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 Tech [3]New York University [4]NVIDIA * Equal Contribution

ABSTRACT

Minimum hyperspherical energy (MHE) has demonstrated its potential in regularizing neural networks and improving the 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 its practical effectiveness, MHE suffers from some difficulties in optimization as the dimensionality of the space becomes higher, therefore limiting the potential to improve network generalization. To address these problems, we propose the compressive minimum hyperspherical energy (CoMHE) as a more effective 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 mapping, we propose two major variants: random projection CoMHE and angle-preserving CoMHE. As a novel extension, We further consider adversarial projection CoMHE and group CoMHE. We also provide some theoretical insights to justify the effectiveness. Our comprehensive experiments show that CoMHE consistently outperforms MHE by a considerable margin, and can be easily applied to improve different 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 [1–3], but their strong approximation ability also makes 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, and instead it regularizes the networks in an implicit manner in order to prevent overfitting and stabilize the training dynamics. A lot of classic methods fall into this category, such as batch normalization [4], dropout [5], weight normalization [6], group normalization [7], etc. Explicit regularization [8–13] 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) [13] 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* [13]). 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 [14, 15]. 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, when the number of neurons is smaller than the dimension of the space (it is often the case in neural networks), MHE defines an ill-posed problem because the neurons can not even fully occupy the space. It will be less meaningful in this case to encourage the hyperspherical diversity. Last, 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. The high-level intuition behind CoMHE is to project neurons to various spaces whose dimensions are significantly smaller than the original one such that the hyperspherical energy can get minimized more effectively. Specifically, CoMHE first maps 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 map the neuron weights to a low-dimensional space while preserving some of the desirable information in high-dimensional space is our major focus. Since MHE regualrizes the directions of neurons, the most important information we care in MHE is the angular similarity between different neuron weights. To this end, we explore multiple novel methods to perform the projection propose and propose two main approaches: *random projection* and *angle-preserving projection*, which can reduce the dimensionality of neurons while still partially preserving the angles.

Random projection (RP) [16] 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 [17, 18] 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. Building upon 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 preserve the angles is usually very difficult even with powerful non-linear functions, which is suggested by the strong conditions required for conformal mapping in complex analysis [20]. Therefore, we adopt a different approach to construct AP by framing the AP construction as an optimization problem which can be solved jointly with the neural network. Furthermore, we propose the *adversarial projection* for CoMHE, which minimizes the maximal energy attained by learning the projection. We formulate it as a min-max optimization and solve it jointly with the neural network.

However, it is inevitable to lose some information in the high-dimensional space and the neurons may only get diverse in some low-dimensional spaces. To address this, we introduce multiple projections to approximate the learning objective of MHE in the original high-dimensional space more accurately. Specifically, we project the neurons to multiple subspaces, compute the MHE loss in each space separately and finally minimize the aggregation (*i.e.*, average or max) of these MHE losses. Besides, we also reinitialize these projection matrix randomly every certain number of iterations to avoid trivial solutions. Our empirical study shows that CoMHE with RP or AP performs consistently better than the standard MHE in nearly all applications. Our contributions can be summarized as:

- To address the drawbacks of MHE, we propose CoMHE as a generic regularization to effectively minimize hyperspherical energy of neurons for better generalizability of neural networks.
- We explore a number of ways to construct the projection for dimensionality reduction. Random projection and angle-preserving projection are proposed to achieve the dimensionality reduction of neurons while being able to partially preserve the angular information in MHE. Moreover, we consider several interesting variants such as adversarial projection CoMHE and group CoMHE.
- We provide partial theoretical insights to show that some projections used in the paper can well preserve the angular similarity between different neurons.
- We apply CoMHE to image recognition and point cloud recognition, and demonstrate consistent and significant improvement over the standard MHE. Notably, a 9-layer plain CNN regularized by CoMHE outperforms a 1001-layer ResNet by nearly 2% on CIFAR-100.

2 Related Work

Diversity-based regularization has been found useful in sparse coding [21, 22], ensemble learning [23, 24], self-paced learning [25], metric learning [26], latent variable models [27], person re-identification [28], face recognition [13], etc. There are a number of ways to characterize diversity in previous works. Early studies in sparse coding [21, 22] model the diversity with the empirical covariance matrix and show that the generalization capability of dictionary can be improved by regularizing its diversity. [29] promotes the uniformity among eigenvalues of the component matrix in a latent space model. [30] encourages the diversity of latent space models by imposing constraints to the pairwise angle among components. [9, 10, 31–33, 33] characterize diversity among neurons with orthogonality, and regularize the neural network using the orthogonality. Inspired by the Thomson problem in physics, MHE [13] 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 [13]: full-space MHE and half-space MHE. Compared to full-space MHE, the half-space variant [13] 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 discussed in [34–41].

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

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

where $\|\cdot\|$ denotes ℓ_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, we write down the gradient of $\boldsymbol{E}_2$ *w.r.t* $\hat{\boldsymbol{w}}_i$ and make the gradient to be zero:

$$\nabla_{\hat{\boldsymbol{w}}_i} \boldsymbol{E}_2 = \sum_{j=1, j\neq i}^N \frac{-2(\hat{\boldsymbol{w}}_i - \hat{\boldsymbol{w}}_j)}{\|\hat{\boldsymbol{w}}_i - \hat{\boldsymbol{w}}_j\|^4} \xRightarrow{\nabla_{\hat{\boldsymbol{w}}_i} \boldsymbol{E}_2 = 0} \hat{\boldsymbol{w}}_i = \frac{\sum_{j=1, j\neq i}^N \alpha_j \hat{\boldsymbol{w}}_j}{\sum_{j=1, j\neq i}^N \alpha_j}, \tag{2}$$

