Adversarial Attack on Hyperdimensional Computing-based NLP Applications

Sizhe Zhang*, Zhao Wang‡, Xun Jiao*
*Villanova University, ‡University of Chicago

***Abstract*—The security and robustness of machine learning algorithms have become increasingly important as they are used in critical applications such as natural language processing (NLP), e.g., text-based spam detection. Recently, the emerging brain-inspired hyperdimensional computing (HDC), compared to deep learning methods, has shown advantages such as compact model size, energy efficiency, and capability of few-shot learning in various NLP applications. While HDC has been demonstrated to be vulnerable to adversarial attacks in image and audio input, there is currently very limited study on its adversarial security to NLP tasks, which is arguable one of the most suitable applications for HDC. In this paper, we present a novel study on the adversarial attack of HDC-based NLP applications. By leveraging the unique properties in HDC, the similarity-based inference, we propose similarity-guided approaches to automatically generate adversarial text samples for HDC. Our approach is able to achieve up to 89% attack success rate. More importantly, by comparing with unguided brute-force approach, similarity-guided attack achieves a speedup of 2.4X in generating adversarial samples. Our work opens up new directions and challenges for future adversarially-robust HDC model design and optimization.**

I. Introduction

Natural language processing (NLP) aims to process and analyze large amounts of natural language data, e.g., text, to understand and extract contextual insights and nuances in the document. Modern NLP is largely developed using the emerging machine learning (ML) methods such as deep neural networks, which has shown superior performance in variety of fields within NLP such as spam detection [5], sentiment analysis [12], and question answering [9]. Popular universal language models include Word2Vec [17], ELMo [21], Glove [20] and BERT [6], which have been developed to extract word semantics into vectors and combined/included the neural network model like LSTM [11] and Transformer [24].

However, recent studies have shown that ML-based NLP applications are vulnerable to adversarial attacks [2], [8], which can become a notable security threat to the security-critical applications such as spam detection. In adversarial attack, the imperceptible perturbations on the input can lead to the wrongfully-predicted results. The adversarial attack problem first raises awareness in the image classification field [7], and then was observed in NLP domain as well. For example, a single character change may result in a change in the meaning of the word [8], as well as changes to the word itself may result in a change in its grammatical meaning [2].

Recently, an emerging brain-inspired method called hyperdimensional computing (HDC) has shown promising accuracy and efficiency in various NLP tasks [15], [22], [23]. This "non-von Neumann" computing scheme aims to imitate human brain functions to process information in high-dimensional space. Compared with DNNs, HDC shows advantages such as compact model size, energy efficiency, and capability of few-shot learning. Nevertheless, HDC also faces security challenges like DNNs, e.g., adversarial samples can fool HDC to make wrong predictions [16]. To the best of our knowledge, there is currently no study to automate the generation of adversarial attacks for HDC-based NLP applications. Thus, the paper aims to provide a novel effort in this direction and raise awareness of the community to jointly design adversarially-robust HDC-based NLP applications.

Our contributions are summarized as follows:

- We present a novel effort in automatically generating adversarial samples for HDC-based NLP applications. Specifically, we develop a similarity-guided approach to automatically replace words with their synonyms, which preserves the original semantic meaning of the text but leads to incorrect classifications.
- To develop the similarity-guided approach, we propose two guide scores, cosine similarity and integrated similarity which synthesizes four different similarity metrics in HDC models.
- We perform adversarial attacks on spam detection benchmarks and compare their advantages and disadvantages based on a variety of datasets. Our experimental result shows that, our guided approach is able to achieve up to 89.49% attack success rate. Besides that, our guided approach is able to generate adversarial samples 2.4X faster than the unguided approach. By using the integrated similarity as the guide score, we are able to further generate 30.4% more adversarial samples and 39.4% faster speed than the cosine similarity approach.

II. Related Work

The generation of adversarial samples for NLP applications has received considerable attention in recent years. Unlike the adversarial image sample, small perturbations, such as a rewording, may significantly alter the semantic meaning of the original sentence. Due to this, attacking the model for NLP tasks is more challenging. Researchers have explored different levels of perturbation in order to generate NLP adversarial samples. As an example, altering characters from plain text can generate adversarial samples [8]. A small spelling mistake, character replacement, or other minor changes in a word are considered typos and will not affect the original semantic meaning. On the word level, as a result of the fact that substituting

synonyms for words is unnoticeable and less risky in this domain, word-level perturbation is more common [2], [14]. Rephrasing the whole sentence is also feasible but challenging to implement [13].

