# Compressive Hyperspherical Energy Minimization

Rongmei Lin[1,*], Weiyang Liu[2,*], Zhen Liu[2], Chen Feng[3], Zhiding Yu[4], James M. Rehg[2], Li Xiong[1], Le Song[2]
[1]Emory University [2]Georgia Institute of Technology [3]New York University [4]NVIDIA
{rongmei.lin@emory.edu wyliu@gatech.edu} * equal contributions



## Abstract

*Recent work on minimum hyperspherical energy (MHE) has demonstrated its potential in regularizing neural networks and improving their generalization. MHE was inspired by the Thomson problem in physics, where the distribution of multiple propelling electrons on a unit sphere can be modeled via minimizing some potential energy. Despite the practical effectiveness, MHE suffers from local minima as their number increases dramatically in high dimensions, limiting MHE from unleashing its full potential in improving network generalization. To address this issue, we propose compressive minimum hyperspherical energy (CoMHE) as an alternative regularization for neural networks. Specifically, CoMHE utilizes a projection mapping to reduce the dimensionality of neurons and minimizes their hyperspherical energy. According to different constructions for the projection matrix, we propose two major variants: random projection CoMHE and angle-preserving CoMHE. Furthermore, we provide theoretical insights to justify its effectiveness. We show that CoMHE consistently outperforms MHE by a significant margin in comprehensive experiments, and demonstrate its diverse applications to a variety of tasks such as image recognition and point cloud recognition.*

## 1. Introduction

Recent years have witnessed the tremendous success of deep neural networks in a variety of tasks. With its over-parameterization nature and hierarchical structure, deep neural networks achieve unprecedented performance on many challenging problems [14, 12, 29], but their strong approximation ability also make it easy to overfit the training set, which greatly affects the generalization on unseen samples. Therefore, how to restrict the huge parameter space and properly regularize the deep networks becomes increasingly important. Regularization of neural networks can be roughly categorized into *implicit* and *explicit*. Implicit regularization usually does not directly impose explicit constraints on the neuron weights, but instead it regularizes the networks in an implicit manner in order to prevent the over-fitting and stabilize the training dynamics. A lot of classic methods fall into this category, such as batch normalization [16], dropout [39], weight normalization [37], group normalization [40], etc. Explicit regularization usually introduces some penalty terms for the neuron weights, and jointly optimizes it along with the other objective functions.

Among many explicit regularizations, minimum hyperspherical energy (MHE) [24] is a simple yet effective regularization that encourages the *hyperspherical diversity* among neurons and significantly improves the network generalization. From geometric perspective, MHE regularizes the directions of neuron weights by minimizing a special potential energy on a unit hypersphere that characterizes the hyperspherical diversity (such energy is defined as *hyperspherical energy* [24]). In contrast, standard weight decay only regularizes the norm of neuron weights, which can be viewed as regularizing one dimension of the neuron weight. In fact, MHE completes an important missing piece in the standard regularization in neural networks by introducing the regularization for the neuron directions (*i.e.*, regularizing the rest dimensions of the neuron weight).

Despite its elegant interpretation and good performance, MHE still has a few critical problems which limit MHE to unleash its full potential. First, MHE suffers from huge number of local minima and stationary points due to its highly non-convex objective function. The problem can get even worse when the dimension gets higher and the number of neurons becomes larger [1, 4]. Second, the gradient *w.r.t* the neuron weight of the original MHE objective is deterministic. Unlike the weight decay whose objective is convex, MHE has a complex and non-convex regularization term. Therefore, deterministic gradients may quickly fall into one of the stationary points and get stuck there. Third, in high-dimensional spaces, neurons are likely to be orthogonal to each other. As a result, in the original measure of diversity in MHE, these high-dimensional neurons can be trivially "diverse", leading to very small gradients that may result in optimization difficulties.

In order to address these problems, we propose the compressive minimum hyperspherical energy (CoMHE) as a more effective regularization for neural networks. CoMHE

first reduces the neuron weight from a high-dimensional space to a low-dimensional one and then applies standard MHE to regularize these neurons. As a result, how to reduce the neuron weights to a low-dimensional space while preserving some of the desirable information in high-dimensional space is our major focus. The most important information we care in MHE is the angular similarity between different neuron weights. To this end, we consider two approaches: *random projection* and *angle-preserving projection*, which can reduce the dimensionality of neurons while still partially preserving the angular information.

Random projection (RP) [2] is a naturally motivated choice to achieve the dimensionality reduction in MHE due to its simplicity and nice theoretical properties. RP can provably preserve the angular information, and most importantly, introduce certain degree of randomness to the gradients, which can help MHE escape from some stationary points. The role that the randomness of RP serves in CoMHE is actually similar to the simulated annealing [43, 42] that is widely used in Physics to solve Thomson problem. Such randomness is also empirically shown to benefit the network generalization [19]. Furthermore, we also prove that using RP can well preserve the pairwise angles between neurons. In addition to RP, we propose the angle-preserving projection (AP) as an alternative way to project the neurons. AP is inspired by the goal that we aim to preserve the pairwise angles between neurons. Constructing an AP that can project neurons to a low-dimensional space and exactly preserve the angles is usually very difficult even with powerful non-linear functions, which might be hinted by the strong conditions required for conformal mapping in complex analysis [31]. Therefore, we adopt a different approach to construct AP by framing the AP construction as an optimization problem which can be solved alternatively with the neural network training.

However, it is still inevitable to lose some information of the high-dimensional space and the neurons may only get diverse in some low-dimensional subspaces. To address this, we introduce multiple projections to approximate the learning objective of MHE in the original high-dimensional space more accurately. Specifically, we use multiple projection to project the neurons to different subspaces, then compute the MHE loss in each space separately and finally minimize the average of these MHE losses. Besides that, we also reinitialize these projection matrix randomly every certain number of iterations to avoid trivial solutions. Most importantly, CoMHE with both RP and AP perform consistently better than standard MHE in different applications.

The high-level intuition behind CoMHE is to project neurons to a space whose dimension is significantly smaller than the original one such that the hyperspherical energy can get minimized more effectively. Our contributions can be summarized in four aspects:

- To address the drawbacks of MHE, we propose CoMHE as a generic framework to effectively minimize hyperspherical energy of neurons in low-dimensional space.
- Two novel approaches (random projection and angle-preserving projection) are proposed to achieve the dimensionality reduction of neurons while still being able to partially preserve the angular information in MHE. Moreover, we also consider several notable CoMHE variants such as group CoMHE, and mixed CoMHE.
- We provide rigorous theoretical analysis to show that the projection used in the paper can well preserve the angular similarity between neurons.
- We apply CoMHE to image recognition and point cloud recognition, and demonstrate consistent and significant improvement over standard MHE.

## 2. Related Work

Diversity-based regularization has been found useful in sparse coding [30, 34], ensemble learning [21, 20], self-paced learning [17], metric learning [47], latent variable models [48], person re-identification [22], face recognition [24], etc. There are a number of ways to characterize diversity in previous works. Early studies in sparse coding [30, 34] model the diversity with the empirical covariance matrix and show that the generalization capability of dictionary can be improved by regularizing its diversity. [46] promotes the uniformity among eigenvalues of the component matrix in a latent space model. [45] encourages the diversity of latent space models by imposing constraints to the pairwise angle among components. [5, 35, 44] characterize diversity among neurons with orthogonality, and regularize the neural network using the orthogonality. Inspired by the Thomson problem in physics, MHE [24] defines the hyperspherical energy to characterize the diversity on a unit hypersphere and shows significant and consistent improvement in supervised learning tasks. There are two MHE variants in [24]: full-space MHE and half-space MHE. Compared to full-space MHE, the half-space variant [24] further eliminates the collinear redundancy by constructing virtual neurons with the opposite direction to the original ones and then minimizing their hyperspherical energy together. The importance of regularizing angular information is also extensively discussed in [27, 28, 26, 25].

## 3. Compressive MHE

### 3.1. Revisiting Standard MHE

MHE characterizes the diversity of $N$ neurons ($\boldsymbol{W}_N = \{\boldsymbol{w}_1, \cdots, \boldsymbol{w}_N \in \mathbb{R}^{d+1}\}$) on a unit hypersphere using hyperspherical energy which is defined as

$$\begin{aligned} \boldsymbol{E}_{s,d}(\hat{\boldsymbol{w}}_i|_{i=1}^N) &= \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} f_s\big(\|\hat{\boldsymbol{w}}_i - \hat{\boldsymbol{w}}_j\|\big) \\ &= \begin{cases} \sum_{i\neq j} \|\hat{\boldsymbol{w}}_i - \hat{\boldsymbol{w}}_j\|^{-s}, & s > 0 \\ \sum_{i\neq j} \log\big(\|\hat{\boldsymbol{w}}_i - \hat{\boldsymbol{w}}_j\|^{-1}\big), & s = 0 \end{cases}, \end{aligned} \tag{1}$$

where $\|\cdot\|$ denotes $\ell_2$ norm, $f_s(\cdot)$ is a decreasing real-valued function (we use $f_s(z)=z^{-s}, s>0$, *i.e.*, Riesz $s$-kernels), and $\hat{\boldsymbol{w}}_i=\frac{\boldsymbol{w}_i}{\|\boldsymbol{w}_i\|}$ is the $i$-th neuron weight projected onto the unit hypersphere $\mathbb{S}^d=\{\boldsymbol{v}\in\mathbb{R}^{d+1}|\,\|\boldsymbol{v}\|=1\}$. For convenience, we denote $\hat{\boldsymbol{W}}_N=\{\hat{\boldsymbol{w}}_1,\cdots,\hat{\boldsymbol{w}}_N\in\mathbb{S}^d\}$, and $\boldsymbol{E}_s=\boldsymbol{E}_{s,d}(\hat{\boldsymbol{w}}_i|_{i=1}^N)$. Note that, each neuron is a convolution kernel in convolutional neural networks (CNNs). MHE minimizes the hyperspherical energy of neurons using gradient descent during back-propagation, and MHE is typically applied to the neural network layer-wise. For example, the gradient of $\boldsymbol{E}_2$ *w.r.t* $\hat{\boldsymbol{w}}_i$ can be written as

$$\nabla_{\hat{\boldsymbol{w}}_i}\boldsymbol{E}_2=\sum_{j=1,j\neq i}^{N}\frac{-2(\hat{\boldsymbol{w}}_i-\hat{\boldsymbol{w}}_j)}{\|\hat{\boldsymbol{w}}_i-\hat{\boldsymbol{w}}_j\|^4}. \tag{2}$$

