

# SpinHash: An $O(1)$ Encoder for Any Vocabulary

Jia Xu  
Rutgers University

Jinhua Wang, Constantin F. Aliferis  
University of Minnesota

## Abstract

The embedding table is a quiet ceiling of nearly every neural network model. Its size grows with the vocabulary, its rows are tied to a single modality, and unseen inputs require adding parameters. This makes the table prohibitively large at scale, impossible to share across modalities, and unable to represent new tokens without retraining.

We remove this ceiling with SPINHASH, an  $O(1)$  encoder in the vocabulary size that stays constant at about 0.2M parameters regardless of how many tokens or modalities it serves. Instead of storing a vocabulary dependent table, SPINHASH hashes each token into a compact *spin* code and decodes it with one shared feedforward network. Our hash is not merely a compression mechanism, but also defines the geometry of the representation space, realized by a similarity preserving Spin-SimHash and an antiferromagnetic Ising Spin-Repulse.

Across text, DNA, and clinical codes, SPINHASH matches embedding tables with 17 to  $283\times$  fewer parameters, reaching parity on IMDB, AG-News, Yelp, and both clinical fusion tasks. For  $k=22$  DNA, where  $4^{22}$  table entries are infeasible, SPINHASH achieves 98.3% promoter accuracy, outperforming same-size HyenaDNA-tiny and the  $1000\times$  larger Nucleotide Transformer.

This work shows that representations need not be tied to a vocabulary, and that we can compute them once and reuse them across vocabularies, modalities, and domains.

## 1 Introduction

Neural networks operate on continuous vectors, but many of the objects they model are discrete, such as words, DNA  $k$ -mers, clinical codes, molecular strings, and program tokens. Nearly all such models bridge this gap with an *embedding table*, a learned matrix  $E \in \mathbb{R}^{|V|\times d}$  that assigns every symbol in a vocabulary  $V$  its own trainable vector in  $d$  dimensions. This simple interface creates a fundamental scaling problem. *The cost of*

*representing a token grows with the vocabulary, not with the model’s capacity to process it.*

Three limitations follow. First, the memory cost is  $O(|V|d)$ , making large vocabularies expensive and, at extreme cardinalities, impossible. For example, DNA windows of length 22 yield  $4^{22}$  possible  $k$ -mers, and a conventional embedding table at ordinary width would require tens of petabytes, ruling out direct modeling of this vocabulary (Nguyen et al., 2023). Second, tables are vocabulary specific. A model combining  $K$  modalities requires  $K$  tables, and representations learned for one vocabulary cannot be reused for another. Third, tables are closed. An unseen token has no row and therefore collapses to a shared unknown-token representation. Supporting a growing vocabulary requires expanding the table and retraining.

These limitations are increasingly consequential as models scale to larger vocabularies and span multiple modalities. Multilingual systems already operate over hundreds of thousands of symbols (Conneau et al., 2020). Clinical models may jointly encode millions of codes across multiple ontologies (Su et al., 2025; Wornow et al., 2023). Biological models face combinatorial vocabularies for  $k$ -mers. In all three settings, the embedding table becomes a bottleneck before the neural network does.

Several approaches remove the explicit table, but do so by changing what is represented rather than eliminating the underlying tradeoff. Byte and character level models such as ByT5 (Xue et al., 2022) and CANINE (Clark et al., 2022) represent a token as a sequence of primitive symbols, avoiding vocabulary dependence and unknown tokens at the cost of much longer sequences and hence substantially higher attention cost. EmByte (Li et al., 2025) keeps tokens intact and decodes their byte representations with a shared network, while hashing methods such as T-Free (Deiseroth et al., 2024) and HashFormers (Xue and Aletras, 2022) map tokens to a fixed set of learned buckets. These

methods share a key limitation. *The token to code geometry is fixed by the token spelling or by an essentially arbitrary hash.*

We instead ask whether token identity can be represented by a *designed discrete code* and decoded by a single vocabulary independent function. SPINHASH realizes this. A fixed hash maps every token, regardless of vocabulary, to a collision-free spin code  $\mathbf{c}_t \in \{\uparrow, \downarrow\}^J$ , and a single shared feedforward network maps that code to its continuous embedding. The encoder therefore has constant size, about 0.2M parameters, independent of both  $|V|$  and the number of vocabularies being fused. New tokens require neither new parameters nor an unknown-token fallback.

