Randomly Weighted Neuromodulation in Neural Networks Facilitates Learning of Manifolds Common Across Tasks

Jinyung Hong[1] **Theodore P. Pavlic**[1,2]
[1]School of Computing and Augmented Intelligence
[2]School of Life Sciences
Arizona State University
Tempe, AZ 85281
{ jhong53, tpavlic }@asu.edu

Editors: Marco Fumero, Emanuele Rodolà, Clementine Domine, Francesco Locatello, Gintare Karolina Dziugaite, Mathilde Caron

Abstract

Geometric Sensitive Hashing functions, a family of Local Sensitive Hashing functions, are neural network models that learn class-specific manifold geometry in supervised learning. However, given a set of supervised learning tasks, understanding the manifold geometries that can represent each task and the kinds of relationships between the tasks based on them has received little attention. We explore a formalization of this question by considering a generative process where each task is associated with a high-dimensional manifold, which can be done in brain-like models with neuromodulatory systems. Following this formulation, we define *Task-specific Geometric Sensitive Hashing (T-GSH)* and show that a randomly weighted neural network with a neuromodulation system can realize this function.[1]

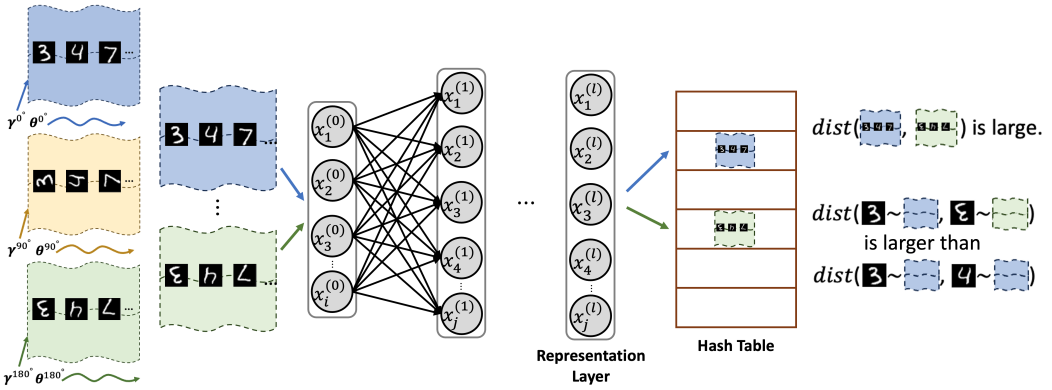
1 Introduction

Deep Neural Networks (DNNs) have shown outstanding performance in various fields due to their ability to learn to transform complex objects into useful representations that can be easily separated in the embedding space. However, due to their "black box" behavior, there is still little known about exactly which information-rich features are captured by a DNN representation and what information-poor variations are eliminated. To address the question, Dikkala et al. [10] considered the manifold geometry of a generative process where each class is associated with a manifold and parameter that shifts class examples along that manifold. Using this perspective, they could show that neural representations in supervised learning could be viewed as a kind of Locality Sensitive Hash functions that they coined Geometric Sensitive Hashing (GSH) [10].

In this work, we extend this geometric interpretation from one classification task to a series of potentially related classification tasks. For example, in a video clip of digits rotating counterclockwise, each possible digit within a single frame is a class associated with its own manifold, as in GSH, and we focus on the relationship between the manfolds induced by the classification problem in

[1]All source codes and figures are available at `https://github.com/PavlicLab/NeurIPS2023-UniReps-Hong-TGSH-Randomly_Weighted_Neuromodulation_in_Neural_Networks.git`.

Proceedings of the I edition of the Workshop on Unifying Representations in Neural Models (UniReps 2023).



(a) Data generating process.

(b) A DNN as a Task-specific Geometry Sensitive Hash (T-GSH) function.

Figure 1: (a) Each task t consists of a set of points of classes on a simple manifold. Furthermore, each point in the task can be characterized by three latent parameters: $\boldsymbol{\gamma}^t, \boldsymbol{\delta}, \boldsymbol{\theta}^t$. Notably, we consider a task-specific manifold broader than the class-level one in [10]. (b) Any set of points on the same task manifold map to (approximately) the same representation (i.e., the penultimate layer feature map), while a set of points from different task manifolds go to far away representations.

one frame (i.e., one rotation angle) and the other frames. Such a manifold-learning perspective allows us to compare representations of tasks that are similar in principle so that we can identify representational kernels that are shared among each individual classification task. In essence, for this example digit-rotation task, we consider the possibility that rotated digit images may in fact map to different points on the same induced digit manifold that is then combined with a rotation manifold to represent the image and its orientation.