By making the gradient to be zero (*i.e.*, $\nabla_{\hat{\boldsymbol{w}}_i}\boldsymbol{E}_2=0$), we can obtain the stationary points. We simplify $\nabla_{\hat{\boldsymbol{w}}_i}\boldsymbol{E}_2=0$ as

$$\hat{\boldsymbol{w}}_i=\frac{\sum_{j=1,j\neq i}^{N}\alpha_j\hat{\boldsymbol{w}}_j}{\sum_{j=1,j\neq i}^{N}\alpha_j} \tag{3}$$

where $\alpha_j=\|\hat{\boldsymbol{w}}_i-\hat{\boldsymbol{w}}_j\|^{-4}$. We use toy and informal examples to show that high dimensional space (*i.e.*, $d$ is large) leads to much more stationary points than low-dimensional one. Assume there are $K=K_1+K_2$ stationary points in total for $\hat{\boldsymbol{W}}_N$ to satisfy Eq. (3), where $K_1$ denotes the number of stationary points in which every element in the solution is distinct and $K_2$ denotes the number of the rest stationary points. We give two examples: *(i)* For $(d+2)$-dimensional space, inserting a zero to the solutions in $(d+1)$-dimensional space satisfies Eq. (3). Because there are $d+2$ ways to insert the zero, we have at least $(d+2)K$ stationary points in $(d+2)$-dimensional space. *(ii)* We denote $K_1'=\frac{K_1}{(d+1)!}$ as the number of unordered sets that construct the stationary points. Then in $(2d+2)$-dimensional space, we have $\hat{\boldsymbol{w}}_j^E=\frac{1}{\sqrt{2}}\{\hat{\boldsymbol{w}}_j;\hat{\boldsymbol{w}}_j\}\in\mathbb{S}^{2d+1},\forall j$ satisfying Eq. (3). Moreover, the order of the elements in $\hat{\boldsymbol{w}}_j^E$ does not matter for Eq. (3).Thus, there are at least $\frac{(2d+2)!}{2^{d+1}}K_1'+K_2$ stationary points for $\hat{\boldsymbol{W}}_N$ in $(2d+2)$-dimensional space, and besides this trivial construction, there are much more stationary points. Now one can know that there are increasingly more stationary points for MHE in higher dimensions.

### 3.2. General Framework

To overcome the drawbacks of high dimensionality, we propose the compressive MHE which first projects the neurons to a low-dimensional space and then minimizes the hyperspherical energy of the projected neurons. The general framework of CoMHE aims to minimize the following form of hyperspherical energy:

$$\boldsymbol{E}_s^C(\hat{\boldsymbol{W}}_N)=\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}f_s\big(\|g(\hat{\boldsymbol{w}}_i)-g(\hat{\boldsymbol{w}}_j)\|\big) \tag{4}$$

where $g:\mathbb{S}^d\rightarrow\mathbb{S}^k$ takes a normalized $(d+1)$-dimensional input and outputs a normalized $(k+1)$-dimensional vector. $g(\cdot)$ can be either linear or nonlinear dimensionality deduction mapping. In this paper, we only focus on the linear mapping. For nonlinear mapping, the simplest case is to apply a multi-layer perceptron. Similar to MHE, CoMHE also serves as a regularization penalty in neural networks.

### 3.3. Random Projection for CoMHE

Random projection is in fact one of the most straightforward way to reduce dimensionality while partially preserving the angular information. More specifically, we use a random mapping $g(\boldsymbol{v})=\frac{\boldsymbol{P}\boldsymbol{v}}{\|\boldsymbol{P}\boldsymbol{v}\|}$ where $\boldsymbol{P}\in\mathbb{R}^{(k+1)\times(d+1)}$ is a Gaussian distributed random matrix (each entry follows i.i.d. normal distribution). In order to reduce the variance of the objective value, we use multiple random projection matrix to project the neurons and compute the MHE objective separately, as shown in the following form:

$$\boldsymbol{E}_s^R(\hat{\boldsymbol{W}}_N)=\frac{1}{C}\sum_{c=1}^{C}\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}f_s\Big(\Big\|\frac{\boldsymbol{P}_c\hat{\boldsymbol{w}}_i}{\|\boldsymbol{P}_c\hat{\boldsymbol{w}}_i\|}-\frac{\boldsymbol{P}_c\hat{\boldsymbol{w}}_j}{\|\boldsymbol{P}_c\hat{\boldsymbol{w}}_j\|}\Big\|\Big) \tag{5}$$

where $\boldsymbol{P}_c,\forall c$ is random matrix with each entry satisfying the normal distribution $\mathcal{N}(0,1)$. According to the properties of normal distribution [6], every normalized row of the random matrix $\boldsymbol{P}$ is uniformly distributed on a hypersphere $\mathbb{S}^d$, which indicates that the projection matrix $\boldsymbol{P}$ is able to cover all the possible subspaces. Multiple projection matrices can also be interpreted as multi-view projection, because we are making use of information from multiple projection views. Note that, we will typically re-initialize the random projection matrices every certain number of iterations. Most importantly, using RP can provably preserve the angular similarity between different neurons.

### 3.4. Angle-preserving Projection for CoMHE

Recalling that our goal is to find a projection to project the neurons to a low-dimensional space while simultaneously preserve some angular information, we propose to explicitly transform this goal to an optimization problem:

$$\boldsymbol{P}^\star=\arg\min_{\boldsymbol{P}}\mathcal{L}_P:=\sum_{i\neq j}(\theta_{(\hat{\boldsymbol{w}}_i,\hat{\boldsymbol{w}}_j)}-\theta_{(\boldsymbol{P}\hat{\boldsymbol{w}}_i,\boldsymbol{P}\hat{\boldsymbol{w}}_j)})^2 \tag{6}$$

where $\boldsymbol{P}\in\mathbb{R}^{(k+1)\times(d+1)}$ is the projection matrix and $\theta_{(\boldsymbol{v}_1,\boldsymbol{v}_2)}$ denotes the angle between $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. For implementation convenience, we can replace the angle with the cosine value (*e.g.*, use $\cos(\theta_{(\hat{\boldsymbol{w}}_i,\hat{\boldsymbol{w}}_j)})$ to replace $\theta_{(\hat{\boldsymbol{w}}_i,\hat{\boldsymbol{w}}_j)}$), so that we can directly use the inner product of normalized vectors to measure the angular similarity. With $\hat{\boldsymbol{P}}$ obtained in Eq. (6), we use a nested loss function shown below:

$$\begin{aligned}\boldsymbol{E}_s^A(\hat{\boldsymbol{W}}_N,\boldsymbol{P}^\star)&=\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}f_s\Big(\Big\|\frac{\boldsymbol{P}^\star\hat{\boldsymbol{w}}_i}{\|\boldsymbol{P}^\star\hat{\boldsymbol{w}}_i\|}-\frac{\boldsymbol{P}^\star\hat{\boldsymbol{w}}_j}{\|\boldsymbol{P}^\star\hat{\boldsymbol{w}}_j\|}\Big\|\Big)\\ \text{s.t.}\quad\boldsymbol{P}^\star&=\arg\min_{\boldsymbol{P}}\sum_{i\neq j}(\theta_{(\hat{\boldsymbol{w}}_i,\hat{\boldsymbol{w}}_j)}-\theta_{(\boldsymbol{P}\hat{\boldsymbol{w}}_i,\boldsymbol{P}\hat{\boldsymbol{w}}_j)})^2\end{aligned} \tag{7}$$

We also propose two different ways to optimize the projection matrix $\boldsymbol{P}$. We can approximate $\boldsymbol{P}^\star$ using a few gradient descent updates. In specific, we use two different ways

to perform the optimization. Naively, we use a few gradient descent steps to update $\boldsymbol{P}$ in order to approximate $\boldsymbol{P}^\star$ and then update $\boldsymbol{W}_N$, which is proceeded alternatively. Besides the naive alternative optimization, we also try a direct optimization of $\boldsymbol{W}_N$ by unrolling the gradient update of $\boldsymbol{P}$.

