Towards Forward-Only Learning for Hyperdimensional Computing

Hyunsei Lee[1], Hyukjun Kwon[1], Jiseung Kim[1], Seohyun Kim[1], Mohsen Imani[2] and Yeseong Kim[1]
[1]DGIST and [2]University of California Irvine
{wwhslee, durwjsdnehd3523, js980408, selenium, yeseongkim}@dgist.ac.kr, m.imani@uci.edu

***Abstract*—Hyperdimensional (HD) Computing is a lightweight representation system that symbolizes data as high-dimensioned vectors. HD computing has been growing in popularity in recent years as an alternative to deep neural networks mainly due to its simple and efficient operations. In HD-based learning frameworks, the encoding of the high dimensional representations are widely cited to be the most contributing procedure to accuracy and efficiency. However, throughout HD computing's history, the encoder has largely remained *static*. In this work, we explore methods for a *dynamic* encoder that yields better representations as training progresses. Our proposed method, SEP, achieves accuracies comparable to state-of-the-art HD-based methods proposed in the literature; more notably, our solutions outperform existing work at lower dimensions while maintaining a relatively small dimension of $D = 3,000$, which equates to an average of $3.32\times$ faster inference.**

***Index Terms*—Hyperdimensional Computing**

I. Introduction

Hyperdimensional (HD) computing, also known as Vector Symbolic Architectures (VSAs), is a computing paradigm modeled after human long-term memory [1]. HD computing has recently been gaining recognition as an attractive alternative to traditional deep learning techniques primarily because of its lightweight computations. However, we observe a significant limitation in the retraining procedures within HD learning models: the sole focus on *fine-tuning class hypervectors*. This approach operates under the assumption that the encoding process will yield sufficiently distinct hypervectors for data associated with different labels. However, this overlooks a critical aspect of HD computing, the role of *hypervector encoding* in determining the overall accuracy of the trained model. The projection matrices generated during the encoding phase remain *static* throughout both the initial training and subsequent retraining procedures and, as a result, the resulting hypervector representations for any given data set are *immutable*. To address the limitations associated with static encoders in HD learning models, prior work have focused on either preprocessing the data [2] or integrating neural networks as inputs to the HD framework [3], [4]. While these approaches can improve performance, they introduce considerable computational overhead, and more critically, they fail to resolve the inherently static nature of the HD encoder.

This rigidity can lead to two critical issues: (i) sub-optimal hypervector representations; especially in scenarios involving complex data, this culminates in ineffective learning, and (ii) the use of excessive dimensions, e.g., $D = 10,000$, to compensate for the error incurred through ineffective projections of the raw data into high-dimensional space, resulting in high computational burdens to perform HD computing.

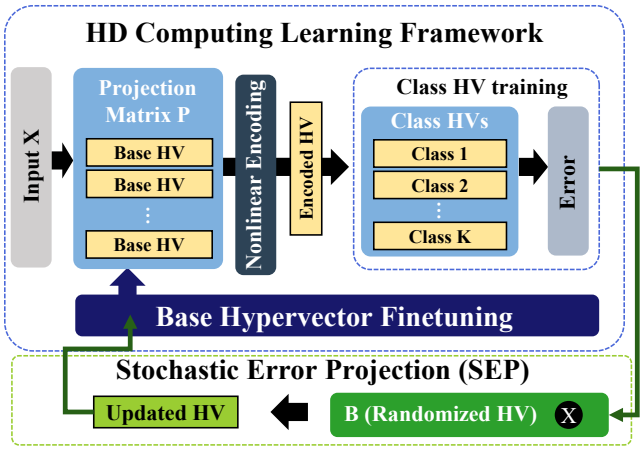


Fig. 1: Overview of the SEP Procedure

In this paper, we introduce *Stochastic Error Projection* (SEP), a novel framework for HD encoder training designed to generate highly accurate hypervector representations learned during the training process. SEP provides implicit feedback to the encoder at each retraining iteration. Though it exhibits a slightly slower convergence rate, it maintains high efficiency and outperforms existing HD-based algorithms. We show an overview of the proposed framework in Figure 1.