The key idea is that the hash itself should be a *design axis*. We study three assignments. Spin-Random is a structureless random assignment, Spin-SimHash preserves token similarity by placing similar tokens at small Hamming distance, and Spin-Repulse uses a repulsive antiferromagnetic Ising objective to separate context-confusable tokens. Unlike conventional hashing, the discrete substrate can be deliberately shaped to preserve similarity or enforce separation. This turns the token to code map from a fixed preprocessing choice into a controllable inductive bias. In effect, whether two tokens are treated as similar becomes a deliberate choice rather than a byproduct of their spelling or an arbitrary hash.

We evaluate SPINHASH across three regimes that expose different weaknesses of embedding tables. On text classification, where conventional tables are already effective, SPINHASH matches table-based models on IMDB, AG-News, and Yelp while using 17 to  $283\times$  fewer embedding parameters. On DNA, where a  $k=22$  table is infeasible, SPINHASH achieves 98.3% accuracy on human-promoter classification, exceeding both a same size HyenaDNA-tiny (Nguyen et al., 2023) and the  $1000\times$  larger Nucleotide Transformer v2-500M (Dalla-Torre et al., 2025). On clinical records, SPINHASH fuses multiple high-cardinality code systems with constant embedding cost and reaches parity with conventional tables on both datasets, namely ClinVar and PharmGKB. Across these settings, the same encoder replaces vocabulary-specific tables without sacrificing token-level expressivity.

Our main contributions are as follows.

1. **Vocabulary-independent spin encoding.** A fixed hash and one shared  $\sim 0.2\text{M}$ -parameter decoder replace the  $|V| \times d$  table, so embedding cost is constant in vocabulary size

and modality count.

2. **Controllable code geometry.** We make the token-to-code hash a design axis, with random, similarity-preserving, and repulsive Ising assignments giving explicit control over which tokens are treated as similar.
3. **Cross-domain performance at constant cost.** One encoder matches or exceeds table-based models across text, DNA, and clinical codes while using orders of magnitude fewer embedding parameters.

## 2 Related Work

### Tokenizer-free and byte-level models.

Tokenizer-free models (Clark et al., 2022; Xue et al., 2022; Tay et al., 2022; Yu et al., 2023; Pagnoni et al., 2025) replace fixed vocabularies with bytes or characters, eliminating out-of-vocabulary tokens but increasing sequence length. Their strongest variants also rely on large pretrained backbones. For example, CANINE has 132M parameters (Clark et al., 2022), so they are not matched-budget baselines for a small from-scratch encoder. EmByte (Li et al., 2025) is closest to SPINHASH: it preserves tokens as single units and decodes their byte representations with a shared feedforward network. SPINHASH instead uses a compact binary spin code, makes token geometry controllable through Spin-SimHash and Spin-Repulse, and extends the same encoder across multiple symbolic modalities.

### Embedding compression and hashing.

Embedding compression reduces  $|V| \times d$  tables through factorization (Lan et al., 2020), hashing and Bloom embeddings (Svenstrup et al., 2017), or quantization (Jacob et al., 2018). These approaches reduce memory or precision but remain dependent on  $V$ , so their cost still grows with vocabulary size and modality count. SPINHASH removes this dependence. Furthermore, it is complementary to these methods and can additionally quantize the downstream model.

