Hierarchical Hyperdimensional Computing for Energy Efficient Classification

Mohsen Imani, Chenyu Huang, Deqian Kong, and Tajana Rosing
CSE Department, UC San Diego, La Jolla, CA 92093, USA
{moimani, chh217, dekong, tajana}@ucsd.edu

ABSTRACT

Brain-inspired Hyperdimensional (HD) computing emulates cognition tasks by computing with hypervectors rather than traditional numerical values. In HD, an encoder maps inputs to high dimensional vectors (hypervectors) and combines them to generate a model for each existing class. During inference, HD performs the task of reasoning by looking for similarities of the input hypervector and each pre-stored class hypervector However, there is not a unique encoding in HD which can perfectly map inputs to hypervectors. This results in low HD classification accuracy over complex tasks such as speech recognition. In this paper we propose MHD, a multi-encoder hierarchical classifier, which enables HD to take full advantages of multiple encoders without increasing the cost of classification. MHD consists of two HD stages: a *main stage* and a *decider stage*. The *main stage* makes use of multiple classifiers with different encoders to classify a wide range of input data. Each classifier in the *main stage* can trade between efficiency and accuracy by dynamically varying the hypervectors' dimensions. The *decider stage*, located before the *main stage*, learns the difficulty of the input data and selects an encoder within the *main stage* that will provide the maximum accuracy, while also maximizing the efficiency of the classification task. We test the accuracy/efficiency of the proposed MHD on speech recognition application. Our evaluation shows that MHD can provide a 6.6× improvement in energy efficiency and a 6.3× speedup, as compared to baseline single level HD.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning approaches**; *Supervised learning*;

KEYWORDS

Brain-inspired computing, Hyperdimensional computing, Machine learning, Energy efficiency

ACM Reference Format:
Mohsen Imani, Chenyu Huang, Deqian Kong, and Tajana Rosing. 2018. Hierarchical Hyperdimensional Computing for Energy Efficient Classification. In *DAC '18: DAC '18: The 55th Annual Design Automation Conference 2018, June 24–29, 2018, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3195970.3196060

1 INTRODUCTION

Today, a large amount of effort is being invested into the design and development of brain inspired computing, which relies on a paradigm of computing starkly different from traditional methods. The recent successes of deep learning networks have contributed a considerable amount in furthering interest into the bio-inspired areas of computing [1]. Neural Networks (NNs), particularly deep neural networks, have demonstrated high accuracy with regards to many cognitive tasks such as speech recognition, and language/text recognition [2, 3]. Although accuracy is an important factor, classification efficiency is becoming more important as many learning algorithms need to be processed on embedded devices with limited resources [4, 5]. However, existing classification algorithms such as neural networks, Support Vector Machine (SVM), and k-Nearest Neighbor (k-NN) are computationally expensive [6]. This eliminates the possibility of using these algorithms on resource limited embedded processors.

Hyperdimensional (HD) computing is a computational approach, which emulates cognitive tasks by computing with vectors in high-dimensional space (hypervectors) as an alternative to traditional deterministic computing with numbers. The idea of HD computing is based on the understanding that brains compute with *patterns of neural activity* that are not readily associated with numbers. Unlike standard computing architectures, HD uses these patterns to process inputs without using a traditional numerical representation [7]. In fact, raw numerical computing ability of HD is relatively feeble when compared against the results achievable by modern machines.

HD models neural activity patterns using hypervectors with dimensionality in the thousands (e.g., D=10,000). HD computing builds upon a well-defined set of operations between randomly generated hypervectors. These sets of operations create a framework that is extremely robust in the presence of failure, and offers a complete computational paradigm that is easily applied to learning problems [7]. The main difference between HD and traditional computing techniques is the way data is represented, in that HD computing represents the data as approximate patterns which can be scaled very efficiently for a wide array of learning applications. HD consists of an encoder and an associative memory. The encoder block maps input data to high dimensional vectors (hypervectors) and combines them to generate a model for each existing class. During inference, an associative memory performs the task of reasoning by looking at the similarity of the input hypervector to each of the stored model hypervectors. Examples include analogy-based reasoning [8], language recognition [9], text classification [10, 11], biosignal processing [12], speech recognition [13], DNA Sequencing [14], and prediction from multimodal sensor fusion [15].

