Encoder-Free Text Classification Using Hyperdimensional Computing

Nikolay Yudin[1]

[1]Independent Researcher, 1@seprotocol.ai

December 22, 2025

Abstract

We investigate the feasibility of text classification using Hyperdimensional Computing (HDC) without neural network encoders. Our method, based on character n-grams hashed to ternary vectors, achieves 94.3% accuracy on language identification (20 classes), 76.8% on topic classification (4 classes), and 70.7% on sentiment analysis (2 classes). We compare against BERT baselines measured on identical test data, confirming that the accuracy gap increases with semantic complexity: 5.1% for pattern-based tasks, 17.3% for keyword-based tasks, and 21.7% for sentiment requiring compositional understanding. Our results demonstrate that encoder-free HDC is well-suited for pattern recognition tasks on resource-constrained edge devices, while identifying the boundaries of the approach for semantically complex tasks. The method requires no training in the traditional sense, uses only integer operations, and can run on microcontrollers with less than 4KB of memory.

Keywords: Hyperdimensional Computing, Text Classification, Edge AI, Encoder-Free, Ternary Vectors

1 Introduction

Modern natural language processing relies heavily on deep neural networks, particularly transformer-based models like BERT [2]. While these models achieve state-of-the-art accuracy, they require significant computational resources: billions of parameters, gigabytes of memory, and GPU acceleration for practical inference times.

This creates a barrier for deploying NLP capabilities on edge devices—smartphones, IoT sensors, embedded systems—where memory, power, and compute are severely constrained. A smart doorbell cannot run BERT locally; it must send data to the cloud, introducing latency, privacy concerns, and dependency on connectivity.

Hyperdimensional Computing (HDC) offers an alternative paradigm [1]. HDC represents information as high-dimensional vectors (typically 1,000–10,000 dimensions) where:

- Similarity corresponds to semantic relatedness
- Simple operations (addition, element-wise multiplication) enable reasoning
- Vectors can be quantized to binary or ternary with minimal information loss
- All operations are embarrassingly parallel and require no multiplication

Previous work has demonstrated HDC for various classification tasks, but typically uses neural network encoders to generate initial embeddings [3, 4]. This defeats the purpose for edge deployment—the encoder becomes the bottleneck.

In this paper, we investigate **encoder-free HDC** for text classification. We ask: *How much accuracy must we sacrifice to eliminate the neural encoder entirely?*

Our contributions:

1. We implement and evaluate HyperEmbed, an encoder-free text classifier using character n-grams hashed to ternary HDC vectors
2. We measure BERT baselines on identical test data for fair comparison (not literature values)
3. We characterize when encoder-free HDC works well (pattern-based tasks) and when it struggles (semantic tasks)
4. We provide reproducible experiments with statistical validation (5 runs per configuration)

2 Related Work

2.1 Hyperdimensional Computing for Classification

Kanerva introduced hyperdimensional computing as a computational framework inspired by neural activity patterns [1]. Rahimi et al. demonstrated practical HDC classifiers for EMG signals, achieving accuracy comparable to SVMs with 100× energy savings [3].

For text, Najafabadi et al. showed that character n-gram HDC achieves 90%+ accuracy on language identification [5]. Kleyko et al. proposed HyperEmbed, using character n-grams with learned projection matrices [6]. Our work differs by using purely hash-based projections with no learned parameters.

2.2 Efficient NLP

Knowledge distillation compresses large models into smaller ones [9]. Quantization reduces precision from 32-bit to 8-bit or lower [10]. However, even distilled and quantized transformers require megabytes of parameters and matrix multiplications.

Our approach is orthogonal: we eliminate the neural architecture entirely, replacing learned representations with deterministic hashing.

2.3 Edge AI for Text

TinyBERT [11] achieves 96% of BERT accuracy with 7.5× smaller model. MobileBERT [12] targets mobile deployment. These remain too large for microcontrollers (ESP32 has 520KB RAM).

HDC with ternary vectors requires approximately 1KB per class prototype (4096 dimensions × 2 bits), enabling deployment on the smallest devices.

3 Method

3.1 Overview

Our classifier, HyperEmbed, operates in three phases:

1. **Encoding**: Convert text to HDC vector via character n-grams

2. **Training**: Bundle all vectors of each class into a prototype
3. **Inference**: Find nearest prototype by cosine similarity

No backpropagation, no gradient descent, no learned parameters.

3.2 Text to HDC Vector

Given text t, we:

1. Convert to lowercase
2. Extract all character n-grams: $G = \{t[i : i + n] \mid i = 0, \ldots, |t| - n\}$
3. Hash each n-gram to a ternary vector
4. Bundle (sum) all n-gram vectors
5. Normalize to unit length

3.3 Hash to Ternary Vector

For n-gram g, we compute:

$$\text{seed} = \text{MD5}(g) \mod 2^{32} \tag{1}$$

Using this seed, we generate a pseudorandom ternary vector:

$$v_i \in \{-1, 0, +1\} \quad \text{with equal probability} \tag{2}$$

This is deterministic: the same n-gram always produces the same vector. No storage of a codebook is required—vectors are generated on demand.