II. Forward-Only HD Learning Method

In this section, we present *Stochastic Error Projection* (SEP). The fundamental intuition behind SEP is premised on the idea that substantial updates to the base hypervectors are needed when the current model displays a high classification error. Unlike existing methods such as the MASS training algorithm [5], which focuses on the classification errors that originate from inaccuracies in the class hypervectors, we further consider the contribution to inaccuracies caused by erroneous base hypervectors. In SEP, the magnitude of adjustments to each element of the base hypervectors is stochastically determined, that is, elements are perturbed with a higher degree of randomness when elevated errors are observed for current samples. In contrast, minimal adjustments are made to the base hypervectors when the overall classification error is low. This is done under the assumption that they are adequately accurate. While the SEP approach bears similarities to the

979-8-3503-4859-0/DATE24/© 2024 EDAA

DFA [6] method proposed for deep learning, we introduce a modification to address a critical shortcoming of DFA, i.e., its tendency to overfit, thereby optimizing its efficiency with higher robustness within the domain of HD computing. Algorithm 1 details the SEP procedure.

Algorithm 1 Stochastic Error Projection (SEP)

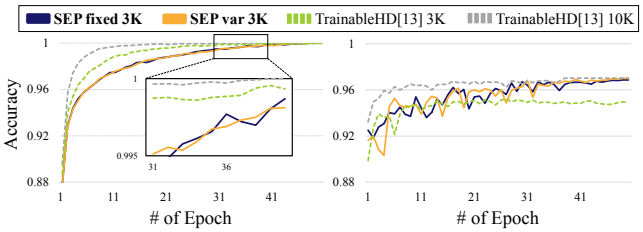
for data $\mathbf{x}$ in a training dataset **do**
  // Ⓐ **Encoding**
  $\overrightarrow{H} \leftarrow \mathbb{P} \otimes \mathbf{x}$
  // Ⓑ **Update class hypervectors** $\mathbb{C}$
  $\vec{s} \leftarrow softmax(\delta(\mathbb{C}, \overrightarrow{H}))$
  $\vec{e} \leftarrow y - \vec{s}$
  $\mathbb{C} \leftarrow \mathbb{C} \oplus \lambda \overrightarrow{H} \vec{e}$
  // Ⓒ **Update projection matrix** $\mathbb{P}$
  updates $\leftarrow$ mean$(\vec{e}) \cdot \lambda \cdot \mathbb{B}$
  $\mathbb{P} \leftarrow \mathbb{P} \oplus$ updates
end for

Given a data sample $\mathbf{x} = \{v_1, v_2, \cdots, v_F\}$, where F is the number of feature values, the projection matrix is generated as $\mathbb{P} = \{\overrightarrow{P}_1, \overrightarrow{P}_2, \cdots, \overrightarrow{P}_F\}^D$, where D is dimensions and each $\overrightarrow{P}$ is drawn from Gaussian distribution. Encoding (Ⓐ) is computed as $\overrightarrow{H} = relu(v_1 \times \overrightarrow{P}_1 + v_2 \times \overrightarrow{P}_2 + \cdots + v_F \times \overrightarrow{P}_F)$. For the retraining procedure (Ⓑ), encoded hypervectors, $\overrightarrow{H}$, are compared for similarity with class hypervectors, $\mathbb{C}$, and normalized with the *softmax* function. We can then obtain the per-class error, $\vec{e}$, with the ground truth one hot encodings, y, and normalized similarities, $\vec{s}$. The class hypervectors are updated by binding the encoded hypervector scaled with the per-class errors and a learning rate, λ.