However, HD computing provides poor accuracy over complex tasks such as speech recognition, as there is not a unique encoder which could perfectly map inputs to hypervectors while preserving all input information. In this paper, we propose a hierarchical Hyperdimensional computing (MHD) which enables HD to take advantage of multiple encoders without increasing the classification cost. MHD consists of two stages: a *main stage* and a *decider stage*. The *main stage* uses classifiers with different encoders to classify a wide range of input data. Each classifier in the *main stage* can trade efficiency-accuracy by dynamically varying the hypervector dimensions. In MHD, the *decider stage* which is located before the *main stage*, learns the difficulty of input data and selects an encoder accordingly in the *main stage* which will provide the maximum accuracy, while also maximizing the efficiency of the classification task. We tested

DAC '18, June 24–29, 2018, San Francisco, CA, USA

ACM ISBN 978-1-4503-5700-5/18/06…$15.00
https://doi.org/10.1145/3195970.3196060

the accuracy/efficiency of the proposed MHD on speech recognition application. Our evaluation shows that MHD can provide a 6.6× improvement in energy efficiency and a 6.3× speedup, as compared to baseline single level HD.

2 HYPERDIMENSIONAL COMPUTING

2.1 HD Overview

HD computing is a computing paradigm involving long vectors with dimensionality in the thousands called hypervectors [7]. In high-dimensional space, there are several nearly orthogonal hypervectors [16]. HD exploits well-defined vector operations to combine these hypervectors, while also preserving most of the information of the hypervectors. Hypervectors are holographic and (pseudo) random with i.i.d. components and full holistic representation, thus no component has more responsibility to store any piece of information than any other hypervector.

2.2 Classification Applications

Classification has applications in many different domains. For instance, speech and object recognition are increasingly common in consumer electronics. There are well-defined pre-processing steps to extract the features of different input types. For example, voice techniques such as Mel-frequency cepstral coefficients (MFCCs) [17] extract and map raw voice information into the frequency domain. Regardless of the input type, the extracted features are ussually represented in a single feature vector with N elements. The goal of learning algorithms is to generate a model which can identify the patterns within the feature vectors.

Figure 1a shows the overview of the classification in high dimensional space. HD consists of an encoder and an associative memory. For all sample data within a class, HD maps data to high dimensional vectors, called hypervectors, then combines them together to create a single hypervector modeling each class. All trained class hypervectors are stored in an associative memory. During the inference process, the same encoding scheme maps test input data to high dimensional space. Associative memory looks at the similarity of the generated query hypervector against all stored class hypervectors. The input then gets the label of that class with which it has the highest similarity with.

The goal of HD is to encode the input data (feature vector) to a single hypervector with D dimensions. Our encoder should consider the impact of each feature position and feature value on the final encoded hypervector. Assume each feature can get a value between F_{min} to F_{max}. Our design divides this feature range into m equal levels $[F_{min}, F_{min} + \Delta Fm, ..., F_{max}]$, where L_i is the i^{th} feature level and $\Delta F = F_{max} - F_{min}$. Level hypervectors need to have correlation such that the neighbor levels provide higher similarity. We randomly generate a bipolar hypervector of the L_1 with D dimensions (-1 and +1 elements). To consider correlation, we randomly select Dm bits of L_1 and flip them to generate L_2 level hypervector. This procedure continues until generating the hypervector of L_m by flipping Dm random bits of the L_{m-1} hypervector. Our design stores the generated level hypervectors in item Memory (iM).

2.3 HD Encoders

We proposed two encoders to map vector features to high dimensional space (shown in Figure 1b,c). The difference between these encoders is in the way they consider the impact of each feature position on the final hypervector. In the following, we explain the encoders functionality in detail:

Encoder I: Record-based encoder

This encoding scheme assigns a unique channel ID to each feature position. These IDs are hypervectors which are randomly generated such that all features will have orthogonal channel IDs, i.e., $\delta ID_i, ID_j < 5,000)$ for $D = 10,000$ and $i \neq j$; where the δ measures the element-wise similarity between the vectors. These hypervectors are stored in a Channel item Memory(CiM) as shown in Figure 1b. Encoder I looks at each position of the feature vector and element-wise multiplies the channel ID (ID_i) with the corresponding level hypervector (hv_i). The following equation shows how the N feature IDs and hypervectors bind together to generate a single data hypervector in i^{th} class:

$$S_1^i = hv_1^i * ID_1^i + hv_2^i * ID_2^i + \ldots + hv_N^i * ID_N^i$$
$$hv_j \in \{L_1, L_2, \ldots, L_m\},\ 1 \leqslant j \leqslant N$$