**Alternative optimization.** The alternative optimization is to optimize $\boldsymbol{P}$ alternatively with the network parameters $\boldsymbol{W}_N$. Specifically, in each iteration of updating the network parameters, we update $\boldsymbol{P}$ every few inner iterations and use it as an approximation to $\boldsymbol{P}^\star$. Essentially, we are alternatively solving two optimization problems for $\boldsymbol{P}$ and $\boldsymbol{W}_N$ with gradient descent.

**Unrolled optimization.** Instead of naively updating $\boldsymbol{W}_N$ with approximate $\boldsymbol{P}^\star$ in the alternative optimization, the unrolled optimization further unrolls the update rule of $\boldsymbol{P}$ and embed it within the optimization of network parameters $\boldsymbol{W}_N$. If we denote the CoMHE loss with a given projection matrix $\boldsymbol{P}$ as $\boldsymbol{E}_s^A(\boldsymbol{W}_N, \boldsymbol{P})$ which takes $\boldsymbol{W}_N$ and $\boldsymbol{P}$ as input, then the unrolled optimization is essentially optimizing $\boldsymbol{E}_s^A(\boldsymbol{W}_N, \boldsymbol{P} - \eta \cdot \frac{\partial \mathcal{L}_P}{\partial \boldsymbol{P}})$. It can also be viewed as minimizing the CoMHE loss after a single step of gradient descent *w.r.t.* the projection matrix. This optimization includes the computation of second-order partial derivatives. Note that, it is also possible to unroll multiple gradient descent steps. Similar unrolling is also applied in [11, 23, 9].

### 3.5. Notable CoMHE Variants

We provide more interesting CoMHE variants as an extension and complement to the previous two major methods. We will have some preliminary empirical study on these variants, but our main focus is still on RP and AP.

**Group CoMHE.** Group CoMHE is a very special case in the CoMHE framework. The basic idea is to divide the weights of each neuron into several groups and then minimize the hyperspherical energy within each group. For example in CNNs, group MHE divides the channels into groups and minimizes within each group the MHE loss. Specifically, the objective function of group CoMHE is

$$\boldsymbol{E}_s^G(\hat{\boldsymbol{W}}_N) = \frac{1}{C}\sum_{c=1}^{C}\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} f_s\Big(\Big\|\frac{\boldsymbol{P}_c\hat{\boldsymbol{w}}_i}{\|\boldsymbol{P}_c\hat{\boldsymbol{w}}_i\|} - \frac{\boldsymbol{P}_c\hat{\boldsymbol{w}}_j}{\|\boldsymbol{P}_c\hat{\boldsymbol{w}}_j\|}\Big\|\Big) \quad (8)$$

where $\boldsymbol{P}_c$ is a diagonal matrix with every diagonal entry being either 0 or 1, and $\sum_c \boldsymbol{P}_c = \boldsymbol{I}$ (in fact, this is optional). There are multiple ways to divide groups for the neuron, and typically we will divide groups according to the channels, similar to [40]. More interesting, one can also divide the groups in a *stochastic* fashion.

**Mixed CoMHE.** Since all the proposed CoMHE variants and the original MHE share the same goal of diversifying the neurons on a unit hypersphere, we can in fact combine any of these MHE variants and use them together. It also has some flavor of multi-view CoMHE, since we optimize hyperspherical energy with different objectives. Experimenting all the possible combination is cumbersome and also out of the main scope of this paper, so we defer it to future investigation.

### 3.6. Shared Projection Basis in Neural Networks

In general, we usually need different projection bases for neurons in different layers of the neural network. However, we find it beneficial to share some projection bases across different layers. We only share the projection matrix for the neurons in different layers that have the same dimensionality. For example in a neural network, if the neurons in the first layer have the same dimensionality with the neurons in the second layer, we will share their projection matrix that reduces the dimensionality. Sharing the projection basis can effectively reduce the number of projection parameters and may also reduce the inconsistency within the hyperspherical energy minimization of projected neurons in different layers. Most importantly, it can further improve the empirical generalization while using much fewer parameters.

## 4. Theoretical Insights

The section mainly discusses some theoretical insights towards understanding CoMHE in terms of angular approximation accuracy and statistical intuitions.

### 4.1. Angle Preservation

We start with some highly relevant properties of random projection and then delve into the angular preservation.

**Lemma 1** (Mean Preservation of Random Projection)**.** *For any $\boldsymbol{w}_1, \boldsymbol{w}_1 \in \mathbb{R}^d$ and any random Gaussian distributed matrix $\boldsymbol{P} \in \mathbb{R}^{k\times d}$ where $\boldsymbol{P}_{ij} = \frac{1}{\sqrt{n}} r_{ij}$, if $r_{ij}, \forall i, j$ are i.i.d. random variables from $\mathcal{N}(0,1)$, we have*

$$\mathbb{E}(\langle \boldsymbol{P}\boldsymbol{w}_1, \boldsymbol{P}\boldsymbol{w}_2\rangle) = \langle \boldsymbol{w}_1, \boldsymbol{w}_2\rangle \quad (9)$$

This lemma indicates that the mean of randomly projected inner product is well preserved, partially justifying why using random projection actually makes senses.

**Lemma 2** (Johnson-Lindenstrauss Lemma [10, 18])**.** *Let $w_1, w_2 \in \mathbb{R}^d$ be vectors, and $\boldsymbol{P} \in \mathcal{R}^{k\times d}, k < d$ be a random projection matrix with entries i.i.d. drawn from a 0-mean $\sigma$-subgaussian distribution. With $\boldsymbol{P}\boldsymbol{w}_1, \boldsymbol{P}\boldsymbol{w}_2 \in \mathbb{R}^k$ being the projected vectors of $\boldsymbol{w}_1, \boldsymbol{w}_2$, then, $\forall \epsilon \in (0,1)$,*

$$\begin{aligned}(1-\epsilon)\left\|\boldsymbol{w}_1 - \boldsymbol{w}_2\right\|^2 k\sigma^2 &< \left\|\boldsymbol{P}\boldsymbol{w}_1 - \boldsymbol{P}\boldsymbol{w}_2\right\|^2 \\ &< (1+\epsilon)\left\|\boldsymbol{w}_1 - \boldsymbol{w}_2\right\|^2 k\sigma^2\end{aligned} \quad (10)$$

*holds with probability at least $1 - 2\exp(-\frac{k\epsilon^2}{8})$.*

Johnson-Lindenstrauss lemma (JLL) establishes a guarantee for the Euclidean distance between randomly projected vectors. Note that, $\mathcal{N}(0,1)$ is in fact 1-subgaussian distributed, so Gaussian random projection satisfies this lemma. However, JLL does not show anything informative about the angle preservation, and it is also nontrivial to give a tight guarantee for angular similarity from JLL.

**Theorem 1** (Angle Preservation I)**.** *Given $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{R}^d$, $\boldsymbol{P} \in \mathbb{R}^{k\times d}$ is a random projection matrix that has i.i.d. 0-mean $\sigma$-subgaussian entries, and $\boldsymbol{P}\boldsymbol{w}_1, \boldsymbol{P}\boldsymbol{w}_2 \in \mathbb{R}^k$ are the randomly projected vectors of $\boldsymbol{w}_1, \boldsymbol{w}_2$ under $\boldsymbol{P}$. Then $\forall \epsilon \in (0,1)$, we have that*

$$\frac{\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) - \epsilon}{1+\epsilon} < \cos(\theta_{(\boldsymbol{P}\boldsymbol{w}_1,\boldsymbol{P}\boldsymbol{w}_2)}) < \frac{\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) + \epsilon}{1-\epsilon} \tag{11}$$

*which holds with probability* $\left(1 - 2\exp(-\frac{k\epsilon^2}{8})\right)^2$.

**Theorem 2** (Angle Preservation II)**.** *Given $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{R}^d$, $\boldsymbol{P} \in \mathbb{R}^{k\times d}$ is a Gaussian random projection matrix where $\boldsymbol{P}_{ij} = \frac{1}{\sqrt{n}} r_{ij}$ ($r_{ij}, \forall i,j$ are i.i.d. random variables from $\mathcal{N}(0,1)$), and $\boldsymbol{P}\boldsymbol{w}_1, \boldsymbol{P}\boldsymbol{w}_2 \in \mathbb{R}^k$ are the randomly projected vectors of $\boldsymbol{w}_1, \boldsymbol{w}_2$ under $\boldsymbol{P}$. Then $\forall \epsilon \in (0,1)$ and $\boldsymbol{w}_1^\top \boldsymbol{w}_2 > 0$, we have that*

$$\begin{aligned} \frac{1+\epsilon}{1-\epsilon}\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) - \frac{2\epsilon}{1-\epsilon} &< \cos(\theta_{(\boldsymbol{P}\boldsymbol{w}_1,\boldsymbol{P}\boldsymbol{w}_2)}) \\ &< \frac{1-\epsilon}{1+\epsilon}\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) + \frac{1+2\epsilon}{1+\epsilon} - \frac{\sqrt{(1-\epsilon^2)}}{1+\epsilon} \end{aligned} \tag{12}$$

*which holds with probability* $1 - 6\exp(-\frac{k}{2}(\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3}))$.