While HDC is an emerging learning method, many researchers have begun to study its vulnerability and security. In the image classification domain, a genetic algorithm was used to generate adversarial samples for handwritten digits with a 78% attack success rate [25]. In voice recognition, with a differential evolution algorithm, researchers can achieve an 85.7% success rate when they launch non-target attacks against HDC [4]. Recently, a framework called HDTest uses differential fuzz testing methods to systematically examine the robustness of HDC model thoroughly [16].

Due to the computational differences between DNNs and HDCs, existing adversarial attack techniques of DNNs cannot be applied directly to HDC models due to the indifferentiable architecture of HDC [16]. Furthermore, to the best of our knowledge, there is currently no study on automatically generating adversarial attacks for HDC-based NLP tasks. This paper presents a novel study in this domain.

III. Background

In this section, we present the background of applying HDC methods to NLP tasks. First, we introduce the fundamental elements and operations of HDC. Then, we describe several stages of applying HDC in NLP tasks, including encoding, one-pass training, inference, and retraining.

A. HDC basic element and operations

The hypervector(HV) is the most fundamental component of HDC models. Training and inference rely on high dimensional vectors as the basic dataflow element. In the HDC model, HVs need to be in a fixed dimension. Additionally, the HDC model is dominated by HV's element-wised operations. Common operations in the HDC model are element-wise addition, multiplication, and HV permutation. The detail is shown in Eq. 1.

$$\begin{aligned} &\vec{H_x} + \vec{H_y} = \langle h_{x_1} + h_{y_1}, h_{x_2} + h_{y_2}, \ldots, h_{x_n} + h_{y_n} \rangle \\ &\vec{H_x} * \vec{H_y} = \langle h_{x_1} * h_{y_1}, h_{x_2} * h_{y_2}, \ldots, h_{x_n} * h_{y_n} \rangle \\ &\rho^1(\vec{H}) = \langle h_n, h_1, h_2, \ldots, h_{n-1} \rangle \end{aligned} \tag{1}$$

B. Encoding

Among the components of HDC, encoding is the most essential process. It is necessary to encode both training and testing samples into HVs before they can be used. Different encodings have been investigated for various tasks. In HDC models for NLP, N-gram encoding is one of the more popular encoding methods [15], [22], [23]. At the start of the encoding, we first randomly generated 37 orthogonal bipolar-1,1 HVs that would serve as the base HVs(item memory) for all characters. These 37 HVs represent a total of 26 alphabetic characters, 10 numbers, and all other signs(including space). Each character will be assigned to their base HVs. The next step of encoding is to calculate each block of N consecutive letters.

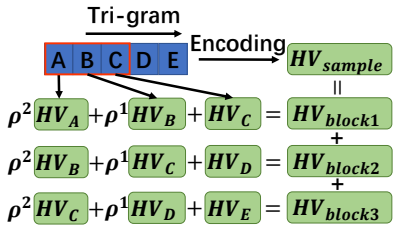

Fig. 1. HDC N-gram Encoding

As an example, consider the case where we compute a tri-gram encoding for the sentence with the character "ABCDE". At the beginning, "ABC","BCD", and "CDE"'s block HV needs to be computed. To calculate the block HV of "ABC", we first select each character's base HV. Next, we perform HV permutation on these base HVs based on their positions in the block. Last, we add them together to generate the HV for the first block. This process can be summarized as $\vec{H_{block}} = \rho^2(\vec{H_A}) * \rho^1(\vec{H_B}) * \vec{H_C}$. By redoing this process for each three characters nearby, we are able to get their block HVs. Last, we add them together to produce the sample HV. The sample HV is the encoded HV for the whole sentence which will be used for training and inference. A summary of this process can be found in Fig. 1.

C. One-pass Training

In the following steps, we encode all training samples into HDC, followed by HDC one-pass training. Every class will be allocated an empty class HV and stored in associative memory. A subsequent step is to perform element-wise additions of the sample HVs of training samples and add them up to the corresponding class HVs. After one-pass training, HDC model is able to get an acceptable level of accuracy.