The geometry of low- and high-level perceptual spaces has been studied in cognitive science, and changes in those behavioral and cognitive states have been associated with neuromodulatory systems in the brain. Neuromodulation has been studied extensively in human and insect brains, where neuromodulatory signals act as a kind of *switchboard* that can remap a neural representation in different ways so as to include adaptive control of behavior based on internal state and environmental context [16, 18]. In response to the importance of neuromodulatory systems in biological brains, neuromodulation-inspired neural networks have been proposed in various domains, such as reinforcement learning [4], modular networks [7], and adaptive behaviors [22]. For example, promising applications of neuromodulated neural networks to the problem of to Continual Learning (CL) [7, 14, 17] can prevent catastrophic forgetting by modulating synapsis that remain persistently stable.

In this work, we connect neuromodulation-inspired neural networks for continual learning and geometric manifold learning in supervised learning. We introduce the definition of *task-specific manifolds* and demonstrate that DNNs with neuromodulators can learn these manifolds.

2 Task-Specific Geometric Sensitive Hashing

2.1 Problem Definition and Proposed Randomly Weighted Neuromodulation Method

Following [14], we consider a supervised continual-learning setup where $\mathcal{T}$ tasks arrive to a learner in sequential order. Let $\mathcal{D}_t = \{\mathbf{x}_{i,t}, y_{i,t}\}_t^{n_t}$ be the dataset of task t, composed of n_t pairs of raw instances and corresponding labels. For simplicity, we adopt a simple task-incremental continual learning setup where $\mathcal{C}$ is the number of classes for every task. Next, we extend the manifolds from Dikkala et al. [10] to represent *task-specific* manifolds. We assume that each input $\mathbf{x}_{i,t} \in \mathbb{R}^d$ is drawn from a mixture distribution over $\mathcal{T}$ manifolds $M_1, \ldots, M_{\mathcal{T}}$ sharing similar topologies (Fig. 1a), and each label $y_{i,t} \in [\mathcal{C}]$ corresponds to the manifold of $\mathbf{x}_{i,t}$. Moreover, each point $\mathbf{x}_t$ on manifold M_t is determined by the latent vectors, $\boldsymbol{\gamma}^t \in \mathbb{R}^s$, $\boldsymbol{\delta} \in \mathbb{R}^p$, and $\boldsymbol{\theta}^t \in \mathbb{R}^k$, where:

- $\boldsymbol{\gamma}^t$ is the manifold identifier for the task t, and there is a one-to-one correspondence between $\boldsymbol{\gamma}^t$ and M_t.

- $\boldsymbol{\delta}$ is a set of manifold identifiers for classes. This is an explicitly shared manifold, representing canonical representation objects and is shared across all task manifolds $\boldsymbol{\gamma}$.
- $\boldsymbol{\theta}^t$ is the transformation in the task t. So, if we fix $\boldsymbol{\gamma}^t$, the manifold M_t can be generated by sampling different values of $\boldsymbol{\theta}^t$.

Intuitively, $\boldsymbol{\gamma}^t$ represents the centroid of a cluster for the task t, $\boldsymbol{\delta}$ includes explicitly shared clusters for labels that every $\boldsymbol{\gamma}$ contains, and $\boldsymbol{\theta}^t$ represents a small perturbation around the centroid. So, the manifold $M_{\boldsymbol{\gamma}^t}$ is comprised of all inputs $\mathbf{x}_t$ of the form $\{\boldsymbol{\gamma}^t + \boldsymbol{\delta} + \boldsymbol{\theta}^t | \ \|\boldsymbol{\theta}^t\| \leq \epsilon\}$. In this setting, we employ a Locality Sensitive Hashing (LSH) to map each input to a hash bucket. With appropriate configuration, we empirically demonstrate that DNNs exhibit LSH-like behavior on the family of manifolds with a shared geometry defined by a set of analytic functions. In particular, extending the work of Dikkala et al. [10], we show that the penultimate "representation layer" r (Fig. 1b) of an appropriately trained network will satisfy the following property:

Definition 1 (Task-specific Geometric Sensitive Hashing (T-GSH)) *We say r is a T-GSH function with respect to a set of task manifolds if:*

a) For any two points on the same task manifold $x_{1,t}, x_{2,t} \in M_t$, the distance $\|r(x_{1,t}) - r(x_{2,t})\|$ is small regardless of the associated classes.

b) For any two points on two well-separated task manifolds $x_{1,t} \in M_t$ and $x_{2,t'} \in M_{t'}$, $\|r(x_{1,t}) - r(x_{2,t'})\|$ is large regardless of the associated classes.

That is, DNNs whose representations satisfy the T-GSH properly capture the shared task manifold geometry similar to how LSH functions capture spatial locality. Finally, by employing the recoverability of the manifold [10], we give empirical evidence to support the recoverability of $\boldsymbol{\gamma}^t$ via simple transformations on top of representations learned by DNNs which behave as T-GSH functions.