Encoder II: Ngram-based encoder

Unlike Encoder I, the second encoder differentiates feature positions by assigning a unique permutation for each feature (shown in Figure 1c). For instance, the level hypervector corresponding to n^{th} feature rotationally permutes by $n-1^{th}$ positions. The following equation shows how hypervectors are combine different features in input data to generate a single hypervector for an input data in i^{th} class:

$$S_1^i = hv_1^i + \rho hv_2^i \ \ldots + \rho^{N-1} hv_N^i$$
$$hv_j \in L_1, L_2, \ldots, L_m,\ 1 \leqslant j \leqslant N$$

For all input data within a class ($\{S_1^i, S_2^i, \ldots, S_K^i\}$), our design generates hypervectors in a similar fashion and then binds them together to generate a single class hypervector.

$$C_i = S_1^i + S_2^i + \ldots + S_K^i$$

After training the HD model, we adjust the HD model based on the retraining algorithm proposed in [13]. This retraining happens for pre-specified number of iteration, unless the model accuracy stays the same from one iteration to the next one. At inference, a test input data is encoded to a hypervector using the same encoding used for training (shown in Figure 1a). HD checks the similarity of this query hypervector against all pre-stored classes in an associative memory. The class with the highest Cosine similarity will be selected as the output class.

3 HD CLASSIFICATION ENHANCEMENT

3.1 Multiple Encoders

In HD, there does not exist a single universal encoder with the ability to map data to high dimensional space while keeping all input information. In fact, each encoder can work properly only for a specific types of input data. To show this, we look at speech recognition problem. For speech recognition, we focus on the Isolet dataset [18] with the goal of recognizing the pre-processed voices among 26 letters of English alphabet.

Table 1 shows the HD classification accuracy using Encoder I, Encoder II using hypervectors with 10,000 dimensions. In speech recognition, we observe that Encoder I does a poor job differentiating similar letters such as $\{T, C, Z, E, \ldots\}$. However, Encoder II can address these classification issues by using permutation instead of ID hypervectors. On the other hand, Encoder II is exclusive and does not properly classify the voice of less similar letters, while Encoder I does. In order to achieve high classification accuracy, we propose MHD, an adaptive hierarchical hyperdimensional computing design which exploits the advantage of multiple encoders to improve classification accuracy.

In addition, the result shows the best HD classification accuracy occurs when the HD benefits from both encoders. The results show

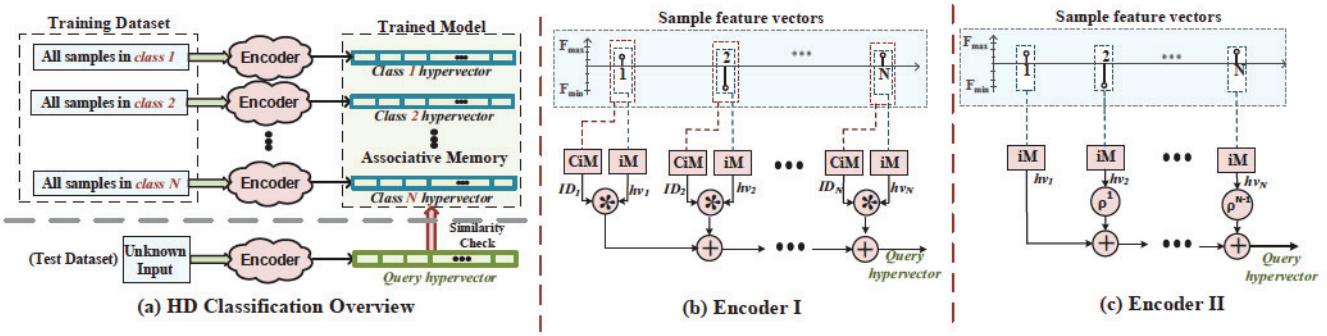

Figure 1: (a) The overview of HD architecture. The functionality of (b) record-based encoder and (c) Ngram-based encoder mapping sample feature vectors to high dimensional space.

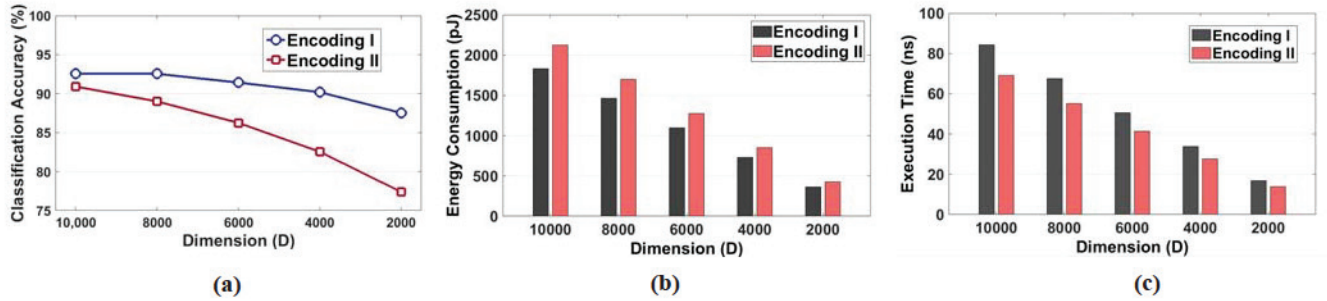

Figure 2: The impact of hypervector dimensions on (a) speech recognition accuracy, (b) energy consumption and (c) execution time of the HD using different encoding modules.