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. 2, 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, we can extend the solutions in $(d+1)$-dimensional space by introducing a new dimension with zero value. The new solutions satisfy Eq. 2. 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. 2. Moreover, the order of the elements in $\hat{\boldsymbol{w}}_j^E$ does not matter for Eq. 2.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 MHE's drawbacks in high dimension, we propose the compressive MHE that projects the neurons to a low-dimensional space and then minimizes the hyperspherical energy of the projected neurons. In general, CoMHE aims to minimize the following form of hyperspherical energy:

$$\boldsymbol{E}_s^C(\hat{\boldsymbol{W}}_N) = \sum_{i=1}^N \sum_{j=1, j\neq i}^N f_s\Big(\|g(\hat{\boldsymbol{w}}_i) - g(\hat{\boldsymbol{w}}_j)\|\Big) \tag{3}$$

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 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, shown as follows:

$$\boldsymbol{E}_s^R(\hat{\boldsymbol{W}}_N)=\frac{1}{C}\sum_{c=1}^{C}\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} f_s\Big(\Big\|\frac{\boldsymbol{P}_c\hat{\boldsymbol{w}}_i}{\|\boldsymbol{P}_c\hat{\boldsymbol{w}}_i\|}-\frac{\boldsymbol{P}_c\hat{\boldsymbol{w}}_j}{\|\boldsymbol{P}_c\hat{\boldsymbol{w}}_j\|}\Big\|\Big) \tag{4}$$

where $\boldsymbol{P}_c,\forall c$ is a random matrix with each entry satisfying the normal distribution $\mathcal{N}(0,1)$. According to the properties of normal distribution [42], 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. In fact, we do not necessarily need to average the energy for multiple projections, and instead we can use maximum operation (or some other meaningful aggregation operations). Then the objective becomes $\max_c\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} f_s(\|\frac{\boldsymbol{P}_c\hat{\boldsymbol{w}}_i}{\|\boldsymbol{P}_c\hat{\boldsymbol{w}}_i\|}-\frac{\boldsymbol{P}_c\hat{\boldsymbol{w}}_j}{\|\boldsymbol{P}_c\hat{\boldsymbol{w}}_j\|}\|)$. Considering that we aim to minimize this objective, the problem is in fact a simple min-max optimization. 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

Recall that our goal is to find a projection to project the neurons to a low-dimensional space while simultaneously preserving angular information. We transform the goal to an optimization problem:

$$\boldsymbol{P}^\star=\arg\min_{\boldsymbol{P}}\mathcal{L}_P:=\sum_{i\neq j}\big(\theta_{(\hat{\boldsymbol{w}}_i,\hat{\boldsymbol{w}}_j)}-\theta_{(\boldsymbol{P}\hat{\boldsymbol{w}}_i,\boldsymbol{P}\hat{\boldsymbol{w}}_j)}\big)^2 \tag{5}$$

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. 5, we use a nested loss function:

$$\begin{gathered}\boldsymbol{E}_s^A(\hat{\boldsymbol{W}}_N,\boldsymbol{P}^\star)=\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} f_s\Big(\Big\|\frac{\boldsymbol{P}^\star\hat{\boldsymbol{w}}_i}{\|\boldsymbol{P}^\star\hat{\boldsymbol{w}}_i\|}-\frac{\boldsymbol{P}^\star\hat{\boldsymbol{w}}_j}{\|\boldsymbol{P}^\star\hat{\boldsymbol{w}}_j\|}\Big\|\Big)\\ \text{s.t.}\quad \boldsymbol{P}^\star=\arg\min_{\boldsymbol{P}}\sum_{i\neq j}\big(\theta_{(\hat{\boldsymbol{w}}_i,\hat{\boldsymbol{w}}_j)}-\theta_{(\boldsymbol{P}\hat{\boldsymbol{w}}_i,\boldsymbol{P}\hat{\boldsymbol{w}}_j)}\big)^2\end{gathered} \tag{6}$$

for which we propose two different ways to optimize the projection matrix $\boldsymbol{P}$. We can approximate $\boldsymbol{P}^\star$ using a few gradient descent updates. Specifically, 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 proceeds alternatively. The number of iteration steps that we use to update $\boldsymbol{P}$ is a hyperparemter and needs to be determined by cross-validation. Besides the naive alternative one, we also use a different 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 number of inner iterations and use it as an approximation to $\boldsymbol{P}^\star$ (the error depends on the number of gradient steps we take). Essentially, we are alternatively solving two separate 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 [43–45].

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.

Adversarial Projection for CoMHE. We consider a very novel CoMHE variant that adversarially learns the projection. The intuition behind is that we want to learn a projection basis that maximizes the hyperspherical energy while the final goal is to minimize this maximal energy. With such intuition, we can naturally construct a min-max optimization problem:

$$\min_{\hat{\boldsymbol{W}}_N}\max_{\boldsymbol{P}}\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} f_s\Big(\Big\|\frac{\boldsymbol{P}\hat{\boldsymbol{w}}_i}{\|\boldsymbol{P}\hat{\boldsymbol{w}}_i\|}-\frac{\boldsymbol{P}\hat{\boldsymbol{w}}_j}{\|\boldsymbol{P}\hat{\boldsymbol{w}}_j\|}\Big\|\Big) \tag{7}$$

which can be solved by gradient descent similar to [46]. From a game-theoretical perspective, $\boldsymbol{P}$ and $\hat{\boldsymbol{W}}_N$ can be viewed as two players that are competing with each other. However, due to the instability of solving the min-max problem, the performance of adversarial projection is also unstable. We will empirically evaluate this approach in experiments.

