Word2HyperVec: From Word Embeddings to Hypervectors for Hyperdimensional Computing

Alaaddin Goktug Ayar
University of Louisiana at Lafayette
Lafayette, LA, USA
alaaddin.ayar1@louisiana.edu

Sercan Aygun
University of Louisiana at Lafayette
Lafayette, LA, USA
sercan.aygun@louisiana.edu

M. Hassan Najafi
University of Louisiana at Lafayette
Lafayette, LA, USA
najafi@louisiana.edu

Martin Margala
University of Louisiana at Lafayette
Lafayette, LA, USA
martin.margala@louisiana.edu

ABSTRACT

Word-aware sentiment analysis has posed a significant challenge over the past decade. Despite the considerable efforts of recent language models, achieving a lightweight representation suitable for deployment on resource-constrained edge devices remains a crucial concern. This study proposes a novel solution by merging two emerging paradigms, the Word2Vec language model and Hyperdimensional Computing, and introduces an innovative framework named Word2HyperVec. Our framework prioritizes model size and facilitates low-power processing during inference by incorporating embeddings into a binary space. Our solution demonstrates significant advantages, consuming only 2.2 W, up to 1.81× more efficient than alternative learning models such as support vector machines, random forest, and multi-layer perceptron.

CCS CONCEPTS

• **Hardware** → **Emerging technologies**.

ACM Reference Format:
Alaaddin Goktug Ayar, Sercan Aygun, M. Hassan Najafi, and Martin Margala. 2024. Word2HyperVec: From Word Embeddings to Hypervectors for Hyperdimensional Computing. In *Great Lakes Symposium on VLSI 2024 (GLSVLSI '24), June 12–14, 2024, Clearwater, FL, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3649476.3658795

1 INTRODUCTION

Language processing systems are in high demand, especially with the advent of Chat Generative Pre-trained Transformer (ChatGPT) by OpenAI, which has sparked immense interest within the machine learning (ML) community. However, it is essential to consider the design aspect of language encoders and classifiers in hardware to create portable models suitable for future use in mobile devices. While large models are effective in most conversation contexts, there is a need for lightweight models, particularly for sentiment-based topics, which can benefit hardware designers and a subset of Large Language Models (LLMs).

GLSVLSI '24, June 12–14, 2024, Clearwater, FL, USA

ACM ISBN 979-8-4007-0605-9/24/06
https://doi.org/10.1145/3649476.3658795

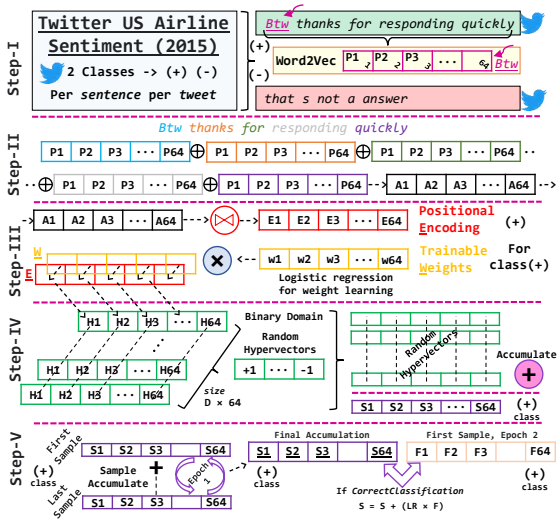

Figure 1: Proposed framework: Word2HyperVec.

Hyperdimensional computing (HDC) systems have obtained significant attention in recent years due to their lightweight architecture and single-pass learning capabilities [4]. By encoding data into binary hypervectors (+1, -1), cumulative data accumulations enable holistic brain-like learning [1]. Once data is represented in random binary hypervectors, learning involves binary logic operations such as shifting, accumulation, population counting, and binary similarity checks [3]. This study employs HDC as an intermediary to represent data in binary form, comprehensively representing multiple linguistic inputs from the language model encoder. Our results demonstrate comparable accuracy with conventional ML models, with reduced model size, shorter interference runtime, and lower power consumption.

2 PROPOSED FRAMEWORK

We propose Word2HyperVec, a framework that integrates two critical concepts: the Word2Vec model and hypervector processing. The framework leverages the Word2Vec model [5], utilizing the neural network-based model to generate linguistic data based on word embeddings within sentences. Each word undergoes encoding, contributing to the overall sentence embedding model. For evaluation, we utilize US Airlines Sentiments sourced from Twitter data [2]. This dataset comprises two classes, positive class (+) and negative class (-), reflecting the sentiment of tweet senders. We disregard neutral sentiments in this study.

As depicted in Step I of Figure 1, the framework first obtains samples (tweets) for each class, representing single sentences. Subsequently, Word2Vec generates encoding values for each word in Step II. These embeddings are then cumulatively added position-wise to form a single sentence embedding, denoted as A1 to A64 with a 64-embedding size. Analogous to HDC and its positional shifting, each sentence embedding undergoes positional embedding ($E1$,...,$E64$), which is then multiplied by trainable weights (w) adjusted via logistic regression for binary learning problems in Step III. This step alleviates the learning burden of HDC.