Table 1: Classification accuracy of speech recognition using different encoding schemes.

	Encoder I	Encoder II	MHD (Both Encoders)
Recognition Accuracy	92.5%	90.9%	95.9%

that HD using multiple encoders can achieve 3.4% higher classification accuracy as compared to single encoder. In order to allow MHD to benefit from multiple encoders, our design needs to be adaptive and must select the proper encoder depending on the input data. In Section 4, we explain how our proposed MHD benefits from both encoding schemes at the same time.

3.2 Dynamic Dimension Reduction

The energy consumption and execution time of the HD block depend on both encoder and associative memory. For speech recognition (using Encoder I), HD uses a large sized encoder with 617 input channels. This large encoder takes 65% and 47% of total energy consumption and execution time respectively of the HD. Reducing the dimension of the hypervectors used is one of the most effective ways to improve HD efficiency. Figure 2a shows the impact of scaling the hypervector dimensions on the speech recognition accuracy using Encoder I and Encoder II. The results show that both encoders have high robustness to dimension reduction. However, this robustness is much higher for Encoder I. For instance, reducing hypervector dimensions to 8,000, the HD using Encoder I can still provide the same accuracy as HD using the full 10,000 dimensions. Further reducing dimensions to 2,000, Encoder I and Encoder II can provide 88.5% and 78.4% recognition accuracy, respectively. This result indicates that the HD using Encoder I and Encoder II can classify 88% and 78% of data correctly, even while using hypervectors with 2,000 dimensions. This means that we do not need to use costly HD with full 10,000 dimensions for classification. Instead we can classify majority of data using HD with 2,000 dimensions while the HD with 10,000 dimensions can be use to classify the more difficult tasks.

Figure 2b,c shows the average energy consumption and execution time of HD when hypervector dimension scales from 10,000 to 2,000. The result shows that HD energy consumption and execution time linearly scales with the hypervector dimensions. For instance, in 2,000 dimensions, HD can achieve a 4.7× energy efficiency improvement and a 3.9× speedup when compared to HD with 10,000 dimensions. One main advantage of HD is that it does not require a different training model for classifiers with smaller dimensions. HD enjoys full holistic hypervector representation, meaning that no component in hypervector is more representative than others. Therefore, using the HD model with $D = 10,000$, we can perform the classification on smaller dimensions of the same model to reduce the classification cost. In other words, we can have the low cost classifiers which are using the same model as Encoder I and Encoder II, but use a part of hypervector dimensions for classification (e.g. 2,000 dimensions). This eliminates the needs to train a separate HD model and also reduces the memory/hardware cost to keep a separated model for low cost classifiers.

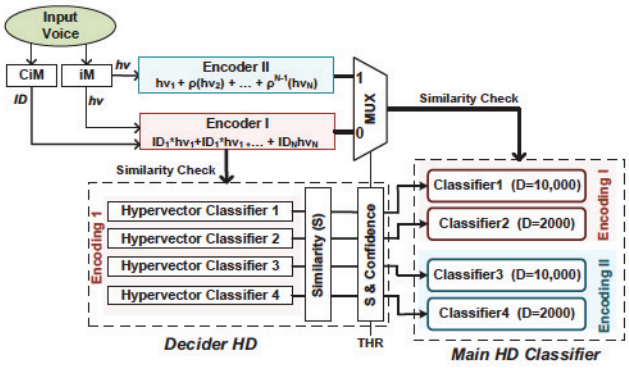

Figure 3: The overview of the proposed MHD architecture.