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

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

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 neurons, and typically we will divide groups according to the channels, similar to [7]. More interestingly, 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 learning [47], 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 empirically improve the network generalizability (as shown in our experiment) while using much fewer parameters and saving more computational overheads.

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 analysis of 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 $\mathbb{E}(\langle\boldsymbol{P}\boldsymbol{w}_1,\boldsymbol{P}\boldsymbol{w}_2\rangle)=\langle\boldsymbol{w}_1,\boldsymbol{w}_2\rangle$.*

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 [48, 49]). *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 σ-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, $\forall\epsilon \in (0,1)$,*

$$(1-\epsilon)\left\|\boldsymbol{w}_1 - \boldsymbol{w}_2\right\|^2 k\sigma^2 < \left\|\boldsymbol{P}\boldsymbol{w}_1 - \boldsymbol{P}\boldsymbol{w}_2\right\|^2 < (1+\epsilon)\left\|\boldsymbol{w}_1 - \boldsymbol{w}_2\right\|^2 k\sigma^2 \tag{9}$$

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 a 1-subgaussian distribution, 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 useful 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 σ-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*

$$\frac{\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) - \epsilon}{1+\epsilon} < \cos(\theta_{(\boldsymbol{P}\boldsymbol{w}_1,\boldsymbol{P}\boldsymbol{w}_2)}) < \frac{\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) + \epsilon}{1-\epsilon} \tag{10}$$

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*

$$\frac{1+\epsilon}{1-\epsilon}\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) - \frac{2\epsilon}{1-\epsilon} < \cos(\theta_{(\boldsymbol{P}\boldsymbol{w}_1,\boldsymbol{P}\boldsymbol{w}_2)}) < \frac{1-\epsilon}{1+\epsilon}\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) + \frac{1+2\epsilon}{1+\epsilon} - \frac{\sqrt{(1-\epsilon^2)}}{1+\epsilon} \tag{11}$$

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 σ 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 [50]. 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. To conclude, Theorem 1 gives tighter bounds when the angle of original vectors is large.

Since AP is randomly initialized every certain number of iterations and learned to minimize the angular difference before and after the projection, AP usually performs better than RP in preserving angles. Without the angle-preserving optimization, AP reduces to RP.

4.2 Statistical Insights

In fact, we can also draw some theoretical intuitions from spherical uniform testing [51] in statistics. Spherical uniform 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 [51] 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 $\boldsymbol{w}_1$, which is specified by Theorem 3.

Theorem 3 (Unique Distribution 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 [35, 52] 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. [10, 32] are also built upon orthogonality. 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 in 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 $N=512, d=3\times3\times512, k=10$, 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 used in training and do not affect inference speed.

6 Experiments and Results

6.1 Image Recognition

We perform image recognition to show the improvement of 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 the same data augmentation as [1, 37]. For ImageNet-2012, we use the same data augmentation in [35]. We train all the networks using SGD with momentum 0.9. All the networks use BN [4] and ReLU if not otherwise specified. By default, all the variants of CoMHE are built upon half-space MHE. More experimental details are given in each subsection and Appendix A.

6.1.1 Ablation Study and Exploratory Experiments

Method	Error (%)
Baseline	28.03
HS-MHE [13]	25.96
G-CoMHE	25.08
Adv-CoMHE	25.09
RP-CoMHE	24.39
RP-CoMHE (max)	24.77
AP-CoMHE (alter.)	24.95
AP-CoMHE (unroll)	**24.33**

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

Variants of CoMHE. We compare different variants of CoMHE with the same plain CNN-9 (see Appendix A). Specifically, we evaluate the baseline CNN without any regularization, half-space MHE (HS-MHE) which is the best MHE variant from [13], random projection CoMHE (RP-CoMHE), RP-CoMHE (max) that uses max instead of average for loss aggregation, angle-preserving projection CoMHE (AP-CoMHE), adversarial projection CoMHE (Adv-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 all of our proposed CoMHE variants can outperform the original half-space MHE by a large margin. The unrolled optimization in AP-CoMHE shows the significant advantage over alternative one and achieves the best accuracy. Both Adv-CoMHE and G-CoMHE achieve decent performance gain over HS-MHE, but not as good as RP-CoMHE and AP-CoMHE. Therefore, we will mostly focus on RP-CoMHE and AP-CoMHE in the remaining experiments.

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.

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 and superiority of the proposed 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.

# 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 of RP-CoMHE 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. It might be because large variance can somehow help the solution to escape more bad local minima in the optimization. Note that, we generally do not use multiple projections in AP-CoMHE, because AP-CoMHE learns the projection and variance reduction is unnecessary. Moreover, 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 [13]	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 with 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 set 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 $t=1,2,4,8,16$. Taking $t=2$ as an example, then the filter numbers in Conv1.x, Conv2.x and Conv3.x 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 performance gains of both RP-CoMHE and AP-CoMHE are very consistent and significant. With wider network, CoMHE also achieves better accuracy. Compared to the strong results of half-space MHE, CoMHE still obtains more than 1% accuracy boost under different network width.

Depth	CNN-6	CNN-9	CNN-15
Baseline	32.08	28.13	Not Converged
HS-MHE [13]	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 with different network depth.

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 while regularizing a plain CNN with different 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 baseline. Specifically for shared projection basis, we share the random projection basis in Conv1.x, Conv2.x and Conv3.x separately. The shared projection yields 24.6% error rate. For independent projection basis, we use separated projection basis for different layers and only obtain 26.05% error rate. The results show that using shared random projection basis for neurons of the same dimensionality improves the network generalization while saving parameters. Note that, all the other experiments use shared projection basis by default.