Theorem 1 is one of our main theoretical results and reveals that the angle between randomly projected vectors is well preserved. Note that, the parameter $\sigma$ of the subgaussian distribution is not related to our bound for the angle, so any Gaussian distributed random matrix has the property of angle preservation. The projection dimension $k$ is related to the probability that the angle preservation bound holds.

Theorem 2 is a direct result from [38]. It again shows that the angle between randomly projected vectors is provably preserved. Both Theorem 1 and Theorem 2 give upper and lower bounds for the angle between randomly projected vectors. If $\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)} > \arccos(\frac{\epsilon+3\epsilon^2}{3\epsilon+\epsilon^2})$, then the lower bound in Theorem 1 is tighter than the lower bound in Theorem 2. If $\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)} > \arccos(\frac{1-3\epsilon^2-(1-\epsilon)\sqrt{1-\epsilon^2}}{3\epsilon-\epsilon^2})$, the upper bound in Theorem 1 is tighter than the upper bound in Theorem 2. In general, Theorem 1 gives tighter bounds when the angle of the original vectors is relatively large.

### 4.2. Statistical Insights

In fact, we can also draw some theoretical intuitions from sphericity testing [7] in statistics. Sphericity testing is a nonparametric statistical hypothesis test that checks whether a set of observed data is generated from a uniform distribution on a hypersphere or not. Random projection is in fact an important tool [7] in statistics to test the uniformity on hyperspheres, while our goal is to promote the same type of hyperspherical uniformity (*i.e.*, diversity). Specifically, we have $N$ random samples $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_N$ of $\mathbb{S}^d$-valued random variables, and the random projection $\boldsymbol{p}$ which is another random variable independent of $\boldsymbol{w}_i, \forall i$ and uniformly distributed on $\mathbb{S}^d$. The projected points of $\boldsymbol{w}_i, \forall i$ is $y_i = \boldsymbol{p}^\top \boldsymbol{w}_i, \forall i$. The distribution of $y_i, \forall i$ uniquely determines the distribution of $\boldsymbol{w}_1$, specified by Theorem 3.

**Theorem 3** (Unique Determination of Random Projection)**.** *Let $\boldsymbol{w}$ be a $\mathbb{S}^d$-valued random variable and $\boldsymbol{p}$ be a random variable that is uniformly distributed on $\mathbb{S}^d$ and independent of $\boldsymbol{w}$. With probability one, the distribution of $\boldsymbol{w}$ is uniquely determined by the distribution of the projection of $\boldsymbol{w}$ on $\boldsymbol{p}$. More specifically, if $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are $\mathbb{S}^d$-valued random variables, independent of $\boldsymbol{p}$ and we have a positive probability for the event that $\boldsymbol{p}$ takes a value $\boldsymbol{p}_0$ such that the two distributions satisfy $\boldsymbol{p}_0^\top \boldsymbol{w}_1 \sim \boldsymbol{p}_0^\top \boldsymbol{w}_2$, then $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are identically distributed.*

Theorem 3 shows that the distributional information is well preserved after random projection, providing the CoMHE framework a statistical intuition and foundation. We emphasize that the randomness here is in fact very crucial. For a fixed projection $\boldsymbol{p}_0$, Theorem 3 does not hold in general. As a result, random projection for CoMHE is well motivated from the statistical perspective.

## 5. Discussions

**Comparison to existing works.** One of the widely used regularizations is the orthonormal regularization [28, 3] that minimizes $\|\boldsymbol{W}^\top \boldsymbol{W} - \boldsymbol{I}\|_F$ where $W$ denotes the weights of a group of neurons with each column being one neuron and $\boldsymbol{I}$ is an identity matrix. In contrast, both MHE and CoMHE do not encourage orthogonality among neurons and instead promote hyperspherical uniformity and diversity.

**The significance of CoMHE.** CoMHE addresses some key problems about MHE, and further boosts the performance on various applications while enjoying elegant theoretical interpretations and properties. By projecting the neurons to a low-dimensional space, CoMHE is able to effectively reduce the number of stationary points and introduce more regularity to the neurons.

**Randomness helps generalization?** Both RP and AP introduce certain degree of randomness to CoMHE, and the empirical results show that such randomness can largely benefit the network generalization. It is well-known that randomness in SGD is one of the key ingredients that help deep models well generalize to unseen samples. Why CoMHE works well may also have some implicit relations to this observation. Beside this, [19] also theoretically shows that randomness can help generalization, which also partially justifies the effectiveness of CoMHE.

**Computational overhead.** One of the significant advantages of CoMHE is its savings of floating-point operations (FLOPs). Assume there are $N$ neurons with dimension $d$ and the projection matrix is of size $k \times d$. Then the FLOPs of original MHE is $N(N-1)(d+2)$, while the FLOPs of CoMHE is $2kd + N(N-1)(k+2)$. In a standard setting where $N = 512, d = 3\times 3\times 512, k = 10$, we have that MHE

has nearly 1200M FLOPs and CoMHE has only 3.32M FLOPs (even with multiple projection matrix, it is still much less than MHE), showing that CoMHE is much more efficient to compute. Both MHE and CoMHE are only involved in training stage and do not affect inference speed.

# 6. Experiments and Results

## 6.1. Image Recognition

We first perform image recognition to show the improvement of generalization by regularizing CNNs with CoMHE. Our goal is to show the superiority of CoMHE rather than achieving stat-of-the-art accuracies on particular tasks. For all the experiments on CIFAR-10 and CIFAR-100 in the paper, we use moderate data augmentation, following [14, 25]. For ImageNet-2012, we follow the same data augmentation in [28]. We train all the networks using SGD with momentum 0.9, and the network initialization follows [13]. All the networks use BN [16] and ReLU if not otherwise specified. By default, all the variants of CoMHE are built upon half-space MHE instead of full-space MHE. Network architectures and experimental details are given in each subsection and Appendix A. More experiments are put in Appendix.

### 6.1.1 Ablation Study and Exploratory Experiments

**Variants of CoMHE**. We compare different variants of CoMHE with the same plain CNN-9. In specific, we evaluate the baseline CNN without any regularization, half-space MHE (HS-MHE) which is the best MHE variant from [24], random projection CoMHE (RP-CoMHE), angle-preserving projection CoMHE (AP-CoMHE), and group CoMHE (G-CoMHE) on CIFAR-100. For RP, we set the projection dimension to 30 (*i.e.*, $k=29$) and the number of projection to 5 (*i.e.*, $C=5$). For AP, the number of projection is 1 and the projection dimension is set to 30. For AP, we evaluate both alternative optimization and unrolled optimization. In alternative optimization, we update the projection matrix every 10 steps of network update. In unrolled optimization, we only unroll one-step gradient in the optimization. For G-CoMHE, we construct a group with every 8 consecutive channels. All these design choices are obtained using cross-validation. We will also study how these hyperparameters affect the performance in the following experiments. The results in Table 1 show that our proposed RP-CoMHE and AP-CoMHE can outperform the original half-space MHE by a large margin. Quite interestingly, the unrolled optimization in AP-CoMHE shows the significant advantage over alternative one and achieves the best accuracy. Compared to HS-MHE, the performance gain of all CoMHE variants is very significant.

| Method | Error (%) |
|---|---|
| Baseline | 28.03 |
| HS-MHE [24] | 25.96 |
| G-CoMHE | 25.08 |
| RP-CoMHE | 24.39 |
| AP-CoMHE (alternative) | 24.95 |
| AP-CoMHE (unrolled) | **24.33** |

Table 1: Testing error (%m) of different CoMHE on CIFAR-100.

**Dimension of projection**. We evaluate how the dimension of projection (*i.e.*, $k$) affects the performance. We use the plain CNN-9 as the backbone network and test on CIFAR-100. We fix the number of projections in RP-CoMHE to 20. Because AP-CoMHE does not need to use multiple projections to reduce variance, we only use one projection in AP-CoMHE. Results are given in Table 2. In general, RP-CoMHE and AP-CoMHE with different projection dimensions can consistently and significantly outperform the half-space MHE, validating the effectiveness of the CoMHE framework. Specifically, we find that both RP-CoMHE and AP-CoMHE usually achieve the best accuracy when the projection dimension is 20 or 30. Since the unrolled optimization in AP-CoMHE is consistently better than the alternative optimization, we will stick to the unrolled optimization for AP-CoMHE in the remaining experiments if not otherwise specified.

| Projection Dimension | 10 | 20 | 30 | 40 | 80 |
|---|---|---|---|---|---|
| RP-CoMHE | 25.48 | 25.32 | 24.60 | **24.75** | 25.46 |
| AP-CoMHE (alter.) | **25.21** | 24.60 | 24.95 | 24.97 | **24.99** |
| AP-CoMHE (unroll.) | 25.32 | **24.59** | **24.33** | 24.93 | 25.12 |

Table 2: Error (%) on CIFAR-100 under different dimension of projection.

| # Projections | 1 | 5 | 10 | 20 | 30 | 80 |
|---|---|---|---|---|---|---|
| RP-CoMHE | 25.11 | **24.39** | 25.11 | 24.6 | 24.82 | 24.92 |
| AP-CoMHE | **24.33** | - | - | - | - | - |

Table 3: Error (%) on CIFAR-100 under different numbers of projections.