4 ADAPTIVE HIERARCHICAL HD

4.1 MHD Overview

In order to benefit from multiple encoder as well as low cost HD classifier, our design needs to pre-recognize input data and assign them to the proper encoder with minimum dimensions. In this section, we explain the functionality of the proposed hierarchical HD classifier, called MHD, supporting multiple encoders as well as low cost classifiers. In the example explained in Section 3, MHD has four classifiers: two main classifiers using Encoder I and Encoder II with high dimensions, e.g. 10,000 bits, and two low cost classifiers using the same encoders but with lower dimensions, e.g. 2,000 bits. Figure 3 shows the architecture of the MHD design consisting of a *Decider* and *Main stages*. *Decider stage* gets general information from input data by mapping the input hypervector using Encoder I. Then, it accordingly selects classifiers in the *main stage* which would possibly classify input data correctly. The *Decider stage* looks at the similarity of a query hypervector to all class hypervectors. All classes with a similarity higher than a decider confidence value (set by user) will be selected as possible target classifiers. However, it is obvious that HD with 10,000 dimensions always outperforms an HD design with 2,000 dimensions (using the same encoder). Thus, the *decider* HD should select a classifier in the *main stage* which results in: (i) maximum accuracy as well as (ii) minimum classification cost. For instance, if the *decider* has high confidence that Encoder I with both 2,000 and 10,000 dimensions can correctly classify as an input data, the *decider* selects low cost classifier with 2,000 dimensions to perform the classification task.

4.2 MHD Training & Confidence

MHD trains in two steps: (i) the *main stage* trains for both encoders in parallel and independently. This training is one-shot learning and gives us two HD models with a full 10,000 dimensions. MHD can use a smaller version of such models for low cost classifiers. (ii) A *decider* HD trains over whole training dataset by initializing multiple hypervectors, each representing a classifier in the *main stage*. The number of hypervectors in *decider* HD depends on the number of available classifiers in the *main stage* (Four in our example). After training all class hypervectors in *main stage*, the training in the decider stage starts by initializing all *decider* hypervectors to zero values. The decide hypervectors train depending on how well the *main stage* can classify data. Therefore, to find *decide* model, our design checks the similarity of each input data to all main stage classifiers. Then our design adds such input hypervector to decide vector in *decider HD*, if the corresponding main stage classifier could correctly classify input data. For example, if an input data in main stage could classify by Encoder I with 2,000 and 10,000 dimensions, our design adds input hypervector to the corresponding hypervectors at the *decider stage* (shown in Figure 3). However, if input data is predicted to wrongly classify by a classifier, no hypervector will be added to corresponding hypervector. This process continues on the training dataset until generating a *decider HD* with multiple hypervectors, each corresponds a classifier in the *main stage*.

Table 2: The impact of the *decider* confidence on the efficiency and accuracy of the MHD in 4-level configuration.

Decider Confidence	75%	80%	85%	90%	95%
Recognition Accuracy	93.5%	94.5%	95.2%	95.7%	95.9%
Energy saving	76.4%	72.1%	62.3%	34.1%	5.7%
Speedup	72.5%	70.6%	67.4%	57.8%	35.4%

During the inference stage, when an input data is loaded into the system, *decider HD* checks its similarity against all stored hypervectors. A class or classes which have the highest Cosine distance similarity to input data (higher than a pre-defined threshold value, THR) can be considered as classifiers which could correctly classify such data. The *Decider HD* looks at all the models in the *main stage* which could possibly classify that particular task and activate a classifier with a lowest dimension. If none of the classes have the confidence level to classify a particular input (i.e., Cosine distance similarity less than a pre-defined threshold over all classes), our design assigns such input data to a class which has the highest similarity. As default, the *decider stage* encodes inputs using Encoder I. If the *decider* selects a classifier which uses Encoder I, our design does not need to pay the cost of encoding again. MHD exploits this characteristic to be biased toward a low cost classifiers. Therefore, after the low dimension classifiers, a classifier which uses the same encoder as the *decider stage* will have second priority.

Table 2 shows the impact of the confidence level on the classification accuracy and efficiency of the MHD design using four classifiers in the *main stage*. The confidence value is a Cosine similarity of an input hypervector with the *decider HD* hypervectors. As our evaluation shows, reducing a threshold confidence below 95% enables classifiers with lower confidence be assigned for the classification task. This reduces the classification accuracy of the MHD design. However, reducing the confidence level improves the efficiency of the classification. The results in Table 2 show the classification accuracy, the average energy savings and the speedup that MHD can achieve when compared to the HD using Encoder I. As results show, reducing the confidence value to 75% improves the performance and energy efficiency of the MHD by 77.4% while providing about 2.4% lower classification accuracy. Note that these accuracies are still higher than the accuracy that HD using single encoding module provides.