Algorithm 1 HyperEmbed Encoding

```
function TextToVector(t, n, d)
    t ← lowercase(t)
    v ← 0_d
    for i = 0 to |t| − n do
        g ← t[i : i + n]
        v ← v + HashToTernary(g, d)
    end for
    return v/‖v‖
end function
```

3.4 Training

For each class c with texts T_c:

$$\text{prototype}_c = \frac{\sum_{t \in T_c} \text{TextToVector}(t)}{\left\| \sum_{t \in T_c} \text{TextToVector}(t) \right\|} \tag{3}$$

This is a single pass through the data—no iterations, no convergence criteria.

3.5 Inference

For input text t:

$$\hat{y} = \arg\max_c \cos(\text{TextToVector}(t), \text{prototype}_c) \tag{4}$$

3.6 Complexity Analysis

- **Memory**: $O(C \cdot d)$ for C classes, dimension d. With $d = 4096$ and ternary encoding (2 bits), each prototype is 1KB.
- **Training time**: $O(N \cdot L)$ for N texts of average length L. Single pass, no iteration.
- **Inference time**: $O(L + C \cdot d)$ per text. No matrix multiplication.

4 Experimental Setup

4.1 Datasets

We evaluate on three text classification tasks of increasing semantic complexity:

1. **Language Identification** (20 classes): Distinguishing languages based on character patterns. Dataset: papluca/language-identification from HuggingFace.
2. **Topic Classification** (4 classes): Categorizing news articles into World, Sports, Business, Sci/Tech. Dataset: AG News [7].
3. **Sentiment Analysis** (2 classes): Classifying movie reviews as positive or negative. Dataset: SST-2 from GLUE benchmark [8].

4.2 Baselines

We evaluate pre-trained BERT models on the **same test data** as HDC:

- Language ID: `papluca/xlm-roberta-base-language-detection`
- AG News: `fabriceyhc/bert-base-uncased-ag_news`
- SST-2: `textattack/bert-base-uncased-SST-2`

This ensures fair comparison under identical conditions, rather than relying on literature values which may use different test splits.

4.3 Configuration

- HDC dimension: $d = 4096$
- N-gram sizes tested: $n \in \{3, 4, 5, 6\}$
- Training samples: up to 20,000 (subsampled with random seed)
- Test samples: up to 5,000
- Validation: 5 independent runs with different random seeds

4.4 Metrics

We report accuracy with mean ± standard deviation over 5 runs, and 95% confidence intervals. Statistical significance is assessed using independent t-tests.

5 Results

5.1 Main Results

Table 1 presents the main results comparing HDC to BERT baselines.

Table 1: Encoder-Free HDC vs BERT Baselines

Task	HDC	BERT	Gap
Language ID (20 cl.)	94.3% ± 0.2%	99.4%	−5.1%
Topic Class. (4 cl.)	76.8% ± 0.2%	94.1%	−17.3%
Sentiment (2 cl.)	70.7% ± 0.9%	92.4%	−21.7%

HDC: HyperEmbed with optimal n-gram, dim=4096. BERT baselines measured on same test data. HDC results: mean ± std over 5 runs.

Key observations:

1. **Language ID**: HDC achieves 94.3%, only 5.1% below BERT. Character-level patterns are highly discriminative for language identification.
2. **Topic Classification**: Gap increases to 17.3%. While keywords help ("stock market" → Business), topic classification requires some semantic understanding.
3. **Sentiment**: Largest gap of 21.7%. Sentiment often depends on subtle cues, negation ("not bad" → positive), and compositional meaning that bag-of-ngrams cannot capture.

5.2 BERT Baseline Validation

Our measured BERT accuracies closely match literature values:

Table 2: BERT Baseline Validation

Task	Measured	Literature	Diff
Language ID	99.4%	∼99%	+0.4%
Topic Class.	94.1%	∼95%	−0.9%
Sentiment	92.4%	∼94%	−1.6%

Minor differences are expected due to different test subsets and model versions. The close agreement validates our experimental setup.

5.3 Effect of N-gram Size

Table 3 shows accuracy for different n-gram sizes.

Interesting pattern:

- **Language ID**: Shorter n-grams work best ($n = 3$). Languages differ in basic character patterns ("the", "le", "der").
- **Topic/Sentiment**: Longer n-grams help ($n = 5$–6). More context needed to capture meaning.

This suggests that optimal n-gram size depends on the nature of the discrimination task.

Table 3: Effect of N-gram Size on Accuracy (%)

Task	$n = 3$	$n = 4$	$n = 5$	$n = 6$
Language ID	**94.3**	89.4	81.7	70.8
Topic Class.	70.3	75.1	76.1	**76.8**
Sentiment	65.6	70.3	**70.7**	68.9

Bold: optimal n-gram for each task.