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 basedline CNN and CNN regularized by orthogonal regularization, HS-MHE, RP-CoMHE and AP-CoMHE during training. Note that, we compute the original hyperspherical energy rather than the energy after projection. All the networks use exactly the same initialization (the initial hyperspherical energy is the same). The results are averaged over five independent runs. We show the hyperspherical energy after the 20000-th iteration, because at the begining of the training, hyperspherical energy fluctuates dramatically and is unstable. Interested readers can refer to Appendix C for the complete energy dynamics. From Fig. 1, one can observe that both RP-CoMHE and AP-CoMHE can better minimize the hyperspherical energy. RP-CoMHE can achieve the lowest energy with smallest standard deviation. From the absolute scale, the optimization gain does not seem to be significant. However, this is not the case. In the high-dimensional space, the variance of hyperspherical energy is usually small (already close to the smallest energy value) and is already very difficult to minimize, so the improvement achieved by our CoMHE is in fact very significant.

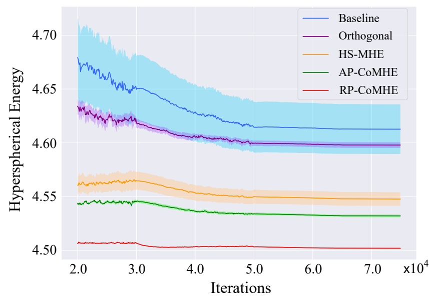

Figure 1: Hyperspherical energy during training. All networks are initialized with the same random weights, so the hyperpsherical energy is the same before the training starts.

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 imposed on the neurons.

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 on CIFAR-100, with the projection dimension being 30 and the number of projection being 20. The iteration number being ∞ in Table 6 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 fix the random projection during training as it may cause trivial solutions and hurt the performance.

# Iterations	1	200	1000	∞
RP-CoMHE	**24.6**	24.84	24.62	26.09

Table 6: Err. of different # iterations for re-initialization.

6.1.2 CIFAR-10 AND CIFAR-100

All the experiments in ablation study are performed using a VGG-like plain CNN, so we use the more powerful ResNet [1] to show that CoMHE is architecture-agnostic. We use the same experimental setting in [53] 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 7, 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

Method	CIFAR-10	CIFAR-100
ResNet-110-original [1]	6.61	25.16
ResNet-1001 [53]	4.92	**22.71**
ResNet-1001 (64 batch) [53]	**4.64**	-
Baseline	5.19	22.87
MHE [13]	4.72	22.19
Half-space MHE [13]	4.66	22.04
RP-CoMHE	4.59	21.82
AP-CoMHE	**4.57**	**21.63**

Table 7: Error (%) on CIFAR-10/100 using ResNets.

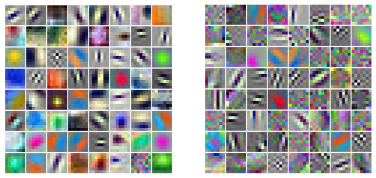

Figure 2: Visualization of the first-layer filters in ResNet.

Method	ResNet-18	ResNet-34	ResNet-50
baseline	32.95	30.04	25.30
Orthogonal [32]	32.65	29.74	25.19
Orthnormal [35]	32.61	29.75	25.21
MHE [13]	32.50	29.60	25.02
HS-MHE [13]	32.45	29.50	24.98
RP-CoMHE	31.90	29.38	**24.51**
AP-CoMHE	**31.80**	**29.32**	24.53

Table 8: Top-1 center crop error (%) on ImageNet-2012.

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 evaluate CoMHE for large-scale image recognition on ImageNet-2012 [54]. We perform the experiment using ResNet-18, ResNet-34 and ResNet-50, and then report the top-1 validation error (center crop) in Table 8. Our results show consistent and significant performance gain of CoMHE in all ResNet variants. 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.

Besides the accuracy improvement, we also visualize in Fig. 2 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

Method	PointNet	PointNet (T)	PointNet++
Original	87.1	89.20	90.07
HS-MHE [13]	87.44	89.41	90.31
RP-CoMHE	87.82	89.69	90.52
AP-CoMHE	**87.85**	**89.70**	**90.56**

Table 9: Testing accuracy (%) on ModelNet-40.

Besides 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 [55] and PointNet++ [56]. 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 [57]. Specifically, since PointNet can be viewed as 1×1 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 9 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.

7 Concluding Remarks

This paper first analyzes some critical problems that the original MHE [13] encounters in high-dimensional space. To address these problems, we propose a compressive hyperspherical energy minimization framework which first projects the neurons to a low-dimensional space and minimizes 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 CoMHE variants: random projection CoMHE and angle-preserving CoMHE. We also provide some theoretical insights for CoMHE. Experiments on image recognition and point cloud recognition show the effectiveness of CoMHE.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 7, 9, 14

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[3] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 1, 7

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014. 1

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

[7] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. 1, 5

[8] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013. 1

[9] Dmytro Mishkin and Jiri Matas. All you need is a good init. In *ICLR*, 2016. 3

[10] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep networks? In *NeurIPS*, 2018. 3, 7

[11] Pau Rodríguez, Jordi Gonzalez, Guillem Cucurull, Josep M Gonfaus, and Xavier Roca. Regularizing cnns with locally constrained decorrelations. *arXiv preprint arXiv:1611.01967*, 2016.

[12] Lei Huang, Xianglong Liu, Bo Lang, Adams Wei Yu, Yongliang Wang, and Bo Li. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *AAAI*, 2018.