D. Inference

HDC inference is based on calculating the similarity between class HVs and sample. As a first step, we encode testing sentences into sample HVs using the same encoding mechanism and parameters as before. The next step is to calculate the cosine similarity between each sample HV $\vec{H_s}$ and all the class HVs $\vec{H_{c_i}}$. The highest score r indicates the most similarity between the sample HV and the class HV, and the class HV's corresponding label represents the classification results.

E. Retraining

After a single pass of training, optional retraining can be performed several times to fine-tune the HDC model if needed. The training sample HVs will be used to perform inference first; if the prediction is wrong, these sample HVs $\vec{H_{H_s}}$ will be subtracted from the incorrectly predicted class HV $\vec{H_{c_w}}$ and added to the right class HV $\vec{H_{c_r}}$. By using this method, the HDC model's accuracy can be increased from about 70% to around 90% with fewer than ten epochs.

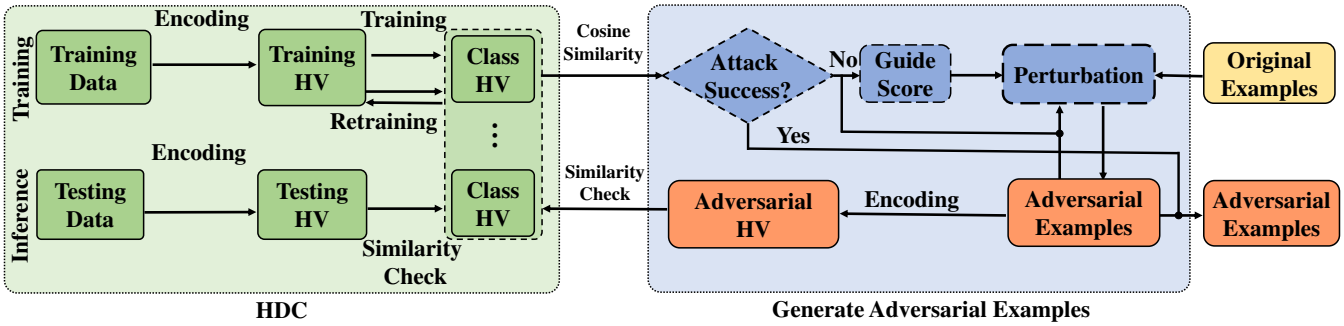

Fig. 2. The overflow of proposed approaches

IV. Adversarial Data Generation

A. Threat Model

To begin with, we will assume that the HDC model is a gray box for attackers. An attacker is not aware of the training dataset, the HDC encoding method, the associative memory value, or any other parameter. An attacker only can access, on the output side, the cosine similarity between the encoded samples (sample HVs) and HDC's class HVs. This setting is widely used in both NLP adversarial samples generation [2] and HDC adversarial attacks [25]. The attacker's goal is to fool HDC in order to make the model falsely classify the samples.

B. Perturbation

The purpose of this paper is to explore word-level replacement as a perturbation, which is the most popular approach to attack NLP neural networks. We aim to replace words in plain sentences with their synonyms. Rather than replacing every word, we first filter all stop words from the NLTK (Natural Language Toolkit [3]) stop words list (e.g., "is", "the" and "or"). In most cases, stop words do not contribute to the semantics of a sentence, however, changing them may result in a break in the original sentence's grammar and a reduction in the readability of the sentence. After removing stop words, we explore the Glove word embedding space [20] to find the nearest neighbors of the remaining non-stop words. Beyond that, we apply the counter-fitting method [18] to post-process the Glove embedding space in order to ensure that the nearest neighbors are synonyms. Once we get the Glove embedding space model post-processed, we select the top N ($N = 10$) closest words to the original word in the embedding space as their synonyms. It is possible that the Glove embedded space dictionary does not cover all of the words we used in our task which means some words cannot find their synonyms. This set of methods of finding synonyms has been extensively studied in the context of generating natural language adversarial samples on neural networks [2]. In contrast to image and signal perturbations, natural language perturbations are difficult to evaluate and nearly impossible to achieve perfection, so we employ mature approaches as our perturbation scheme.