We consider task manifolds as subsets of points in $\mathbb{R}^d$. Every manifold $M_{\boldsymbol{\gamma}^t}$ has an associated latent vector $\boldsymbol{\gamma}^t \in \mathbb{R}^s$ (where $s \leq d$) which acts as an identifier of $M_{\boldsymbol{\gamma}^t}$. The task manifold is then defined to be the set of points $\mathbf{x}_t = \mathbf{f}(\boldsymbol{\gamma}^t, \boldsymbol{\delta}, \boldsymbol{\theta}^t) = (f_1(\boldsymbol{\gamma}^t, \boldsymbol{\delta}, \boldsymbol{\theta}^t), \ldots, f_d(\boldsymbol{\gamma}^t, \boldsymbol{\delta}, \boldsymbol{\theta}^t))$ for $\boldsymbol{\delta} \in \Delta \subseteq \mathbb{R}^p, p < d,$ and for $\boldsymbol{\theta}^t \in \Theta \subseteq \mathbb{R}^k, k < d$. Here, the manifold generating function $\mathbf{f} = \{f_i(\cdot, \cdot)\}_{i=1}^d$ where the f_i are all analytic functions. Without significant loss of generality, we assume our inputs $\mathbf{x}$ and $\boldsymbol{\gamma}$ are normalized and lie on S^{d-1}, and S^{s-1}, the d and s-dimensional unit spheres, respectively. Given the above generative process, we leverage the Assumption 1, which allows us to use a special form of the DNN configuration used by Dikkala et al. [10].

Assumption 1 (Modified Invertibility [10]) *There is an analytic function $g(\cdot) : \mathbb{R}^d \to \mathbb{R}^s$ with bounded norm Taylor expansion such that for every point $\mathbf{x}_t = \mathbf{f}(\boldsymbol{\gamma}^t, \boldsymbol{\delta}, \boldsymbol{\theta}^t)$ on $M_{\boldsymbol{\gamma}^t}$, $g(\mathbf{x}_t) = \boldsymbol{\gamma}^t$.*

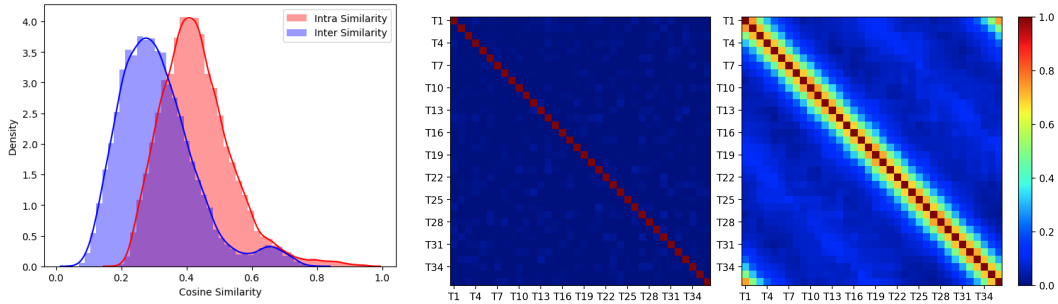
The conventional GSH neural network [10] is defined as a single-hidden-layer network, $\mathbf{y} = A \cdot B \cdot \sigma(C\mathbf{x})$, where the input $\mathbf{x} \in \mathbb{R}^d$ passes through a wide, randomly initialized, fully connected, non-trainable layer $C \in \mathbb{R}^{D \times d}$ followed by a ReLU activation $\sigma(\cdot)$. Then, there are two trainable, fully connected layers $A \in \mathbb{R}^{\mathcal{C} \times T}$, $B \in \mathbb{R}^{T \times D}$ with no non-linearity between them. Following the above configuration, we define a special-case T-GSH neural network as follows:

$$\hat{\mathbf{y}}_t = \mathbf{R} \cdot B^t \cdot \sigma(C\mathbf{x}) \tag{1}$$

for each task t. Thus, not only does a T-GSH differ from a GSH by the introduction of multiple task labels, but the learnable matrix A from a standard GSH is replaced with another randomly weighted matrix $\mathbf{R}$ that acts as the explicitly shared manifold $\boldsymbol{\delta}$ in our definition (i.e., it is shared across all tasks). Intuitively, a randomly weighted matrix is nearly orthogonal, and so learning task-specific B^t must be enforced to learn the commonality in the task t. In the upcoming section, we connect a T-GSH neural network with one of the examples of neuromodulation-inspired DNNs for continual learning.