[13] Weiyang Liu, Rongmei Lin, Zhen Liu, Lixin Liu, Zhiding Yu, Bo Dai, and Le Song. Learning towards minimum hyperspherical energy. *NeurIPS*, 2018. 1, 3, 7, 8, 9, 10, 14, 15

[14] J Batle, Armen 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

[15] Matthew Calef, Whitney Griffiths, and Alexia Schulz. Estimating the number of stable configurations for the generalized thomson problem. *Journal of Statistical Physics*, 160(1):239–253, 2015. 1

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

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

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

[19] Kenji Kawaguchi, Bo Xie, and Le Song. Deep semi-random features for nonlinear function approximation. In *AAAI*, 2018. 2, 7

[20] Zeev Nehari. *Conformal mapping*. Courier Corporation, 2012. 2

[21] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *ICML*, 2009. 2, 3

[22] Ignacio Ramirez, Pablo Sprechmann, and Guillermo Sapiro. Classification and clustering via dictionary learning with structured incoherence and shared features. In *CVPR*, 2010. 2, 3

[23] Nan Li, Yang Yu, and Zhi-Hua Zhou. Diversity regularized ensemble pruning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2012. 2

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

[25] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. Self-paced learning with diversity. In *NIPS*, 2014. 2

[26] Pengtao Xie, Wei Wu, Yichen Zhu, and Eric P Xing. Orthogonality-promoting distance metric learning: convex relaxation and theoretical analysis. In *ICML*, 2018. 2

[27] Pengtao Xie, Jun Zhu, and Eric Xing. Diversity-promoting bayesian learning of latent variable models. In *ICML*, 2016. 2

[28] Shuang Li, Slawomir Bak, Peter Carr, and Xiaogang Wang. Diversity regularized spatiotemporal attention for video-based person re-identification. In *CVPR*, 2018. 3

[29] Pengtao Xie, Aarti Singh, and Eric P Xing. Uncorrelation and evenness: a new diversity-promoting regularizer. In *ICML*, 2017. 3

[30] Pengtao Xie, Yuntian Deng, Yi Zhou, Abhimanu Kumar, Yaoliang Yu, James Zou, and Eric P Xing. Learning latent space models with angular constraints. In *ICML*, 2017. 3

[31] Michael Cogswell, Faruk Ahmed, Ross Girshick, Larry Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations. In *ICLR*, 2016. 3

[32] Pau Rodríguez, Jordi Gonzalez, Guillem Cucurull, Josep M Gonfaus, and Xavier Roca. Regularizing cnns with locally constrained decorrelations. In *ICLR*, 2017. 7, 10

[33] Di Xie, Jiang Xiong, and Shiliang 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. 3

[34] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, 2016. 3

[35] Weiyang Liu, Yan-Ming Zhang, Xingguo Li, Zhiding Yu, Bo Dai, Tuo Zhao, and Le Song. Deep hyperspherical learning. In *NIPS*, 2017. 7, 10

[36] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*, 2017.

[37] Weiyang Liu, Zhen Liu, Zhiding Yu, Bo Dai, Rongmei Lin, Yisen Wang, James M Rehg, and Le Song. Decoupled networks. *CVPR*, 2018. 7

[38] Jiankang Deng, Jia Guo, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *arXiv preprint arXiv:1801.07698*, 2018.

[39] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Zhifeng Li, Dihong Gong, Jingchao Zhou, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. *arXiv preprint arXiv:1801.09414*, 2018.

[40] Feng Wang, Xiang Xiang, Jian Cheng, and Alan L Yuille. Normface: L2 hypersphere embedding for face verification. *arXiv preprint arXiv:1704.06369*, 2017.

[41] Feng Wang, Weiyang Liu, Haijun Liu, and Jian Cheng. Additive margin softmax for face verification. *arXiv preprint arXiv:1801.05599*, 2018. 3

[42] Harald Cramér. *Mathematical methods of statistics (PMS-9)*, volume 9. Princeton university press, 2016. 4

[43] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 4

[44] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[45] Bo Dai, Hanjun Dai, Niao He, Weiyang Liu, Zhen Liu, Jianshu Chen, Lin Xiao, and Le Song. Coupled variational bayes via optimization embedding. In *NIPS*, 2018. 4

[46] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 5

[47] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*, 2013. 5

[48] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003. 6

[49] Ata Kaban. Improved bounds on the dot product under random projection and random sign projection. In *KDD*, 2015. 6, 16

[50] Qinfeng Shi, Chunhua Shen, Rhys Hill, and Anton van den Hengel. Is margin preserved after random projection? *arXiv preprint arXiv:1206.4651*, 2012. 6, 18

[51] Juan A Cuesta-Albertos, Antonio Cuevas, and Ricardo Fraiman. On projection-based tests for directional and compositional data. *Statistics and Computing*, 19(4):367, 2009. 6

[52] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016. 7

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 9

[54] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, pages 1–42, 2014. 10

[55] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 10, 15

[56] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 10, 15

[57] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 10

[58] Juan Antonio Cuesta-Albertos, Ricardo Fraiman, and Thomas Ransford. A sharp form of the cramer–wold theorem. *Journal of Theoretical Probability*, 20(2):201–209, 2007. 18

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 10: 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 3×3, Max Pooling, Stride 2	[7×7, 64], Stride 2 3×3, Max Pooling, Stride 2
Conv1.x	[3×3, 64]×1 $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times 5$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times 3$
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 11: 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 [1], 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 10 and Table 11. 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. [13] 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 ℓ_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*. All the network architectures are implemented by ourselves, so there might be performance difference between our implementation and the original paper due to some different network and optimization hyperparameters. For example, due to limitation of computation resources, our batch size for ImageNet-2012 is 64 batch size, which might has some performance loss. However, the network and training settings for the baseline and all the compared regularizations are the same, which ensures the fairness of our experiments.