### Multimodal representation learning.

Multimodal systems such as CLIP (Radford et al., 2021), Flamingo (Alayrac et al., 2022), ImageBind (Girdhar et al., 2023), Unified-IO (Lu et al., 2023), and Gato (Reed et al., 2022), as well as multimodal clinical models (Chen et al., 2022; Schulz et al., 2021), generally align modality-specific encoders above the input representation layer. Thus, adding a high-cardinality symbolic modality introduces new

vocabulary-dependent parameters. SPINHASH instead shares the input encoder itself, so one encoder maps tokens from all modalities into a common representation space, enabling scalable fusion at constant embedding cost.

### 3 Method

#### 3.1 Background

A multimodal task over  $K$  modalities predicts a single label or scalar  $\hat{y} = \mathbb{F}(\{x_i^{(1)}, \dots, x_i^{(K)}\}_{i=1}^I)$ , where  $\mathbb{F}$  is the model and  $x_i^{(k)}$  is the token at position  $i \in \{1, \dots, I\}$  of modality  $k$ , such as text, gene symbol, or structured code. The standard recipe assigns each modality its own embedding table  $E^{(k)} \in \mathbb{R}^{|V_k| \times d}$ , where  $V_k$  is the vocabulary and  $d$  is the embedding dimension, and then fuses the resulting embeddings. The aggregate cost is therefore  $\sum_k |V_k|d$ , which grows with *both* the vocabulary size and the number of modalities  $K$ .

This embedding cost is the bottleneck. A single language vocabulary is already large. BERT (Devlin et al., 2019) uses 30,522 tokens, GPT-2 (Radford et al., 2019) uses 50,257, and XLM-R uses 250,002 across many languages (Conneau et al., 2020). Thus, even one embedding table contains  $\approx 192\text{M}$  parameters at  $d=768$ . Stacking a separate table for each symbol system, such as ICD, SNOMED, RxNorm, drug names, and free text, reaches gigabyte scale and can exceed the memory of one accelerator before training even begins.

For example, for  $k$ -mer vocabularies, the problem is not merely expensive but impossible. Their vocabulary size is  $4^k$ , so  $k=22$  requires  $4^{22} \approx 1.8 \times 10^{13}$  rows, far too many to store. Moreover, the human genome contains only  $\sim 3.2 \times 10^9$  bp, meaning that over 99.9% of these  $k$ -mers are never observed. Consequently, the table cannot even be meaningfully filled.

#### 3.2 Architecture Overview

SPINHASH removes this dependence. It replaces all  $K$  tables with *one* shared encoder applied to a vocabulary-independent spin code, so cost is constant in both vocabulary size and modality count. The only remaining choice is how to assign each token its code, the hash, which we make a first-class design axis. SPINHASH runs in two stages:

- **Hashing**, §3.3. Map each token to a spin code of length  $J$ ,  $\mathbf{c}(t)$ , via a hash  $\mathcal{H}$ . We study three choices, Spin-Random, Spin-SimHash, and our Spin-Repulse.

- **Multimodal encoder**, §3.4. Turn the spin codes of the concatenated modalities into one vector through *spin embedding* in Eq. 6, *aggregation* in Eq. 7, and *compression* in Eq. 8. A linear *predictor* then maps that vector to the class label or scalar.

#### 3.3 Hashing

SPINHASH is vocabulary-independent, at  $\mathcal{O}(1)$  in the vocabulary size  $|V|$  and the modality count  $K$ . This leaves a single question, *how to assign each token its spin code?*

Formally, let  $x$  be a token, a subword from any vocabulary with  $x \in V$ . Following EmByte (Li et al., 2025), we identify  $x$  with its UTF-8 byte vector  $t = b(x) \in \{0, \dots, 255\}^{L(x)}$ , where  $L(x) \geq 1$  is the number of bytes. A hash  $\mathcal{H} : \bigcup_{L \geq 1} \{0, \dots, 255\}^L \rightarrow \{\uparrow, \downarrow\}^J$  maps the byte vector to a *spin code* of length  $J$ , with  $J=64$ . Each spin  $c_j(t) = [\mathcal{H}(t)]_j \in \{\uparrow, \downarrow\}$ , which we identify with  $+1$  and  $-1$  respectively, with spin index  $j \in \{1, \dots, J\}$ , is set by one of the variants below in Eqs. 2–5, and the code collects the  $J$  spins:

$$\begin{aligned} \mathbf{c}(t) &= (c_1(t), \dots, c_J(t)) \\ &= \mathcal{H}(t) \in \{\uparrow, \downarrow\}^J. \end{aligned} \quad (1)$$