5.4 Statistical Significance

We performed pairwise t-tests to confirm that accuracy differences between tasks are statistically significant:

- Language ID vs AG News: $t = 147.3$, $p < 0.001$
- AG News vs SST-2: $t = 14.8$, $p < 0.001$
- Language ID vs SST-2: $t = 61.4$, $p < 0.001$

Effect sizes (Cohen's d) exceed 8.0 for all comparisons, indicating very large effects. The accuracy ordering Language ID > Topic > Sentiment is robust.

5.5 Per-Class Analysis

Confusion matrices reveal class-specific performance:

Language ID: Near-diagonal matrix with most confusion between related languages (e.g., Spanish/Portuguese, Dutch/German).

AG News: Sports achieves highest accuracy (90%), likely due to distinctive vocabulary. Business shows most confusion with Sci/Tech.

SST-2: Negative sentiment (79%) is easier to detect than positive (62%). Negative reviews contain explicit markers ("terrible", "boring"), while positive sentiment is often expressed implicitly.

5.6 Computational Efficiency

Table 4: Computational Requirements

Metric	**HDC**	**BERT**
Parameters	0	110M
Memory (model)	4KB/class	440MB
Operations	Add, compare	MatMul
GPU required	No	Yes (practical)
Training	Single pass	Hours/days

HDC requires approximately 100,000× less memory than BERT, with no learned parameters and no GPU requirement.

6 Discussion

6.1 When Does Encoder-Free HDC Work?

Our results suggest a clear pattern:

1. **Pattern-based tasks**: Excellent performance. Language identification relies on character frequency patterns that n-grams capture directly.

2. **Keyword-based tasks**: Good performance. Topic classification benefits from distinctive vocabulary, though some semantic understanding helps.

3. **Semantic tasks**: Significant gap. Sentiment analysis requires understanding composition ("not bad" = positive), negation, sarcasm, and implicit meaning.

6.2 Implications for Edge Deployment

Many practical edge AI tasks are pattern-based:

- Anomaly detection in sensor data
- Activity recognition from accelerometer patterns
- Wake word detection in audio
- Language/locale detection

For these applications, encoder-free HDC offers a viable path to on-device intelligence with minimal resources.

6.3 Bridging the Gap

Several approaches could improve HDC accuracy on semantic tasks:

1. **Negation handling**: Special encoding for "not", "never", "no"

2. **Word-level HDC**: Using word tokens instead of character n-grams

3. **Positional encoding**: Incorporating word position information

4. **Mesh voting**: Combining predictions from multiple devices

We leave these extensions for future work.

6.4 Limitations

- We evaluated on English datasets only; performance may vary for other languages
- The test set sizes (up to 5,000) are smaller than standard benchmarks
- We used pre-trained BERT models, not fine-tuned versions which may achieve higher accuracy

7 Conclusion

We demonstrated that encoder-free Hyperdimensional Computing can achieve competitive text classification accuracy on pattern-based tasks while requiring orders of magnitude less computational resources than neural approaches.

Our key findings:

1. HDC achieves 94.3% on language identification (5.1% below BERT)

2. Accuracy gap increases with semantic complexity: 17.3% for topic classification, 21.7% for sentiment

3. Optimal n-gram size depends on task: shorter for pattern matching, longer for semantic tasks

4. The method requires no training, no GPU, and less than 4KB memory per class

Encoder-free HDC is well-suited for edge devices performing pattern recognition tasks. For semantically complex tasks, hybrid approaches or collective inference through device meshes may bridge the gap.

Code and data are available at: https://github.com/nick-yudin/SEP

Acknowledgments

This work is part of the Semantic Event Protocol (SEP) project, developing distributed AI systems for edge devices. Website: https://seprotocol.ai

References

[1] 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.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.

[3] A. Rahimi, P. Kanerva, and J. M. Rabaey, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing," in *Proc. ISLPED*, 2016, pp. 64–69.

[4] M. Imani, D. Kong, A. Rahimi, and T. Rosing, "VoiceHD: Hyperdimensional computing for efficient speech recognition," in *Proc. IEEE ICRC*, 2017, pp. 1–8.

[5] M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, no. 1, pp. 1–21, 2015.

[6] D. Kleyko, E. Osipov, and R. W. Gayler, "Holographic graph neuron: A bioinspired architecture for pattern processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 6, pp. 1250–1262, 2017.

[7] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. NeurIPS*, 2015, pp. 649–657.

[8] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proc. EMNLP*, 2018.

[9] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," in *Proc. EMC2 Workshop at NeurIPS*, 2019.

[10] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, "Q8BERT: Quantized 8bit BERT," in *Proc. EMC2 Workshop at NeurIPS*, 2019.

[11] X. Jiao et al., "TinyBERT: Distilling BERT for natural language understanding," in *Proc. EMNLP Findings*, 2020, pp. 4163–4174.

[12] Z. Sun et al., "MobileBERT: a compact task-agnostic BERT for resource-limited devices," in *Proc. ACL*, 2020, pp. 2158–2170.