4.3 Decider Stage Relaxation

Running the *decider HD* at the top of the *main stage* is not always cheap. For MHD with many encoders in the *main stage*, a *decider HD* requires a large associative memory to store hypervectors corresponding to each class. This reduces the overall advantages that our design can provide. For the speech recognition example, the *main stage* with four classifiers requires a *decider HD* with four hypervectors. In this configuration, the *decider* adds 426 $\approx 15\%$ energy overhead to classification. As HD with $D = 2,000$ takes about $5\times$ lower energy than HD with $D = 10,000$, our design needs to run at least 18% of inputs on HD with $D = 2,000$ in order to compensate the *decider HD* overhead. Our design uses two approximations to

Table 3: Model size and classification accuracy of MHD in different configurations.

		1-level	2-level	4-level	6-level	8-level
Configuration	Encoder	Encoding 1	Encoder I, Encoder II	Encoder I, Encoder II	Encoder I, Encoder II	Encoder I, Encoder II
	Dimensions	10,000	10,000	2,000, 10,000	2,000, 4,000 10,000	2,000, 4,000 6,000, 10,000
Speech Recognition	Effective Dimension	10,000	10,000	6,000	5,400	4,600
	Model Size	67.5KB	70KB	72.5KB	75KB	77.5KB
	Classification Accuracy	93.6%	95.9%	95.9%	95.9%	95.9%

improve the efficiency of the *decider HD*:
(i) Sampling: sampling data features to generate input hypervector. We do not need to use all input features to generate a voice hypervector. Instead, we can use a part of input frequencies to generate input data.
(ii) Dimension reduction: reducing dimensionality of the *decider HD* could significantly improve the classification efficiency.
In Section 5.3, we will explore the impact of the *decider HD* relaxation on the overall MHD efficiency and accuracy.

5 EXPERIMENTAL RESULTS

5.1 Experimental Setup

To estimate the cost of digital design, we use a standard cell-based flow to design dedicated hardware for MHD. We describe the proposed designs using RTL System-Verilog. For the synthesis, we use *Synopsys Design Compiler* with the TSMC 45 nm technology library with the general purpose process and high V_{TH} cells. We extract its switching activity using *ModelSim* by applying the test sentences. We measure the power consumption of HD designs using *Synopsys PrimeTime* at (1 V, 25°C, TT) corner.

We describe the functionality of the proposed MHD using C++ implementation. We compare the efficiency and accuracy of MHD architectures with state-of-the-art classification techniques running on Intel Core i7 processor with 16 GB memory (4-core, 2.8GHz). For the measurement of CPU power, we use Hioki 3334 power meter. We use this to test the efficiency of the proposed design on speech recognition application, where the goal is to recognize voice audio of the 26 letters of the English alphabet. The training and testing datasets are taken from the Isolet dataset [18]. This dataset consists of 150 subjects speaking each letter of the alphabet twice. The speakers are grouped into sets of 30 speakers. The training of hypervectors is performed on *Isolet*1,2,3,4, and tested on *Isolet* 5.

5.2 MHD Efficiency-Accuracy Trade-off

We explore the accuracy and efficiency of the MHD when the confidence of the HD decider is changed. Table 3 shows different MHD configurations. The baseline is an HD using a single encoder (Encoder I). The second configuration is an HD with two different encoding schemes, both working with 10,000 dimensions. In the third configuration, our design can select hypervectors with 2,000 dimensions over each encoding. The 6-level (8-level) configuration has two (four) more intermediate classifiers with $D = 4,000$ ($D = 4,000$ and $D = 6,000$) dimensions over both encoding modules.

As explained before, our design uses the same model as $D = 10,000$ for classifiers with lower dimensions. Table 3 shows the MHD model size in different configurations. Our results show that there is a jump in model size, while going from 1-level to 2-level configuration. However, increasing the number of levels does not change the model size significantly. This is because in multi-level design, the size of main classifier is fixed over all configurations (i.e., fixed encoding). Thus, using a larger number of levels only increases the *decider HD* model size.

Table 3 shows the classification accuracy of MHD in different configurations during recognition task. The results are obtained while setting the confidence value to 95% for the decider HD. MHD in 2-level configuration can improve the classification accuracy by 3.5%, by adaptively switching between the encoding modules. This improvement is achieved at the cost of adding extra decider block with a 27% energy and a 4% performance penalty. MHD in 4-level configuration does not further improve the classification accuracy because the two new classes simply use the same encoder in different dimensions. However, MHD in this configuration (using classifiers with $D = 2,000$ dimensions) can provide 5.7% higher energy efficiency and a 35.4% speedup as compared to single stage HD. Increasing the number of levels to six gives more flexibility to MHD to select a better low cost encoder to classify incoming input data. MHD in this configuration can provide a similar accuracy to the original HD with 42.5% and 52.8% energy efficiency improvement and speedup respectively when compared to baseline HD (using single encoder). Increasing the number of levels to eight does not further improve the classification efficiency because MHD in 8-level configuration requires a large and costly *decider HD*.