Step IV in Figure 1 illustrates the HDC learning phase, where binary data is derived from random hypervector generation using Ew ($embeddings \times weights$). Each Ew is converted into a D-size binary data (H) comprising random +1s and -1s. Similar to the accumulation step in baseline HDC, positional accumulations are applied over each H. The accumulated vector (S) in scalar form is updated through each sample until the end of the first epoch, serving as the reference epoch for validation purposes.

Once both positive and negative class labels generate their corresponding class vectors (S), starting from the second epoch, each incoming S undergoes cosine similarity evaluation to determine whether it aligns correctly with the expected label class. Correct classifications prompt the second epoch to adjust the sample effect using a learning rate (LR) and accumulate it back onto the corresponding class hypervector, depicted as $S_{\text{new}} = S_{\text{old}} + (\text{LR} \times F)$.

3 EVALUATION RESULTS

Simulation results shown in Figure 2 illustrate how epoch-based training benefits HDC models. As with many ML processes, the initial training phase is somewhat turbulent but quickly stabilizes. As we can see, in both scenarios of Figure 2 (a) with an LR of 1 and Figure 2 (b) with an LR of 10, the training accuracy improves consistently over epochs. Meanwhile, cosine similarity, an indicator of class separability in the model, decreases correspondingly, suggesting that the classes are becoming more distinct. This trend is reminiscent of the familiar loss curves in other ML training processes, reinforcing that the model genuinely learns to differentiate between classes. By reinforcing correctly predicted classifications, we effectively enhance the model's ability to form robust hyperdimensional representations for each class.

Table 1 reports the performance results. The most significant observation is the tangible improvement in test accuracy, achieving up to 82% accuracy. This leap in performance, accomplished without increasing model size or inference time, indicates a successful learning trajectory rather than mere data memorization. The plots in Figure 2 confirm that the model is internalizing the underlying structure of the data, evidenced by the steady learning and meaningful cosine similarity metrics. We observe that HDC surpasses other algorithms in efficiency. It exhibits inference times that are roughly 2.76× to 10.36× faster, is 1.77× to 1.82× more power-efficient, and maintains a model size that is approximately 20.42× to 1050.88× smaller than traditional ML algorithms. This underscores the potential of HDC, especially when optimized with an epoch-based training approach, to deliver high accuracy without increasing computational costs, making it a prime candidate for applications where both efficiency and accuracy are crucial.

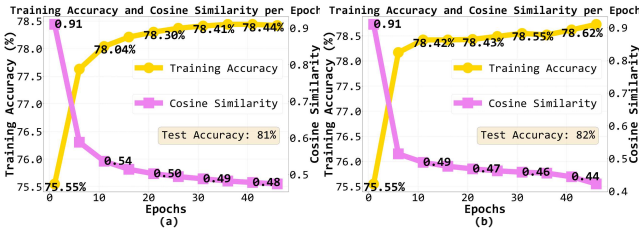

Figure 2: HDC model training with two different LRs. (a) a model with LR=1, (b) a model with LR=10.

Table 1: Word2HyperVec vs. Machine Learning Algorithms

Embedding❖	Algorithm❖	Accuracy	Mod. Size	Infer. Time (sec.)★	Power(W)●
word2vec	SVM	0.80	2.14 MB	0.0152	3.9 W
	RF	0.82	8.21 MB	0.0570	4 W
	MLP	0.82	163.39 KB	0.0124	3.3 W
	HDC	**0.82**	**8 KB**	**0.0055**	**2.2 W**

★ Inference conducted on an edge device equipped with a 1.2GHz quad-core ARM Cortex-A53 CPU, 1GB of RAM, and 64-bit architecture.
❖Embeddings are created using the Word2vec model for this study. Embedding model, HDC (D=1000), and ML models trained on a 12th Gen Intel® Core™ i7-12800H CPU with 20 logical cores.
● Power value for each algorithm measured with USB power-meter.

4 CONCLUSIONS

Examining HDC alongside established ML algorithms highlights HDC's remarkable potential for efficient computing. With an accuracy on par with robust models like RF and MLP and superior to SVM, it stands out for its exceptionally small model size and fast inference while maintaining the lowest power consumption among the tested algorithms. The implementation of an epoch-based training approach further enhances HDC. Remarkably, accuracy improvements do not come at the cost of model size or inference speed, both of which remain exceptionally efficient. These attributes make HDC a highly attractive option for devices with limited memory and energy, showcasing the HDC's important role in embedded computing, where hardware efficiency is paramount.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under Grants 2339701 and 2019511; University of Louisiana at Lafayette Computer Science Endowed Chair Fund; and generous gifts from Cisco and Nvidia.

REFERENCES

[1] Sercan Aygun, Mehran Shoushtari Moghadam, M. Hassan Najafi, and Mohsen Imani. 2023. Learning from Hypervectors: A Survey on Hypervector Encoding. arXiv, 2308.00685.
[2] Crowdflower. 2023. Twitter Airline Sentiment. https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment. Accessed: 2024-03-17.
[3] Lulu Ge and Keshab K. Parhi. 2020. Classification Using Hyperdimensional Computing: A Review. *IEEE Circuits and Systems Magazine* 20, 2 (2020).
[4] Denis Kleyko, Dmitri Rachkovskij, Evgeny Osipov, and Abbas Rahimi. 2023. A Survey on Hyperdimensional Computing Aka Vector Symbolic Architectures, Part II. *ACM Comput. Surv.* 55, 9 (2023).
[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *CoRR* abs/1310.4546 (2013).