C. Adversarial Generation Algorithm

To automatically generate adversarial samples, we propose two strategies, unguided generation and guided generation. The former strategy is based on a brute-force search without any guidance. The latter approach is a greedy approach guided by the similarity scores we propose. Because our task is a binary classification, we call the plain text prediction result as the original label. If the attack is successful, the flipped label which we call it adversarial label. The whole HDC model and attack overflow is shown in Fig. 2. The details are as follows.

Brute-force Algorithm We first develop a brute-force search method to generate adversarial samples. In order to generate as many adversarial samples as possible, it will attempt to search for every possible word replacement combination for the plain sentence. Specifically, we first generate and test all possible adversarial samples which is generated by only one word substituted on the plain sentence. If any of these adversarial samples can not "fool" the model, we will try every possible two-word substitution. In addition, if too many words are able to be replaced, the number of possible combinations may be enormous. We, therefore, set a time limit for generating each adversarial sample. If it finds an adversarial sample that can fool the model or reaches the process time limit, it will stop and try to generate an adversarial sample based on the next plain sentence. Besides that, if the number of word changes meet our max limit(n), it will give up this try and attempt the next one. This process is explained in Algorithm 1.

Algorithm 1 Brute-force

```
for y = 1, ..., S in x_orig do
  if y is not in stop words list then
    A[y] ← find synonym(y, N)
for i = 1, 2...n do
  for C in combination(i, x_orig) do
    for CC in Cartesian product(i, x_adv) do
      x_adv = perturb(x_orig, A[C][CC])
      Label_{x_adv} = test(x_adv)
      if Label_{x_adv} != Label_{x_orig} then
        return x_adv {Attack success}
  i = i + 1
```

Greedy Algorithm An alternative method for generating adversarial samples is to utilize the output of the model as guidance, which are cosine similarities between sample HV and class HVs in HDC model. We develop a greedy algorithm using cosine similarity and integrated similarity as the guid-

ance scores(G). In general, a higher score indicates that the adversarial samples are closer to our adversarial label class HV hence is more promising to lead to a successful attack. First, the algorithm generates all possible adversarial samples with only one-word replacements. After that, we determine whether any of these adversarial samples attack successful or not. If not, as a result, we will select the sample with the highest score G_{max} as the new sample and repeat this procedure until attack success. However, we only replace each word in the original sentence once. Similar to the brute-force approach, we set limits for the number of words(n) to be replaced, and the maximum time it takes to generate each adversarial sample. The details of the algorithms are provided in the Algorithm 2.

Algorithm 2 Greedy Algorithm

```
x_adv = x_orig
for y = 1, ..., S in x_adv do
  while i¡=n do
    if y is not in stop words list and not replaced then
      A[y] ← find synonym(y, N)
      x_adv = perturb(x_orig, A[y])
      Label x_adv, G = test(x_adv)
      if Label_{x_adv} != Label_{x_orig} then
        return x_adv {Attack success}
      G_max ← max(G)
      if G_max > 0 then
        y_max = A[argmax(G)]
        x_adv = Perturb(x_adv, A[y_max])
        i = i + 1
```

D. Guidance Scores

To make the algorithm perform better, the algorithm needs guidance from the HDC model to better replace words to attack. As mentioned in Section IV-A, we assume the HDC model as a grey box in this paper. The attacker will have access to the cosine similarity between the encoded sample (sample HVs) and the class HVs. Based on these, we propose two guidance scores to guide our greedy algorithm.

Cosine Similarity Current HDC adversarial attacks [4], [16], [25] generally use cosine similarity to guide their decision making. An increase in similarity indicates that the samples are more likely to be classified as belonging to the same class as the adversarial samples. The difference between the adversarial sample and the original sample similarity to the adversarial class HV on the adversarial sample is calculated, if it is negative, the adversarial sample is less similar to the adversarial class, which is undesirable. When it is positive, it indicates a more likely chance of attacking success for the adversarial sample. Due to HDC inference results being obtained from comparing the most similar vectors. It is a very useful and straightforward score.