Figure 4 shows the impact of the *decider* confidence on the accuracy classification, energy consumption and speedup of MHD. The energy and speedup are normalized to a 1-level HD. Our result shows that in all the configurations, MHD with higher *decider* confidence provides higher classification accuracy. Although reducing the confidence level below 95% slightly degrades the classification accuracy, it significantly improves the classification efficiency by assigning a greater portion of tasks to low dimension classifiers, and therefore reducing the effective MHD dimension (shown in Table V). MHD with a larger number of levels has higher robustness to reduction in *confidence* level. For example, reducing the confidence level from 95% to 75% degrades the classification accuracy of 4-level MHD by 2.4%, while only having a 1.1% impact on accuracy for MHD with 6-level configurations. The higher robustness of 8-level MHD is due to the availability of a greater number of intermediate levels that inputs can be classified to. In terms of efficiency, 4-level MHD shows higher potential for energy savings, as compared to 6-level and 8-level designs. While accepting a 1% quality loss, MHD in 4-level configuration (using 80% confidence) can achieve 8.1× energy-delay product (EDP) improvement as compared to 1-level HD. This EDP improvement increases to 27.2× and 14.3× for MHD in 6-level and 8-level configurations.

5.3 Decider Relaxation

As explained before, the *decider HD* can be an energy bottleneck of MHD design when the number of classifiers in the *main stage* increases over six. In general, for MHD with a large number of classes, the *decider HD* takes a considerable part of total MHD energy and execution time. For instance, in 6-level MHD (95% decider confidence), the *decider* takes in an average of 37% of total MHD energy consumption and 58% of total execution time. Therefore, relaxing the computational complexity of the *decider* can further improve MHD classification efficiency. However, our

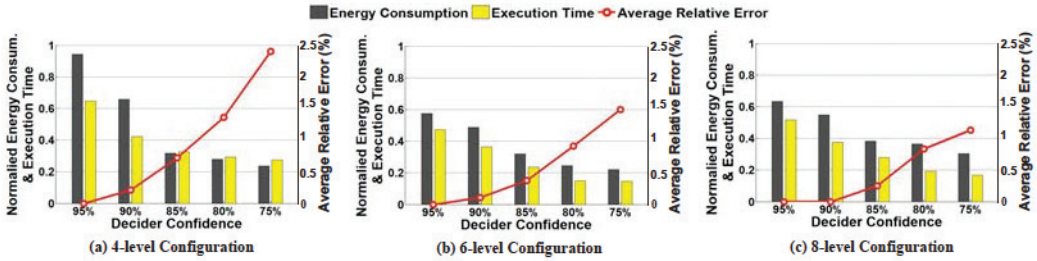

TABLE V
CONFIGURATION AND EFFICIENCY OF MHD ACCEPTING DIFFERENT ERRORS

Configurations		4-Level	6-Level	8-level
@ 0% Error	*Confident*	95%	95%	90%
	Effective D	6000	5400	4600
	Speedup	35.4%	52.8%	62.4%
	EDP Improv.	1.6×	3.6×	4.8×
@ 1% Error	*Confident*	80%	80%	80%
	Effective D	2400	2100	2060
	Speedup	67.4%	84.9%	80.7%
	EDP Improv.	8.1×	27.2×	14.3×

Figure 4: Impact of the *decider* confidence on the speech recognition accuracy and efficiency (normalized to 1-level HD) .

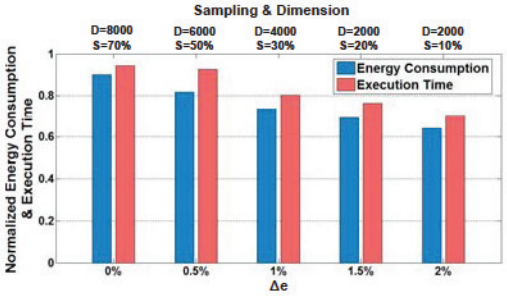

Figure 5: Normalized energy consumption and execution time of MHD using sampling and dimension reduction in the *decider HD*.

design should ensure that the changes have minor impact on MHD accuracy.

