularizations. Values are Top-1 accuracy for various λ_2/λ_1. We use ResNet18 and CIFAR-100 for this experiment.

Chapter 6

Conclusion

Many approaches have been developed to give orthogonality to the matrix from convolutional kernel weights to deal with the gradient vanishing or explosion issue caused by the depth of CNN and to avoid redundant features. However, these approaches do not regularize convolutional kernel as well as we wish. In this thesis, we analyze whether these existing methods effectively conserve energy and disperse angles, and we show that this is not the case through experiments at low dimensional case. To resolve these problems, we introduce the concept of evenly dispersed state in a compact metric space and show that this can generalize the concept of orthogonality. And based on this dispersing concept, we propose **MST**, **EADK** and **EADC** regularizations. Through our experiments, we show that our methods actually cause a set of points to be evenly dispsersed. In addition, we can infer the ideal adjacent angle of an evenly dispersed state of points on $\mathbb{S}^2$. In particular, since our **EADK** is much fast regularization and a matrix from convolutional kernel weight could be of hundreds to thousands of dimensions, our **EADK** has not only good performances but also fast learning speed. We expect the ideas presented

in this thesis to give a good inspiration for regularization problem of convolutional kernel weights and even dispersion problem of points in a high dimensional compact metric. For future work, we need to study theorems that can better estimate the expected adjacent distance and more appropriate ratio of dispersing coefficient to normalizing coefficient.

Bibliography

[1] R. Balestriero and R. Baraniuk, *Mad max: Affine spline insights into deep learning*, arXiv preprint arXiv:1805.06576, (2018).

[2] N. Bansal, X. Chen, and Z. Wang, *Can we gain more from orthogonality regularizations in training deep networks?*, Advances in Neural Information Processing Systems, 31 (2018).

[3] T. Birdal, A. Lou, L. J. Guibas, and U. Simsekli, *Intrinsic dimension, persistent homology and generalization in neural networks*, Advances in Neural Information Processing Systems, 34 (2021).

[4] A. Böttcher and S. M. Grudsky, *Toeplitz matrices, asymptotic linear algebra and functional analysis*, vol. 67, Springer, 2000.

[5] E. J. Candes and T. Tao, *Decoding by linear programming*, IEEE transactions on information theory, 51 (2005), pp. 4203–4215.

[6] G. Carlsson, *Topology and data*, Bulletin of the American Mathematical Society, 46 (2009), pp. 255–308.

[7] P. J. Davis, *Circulant Matrices*, Wiley, New York, 1970.

[8] D. L. Donoho, *Compressed sensing*, IEEE Transactions on information theory, 52 (2006), pp. 1289–1306.

[9] V. Dumoulin and F. Visin, *A guide to convolution arithmetic for deep learning*, ArXiv e-prints, (2016).

[10] H. Edelsbrunner and J. L. Harer, *Computational topology: an introduction*, American Mathematical Society, 2022.

[11] R. L. Graham and P. Hell, *On the history of the minimum spanning tree problem*, Annals of the History of Computing, 7 (1985), pp. 43–57.

[12] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al., *Recent advances in convolutional neural networks*, Pattern Recognition, 77 (2018), pp. 354–377.

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

[14] F. Heide, W. Heidrich, and G. Wetzstein, *Fast and flexible convolutional sparse coding*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5135–5143.

[15] P. G. Huray, *Maxwell's equations*, John Wiley & Sons, 2011.

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

[17] I. M. James, *The topology of Stiefel manifolds*, vol. 24, Cambridge University Press, 1976.

[18] J. Jaquette and B. Schweinhart, *Fractal dimension estimation with persistent homology: a comparative study*, Communications in Nonlinear Science and Numerical Simulation, 84 (2020), p. 105163.

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

[20] S. Liu, X. Li, Y. Zhai, C. You, Z. Zhu, C. Fernandez-Granda, and Q. Qu, *Convolutional normalization: Improving deep convolutional network robustness and training*, Advances in Neural Information Processing Systems, 34 (2021).