2.2 Revising Configurable Random Weight Networks

Configurable Random Weight Networks (CRWNs) [14] were proposed as a simple yet effective form of artificial neuromodulatory systems for continual learning. The proposed approach consists of two types of artificial neuromodulation, *Global* and *Local* modulation, and shows that their



(a) A comparison of cosine similarities of the points. (**Intra similarity**): the cosine similarities of the points with the *different labels* on the *same task manifold*. (**Inter similarity**): the cosine similarities of the points with the *same label* on the *different task manifolds*.

(b) A confusion matrix of intra (same task manifold) VS inter (different task manifolds) cosine similarity of task representations trained on *RotationMNIST*. Cosine similarity between task context vectors before (**left**) and after training (**right**). Notably, adjacent tasks are expected to be the most similar to each other as they represent the smallest angular difference in images.

Figure 2: Experiment results on RotationMNIST.

neuromodulatory signals learn how to adjust long-lasting/unchanging random synaptic weights for specific tasks, enabling task-specific learning. Furthermore, the proposed neuromodulation can be applied in a layer-wise fashion. To improve the efficiency of training and the applicability to any network architecture, Hong and Pavlic [14] proposed two kinds of model architectures, which we summarize in the single expression in Eq. (2),

$$\begin{aligned} \hat{\mathbf{y}}_t &= \alpha_t \cdot \mathbf{R} \cdot (v^t \odot \sigma(C\mathbf{x})) \\ &= \mathbf{R} \cdot ((\alpha_t \cdot v^t) \odot \sigma(C\mathbf{x})) \end{aligned} \tag{2}$$

where:

- $\mathbf{R} \in \mathbb{R}^{\mathcal{C} \times D}$ and $C \in \mathbb{R}^{D \times d}$ are non-trainable, randomly weighted matrices.
- $\alpha_t \in \mathbb{R}$ is a learnable constant acting as *global* neuromodulation.
- $v^t \in \mathbb{R}^D$ is a learnable vector mimicking *local* neuromodulation.

In the following, we view Eq. (2) through the lens of T-GSH in Eq. (1).

CRWN as a T-GSH function. CRWNs were first proposed as an efficient CL method by highlighting the approach's benefits concerning computational efficiency. However, we highlight here that CRWNs are a T-GSH function achieved by the proposed artificial neuromodulatory systems and thus can be used to shed light on the manifold learning facilitated by neuromodulatory processes. Below, we compare Eq. (1) and (2).

$$\hat{\mathbf{y}}_t = \overbrace{\mathbf{R} \cdot B^t \cdot \sigma(C\mathbf{x})}^{\text{Eq. (1)}} = \overbrace{\mathbf{R} \cdot (\underbrace{(\alpha_t \cdot v^t)}_{B^t} \odot \sigma(C\mathbf{x}))}^{\text{Eq. (2)}}$$

Thus, $\alpha_t \cdot v^t$ from the CRWN in Eq. (2) plays the role of B^t in the T-GSH in Eq. (1), and so a CRWN is a simple realization of a T-GSH. Following Dikkala et al. [10], it is important to use a proper regularization term on the weights B^t in the classification loss for training a T-GSH function, and the learnable constant α_t in CRWNs acts as a similar regularizer for v^t.

3 Experiments

In this section, we demonstrate the CRWN is a T-GSH function using variants of the MNIST [15] dataset, including *RotationMNIST*, and our newly proposed experimental setup, *ShiftMNIST* and *AugmentMNIST*.

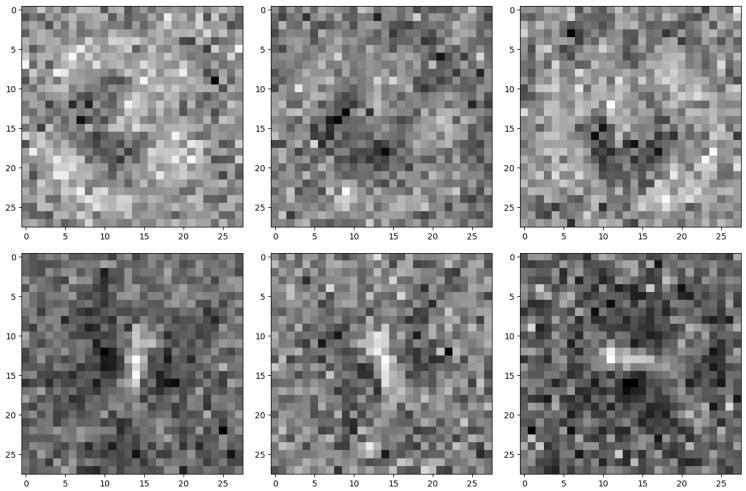


Figure 3: Empirical evidence of approximate data generating process with our defined invertibility assumption on RotationMNIST. **(Top Row)** Reconstructed samples of digit "3". **(Bottom Row)** Reconstructed samples of digit "1". **(Left Column)** Reconstructed samples of digits from the task manifold M^{T1}, which is $0°$rotation. **(Middle Column)** the samples of digits from the task manifold M^{T5}, which is 40 °counterclockwise rotation. **(Right Column)** the samples of digits from the task manifold M^{T10}, which is 90 °counterclockwise rotation.