Point cloud recognition settings. For all the PointNet and PointNet++ experiments, we exactly follow the same setting in the original papers [55, 56] and their official repositories[1] [2]. Specifically, we combine CoMHE regularization to neurons in all the 1×1 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 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 [13].

Besides PointNet, we combine CoMHE to PointNet++ [56] 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 [56] 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 [13] only has about 0.2% accuracy gain.

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

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:

$$\begin{aligned}\mathbb{E}(\langle \boldsymbol{P}\boldsymbol{w}_1, \boldsymbol{P}\boldsymbol{w}_2\rangle) &= \frac{1}{n}\mathbb{E}\Big(\sum_{l=1}^{n}(\sum_{j=1}^{d} r_{lj}\{\boldsymbol{w}_1\}_j \sum_{i=1}^{d} r_{li}\{\boldsymbol{w}_2\}_i)\Big)\\ &= \frac{1}{n}\sum_{l=1}^{n}\Big(\sum_{j=1}^{d}\mathbb{E}(r_{lj}^2)\{\boldsymbol{w}_1\}_j\{\boldsymbol{w}_2\}_j + \sum_{j=1}^{d}\mathbb{E}(r_{lj})\{\boldsymbol{w}_1\}_j \cdot \sum_{i\neq j:i=1}^{d}\mathbb{E}(r_{li})\{\boldsymbol{w}_2\}_i\Big)\\ &= \langle \boldsymbol{w}_1, \boldsymbol{w}_2\rangle\end{aligned} \tag{12}$$

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 [49].

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 σ^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, $\forall\epsilon \in (0, 1)$:*

$$\boldsymbol{w}_1^\top \boldsymbol{w}_2 k\sigma^2 - \epsilon k\sigma^2 \|\boldsymbol{w}_1\|\|\boldsymbol{w}_2\| < (\boldsymbol{P}\boldsymbol{w}_1)^\top \boldsymbol{P}\boldsymbol{w}_2 < \boldsymbol{w}_1^\top \boldsymbol{w}_2 k\sigma^2 + \epsilon k\sigma^2 \|\boldsymbol{w}_1\|\|\boldsymbol{w}_2\| \tag{13}$$

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

From Lemma 2, we have that

$$\begin{aligned}(1-\epsilon)\|\boldsymbol{w}_1\|^2 k\sigma^2 < \|\boldsymbol{P}\boldsymbol{w}_1\|^2 < (1+\epsilon)\|\boldsymbol{w}_1\|^2 k\sigma^2\\ (1-\epsilon)\|\boldsymbol{w}_2\|^2 k\sigma^2 < \|\boldsymbol{P}\boldsymbol{w}_2\|^2 < (1+\epsilon)\|\boldsymbol{w}_2\|^2 k\sigma^2\end{aligned} \tag{14}$$

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

Then we combine Eq. 14 to Lemma 3 and obtain that

$$\frac{\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) - \epsilon}{1+\epsilon} < \cos(\theta_{(\boldsymbol{P}\boldsymbol{w}_1,\boldsymbol{P}\boldsymbol{w}_2)}) < \frac{\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) + \epsilon}{1-\epsilon} \tag{15}$$

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

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 $\epsilon \in (0, 1)$*

$$\Pr\Big((1-\epsilon) \le \frac{\|\boldsymbol{P}\boldsymbol{w}\|^2}{\|\boldsymbol{w}\|^2} \le (1+\epsilon)\Big) \ge 1 - 2\exp\big(-\frac{n}{2}(\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3})\big) \tag{16}$$

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

$$\|\boldsymbol{P}\boldsymbol{w}\|^2 = \frac{1}{n}\boldsymbol{w}^2 \sum_{l=1}^{n} z_l^2 \tag{17}$$

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

$$\begin{aligned}\Pr\Big(\|\boldsymbol{P}\boldsymbol{w}\|^2 \leq (1-\epsilon)\left\|\boldsymbol{w}^2\right\| \Big) &\leq \exp\left(\frac{n}{2}(1-(1-\epsilon)+\ln(1-\epsilon))\right) \\ &\leq \exp(-\frac{n}{4}\epsilon^2)\end{aligned} \tag{18}$$

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

$$\begin{aligned}\Pr\Big(\|\boldsymbol{P}\boldsymbol{w}\|^2 \leq (1+\epsilon)\left\|\boldsymbol{w}^2\right\| \Big) &\leq \exp\left(\frac{n}{2}(1-(1+\epsilon)+\ln(1+\epsilon))\right) \\ &\leq \exp(-\frac{n}{2}(\frac{\epsilon^2}{2}-\frac{\epsilon^3}{3}))\end{aligned} \tag{19}$$

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

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

$$\begin{aligned}(1-\epsilon) \leq \frac{\|\boldsymbol{P}\boldsymbol{w}_1\|^2}{\|\boldsymbol{w}_1\|^2} \leq (1+\epsilon) \\ (1-\epsilon) \leq \frac{\|\boldsymbol{P}\boldsymbol{w}_2\|^2}{\|\boldsymbol{w}_2\|^2} \leq (1+\epsilon)\end{aligned} \tag{20}$$

which holds with probability at least $1 - 4\exp(-\frac{n}{2}(\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3}))$. Using Eq. 20, we can have that

$$\left\|\frac{\boldsymbol{P}\boldsymbol{w}_1}{\|\boldsymbol{P}\boldsymbol{w}_1\|} - \frac{\boldsymbol{P}\boldsymbol{w}_2}{\|\boldsymbol{P}\boldsymbol{w}_2\|}\right\|^2 \leq \left\|\frac{\boldsymbol{P}\boldsymbol{w}_1}{\sqrt{1-\epsilon}\,\|\boldsymbol{w}_1\|} - \frac{\boldsymbol{P}\boldsymbol{w}_2}{\sqrt{1-\epsilon}\,\|\boldsymbol{w}_2\|}\right\|^2 \tag{21}$$