Figure 5 shows the impact of sampling and dimension reduction on the energy consumption and execution time of MHD. The graph shows the result when MHD accepts 0%, 0.5%, 1%, 1.5% and 2% quality losses when compared to the baseline MHD where the decider is not relaxed ($\Delta e = e_{relaxed} - e_{baseline}$). The top x-axis on Figure 5 shows the best values for sampling and hypervector dimensions which correspond to the Δe error reported on bottom x-axis. Our evaluation shows that relaxed MHD can achieve same accuracy as that of the baseline (no sampling and $D = 10,000$) when the *decider* dimension is reduced to 8,000 and samples 70% of the inputs data. In this configuration, MHD can achieve 10.3% energy efficiency improvement and a 6.4% speedup when compared to the baseline. Reducing the sampling rate and dimension of decider HD further improves the MHD efficiency at the cost of lower classification accuracy. For example, while accepting a 2% loss in accuracy, MHD can improve the classification energy and execution time by 6.6× and 6.3× (2.7× and 3.3×) as compared to single level HD (baseline 6-level MHD). Note that this accuracy is still 0.3% higher than the accuracy for a single stage HD. Relaxing the decider HD ($D = 4,000$ dimensions and 30% sampling) and using a decider with 80% confidence improves the energy consumption and execution time of MHD by 9.3× and 19.4×, while providing 94.0% classification accuracy.

6 CONCLUSION

In this paper we propose a novel hierarchical hyperdimensional (HD) classifier, which enables the classifier to take advantage of multiple encoders without increasing the classification cost. MHD consists of two stages: a *main stage* and a *decider HD*. The *main stage* uses multiple classifiers with different encoders to classify a wide range of input data. Each classifier in the *main stage* can trade efficiency-accuracy by dynamically varying the hypervector dimensions. The *decider stage*, which is located before the *main stage*, learns the difficulty of the input data and accordingly selects an encoder in the *main stage* which provides the maximum accuracy, while maximizing the efficiency of the classification task. We test the accuracy/efficiency of the proposed MHD on speech recognition application. Our evaluation shows that MHD can provide a 6.6× improvement in energy efficiency and a 6.3× speedup, as compared to baseline single level HD.

ACKNOWLEDGMENT

This work was partially supported by CRISP, one of six centers in JUMP, an SRC program sponsored by DARPA, and also NSF grants #1730158 and #1527034.

REFERENCES

[1] Y. LeCun *et al.*, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] D. Amodei *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *ICML*, pp. 173–182, 2016.
[3] S. I. Venieris *et al.*, "fpgaconvnet: A framework for mapping convolutional neural networks on fpgas," in *FCCM*, pp. 40–47, IEEE, 2016.
[4] J. Venkatesh *et al.*, "Scalable-application design for the iot," *IEEE Software*, vol. 34, no. 1, pp. 62–70, 2017.
[5] M. Shafique *et al.*, "Adaptive and energy-efficient architectures for machine learning: Challenges, opportunities, and research roadmap," in *IEEE ISVLSI*, pp. 627–632, IEEE, 2017.
[6] S. Suthaharan, "Big data classification: Problems and challenges in network intrusion prediction with machine learning," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 4, pp. 70–73, 2014.
[7] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, 2009.
[8] P. Kanerva, "What we mean when we say "whatâĂŹs the dollar of mexico?": Prototypes and mapping in concept space," in *AAAI Fall Symposium*, pp. 2–6, 2010.
[9] A. Joshi *et al.*, "Language geometry using random indexing," *Quantum Interaction 2016 Conference Proceedings*, In press.
[10] M. Imani *et al.*, "Low-power sparse hyperdimensional encoder for language recognition," *IEEE Design & Test*, vol. 34, no. 6, pp. 94–101, 2017.
[11] M. Imani *et al.*, "Exploring hyperdimensional associative memory," in *IEEE HPCA*, pp. 445–456, IEEE, 2017.
[12] A. Rahimi *et al.*, "Hyperdimensional biosignal processing: A case study for emg-based hand gesture recognition," in *IEEE ICRC*, October 2016.
[13] M. Imani *et al.*, "Voicehd: Hyperdimensional computing for efficient speech recognition,"
[14] M. Imani *et al.*, "Hdna: Energy-efficient dna sequencing using hyperdimensional computing," *IEEE BHI*, 2018.
[15] O. Rasanen *et al.*, "Sequence prediction with sparse distributed hyperdimensional coding applied to the analysis of mobile phone use patterns," *IEEE TNNLS*, vol. PP, no. 99, pp. 1–12, 2015.
[16] P. Kanerva, "Encoding structure in boolean space," in *ICANN 98*, pp. 387–392, Springer, 1998.
[17] B. Logan *et al.*, "Mel frequency cepstral coefficients for music modeling.," in *ISMIR*, 2000.
[18] "Uci machine learning repository." http://archive.ics.uci.edu/ml/datasets/ISOLET.