3.1 Experiments for showing that CRWN is a T-GSH function.

We first validate the CRWN on RotationMNIST to show it is a T-GSH function. RotationMNIST is one of the popular CL benchmark datasets. Following Hong and Pavlic [14], we define the dataset so that each of the 36 tasks corresponds to images counterclockwise rotated by a multiple of 10 degrees.

CRWNs as a GSH and T-GSH function. Using experimental results of CRWNs for continual learning, we can empirically demonstrate that CRWNs are T-GSH functions. In particular, Hong and Pavlic [14, Table 1] showed that CRWNs achieved about 95% test accuracy average over all 36 RotationMNIST tasks (`FlyNet`: 94.9% and `NeuroModNet`: 95.5%, respectively). This indicates that CRWNs perform well as a GSH function that successfully classifies all labels for every task.

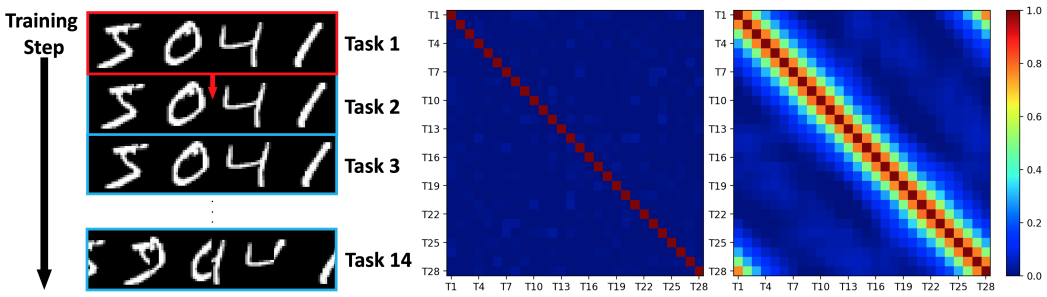
Figure 2a depicts a comparison of cosine similarities of latent CRWN task representations. *Intra similarity* indicates the cosine similarities of the points with different labels on the same task manifold (i.e., same RotationMNIST rotation angle). In contrast, *Inter similarity* means the cosine similarity of the points with the same label, but on the different task manifolds (i.e., different RotationMNIST rotational angles). This shows CRWN is a T-GSH function in Fig. 1b, showing that the similarity of the points having different labels, but on the same task manifold must be larger than one of the points with the same label, but on the different task manifolds.

Satisfaction of Assumption 1. We can also show that the inverse function of CRWNs can mimic the data-generating process, satisfying our Assumption 1. Figure 3 depicts the reconstructed image samples from CRWNs from the learned task manifolds.

Because of the ReLU function, we cannot obtain the complete inverse function of CRWNs. Thus, we omitted ReLU and computed the pseudo-inverse of the weights of the learned CRWNs. Because we fixed the manifolds for classes $\boldsymbol{\delta}$ using the non-trainable matrix $\mathbf{R}$, each row in $\mathbf{R}$ can be seen as the canonical representations of objects. As shown in Fig. 3, the canonical representation for the digit "3" is the 4th row, and for the digit "1" is the 2nd row. Thus, let $\mathbf{f}$ be the inverse CRWNs omitting the ReLU function. We can reconstruct a digit sample with class i as follows:

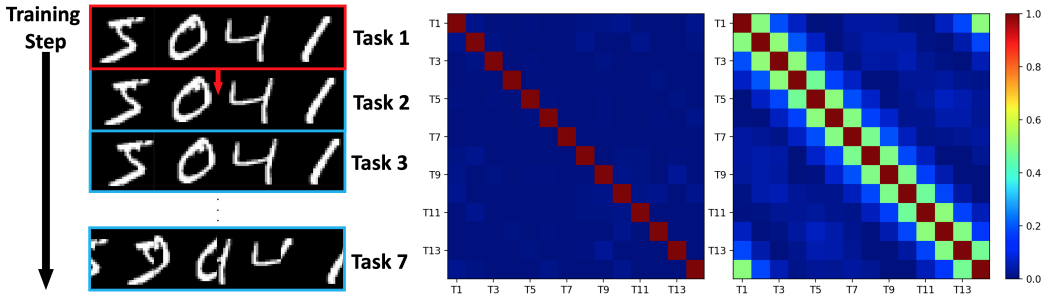
$$\bar{\mathbf{x}}_{i,t} = \mathbf{f}(\mathbf{s}) \text{ where } \mathbf{s} \sim \mathcal{N}(\mu_{i,t}, \sigma_{i,t}), \mu_{i,t} = R_i^\top \odot (\alpha_t \cdot v_t), \sigma_{i,t} = \mathbf{1}_D \cdot 1/D \tag{3}$$

R_i is i-th row of the matrix $\mathbf{R}$ and $\mathbf{1}_D$ is the vector containing ones with the size D. In Fig. 3, despite approximate reconstruction, the shape of digits and the rotation can be shown, indicating the CRWN satisfies Assumption 1.