From Eq. 20 and the condition that $\boldsymbol{w}_1^\top \boldsymbol{w}_2 > 0$, we further have that

$$\begin{aligned}\left\|\frac{\boldsymbol{P}\boldsymbol{w}_1}{\|\boldsymbol{w}_1\|} - \frac{\boldsymbol{P}\boldsymbol{w}_2}{\|\boldsymbol{w}_2\|}\right\|^2 &\leq \left\|\sqrt{1+\epsilon} - \sqrt{1-\epsilon}\right\|^2 \\ &\leq \left\|\sqrt{1+\epsilon}\Big(\frac{\boldsymbol{P}\boldsymbol{w}_1}{\|\boldsymbol{P}\boldsymbol{w}_1\|} - \frac{\boldsymbol{P}\boldsymbol{w}_2}{\|\boldsymbol{P}\boldsymbol{w}_2\|}\Big)\right\|^2 + \left\|\sqrt{1+\epsilon} - \sqrt{1-\epsilon}\right\|^2\end{aligned} \tag{22}$$

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

$$(1-\epsilon)\left\|\frac{\boldsymbol{w}_1}{\|\boldsymbol{w}_1\|} - \frac{\boldsymbol{w}_2}{\|\boldsymbol{w}_2\|}\right\|^2 \leq \left\|\frac{\boldsymbol{P}\boldsymbol{w}_1}{\|\boldsymbol{w}_1\|} - \frac{\boldsymbol{P}\boldsymbol{w}_2}{\|\boldsymbol{w}_2\|}\right\|^2 \leq (1+\epsilon)\left\|\frac{\boldsymbol{w}_1}{\|\boldsymbol{w}_1\|} - \frac{\boldsymbol{w}_2}{\|\boldsymbol{w}_2\|}\right\|^2 \tag{23}$$

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

$$\begin{aligned}\frac{\langle \boldsymbol{w}_1, \boldsymbol{w}_2\rangle}{\|\boldsymbol{w}_1\|\,\|\boldsymbol{w}_2\|} &= 1 - \frac{1}{2}\left\|\frac{\boldsymbol{w}_1}{\|\boldsymbol{w}_1\|} - \frac{\boldsymbol{w}_2}{\|\boldsymbol{w}_2\|}\right\|^2, \\ \frac{\langle \boldsymbol{P}\boldsymbol{w}_1, \boldsymbol{P}\boldsymbol{w}_2\rangle}{\|\boldsymbol{P}\boldsymbol{w}_1\|\,\|\boldsymbol{P}\boldsymbol{w}_2\|} &= 1 - \frac{1}{2}\left\|\frac{\boldsymbol{P}\boldsymbol{w}_1}{\|\boldsymbol{P}\boldsymbol{w}_1\|} - \frac{\boldsymbol{P}\boldsymbol{w}_2}{\|\boldsymbol{P}\boldsymbol{w}_2\|}\right\|^2.\end{aligned} \tag{24}$$

From Eq. 21, Eq. 22 and Eq. 23, 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. 24, we have that

$$\frac{1+\epsilon}{1-\epsilon}\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) - \frac{2\epsilon}{1-\epsilon} < \cos(\theta_{(\boldsymbol{P}\boldsymbol{w}_1,\boldsymbol{P}\boldsymbol{w}_2)}) < \frac{1-\epsilon}{1+\epsilon}\cos(\theta_{(\boldsymbol{w}_1,\boldsymbol{w}_2)}) + \frac{1+2\epsilon}{1+\epsilon} - \frac{\sqrt{(1-\epsilon^2)}}{1+\epsilon} \tag{25}$$

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 $\boldsymbol{w}_2$. The unorganized original proof is given in [50]. □

B.4 Theorem 3

We first introduce a lemma before proving the theorem.

Lemma 5. *[Direct Result from [58]] Let $\mathcal{H}$ be a separable Hilbert space, and let μ 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 μ-measure.*

Then we have $P = Q$.

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 a Gaussian distribution, and then the condition on the moments of $\boldsymbol{w}$ in Lemma 5 holds. Then with the following lemma, we can easily have the desired result.

□

C More Discussion on the Effectiveness of CoMHE

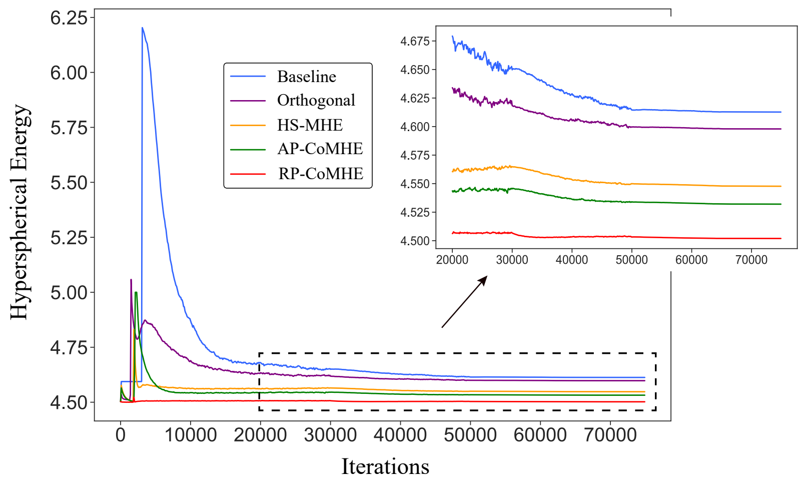

Figure 3: Hyperspherical energy during the entire training. Note that, all networks are initialized with the same weights and therefore have the same hyperspherical energy at the beginning.