[21] R. Mises and H. Pollaczek-Geiringer, *Praktische verfahren der gleichungsauflösung.*, ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik, 9 (1929), pp. 58–77.

[22] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, *Reading digits in natural images with unsupervised feature learning*, (2011).

[23] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, *Deep learning for anomaly detection: A review*, ACM Computing Surveys (CSUR), 54 (2021), pp. 1–38.

[24] R. Priemer, *Introductory Signal Processing. Advanced Series in Electrical and Computer Engineering*, vol. 6, World Scientific Pub Co Inc., 1991.

[25] V. Udayashankara, *Real Time Digital Signal Processing*, Prentice-Hall, 2010.

[26] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, *Residual attention network for image classification*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 3156–3164.

[27] J. Wang, Y. Chen, R. Chakraborty, and S. X. Yu, *Orthogonal convolutional neural networks*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 11505–11515.

[28] D. Xie, J. Xiong, and S. Pu, *All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6176–6185.

[29] K. Yanai, R. Tanno, and K. Okamoto, *Efficient mobile implementation of a cnn-based object recognition system*, in Proceedings of the 24th ACM international conference on Multimedia, 2016, pp. 362–366.

[30] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, *Generative image inpainting with contextual attention*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 5505–5514.

[31] S. Zagoruyko and N. Komodakis, *Wide residual networks*, arXiv preprint arXiv:1605.07146, (2016).

[32] W. Zhang, J. Tanida, K. Itoh, and Y. Ichioka, *Shift-invariant pattern recognition neural network and its optical architecture*, in Proceedings of an-

nual conference of the Japan Society of Applied Physics, Montreal, CA, 1988, pp. 2147–2151.

국문초록

이 논문에서는 합성곱커널에 대한 새로운 정규화 방법들을 제안한다. 딥러닝의 발달과 더불어 신경망의 가장 기본적인 모듈인 합성곱 레이어를 효과적으로 정규화하려는 시도들이 있어 왔다. 합성곱신경망는 인풋데이터를 추상화하는데 탁월하지만 네트워크의 깊이가 깊어지면 그레디언트 소멸이나 폭발 문제를 일으키고 중복된 피쳐들을 만든다. 이러한 문제들을 해결하기 위한 접근법 중 하나는 직접 합성곱 신경망의 합성곱커널을 직접 정규화 하는 것이다. 이 방법은 합성곱커널을 어떤 행렬로 변환하고 행렬의 행 또는 열들의 벡터들을 직교시키는 것이다. 그러나 이러한 접근법은 몇가지 단점이 있다. 첫째로, 벡터의 수가 벡터의 차원보다 많을 때는 모든 벡터를 직교화 시킬 수 없게 되므로 적절한 기법들을 필요로 한다. 이 문제를 다루기 위한 한 가지 방법으로 우리는 분산 상태라는 개념을 정의하고 이 개념을 활용한 **PH0**와 **MST** 정규화법을 제안한다. 둘째로, 그람행렬을 항등행렬로 근사시키는 방법을 사용하는 기존 정규화법이 벡터들을 직교화시키는 최적의 방법이 아닐 수 있다는 점이다. 즉, 기존의 정규화법이 두 벡터가 가까울 때는 오히려 각도의 업데이트를 줄이게 된다. 따라서 이를 보완하기 위하여 우리는 각도를 직접 업데이트하는 **EADK**와 **EADC** 정규화법을 제안한다. 그리고 다양한 실험을 통해 **EADK**와 **EADC** 정규화법이 다수의 신경망구조에서는 기존의 방법들보다 우수한 성능을 보이고 특히 **EADK**는 빠른 학습시간을 가진다는 것을 확인한다.

주요어휘: 딥러닝, 합성곱, 커널, 정규화, 직교화, 고른분산상태

학번: 2012-23024