(a) ShiftMNIST-1px tasks (b) ShiftMNIST-1px results

Figure 4: ShiftMNIST-1px. (a) The difference between the datasets of the two adjacent tasks, i and $i+1$, is that the images of task $i+1$ are the images of task i shifted a single pixel to the right. Thus, the total number of tasks to learn is 28. (b) Cosine similarity between the context vectors for each task before (**left**) and after training (**right**).



(a) ShiftMNIST-2px tasks (b) ShiftMNIST-2px results

Figure 5: ShiftMNIST-2px. (a) The difference between the datasets of the two adjacent tasks, i and $i+1$, is that the images of task $i+1$ are the images of task i shifted two pixels to the right. Thus, the total number of tasks to learn is 14. (b) Cosine similarity between the context vectors for each task before (**left**) and after training (**right**).

3.2 Experiments For Task Similarities

In this section, due to the architectural characteristic of a T-GSH function, we demonstrate unique features of the CRWN that allow for computing the relationship among tasks. In particular, we introduce the scalar–vector product $c_t \triangleq \alpha_t \cdot v^t$ so that the CRWN expression becomes:

$$\hat{\mathbf{y}}_t = \mathbf{R} \cdot ((\alpha_t \cdot v^t) \odot \sigma(C\mathbf{x})) = \mathbf{R}(c_t \odot \sigma(C\mathbf{x})).$$

We refer to c_t as a *context vector.* The following experiments on RotationMNIST, ShiftMNIST, and AugmentMNIST demonstrate the semantic interpretation of the context vector c_t among tasks.

RotationMNIST. Figure 2b shows the pairwise cosine similarity[2] between context-vector pairs (c_j, c_k) for every (j, k) pair of RotationMNIST tasks, shown both before training and after training. In the task ordering, adjacent tasks are expected to be the most similar to each other as they represent the smallest angular difference in images. In other words, networks trained for task i should have minimal degradation in performance for test data from tasks $i-1$ and $i+1$. Consistent with this functional expectation, the mechanistic pattern Fig. 2b shows greatest similarity between adjacent context vectors after training, and so trained context vectors exist in a representational space that captures fundamental similarities among tasks.

ShiftMNIST. To test whether geometric adjacency generally implies similarity of trained vectors, we introduce a new sequence of tasks, *ShiftMNIST*, as shown in Figs 4a and 5a. Whereas RotationMNIST rotates each of the MNIST characters, ShiftMNIST translates them laterally by a number of

[2]We compute $1 -$ `scipy.spatial.distance.cosine`.

Table 1: PyTorch routines used for AugmentMNIST. Here, `TT` is shorthand for `torchvision.transforms`.

AugmentMNIST Task:	**PyTorch method and configuration**
Horizontal Flip (T2):	`TT.RandomHorizontalFlip(p=1.0)`
Vertical Flip (T3):	`TT.RandomVerticalFlip(p=1.0)`
Gaussian Blur (T4):	`TT.GaussianBlur(kernel_size=5, sigma=2)`
Perspective (T5):	`TT.RandomPerspective(distortion_scale=0.5, p=1.0)`
Random Erasing (T6):	`TT.RandomErasing(p=1.0)`
Invert (T7):	`TT.RandomInvert(p=1.0)`
RandomResizedCrop (T8):	`TT.RandomResizedCrop(size=28)`

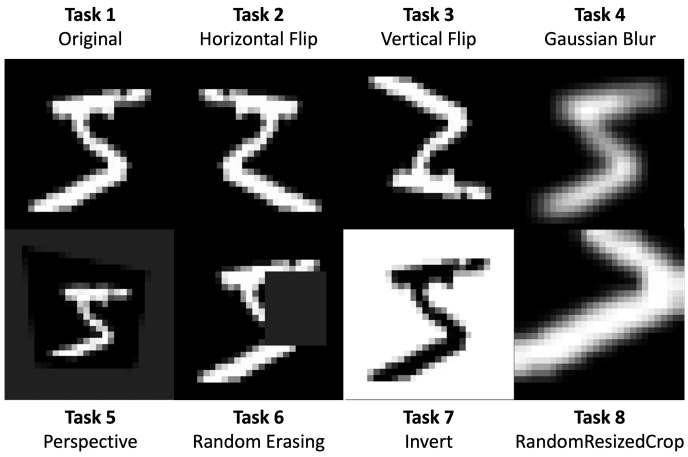


Figure 6: AugmentMNIST task. The list of variants of augmentation methods applied to produce the task. See the implementation details in Table 1.