**Number of projections**. We evaluate RP-CoMHE under different numbers of projections. We use the plain CNN-9 as the baseline and test on CIFAR-100. Results in Table 3 show that the performance is generally not very sensitive to the number of projections. Surprisingly, we find that it is not necessarily better to use more projections for variance reduction. Our experiment show that using 5 projections can achieve the best accuracy. We think that it may be because large variance can somehow help the optimization escape more bad local minima. Note that, we generally do not use multiple projections in AP-CoMHE, because variance reduction is not needed in AP-CoMHE, and our empirical results do not show any noticeable performance gain by using multiple projections in AP-CoMHE.

| Width | $t=1$ | $t=2$ | $t=4$ | $t=8$ | $t=16$ |
|---|---|---|---|---|---|
| Baseline | 47.72 | 38.64 | 28.13 | 24.95 | 25.45 |
| HS-MHE [24] | 35.16 | 29.33 | 25.96 | 23.38 | 21.83 |
| RP-CoMHE | **34.73** | **28.92** | 24.39 | **22.44** | 20.81 |
| AP-CoMHE | 34.89 | 29.01 | **24.33** | 22.6 | **20.72** |

Table 4: Error (%) on CIFAR-100 under different network width.

**Network width.** We evaluate RP-CoMHE and AP-CoMHE with different network width on CIFAR-100. We use the plain CNN-9 as our backbone network architecture, and change its filter number in Conv1.x, Conv2.x

and Conv3.x (see Appendix A) to $16 \times t$, $32 \times t$ and $64 \times t$, respectively. Specifically, we test the cases where $t = 1, 2, 4, 8, 16$. For example, if $t = 2$, then the filter numbers are 32, 64 and 128, respectively. For RP, we set the projection dimension to 30 and the number of projection to 5. For AP, the number of projection is 1 and the projection dimension is set to 30. The results are shown in Table 4. Note that, we use the unrolled optimization in AP-CoMHE. From Table 4, one can observe that the accuracy gain of both RP-CoMHE and AP-CoMHE are very consistent and significant. Compared to the very strong results of half-space MHE, CoMHE can still further obtain more than 1% accuracy boost under different network width.

**Network depth.** We evaluate RP-CoMHE and AP-CoMHE with different network depth on CIFAR-100. We use three plain CNN with 6, 9 and 15 convolution layers, respectively. For all the networks, we set the filter number in Conv1.x, Conv2.x and Conv3.x to 64, 128 and 256, respectively. Detailed network architectures are given in Appendix A. For RP, we set the projection dimension to 30 and the number of projection to 5. For AP, the number of projection is 1 and the projection dimension is set to 30. The results in Table 5 show that both RP-CoMHE and AP-CoMHE can outperform half-space MHE by a considerable margin under CNN with different depth.

| Depth | CNN-6 | CNN-9 | CNN-15 |
|---|---|---|---|
| Baseline | 32.08 | 28.13 | Not Converged |
| HS-MHE [24] | 27.56 | 25.96 | 25.84 |
| RP-CoMHE | 26.73 | 24.39 | **24.21** |
| AP-CoMHE | **26.55** | **24.33** | 24.55 |

Table 5: Error (%) on CIFAR-100 under different network depth.

**Shared projection basis.** We take RP-CoMHE as an example to empirically verify the advantages of shared projection basis across different layers. We set the projection dimension to 20 and the number of projections to 30. The plain CNN-9 is used as the baseline network. For the case of shared projection basis, we share the random projection basis in Conv1.x, Conv2.x and Conv3.x separately. The shared projection case yields 24.6% error rate. For the case of independent projection basis, we use separated projection basis for different layer and only obtain 26.05% error rate. The results show that using shared random projection basis for neurons of the same dimensionality is beneficial to the network generalization. It also saves some parameters.

**Effectiveness of optimization.** In order to verify that our CoMHE can better minimize the hyperspherical energy, we compute the hyperspherical energy $\boldsymbol{E}_2$ (Eq. (1)) for HS-MHE regularized CNN, RP-CoMHE regularized CNN and AP-CoMHE regularized CNN before and after training. From Table 6, one can observe that both RP-CoMHE and AP-CoMHE can better minimize the hyperspherical energy. From the absolute scale, the optimization gain does not seem to be significant. However, this is not the case.

| Method | Baseline | HS-MHE | RP-CoMHE | AP-CoMHE |
|---|---|---|---|---|
| Beginning | **4.5470** | 4.5355 | 4.5403 | 4.5306 |
| End | 4.5485 | **4.5095** | **4.5044** | **4.5042** |

Table 6: Hyperspherical energy at the beginning and the end of the training.

In the high-dimensional space, the hyperspherical energy is usually small (close to the smallest energy value) and is already very difficult to minimize, so the improvement in the hyperspherical energy is in fact very significant.

**Naively learning projection basis from training data**. We study the case where we enable the back-propagation gradient to flow back to the projection basis. That is to say, the model learns the projection basis naively using training data. We find that naively learning the projection basis yields much worse performance (26.5%), compared to RP-CoMHE (24.6%). It is even worse than our baseline half-space MHE (25.96%). The results show that naively learning projection basis from training data leads to inferior performance. Allowing the projection basis to be updated according to the training data could undermine the strength of CoMHE regularization that is imposed on the neurons.

| # Iterations | 1 | 200 | 1000 | $\infty$ |
|---|---|---|---|---|
| RP-CoMHE | **24.6** | 24.84 | 24.62 | 26.09 |

Table 7: Error (%) with different numbers of iteration for re-initialization.

**Frequency of re-initialization in RP-CoMHE.** In RP-CoMHE, we need to re-initialize the random projections every certain number of iterations to avoid trivial solutions caused by bad initialization. Here, we test how the frequency of re-initialization will affect the accuracy, with the projection dimension being 30 and the number of projection being 20. The iteration number being $\infty$ in Table 7 represents that the random projection is fixed throughout the training once it is initialized. The results shows the performance is not very sensitive to the frequency of re-initialization, but we cannot use fixed random projection as it may cause trivial solutions and hurt the performance.

### 6.1.2 CIFAR-10 and CIFAR-100

| Method | CIFAR-10 | CIFAR-100 |
|---|---|---|
| ResNet-110-original [14] | 6.61 | 25.16 |
| ResNet-1001 [15] | 4.92 | **22.71** |
| ResNet-1001 (64 batch) [15] | **4.64** | - |
| Baseline | 5.19 | 22.87 |
| MHE [24] | 4.72 | 22.19 |
| Half-space MHE [24] | 4.66 | 22.04 |
| RP-CoMHE | 4.59 | 21.82 |
| AP-CoMHE | **4.57** | **21.63** |

Table 8: Error (%) on CIFAR-10/100 using ResNets.

All the experiments in ablation study are performed using a VGG-like plain CNN, so we use the more powerful ResNet [14] to show that CoMHE is architecture-agnostic. We use the same experimental setting in [15] for fair comparison. We use a standard ResNet-32 as our baseline and the network architecture is specified in Appendix A. From

the results in Table 8, one can observe that both RP-CoMHE and AP-CoMHE can consistently outperform half-space MHE, showing that CoMHE can boost the performance across different network architectures. More interestingly, ResNet-32 regularized by CoMHE achieves impressive accuracy and is able to outperform 1001-layer ResNet by a large margin. Additionally, we note that from Table 4, we can regularize a plain VGG-like 9-layer CNN with CoMHE and achieve 20.81% error rate, which is nearly 2% improvement over the 1001-layer ResNet.

#### 6.1.3 ImageNet-2012

We also evaluate CoMHE for large-scale image recognition on ImageNet-2012 [36]. We perform the experiment using both ResNet-18 and ResNet-34, and then report the top-1 validation error (center crop) in Table 9. Our results show consistent and significant performance gain in both ResNet-18 and ResNet-34. Compared to the baselines, CoMHE can reduce the top-1 error for more than 1%. Since the computational overhead of CoMHE is almost neglectable, the performance gain is obtained without many efforts. Most importantly, as a plug-in regularization, CoMHE is shown to be architecture-agnostic and produces considerable accuracy gain in most circumstances.

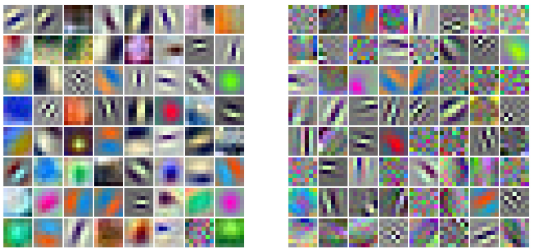
| Method | ResNet-18 | ResNet-34 |
|---|---|---|
| baseline | 32.95 | 30.04 |
| Orthogonal [35] | 32.65 | 29.74 |
| Orthnormal [28] | 32.61 | 29.75 |
| MHE [24] | 32.50 | 29.60 |
| HS-MHE [24] | 32.45 | 29.50 |
| RP-CoMHE | 31.90 | 29.38 |
| AP-CoMHE | **31.80** | **29.32** |

Table 9: Top-1 center crop error (%) on ImageNet-2012.