Integrated Similarity In order to guide the adversarial attack more effectively and comprehensively, we develop an integrated similarity as a guidance score. This score is based on four similarities rather than two. There are four cosine similarities between adversarial sample encoded sample HV and the adversarial class HVs: C_{ar} (cosine similarity between adversarial sample encoded sample HV and the adversarial label class HV), C_{or} (cosine similarity between original sample encoded sample HV and the adversarial label class HV), C_{ow} (cosine similarity between original sample encoded sample HV and the original class HV), and C_{aw} (cosine similarity between adversarial sample encoded sample HV and the original class HV). In order to maximize the similarity between the encoded sample HV and the adversarial class HV, as well as the minimum similarity between the sample HV and the original label class HV, we mixed these four similarities differences. The details of the calculation are explained in the Eq. 2.

$$\mathbf{S} = (C_{ar} - C_{or}) + (C_{ow} - C_{aw}) \tag{2}$$

V. Experimental Results

A. Experiment Setup

We pre-train three HDC models for three text spam detection datasets, **SMS text** [10], **YouTube comments** [1] and **Hotel reviews** [19], using 70% of training data in each dataset. Then for each model, we use the remaining 30% data as the base (validation) data to generate adversarial samples. According to the section above, we use the n-gram encoding to train our HDC models [22], [23]. A counterfitting method [18] was used to post-process the vectors using the GloVe embedding space [20]. In order to minimize the disruption of original sentences, we limited the number of word replacements (n) to 30% of each sample across all our approaches. We test our three unguided and guided approaches including: brute-force search, greedy with cosine similarity, and greedy with integrated similarity. We test the number of adversarial samples that are generated against the generation time. As part of our experiment, we also conducted experiments with various time limits (5s,10s,30s) for generating each adversarial sample in order to better understand the benefits of each method. Table II shows our attack success rate across different datasets and approaches. Figure 3 illustrates the results.

B. Comparisons Between Different Approaches

Attack Success Rate The attack success rate with different approaches across different datasets is shown in Table II. Our first finding is that no matter what approach we use, the attack success rate is relatively low (3%-18%) for the SMS and Youtube datasets. However, we are able to achieve an attack success rate of 89% for the Hotel Review dataset. This is reasonable because it is more challenging to attack the dataset with shorter-length sentences due to the limited word substitutions we can make. As an example, SMS and YouTube samples contain 18 and 20 words on average, respectively. However, the average number of words in the hotel reviews dataset is 170. By having more words in each sample, it is easier to have more options for word substitutions, which will lead to a higher attack success rate. The same phenomenon has been observed in DNN-based NLP applications [2]. It is also possible

TABLE I
EXAMPLES OF ADVERSARIAL SAMPLES FOR DIFFERENT DATASETS. MODIFIED WORDS ARE IN RED AND ORIGINAL WORDS ARE IN GREEN.

SMS Texts
Original Text Prediction = **Ham**. (Cosine Similarity: 0.1225)
I **liked** the new mobile
Adversarial Text Prediction = **Spam**. (Cosine Similarity: 0.2499)
I **enjoyed** the new mobile

YouTube Comments
Original Text Prediction = **Spam**. (Cosine Similarity:0.2623)
Eminem is the king of **rap** Micheal Jackson is the king of pop If you also wanna go hard and wanna be the person of first class fame just check out Authenticviews*com and be famous just within days !! yO
Adversarial Text Prediction = **Ham**. (Cosine Similarity:0.2644)
Eminem is the king of **rapper** Micheal Jackson is the king of pop If you also wanna go hard and wanna be the person of first class fame just check out Authenticviews*com and be famous just within days !! yO