pixels (with periodic boundaries). Adjacent ShiftMNIST-1px tasks in Fig. 4a are separated by 1 pixel, and so an image in Task 3 is the same as an image in Task 2 shifted to the right by 1 pixel. Similarly, adjacent ShiftMNIST-2px tasks in Fig. 5a are separated by 2 pixels. Because MNIST images are each 28 pixels wide, there are 28 distinct ShiftMNIST-1px tasks and 14 distinct ShiftMNIST-2px tasks. We conduct two experimental setups separately using different random seeds and check whether meaningful relationships with each other can be derived. The cosine similarities of context vectors before and after training for ShiftMNIST-1px and ShiftMNIST-2px are shown in Figs 4b and 5b, respectively. The diagonal pattern in both cases matches that of Fig. 2b, which confirms that the similarity vectors for ShiftMNIST have encoded geometric relationships among tasks in a similar way as with RotationMNIST. Furthermore, Fig. 5b appears as a more coarsely subsampled version of Fig. 4b, which demonstrates that cosine similarity differences are repeatable and a function of the tasks themselves and not an artifact of the architecture of the network. Thus, training these compact context vectors extracts meaningful semantic information about individual tasks.

AugmentMNIST. Here, we propose AugmentMNIST, which employs a sequence of 8 off-the-shelf, commonly used data-augmentation tasks as shown in Fig. 6 (e.g., Random Erasing [24] that is provided by PyTorch). We list the details of the setting in Table 1. After training on each of the 8 tasks, we use hierarchical agglomerative clustering to sort the tasks so that adjacent tasks tend to have highest similarity. Fig. 7 shows the resulting cosine similarities as well as a dendrogram of the hierarchical clustering by cosine similarity. The results show that context-vector representations of Random Erasing (T6), Gaussian Blur (T4), and Perspective (T5) are very close to that of the original image (T1); in other words, the four tasks that maintain the shape, color, and orientation of the image are grouped together. Furthermore, the inverted-image task (T7) is an outgroup as it has negative cosine similarity with all other tasks, which is consistent with it being the only task operating on a white background. Furthermore, the cosine similarities suggest a closer relationship between T1 and T7 than between T2 and T7, indicating that the horizontal flip combined with the color inversion

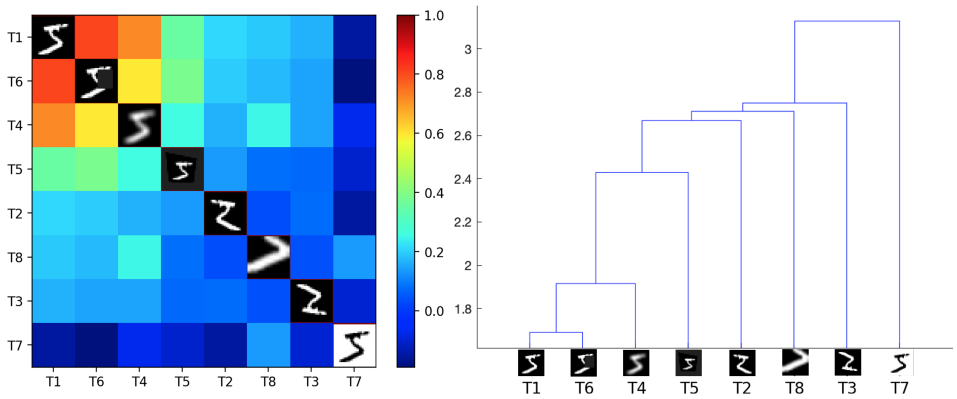


Figure 7: AugmentMNIST results. Sorted cosine similarity between the context vectors for each task after training (**left**) and task-similarity clustering (**right**).

has an additive effect in terms of context vector dissimilarity. Thus, the geometry of the trained context-vector space encodes semantic information about comparisons between tasks themselves.

4 Related Work

There have been various works on studying classification problems as manifold learning [3, 20]. Furthermore, manifold learning has been adapted in various literature, such as feature learning with different modalities [19], explainability [11], speech recognition [21] and neuroscience [5]. We leverage the concept of Locality Sensitive Hashing, and this property has been extensively studied in insect-brain-like architecture [8], and several works have been proposed in novelty detection [9] and relational learning [12, 13]. The connection between DNNs and Hash Functions was explored before [23]. Another related work is the benefits of wide non-linear layers [6] and over-parameterized networks [1, 2]. We empirically demonstrate that T-GSH holds for certain architectures under the manifold data assumption given a series of classification tasks.