Fig. 3 shows the entire training dynamics (from initialization to the end of training) of the hyperspherical energy of baseline CNN and CNN regularized by orthogonal regularization, HS-MHE, AP-CoMHE and RP-CoMHE. All the networks use exactly the same initialized weights to ensure the hyperspherical energy is the same at the beginning. One can observe that the hyperpsherical energy is actually very low for the initialized weights. This is because the initialized weights follows Gaussian distribution and the hyperspherical energy is computed with normalized weights. The normalized weights (sampled from Gassuain distribution) follows the uniform distribution on the hypersphere (see the theorem below), which can obtain the lowest hyperspherical distribution in expectation. However, when the weights of the neural network start to fit the data and minimize the data approximation loss, the neuron weights no longer follow the hyperspherical uniform distribution. Therefore the hyperspherical energy will quickly get large. This is when MHE and CoMHE are useful. From Fig. 3, one can see that without any regularization on hyperspherical energy, the hyperspherical energy of the baseline network gets extremely large at the beginning and then decreases as the training goes. However, the final hyperspherical energy of the baseline network is still way higher than the CNNs regularized by MHE and CoMHE. Notice that, the orthogonal regularized CNN also obtain high hyperspherical energy at the end (similar to the baseline network). In contrast to MHE, we can observe that CoMHE can effectively minimize the hyperspherical energy and RP-CoMHE achieves significantly lower hyperspherical energy in the end, which well verifies the superiority of the proposed CoMHE.

Theorem 4. *The normalized vector of Gaussian variables is uniformly distributed on the sphere. Formally, let $x_1, x_2, \cdots, x_n \sim \mathcal{N}(0,1)$ and be independent. Then the vector*

$$\boldsymbol{x} = \left[\frac{x_1}{z}, \frac{x_2}{z}, \cdots, \frac{x_n}{z}\right] \tag{26}$$

follows the uniform distribution on $\mathbb{S}^{n-1}$, where $z = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$ is a normalization factor.

Proof. A random variable has distribution $\mathcal{N}(0,1)$ if it has the density function

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}. \tag{27}$$

A n-dimensional random vector $\boldsymbol{x}$ has distribution $\mathcal{N}(0,1)$ if the components are independent and have distribution $\mathcal{N}(0,1)$ each. Then the density of $\boldsymbol{x}$ is given by

$$f(x) = \frac{1}{(\sqrt{2\pi})^n} e^{-\frac{1}{2}\langle x,x \rangle}. \tag{28}$$

Then we introduce the following lemma about the orthogonal-invariance of the normal distribution.

Lemma 6. *Let $\boldsymbol{x}$ be a n-dimensional random vector with distribution $\mathcal{N}(0,1)$ and $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ be an orthogonal matrix ($\boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}$). Then $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{x}$ also has distribution $\mathcal{N}(0,1)$.*

Proof. For any measurable set $A \subset \mathbb{R}^n$, we have that

$$\begin{aligned} P(Y \in A) &= P(X \in U^\top A) \\ &= \int_{U^\top A} \frac{1}{(\sqrt{2\pi})^n} e^{-\frac{1}{2}\langle x,x\rangle} \\ &= \int_{A} \frac{1}{(\sqrt{2\pi})^n} e^{-\frac{1}{2}\langle Ux,Ux\rangle} \\ &= \int_{A} \frac{1}{(\sqrt{2\pi})^n} e^{-\frac{1}{2}\langle x,x\rangle} \end{aligned} \tag{29}$$

because of orthogonality of U. Therefore the lemma holds. ☐

Because any rotation is just a multiplication with some orthogonal matrix, we know that normally distributed random vectors are invariant to rotation. As a result, generating $\boldsymbol{x} \in \mathbb{R}^n$ with distribution $\mathbb{N}(0,1)$ and then projecting it onto the hypersphere $\mathbb{S}^{n-1}$ produces random vectors $U = \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}$ that are uniformly distributed on the hypersphere. Therefore the theorem holds. ☐

D TRAINING RUNTIME COMPARISON

We also provide runtime comparison for all the proposed CoMHE. We use the plain CNN-9 for all the methods in this experiment. 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. This hyperparameter setting for CoMHE can achieve the best testing accuracy on CIFAR-100. The results in Table 12 are computed using the total runtime of runing 100 iterations. We can see that the runtime of RP-CoMHE, AP-CoMHE and Adv-CoMHE is comparable to HS-MHE and the baseline. Without any code optimization, RP-CoMHE is 36% slower than the baseline and 18% slower than the HS-MHE, and AP-CoMHE is 34% slower than the baseline and 17% slower than the HS-MHE. Note that, although CoMHE is relatively slower in terms of training runtime, it will affect the testing runtime of a trained model. That is to say, CoMHE-regularized CNN has the same inference speed with its baseline CNN counterpart.

Method	Runtime (s)
Baseline	5.61
HS-MHE	6.46
RP-CoMHE	7.62
AP-CoMHE	7.48
Adv-CoMHE	6.27
Group CoMHE	11.12

Table 12: Training Runtime (s / 100 iterations) comparison on CIFAR-100.