Besides the accuracy improvement, we also visualize in Fig. 1 the first-layer filters learned by the baseline ResNet and the CoMHE-regularized ResNet. The filters look quite different after we regularize the network using CoMHE. Each filter learned by baseline focuses on a particular local pattern (*e.g.*, edge, color and shape) and each one has a clear local semantic meaning. In contrast, filters learned by CoMHE focuses more on edges, textures and global patterns which do not necessarily have a clear local semantic meaning. However, from a representation basis perspective, it seems that having such global patterns is beneficial to the recognition accuracy. We also observe that filters learned by CoMHE does not pay too much attention to the color information, which is reasonable since humans generally do not need to use color to identify an object.

### 6.2. Point Cloud Recognition

In addition to image recognition, we apply CoMHE to improve point cloud recognition. Our goal is to validate the effectiveness of CoMHE on a different network architecture with a different form of input data structure, rather than achieving state-of-the-art performance on point cloud recognition. To this end, we conduct experiments on widely used neural networks that handles point clouds: PointNet [32] and PointNet++ [33]. We combine half-space MHE, RP-CoMHE and AP-CoMHE into PointNet (without T-Net), PointNet (with T-Net) and PointNet++. More experimental details are given in Appendix A. We evaluate the performance on ModelNet-40 [41]. Specifically, since PointNet can be viewed as $1 \times 1$ convolutions before the max pooling layer, we can apply all these MHE variants similarly to CNN. After the max pooling layer, there is a standard fully connected network where we can still apply the MHE variants. We compare the performance of regularizing PointNet and PointNet++ with half-space MHE, RP-CoMHE or AP-CoMHE. Table 10 shows that all MHE variants consistently improve PointNet and PointNet++, while RP-CoMHE and AP-CoMHE again perform the best among all. We demonstrate that CoMHE is generally useful for different kinds of neural networks, not limited to CNNs.



Baseline CoMHE

Figure 1: Visualization of the first-layer filters in ResNet.

| Method | PointNet | PointNet (T-Net) | PointNet++ |
|---|---|---|---|
| Original | 87.1 | 89.20 | 90.07 |
| HS-MHE [24] | 87.44 | 89.41 | 90.31 |
| RP-CoMHE | 87.82 | 89.69 | 90.52 |
| AP-CoMHE | **87.85** | **89.70** | **90.56** |

Table 10: Testing accuracy (%) on ModelNet-40.

## 7. Concluding Remarks

This paper first analyzes some critical problems that the original MHE [24] encounters in high-dimensional space, including increasingly more stationary points that results in bad solutions, and deterministic and small MHE gradients that lead to difficulty in optimization. To address these problems, we propose a compressive hyperspherical energy minimization framework which first projects the neurons to a low-dimensional space and minimize the hyperspherical energy for the projected neurons. The CoMHE gradients will flow from the projected space to the original space via the projection mapping and then update the original neurons. Specifically, we propose two important linear CoMHE variants: random projection CoMHE and angle-preserving CoMHE. We also provide some insights from a theoretical viewpoint. Experimental results on image recognition and point cloud recognition show the effectiveness of CoMHE.

## References


[1] J. Batle, A. Bagdasaryan, M. Abdel-Aty, and S. Abdalla. Generalized thomson problem in arbitrary dimensions and non-euclidean geometries. *Physica A: Statistical Mechanics and its Applications*, 451:237–250, 2016. 1

[2] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *SIGKDD*, 2001. 2

[3] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016. 5

[4] M. Calef, W. Griffiths, and A. Schulz. Estimating the number of stable configurations for the generalized thomson problem. *Journal of Statistical Physics*, 160(1):239–253, 2015. 1

[5] M. Cogswell, F. Ahmed, R. Girshick, L. Zitnick, and D. Batra. Reducing overfitting in deep networks by decorrelating representations. In *ICLR*, 2016. 2

[6] H. Cramér. *Mathematical methods of statistics (PMS-9)*, volume 9. Princeton university press, 2016. 3

[7] J. A. Cuesta-Albertos, A. Cuevas, and R. Fraiman. On projection-based tests for directional and compositional data. *Statistics and Computing*, 19(4):367, 2009. 5

[8] J. A. Cuesta-Albertos, R. Fraiman, and T. Ransford. A sharp form of the cramer–wold theorem. *Journal of Theoretical Probability*, 20(2):201–209, 2007. 15

[9] B. Dai, H. Dai, N. He, W. Liu, Z. Liu, J. Chen, L. Xiao, and L. Song. Coupled variational bayes via optimization embedding. In *NIPS*, 2018. 4

[10] S. Dasgupta and A. Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003. 4

[11] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 4

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1

[13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 6

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 6, 7, 11

[15] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 7

[16] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 1, 6

[17] L. Jiang, D. Meng, S.-I. Yu, Z. Lan, S. Shan, and A. Hauptmann. Self-paced learning with diversity. In *NIPS*, 2014. 2

[18] A. Kaban. Improved bounds on the dot product under random projection and random sign projection. In *KDD*, 2015. 4, 13

[19] K. Kawaguchi, B. Xie, and L. Song. Deep semi-random features for nonlinear function approximation. In *AAAI*, 2018. 2, 5

[20] L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207, 2003. 2

[21] N. Li, Y. Yu, and Z.-H. Zhou. Diversity regularized ensemble pruning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2012. 2

[22] S. Li, S. Bak, P. Carr, and X. Wang. Diversity regularized spatiotemporal attention for video-based person re-identification. In *CVPR*, 2018. 2

[23] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 4

[24] W. Liu, R. Lin, Z. Liu, L. Liu, Z. Yu, B. Dai, and L. Song. Learning towards minimum hyperspherical energy. *NeurIPS*, 2018. 1, 2, 6, 7, 8, 11, 12

[25] W. Liu, Z. Liu, Z. Yu, B. Dai, R. Lin, Y. Wang, J. M. Rehg, and L. Song. Decoupled networks. *CVPR*, 2018. 2, 6

[26] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*, 2017. 2

[27] W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, 2016. 2

[28] W. Liu, Y.-M. Zhang, X. Li, Z. Yu, B. Dai, T. Zhao, and L. Song. Deep hyperspherical learning. In *NIPS*, 2017. 2, 5, 6, 8

[29] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[30] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *ICML*, 2009. 2

[31] Z. Nehari. *Conformal mapping*. Courier Corporation, 2012. 2

[32] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 8, 11

[33] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 8, 11, 12

[34] I. Ramirez, P. Sprechmann, and G. Sapiro. Classification and clustering via dictionary learning with structured incoherence and shared features. In *CVPR*, 2010. 2

[35] P. Rodríguez, J. Gonzalez, G. Cucurull, J. M. Gonfaus, and X. Roca. Regularizing cnns with locally constrained decorrelations. In *ICLR*, 2017. 2, 8

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, pages 1–42, 2014. 8

[37] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, 2016. 1

[38] Q. Shi, C. Shen, R. Hill, and A. v. d. Hengel. Is margin preserved after random projection? *arXiv preprint arXiv:1206.4651*, 2012. 5, 13

[39] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014. 1

[40] Y. Wu and K. He. Group normalization. In *ECCV*, 2018. 1, 4

[41] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 8

[42] Y. Xiang and X. Gong. Efficiency of generalized simulated annealing. *Physical Review E*, 62(3):4473, 2000. 2

[43] Y. Xiang, D. Sun, W. Fan, and X. Gong. Generalized simulated annealing algorithm and its application to the thomson model. *Physics Letters A*, 233(3):216–220, 1997. 2

[44] 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. *arXiv:1703.01827*, 2017. 2

[45] P. Xie, Y. Deng, Y. Zhou, A. Kumar, Y. Yu, J. Zou, and E. P. Xing. Learning latent space models with angular constraints. In *ICML*, 2017. 2

[46] P. Xie, A. Singh, and E. P. Xing. Uncorrelation and evenness: a new diversity-promoting regularizer. In *ICML*, 2017. 2

[47] P. Xie, W. Wu, Y. Zhu, and E. P. Xing. Orthogonality-promoting distance metric learning: convex relaxation and theoretical analysis. In *ICML*, 2018. 2

[48] P. Xie, J. Zhu, and E. Xing. Diversity-promoting bayesian learning of latent variable models. In *ICML*, 2016. 2

# Appendix

## A. Experimental Details

| Layer | CNN-6 | CNN-9 | CNN-15 |
|---|---|---|---|
| Conv1.x | [3×3, 64]×2 | [3×3, 64]×3 | [3×3, 64]×5 |
| Pool1 | 2×2 Max Pooling, Stride 2 | | |
| Conv2.x | [3×3, 128]×2 | [3×3, 128]×3 | [3×3, 128]×5 |
| Pool2 | 2×2 Max Pooling, Stride 2 | | |
| Conv3.x | [3×3, 256]×2 | [3×3, 256]×3 | [3×3, 256]×5 |
| Pool3 | 2×2 Max Pooling, Stride 2 | | |
| Fully Connected | 256 | 256 | 256 |

Table 11: Our plain CNN architectures with different convolutional layers. Conv1.x, Conv2.x and Conv3.x denote convolution units that may contain multiple convolution layers. E.g., [3×3, 64]×3 denotes 3 cascaded convolution layers with 64 filters of size 3×3.

| Layer | ResNet-32 for CIFAR-10/100 | ResNet-18 for ImageNet-2012 | ResNet-34 for ImageNet-2012 |
|---|---|---|---|
| Conv0.x | N/A | [7×7, 64], Stride 2<br>3×3, Max Pooling, Stride 2 | [7×7, 64], Stride 2<br>3×3, Max Pooling, Stride 2 |
| Conv1.x | [3×3, 64]×1<br>$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 5$ | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$ | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$ |
| Conv2.x | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 5$ | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$ | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$ |
| Conv3.x | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 5$ | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$ | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$ |
| Conv4.x | N/A | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$ | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$ |
| | Average Pooling | | |

Table 12: Our ResNet architectures with different convolutional layers. Conv0.x, Conv1.x, Conv2.x, Conv3.x and Conv4.x denote convolution units that may contain multiple convolutional layers, and residual units are shown in double-column brackets. Conv1.x, Conv2.x and Conv3.x usually operate on different size feature maps. These networks are essentially the same as [14], but some may have a different number of filters in each layer. The downsampling is performed by convolutions with a stride of 2. E.g., [3×3, 64]×4 denotes 4 cascaded convolution layers with 64 filters of size 3×3, and S2 denotes stride 2.