5 Discussion

We studied the problem of a sequence of multiple supervised classification tasks as a task manifold-learning problem under a specific generative process wherein the manifolds share geometry. We demonstrated empirically that properly trained DNNs with a neuromodulation-inspired architecture satisfy the T-GSH property by recovering each task's semantically meaningful latent representation γ. Our work provides new geometric intuitions about the functioning of randomly weighted neuromodulation systems found throughout nature and allows us to better understand how randomness helps exploit finite representation spaces. We plan to extend our work to provide theoretical evidence in various continual-learning setups, such as class-incremental and domain-incremental learning.

Acknowledgements

This work was supported in part by NSF award 2223839 and USACE ERDC award W912HZ-21-2-0040.

References

[1] Z. Allen-Zhu, Y. Li, and Y. Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *Advances in neural information processing systems*, 32, 2019.

[2] S. Arora, S. Du, W. Hu, Z. Li, and R. Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

[3] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(11), 2006.

[4] M. Botvinick, J. X. Wang, W. Dabney, K. J. Miller, and Z. Kurth-Nelson. Deep reinforcement learning and its neuroscientific implications. *Neuron*, 107(4):603–616, 2020.

[5] E. L. Busch, J. Huang, A. Benz, T. Wallenstein, G. Lajoie, G. Wolf, S. Krishnaswamy, and N. B. Turk-Browne. Multi-view manifold learning of human brain-state trajectories. *Nature Computational Science*, 3(3):240–253, 2023.

[6] A. Daniely, R. Frostig, and Y. Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. *Advances in neural information processing systems*, 29, 2016.

[7] A. Daram, A. Yanguas-Gil, and D. Kudithipudi. Exploring neuromodulation for dynamic learning. *Front. Neurosci.*, page 928, 2020.

[8] S. Dasgupta, C. F. Stevens, and S. Navlakha. A neural algorithm for a fundamental computing problem. *Science*, 358(6364):793–796, 2017.

[9] S. Dasgupta, T. C. Sheehan, C. F. Stevens, and S. Navlakha. A neural data structure for novelty detection. *PNAS USA*, 115(51):13093–13098, 2018.

[10] N. Dikkala, G. Kaplun, and R. Panigrahy. For manifold learning, deep neural networks can be locality sensitive hash functions. *arXiv preprint arXiv:2103.06875*, 2021.

[11] H. Han, W. Li, J. Wang, G. Qin, and X. Qin. Enhance explainability of manifold learning. *Neurocomputing*, 500:877–895, 2022.

[12] J. Hong and T. Pavlic. Representing prior knowledge using randomly, weighted feature networks for visual relationship detection. In *The First International Workshop on Combining Learning and Reasoning: Programming Languages, Formalisms, and Representations (CLeaR) at the AAAI Conference on Artificial Intelligence*, 2022.

[13] J. Hong and T. P. Pavlic. An insect-inspired randomly, weighted neural network with random fourier features for neuro-symbolic relational learning. In *Proceedings of the 15th International Workshop on Neuro-Symbolic Learning and Reasoning (NeSy 20/21)*, October 25–27, 2021.

[14] J. Hong and T. P. Pavlic. Learning to modulate random weights: Neuromodulation-inspired neural networks for efficient continual learning. *arXiv preprint arXiv:2204.04297*, 2022.

[15] Y. LeCun. The MNIST database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

[16] J. Mei, E. Muller, and S. Ramaswamy. Informing deep neural networks by multiscale principles of neuromodulatory systems. *Trends Neurosci.*, 2022.

[17] T. Miconi, A. Rawal, J. Clune, and K. O. Stanley. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. *arXiv:2002.10585*, 2020.

[18] M. N. Modi, Y. Shuai, and G. C. Turner. The *drosophila* mushroom body: from architecture to algorithm in a learning circuit. *Annu. Rev. Neurosci.*, 43:465–484, 2020.

[19] N. D. Nguyen, J. Huang, and D. Wang. A deep manifold-regularized learning model for improving phenotype prediction from multi-modal data. *Nature computational science*, 2(1): 38–46, 2022.

[20] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[21] V. S. Tomar and R. C. Rose. Efficient manifold learning for speech recognition using locality sensitive hashing. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6995–6999. IEEE, 2013.

[22] N. Vecoven, D. Ernst, A. Wehenkel, and G. Drion. Introducing neuromodulation in deep neural networks to learn adaptive behaviours. *PloS ONE*, 15(1):e0227922, 2020.

[23] J. Wang, T. Zhang, N. Sebe, H. T. Shen, et al. A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):769–790, 2017.

[24] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. Random erasing data augmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):13001–13008, 2020. doi: 10.1609/aaai.v34i07.7000.