To update our projection matrix (Ⓒ), we generate an additional hypermatrix, $\mathbb{B}$, with the same dimensions as $\mathbb{P}$. The $\mathbb{B}$ hypermatrix is initialized with He uniform distribution [7], $\mathbb{B} \sim \mathcal{U}(-\sqrt{\frac{6}{F}}, +\sqrt{\frac{6}{F}})$, where F is the number of features. This random $\mathbb{B}$ matrix can be fixed for subsequent iterations of retraining. To update the encoder we simply bind to $\mathbb{P}$ the $\mathbb{B}$ hypermatrix scaled with the per-class errors averaged and λ, the learning rate.

In the back-propagation inspired TrainableHD [8], the projection matrix is given explicit feedback through class errors. However, in SEP, the random hypermatrix $\mathbb{B}$ presents implicit feedback to the encoder with only an approximate update direction and angle through averaged errors, $\vec{e}$. A *fixed* $\mathbb{B}$ matrix pushes updates to be guided by changes in the projection matrix $\mathbb{P}$. Although convergence of the model, especially during earlier iterations of training, may be slower than giving explicit feedback, the updates quickly improve to be on par with explicit updates.

Another method of implementing the $\mathbb{B}$ hypermatrix is newly generating the random matrix at every iteration, unlike the DFA method [6]. Because the matrix is not fixed, it is slightly less sensitive to changes in the projection matrix and therefore, is slower to converge. However, this method is more robust and still manages to reach the same performance as the fixed variant of SEP at the end of training.



(a) Training accuracy **(b) Test accuracy**

Fig. 2: SEP Accuracy Changes over Iterations

III. Experimental Results

We implement the SEP and IMP procedures on an NVIDIA GeForce RTX 2080 SUPER GPU with an Intel Core i9-10900K CPU using the PyTorch framework. We evaluate the two versions of the SEP algorithm, one with a fixed $\mathbb{B}$ (SEP fixed) and another with a newly generated $\mathbb{B}$ at each iteration (SEP var). For comparison, we chose TrainableHD [8] which uses a backpropagation-like method to implement a dynamic encoder. Each model is evaluated based on 50 retraining iterations.

Figure 2 shows accuracy changes during training and inference on the MNIST dataset for each retraining epoch. As expected, both fixed and variable versions of SEP start off with seemingly random updates and in the very initial iterations, this is even more so with SEP var. However, as iterations pass, both SEP models quickly find representations that best fit data features and shows performance comparable to that of TrainableHD 10K. SEP var and SEP fixed show final inference accuracies of 96.99% and 96.87% while TrainableHD results in 94.91% and 97.02% accuracy, respectively. SEP is able to achieve this through implicit feedback of rough update directions provided by the error projected matrix.

References

[1] P. Kanerva, *Sparse distributed memory*. MIT press, 1988.

[2] M. Imani, A. Zakeri, H. Chen, T. Kim, P. Poduval, H. Lee, Y. Kim, E. Sadredini, and F. Imani, "Neural computation for robust and holographic face detection," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, ser. DAC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 31–36. [Online]. Available: https://doi.org/10.1145/3489517.3530653

[3] P. Neubert, S. Schubert, and P. Protzel, "An introduction to hyperdimensional computing for robotics," *KI - Künstliche Intelligenz*, vol. 33, no. 4, pp. 319–330, Dec 2019. [Online]. Available: https://doi.org/10.1007/s13218-019-00623-z

[4] P. Sutor, D. Yuan, D. Summers-Stay, C. Fermuller, and Y. Aloimonos, "Gluing neural networks symbolically through hyperdimensional computing," 2022.

[5] Y. Kim, J. Kim, and M. Imani, "Cascadehd: Efficient many-class learning framework using hyperdimensional computing," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 775–780.

[6] A. Nøkland, "Direct feedback alignment provides learning in deep neural networks," 2016.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," 2015.

[8] J. Kim, H. Lee, M. Imani, and Y. Kim, "Efficient hyperdimensional learning with trainable, quantizable, and holistic data representation," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023, pp. 1–6.