**Image recognition settings.** The network architectures used in the paper are elaborated in Table 11 and Table 12. For CIFAR-10 and CIFAR-100, we use batch size 128. We start with learning rate 0.1, divide it when the performance is saturated. For ImageNet-2012, we use batch size 64 and start with learning rate 0.1. The learning rate is divided by 10 when the performance is saturated, and the training is terminated at 500k iterations. Note that, for all the compared methods, we always use the best possible hyperparameters to make sure that the comparison is fair. The baseline has exactly the same architecture and training settings as the one that CoMHE uses. For both half-space MHE and all the variants of CoMHE in hidden layers, we set the weighting hyperparameter as 1 in all experiments. [24] already shows that MHE type of losses are not senstitive to the weighting hyperparameter. We use $1e-5$ for the orthonormal regularization. If not otherwise specified, standard $\ell_2$ weight decay ($1e-4$) is applied to all the neural network including baselines and the networks that use MHE regularization. Note that, all the neuron weights in the neural networks used in the paper are not normalized (unless otherwise specified), but both MHE and CoMHE will normalize the neuron weights while computing the regularization loss. For all experiments, we use $s = 2$ in both MHE and CoMHE. As a result, *CoMHE does not need to modify any component of the original neural networks, and it can simply be viewed as an extra regularization loss that can boost the performance*.

**Point cloud recognition settings.** For all the PointNet and PointNet++ experiments, we exactly follow the same setting in the original papers [32, 33] and their official repositories[1] [2]. Specifically, we combine CoMHE regularization to neurons in all the $1 \times 1$ convolution layers before the max pooling layer and the multi-layer perceptron classifier after the max pooling layer. All the regularization is added without changing any components in PointNet. For PointNet experiments, we use point number 1024, batch size 32 and Adam optimizer started with learning rate 0.001, the learning rate will decay by 0.7 every 200k iterations, and the training is terminated at 250 epochs. For PointNet++ experiments, since the MRG (multi-resolution grouping) model is not provided in the official repository, we use the SSG (single scale grouping) model as baseline. Specifically, we use point number 1024, batch size 16 and Adam optimizer started with learning rate 0.001, the

[1] https://github.com/charlesq34/pointnet
[2] https://github.com/charlesq34/pointnet2

learning rate will decay by 0.7 every 200k iterations, and the training is terminated at 251 epochs. For all experiments, we use $s = 2$ in both MHE and CoMHE.

We evaluate on PointNet with T-Net and without T-Net in order to demonstrate that CoMHE is not sensitive to architecture modifications. We follow all the default hyperparameters used in the official released code, and the only difference is that we further combine an additional regularization loss for the neurons in each layer. One can observe that CoMHE consistently performs better than half-space MHE [24].

Besides PointNet, we combine CoMHE to PointNet++ [33] and further show the improvement of generalization introduced by CoMHE is agnostic to the architecture. We evaluate PointNet++ with and without CoMHE on ModelNet-40. Note that, we exactly follow the released code in the official repository where PointNet++ uses the single scale grouping model. Because the original paper [33] uses the multi-resolution grouping model, the baseline performance reported in our paper is not as good as the accuracy reported in the original paper. However, our purpose is to validate the effectiveness of CoMHE, so we only focus on the performance gain. One can observe that CoMHE achieves about 0.5% accuracy gain, while half-space MHE [24] only has about 0.2% accuracy gain.

## B. Proofs

In the section, we aim to provide the complete proof for self-containedness. We note that some of these proofs below are not our contributions.

### B.1. Lemma 1

We take the expectation of the inner product between projected vectors:

$$\begin{aligned}\mathbb{E}(\langle \boldsymbol{P}\boldsymbol{w}_1, \boldsymbol{P}\boldsymbol{w}_2\rangle) &= \frac{1}{n}\mathbb{E}\Big(\sum_{l=1}^{n}(\sum_{j=1}^{d} r_{lj}\{\boldsymbol{w}_1\}_j \sum_{i=1}^{d} r_{li}\{\boldsymbol{w}_2\}_i)\Big) \\ &= \frac{1}{n}\sum_{l=1}^{n}\Big(\sum_{j=1}^{d}\mathbb{E}(r_{lj}^2)\{\boldsymbol{w}_1\}_j\{\boldsymbol{w}_2\}_j + \sum_{j=1}^{d}\mathbb{E}(r_{lj})\{\boldsymbol{w}_1\}_j \cdot \sum_{i\neq j:i=1}^{d}\mathbb{E}(r_{li})\{\boldsymbol{w}_2\}_i\Big) \\ &= \langle \boldsymbol{w}_1, \boldsymbol{w}_2\rangle\end{aligned} \tag{13}$$

where $\{\boldsymbol{w}_1\}_i$ is the $i$-th element of the vector $\boldsymbol{w}_1$, and $\{\boldsymbol{w}_2\}_i$ is the $i$-th element of the vector $\boldsymbol{w}_2$. From the equation, we see that the lemma is proved. □

### B.2. Theorem 1

Before proving the the main theorem, we first show a lemma from [18].

**Lemma 3** (Dot Product under Random Projection)**.** *Let $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{R}^d$, $\boldsymbol{P} \in \mathbb{R}^{k\times d}, k < d$ be a random projection matrix having i.i.d. 0-mean subgaussian entries with parameter $\sigma^2$, and $\boldsymbol{P}\boldsymbol{w}_1, \boldsymbol{P}\boldsymbol{w}_2$ be the images of $\boldsymbol{w}_1, \boldsymbol{w}_2$ under projection $\boldsymbol{P}$. Then, $\forall\epsilon \in (0,1)$:*

$$\boldsymbol{w}_1^\top\boldsymbol{w}_2 k\sigma^2 - \epsilon k\sigma^2\|\boldsymbol{w}_1\|\|\boldsymbol{w}_2\| < (\boldsymbol{P}\boldsymbol{w}_1)^\top\boldsymbol{P}\boldsymbol{w}_2 < \boldsymbol{w}_1^\top\boldsymbol{w}_2 k\sigma^2 + \epsilon k\sigma^2\|\boldsymbol{w}_1\|\|\boldsymbol{w}_2\| \tag{14}$$

*holds with probability* $1 - 2\exp(-\frac{k\sigma^2}{8})$.

From Lemma 2, we have that

$$\begin{aligned}(1-\epsilon)\|\boldsymbol{w}_1\|^2 k\sigma^2 < \|\boldsymbol{P}\boldsymbol{w}_1\|^2 < (1+\epsilon)\|\boldsymbol{w}_1\|^2 k\sigma^2 \\ (1-\epsilon)\|\boldsymbol{w}_2\|^2 k\sigma^2 < \|\boldsymbol{P}\boldsymbol{w}_2\|^2 < (1+\epsilon)\|\boldsymbol{w}_2\|^2 k\sigma^2\end{aligned} \tag{15}$$

which holds with probability $\big(1 - 2\exp(-\frac{k\epsilon^2}{8})\big)^2$.

Then we combine Eq. (15) to Lemma 3 and obtain that

$$\frac{\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) - \epsilon}{1+\epsilon} < \cos(\theta_{(\boldsymbol{P}\boldsymbol{w}_1,\boldsymbol{P}\boldsymbol{w}_2)}) < \frac{\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) + \epsilon}{1-\epsilon} \tag{16}$$

which holds with probability $\big(1 - 2\exp(-\frac{k\epsilon^2}{8})\big)^2$. $\theta_{(\boldsymbol{P}\boldsymbol{w}_1,\boldsymbol{P}\boldsymbol{w}_2)}$ denotes the angle between $\boldsymbol{P}\boldsymbol{w}_1$ and $\boldsymbol{P}\boldsymbol{w}_2$, and $\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}$ denotes the angle between $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$. □

### B.3. Theorem 2

We reorganize the original proof in [38] below. Before proving our main theorem, we first show a lemma below:

**Lemma 4.** *For any $w \in \mathbb{R}^d$, any random Gaussian matrix $\boldsymbol{P} \in \mathbb{R}^{k\times d}$ where $\boldsymbol{P}_{ij} = \frac{1}{\sqrt{n}}r_{ij}$ and $r_{ij}, \forall i,j$ are i.i.d. random variables from $\mathcal{N}(0,1)$, and $\epsilon \in (0,1)$*

$$\Pr\Big((1-\epsilon) \le \frac{\|\boldsymbol{P}\boldsymbol{w}\|^2}{\|\boldsymbol{w}\|^2} \le (1+\epsilon)\Big) \ge 1 - 2\exp\big(-\frac{n}{2}(\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3})\big) \tag{17}$$