Hotel Review
Original Text Prediction = **Truthful**. (Cosine Similarity:0.6231)
The Sheraton Chicago Hotel and Towers is a **nice** place to stay if you **need** a place to stay on quick notice, but it certainly does not 'exceed expectations' as touted on their website. Their Starpoints system is somewhat complicated and not helpful for the frequent traveler. 'Chic but not fussy' is an overstatement. The room was clean, although the bed covering was wrinkled and the bathroom counter had water lying on it that appeared as though it hadn't been cleaned since the last guest. There was hand lotion and shampoo samples, but no soap sample. The bathroom was short on two towels and it took two calls to housekeeping to get this fixed. 'Cheap' might have been a better adjective. Check out was simple and not much hassle. Overall, the Sheraton Chicago Hotel and Towers is fine in a pinch, but next time, I will research hotels a little better before making a decision and reserving a room.
Adversarial Text Prediction = **Deceptive**. (Cosine Similarity:0.6237)
The Sheraton Chicago Hotel and Towers is a **lovely** place to stay if you **needs** a place to stay on quick notice, but it certainly does not 'exceed expectations' as touted on their website. Their Starpoints system is somewhat complicated and not helpful for the frequent traveler. 'Chic but not fussy' is an overstatement. The room was clean, although the bed covering was wrinkled and the bathroom counter had water lying on it that appeared as though it hadn't been cleaned since the last guest. There was hand lotion and shampoo samples, but no soap sample. The bathroom was short on two towels and it took two calls to housekeeping to get this fixed. 'Cheap' might have been a better adjective. Check out was simple and not much hassle. Overall, the Sheraton Chicago Hotel and Towers is fine in a pinch, but next time, I will research hotels a little better before making a decision and reserving a room.

TABLE II
ATTACK SUCCESS RATE ACROSS ALL APPROACHES AND DATASETS

	Brute-force	Cosine Similarity	Integrated Similarity
SMS	7.27%	3.31%	4.14%
Youtube	18.93%	13.84%	16.38%
Hotel	25.72%	60.51%	89.49%

that the rewording cannot effectively attack HDC in short-length sentence tasks and that character-level replacements or sentence rephrases would be more effective.

Attack Speed Additionally, we have observed that our guided approaches are capable of generating adversarial samples much more quickly than the unguided approach. It can be seen from Figure 3 that the slope of the guided approach is always steeper. As a result, our guide approach is able to generate adversarial samples 2.4X faster than the brute-force approach. In theory, the brute-force approach is able to achieve the highest attack success rate since it will search the entire space, but practically, it is not realistic, especially for long sentence samples. In this regard, we believe it is essential to take into account the speed of the attack.

Different Guidance Score Lastly, the performance of the approach with integrated similarity is better than the cosine similarity one, both in terms of success rate and speed. As shown in the table and figure, our guide approach with integrated similarity generates 30.4% more adversarial samples and 39.4% faster than cosine similarity. This proves that our proposed integrated similarity is able to guide an adversarial attack on HDC more effectively than cosine similarity with nearly no additional overhead.

C. Perturbations of Adversarial Sample

As we mentioned in the introduction, reword as perturbation is clearly perceptible in the NLP task. At the same time, unlike images or voice, the perturbation in the NLP task is challenging to evaluate because there is no widely-used objective metric. Many existing studies require volunteers to perform user studies as part of the evaluation of the perturbations [2], [14], in which their settings are varied, and volunteers' evaluations are subjective. In order to minimize the side effects of the perturbation, we utilize widely used perturbation approaches as we described in the Section IV-B. Even though we cannot conduct some user studies, we still believe this is a reasonable approach. Table I illustrates some adversarial samples we generated across different datasets.

VI. CONCLUSION

Brain-inspired HDC as a novel computing paradigm has shown promising performance in NLP tasks. This paper presents a novel study on automatically generating adversarial attacks for HDC-based NLP applications. We propose similarity score-guided greedy algorithms to automatically generate adversarial samples for text data. Experimental results on three spam detection datasets show that our approach can achieve up to 89% attack success rate and similarity-guided attack achieves a speedup of 2.4X in generating adversarial samples than the brute-force approach. Our future work will focus on leveraging the automatically-generated adversarial samples to