*Proof of Lemma 4.* From Lemma 1, we have that $\mathbb{E}(\|\boldsymbol{P}\boldsymbol{w}\|^2) = \|\boldsymbol{w}\|^2$. Due to 2-stability of the Gaussian distribution, we have that $\sum_{j=1}^{d} r_{lj} w_j = \|\boldsymbol{w}\| z_l$ where $z_l \sim \mathcal{N}(0,1)$. As a result, we have that

$$\|\boldsymbol{P}\boldsymbol{w}\|^2 = \frac{1}{n}\boldsymbol{w}^2 \sum_{l=1}^{n} z_l^2 \tag{18}$$

where $\sum_{l=1}^{n} z_l^2$ is chi-square distributed with $n$-degree freedom. Then we apply the standard tail bound of the chi-square distribution and obtain

$$\begin{aligned} \Pr\Big( \|\boldsymbol{P}\boldsymbol{w}\|^2 \le (1-\epsilon)\left\|\boldsymbol{w}^2\right\| \Big) &\le \exp\left(\frac{n}{2}\big(1-(1-\epsilon)+\ln(1-\epsilon)\big)\right) \\ &\le \exp(-\frac{n}{4}\epsilon^2) \end{aligned} \tag{19}$$

where the inequality $\ln(1-\epsilon) \le -\epsilon - \frac{\epsilon^2}{2}$ is applied. Similarly, one can have

$$\begin{aligned} \Pr\Big( \|\boldsymbol{P}\boldsymbol{w}\|^2 \le (1+\epsilon)\left\|\boldsymbol{w}^2\right\| \Big) &\le \exp\left(\frac{n}{2}\big(1-(1+\epsilon)+\ln(1+\epsilon)\big)\right) \\ &\le \exp(-\frac{n}{2}(\frac{\epsilon^2}{2}-\frac{\epsilon^3}{3})) \end{aligned} \tag{20}$$

where the inequality $\ln(1+\epsilon) \le \epsilon - \frac{\epsilon^2}{2} + \frac{\epsilon^3}{3}$ is used. □

From the lemma above, we apply the union bound and have that

$$\begin{aligned} (1-\epsilon) \le \frac{\|\boldsymbol{P}\boldsymbol{w}_1\|^2}{\|\boldsymbol{w}_1\|^2} \le (1+\epsilon) \\ (1-\epsilon) \le \frac{\|\boldsymbol{P}\boldsymbol{w}_2\|^2}{\|\boldsymbol{w}_2\|^2} \le (1+\epsilon) \end{aligned} \tag{21}$$

which holds with probability at least $1 - 4\exp(-\frac{n}{2}(\frac{\epsilon^2}{2}-\frac{\epsilon^3}{3}))$. Using Eq. (21), we can have that

$$\left\| \frac{\boldsymbol{P}\boldsymbol{w}_1}{\|\boldsymbol{P}\boldsymbol{w}_1\|} - \frac{\boldsymbol{P}\boldsymbol{w}_2}{\|\boldsymbol{P}\boldsymbol{w}_2\|} \right\|^2 \le \left\| \frac{\boldsymbol{P}\boldsymbol{w}_1}{\sqrt{1-\epsilon}\,\|\boldsymbol{w}_1\|} - \frac{\boldsymbol{P}\boldsymbol{w}_2}{\sqrt{1-\epsilon}\,\|\boldsymbol{w}_2\|} \right\|^2 \tag{22}$$

From Eq. (21) and the condition that $\boldsymbol{w}_1^\top \boldsymbol{w}_2 > 0$, we further have that

$$\begin{aligned} \left\| \frac{\boldsymbol{P}\boldsymbol{w}_1}{\|\boldsymbol{w}_1\|} - \frac{\boldsymbol{P}\boldsymbol{w}_2}{\|\boldsymbol{w}_2\|} \right\|^2 &\le \left\|\sqrt{1+\epsilon} - \sqrt{1-\epsilon}\right\|^2 \\ &\le \left\| \sqrt{1+\epsilon}\left( \frac{\boldsymbol{P}\boldsymbol{w}_1}{\|\boldsymbol{P}\boldsymbol{w}_1\|} - \frac{\boldsymbol{P}\boldsymbol{w}_2}{\|\boldsymbol{P}\boldsymbol{w}_2\|} \right) \right\|^2 + \left\|\sqrt{1+\epsilon} - \sqrt{1-\epsilon}\right\|^2 \end{aligned} \tag{23}$$

Then we apply Lemma 4 to the vector $(\frac{\boldsymbol{w}_1}{\|\boldsymbol{w}_1\|} - \frac{\boldsymbol{w}_2}{\|\boldsymbol{w}_2\|})$ and see that

$$(1-\epsilon)\left\| \frac{\boldsymbol{w}_1}{\|\boldsymbol{w}_1\|} - \frac{\boldsymbol{w}_2}{\|\boldsymbol{w}_2\|} \right\|^2 \le \left\| \frac{\boldsymbol{P}\boldsymbol{w}_1}{\|\boldsymbol{w}_1\|} - \frac{\boldsymbol{P}\boldsymbol{w}_2}{\|\boldsymbol{w}_2\|} \right\|^2 \le (1+\epsilon)\left\| \frac{\boldsymbol{w}_1}{\|\boldsymbol{w}_1\|} - \frac{\boldsymbol{w}_2}{\|\boldsymbol{w}_2\|} \right\|^2 \tag{24}$$

which holds with probability $1 - 2\exp\big(-\frac{n}{2}(\frac{\epsilon^2}{2}-\frac{\epsilon^3}{3})\big)$. Then we have that

$$\begin{aligned} \frac{\langle \boldsymbol{w}_1, \boldsymbol{w}_2 \rangle}{\|\boldsymbol{w}_1\|\,\|\boldsymbol{w}_2\|} &= 1 - \frac{1}{2}\left\| \frac{\boldsymbol{w}_1}{\|\boldsymbol{w}_1\|} - \frac{\boldsymbol{w}_2}{\|\boldsymbol{w}_2\|} \right\|^2, \\ \frac{\langle \boldsymbol{P}\boldsymbol{w}_1, \boldsymbol{P}\boldsymbol{w}_2 \rangle}{\|\boldsymbol{P}\boldsymbol{w}_1\|\,\|\boldsymbol{P}\boldsymbol{w}_2\|} &= 1 - \frac{1}{2}\left\| \frac{\boldsymbol{P}\boldsymbol{w}_1}{\|\boldsymbol{P}\boldsymbol{w}_1\|} - \frac{\boldsymbol{P}\boldsymbol{w}_2}{\|\boldsymbol{P}\boldsymbol{w}_2\|} \right\|^2. \end{aligned} \tag{25}$$

From Eq. (22), Eq. (23) and Eq. (24), we can learn that $\left\| \frac{\boldsymbol{P}\boldsymbol{w}_1}{\|\boldsymbol{P}\boldsymbol{w}_1\|} - \frac{\boldsymbol{P}\boldsymbol{w}_2}{\|\boldsymbol{P}\boldsymbol{w}_2\|} \right\|^2$ is bounded below and above. Further combining Eq. (25), we have that

$$\frac{1+\epsilon}{1-\epsilon}\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) - \frac{2\epsilon}{1-\epsilon} < \cos(\theta_{(\boldsymbol{P}\boldsymbol{w}_1,\boldsymbol{P}\boldsymbol{w}_2)}) < \frac{1-\epsilon}{1+\epsilon}\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) + \frac{1+2\epsilon}{1+\epsilon} - \frac{\sqrt{(1-\epsilon^2)}}{1+\epsilon} \tag{26}$$

where $\theta_{(\boldsymbol{P}\boldsymbol{w}_1,\boldsymbol{P}\boldsymbol{w}_2)}$ denotes the angle between $\boldsymbol{P}\boldsymbol{w}_1$ and $\boldsymbol{P}\boldsymbol{w}_2$, and $\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}$ denotes the angle between $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$. □

### B.4. Theorem 3

If we consider $\boldsymbol{w} \in \mathbb{R}^d$ as a bounded variable, and without loss of generality, we assume that $\boldsymbol{p} = \boldsymbol{z}/\|\boldsymbol{z}\|$ where $\boldsymbol{z}$ is Gaussian distributed, and then using Theorem 4.1 in [8] (shown in Lemma 5), we can easily have the desired result.

**Lemma 5.** *Let $\mathcal{H}$ be a separable Hilbert space, and let $\mu$ be a non-degenerate Gaussian measure on $\mathcal{H}$. Let $P, Q$ be Borel probability measures on $\mathcal{H}$. Assume that:*

- *The abosolute moments $m_n := \int \|x\|^n \, dP(x)$ are finite and satisfy $\sum_{n\geq 1} m_n^{\frac{-1}{n}} = \infty$;*
- *The set $\varepsilon(P,Q) := \{x \in \mathcal{H} : P_{\langle x\rangle} = Q_{\langle x\rangle}\}$, where $\langle x\rangle$ denotes the one-dimensional subspace spanned by $x$, is of positive $\mu$-measure.*

*Then we have $P = Q$.*

□