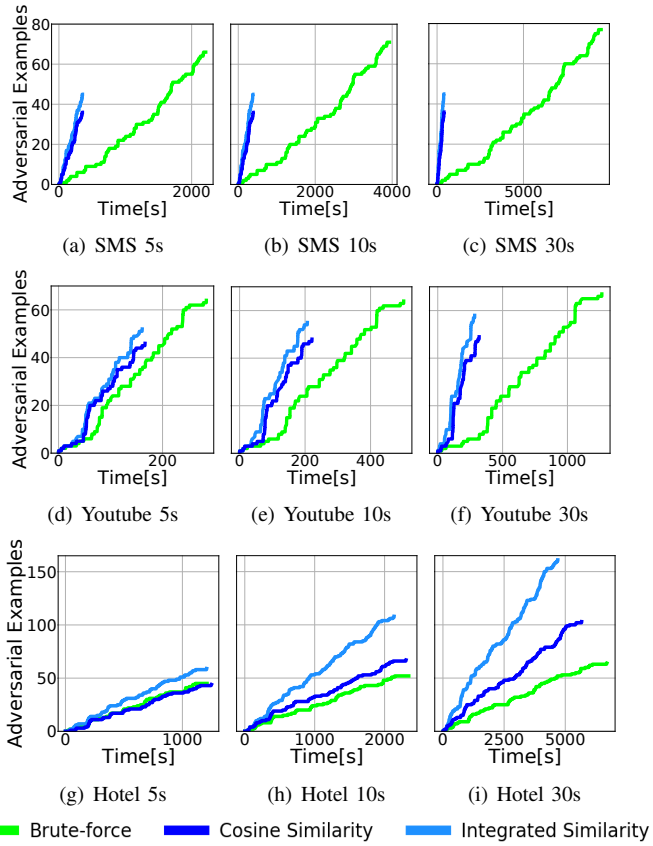

Fig. 3. Number of generated adversarial samples versus time across different approaches, datasets and time limits

enhance the robustness of HDC-based NLP applications to adversarial attacks.

Acknowledgments. This work was partially supported by NSF grant #2202310. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Túlio C Alberto, Johannes V Lochter, and Tiago A Almeida. Tubespam: Comment spam filtering on youtube. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*, pages 138–143. IEEE, 2015.

[2] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[3] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[4] Wencheng Chen and Hongyu Li. Adversarial attacks on voice recognition based on hyper dimensional computing. *Journal of Signal Processing Systems*, 93(7):709–718, 2021.

[5] Michael Crawford, Taghi M Khoshgoftaar, Joseph D Prusa, Aaron N Richter, and Hamzah Al Najada. Survey of review spam detection using machine learning techniques. *Journal of Big Data*, 2(1):1–24, 2015.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.

[8] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE, 2018.

[9] Sarik Ghazarian, Ralph Weischedel, Aram Galstyan, and Nanyun Peng. Predictive engagement: An efficient metric for automatic evaluation of open-domain dialogue systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7789–7796, 2020.

[10] José María Gómez Hidalgo, Tiago A Almeida, and Akebo Yamakami. On the validity of a new sms spam collection. In *2012 11th International Conference on Machine Learning and Applications*, volume 2, pages 240–245. IEEE, 2012.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] Tomoki Ito, Kota Tsubouchi, Hiroki Sakaji, Tatsuo Yamashita, and Kiyoshi Izumi. Word-level contextual sentiment analysis with interpretability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4231–4238, 2020.

[13] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. *arXiv preprint arXiv:1804.06059*, 2018.

[14] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025, 2020.

[15] Fangxin Liu, Haomin Li, and Li Jiang. L3e-hd: A framework enabling efficient ensemble in high-dimensional space for language tasks. In *Proceedings of the International ACM Sigir Conference on Research and Development in Information Retrieval (SIGIR)*, 2022.

[16] Dongning Ma, Jianmin Guo, Yu Jiang, and Xun Jiao. Hdtest: Differential fuzz testing of brain-inspired hyperdimensional computing. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 391–396. IEEE, 2021.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[18] Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*, 2016.

[19] Myle Ott, Claire Cardie, and Jeffrey T Hancock. Negative deceptive opinion spam. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: human language technologies*, pages 497–501, 2013.

[20] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[21] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[22] Abbas Rahimi, Pentti Kanerva, and Jan M Rabaey. A robust and energy-efficient classifier using brain-inspired hyperdimensional computing. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, pages 64–69, 2016.

[23] Rahul Thapa, Bikal Lamichhane, Dongning Ma, and Xun Jiao. Spamhd: Memory-efficient text spam detection using brain-inspired hyperdimensional computing. In *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 84–89. IEEE, 2021.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[25] Fangfang Yang and Shaolei Ren. On the vulnerability of hyperdimensional computing-based classifiers to adversarial attacks. In *International Conference on Network and System Security*, pages 371–387. Springer, 2020.