The hash  $\mathcal{H}$  takes the byte vector  $t$ . The hyperplanes  $\mathbf{r}_j$  and couplings  $S$  introduced below are parameters of  $\mathcal{H}$ , fixed once before encoding, not arguments. In the Ising energy of Eq. 5 we read  $\uparrow$  as  $+1$  and  $\downarrow$  as  $-1$ , so  $s_j(t) \in \{-1, +1\}$ .

In this way, SPINHASH turns this assignment into a first-class, controllable design axis, shown in Fig. 1, and is **hash-agnostic**. It maps each token to a spin code through a chosen hash, then learns it with one shared neural network encoder in §3.4 that accepts whatever code  $\mathcal{H}$  produces. Different hashes trade off code similarity against separation, and we show this choice matters as much as the encoder itself. We consider three ways to define the hash  $\mathcal{H}$ .

**Spin-Random.** Uniform sampling assigns

$$\begin{aligned} c_j(t) &\sim \mathcal{U}\{\uparrow, \downarrow\}, \\ &\text{i.i.d. over tokens } t \text{ and spins } j, \end{aligned} \quad (2)$$

with resampling on collision. Two distinct tokens collide only with probability  $2^{-J}$ , about  $5.4 \times 10^{-20}$  at  $J=64$ , giving maximal expected separation but high variance.

**Spin-SimHash.** Our most stable member adapts the SimHash LSH of Charikar (2002) to the spin substrate. With a feature map  $\psi(t) \in \mathbb{R}^F$ , a

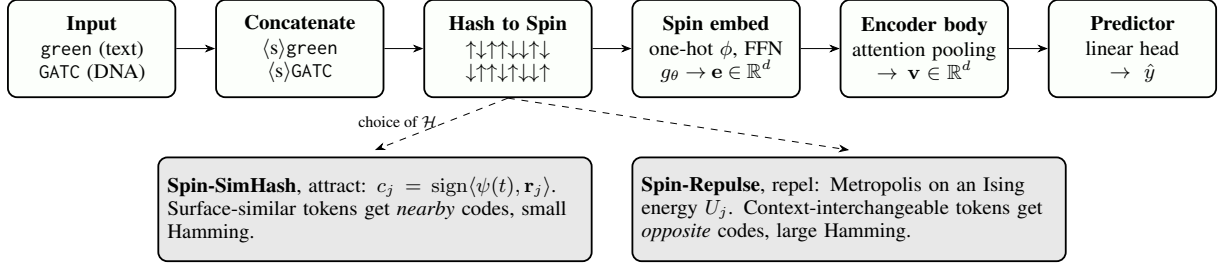


Figure 1: **SPINHASH pipeline with a worked example.** A text token green and a DNA token GATC are concatenated, and a hash  $\mathcal{H}$  maps each token to a spin code of length  $J$ , shown at  $J=8$  and  $64$  in practice. Each spin is one-hot lifted by  $\phi$  and decoded by *one shared* FFN  $g_\theta$  into a spin embedding  $\mathbf{e}$ . The encoder body then pools the token embeddings into a single vector  $\mathbf{v}$  by attention pooling, and a linear predictor maps  $\mathbf{v}$  to the output  $\hat{y}$ . The two stages are hashing and the multimodal encoder, and only  $g_\theta$  with a small predictor head is learned, so cost is constant in vocabulary size and modality count. SPINHASH is hash-agnostic, shown at the bottom: **Spin-SimHash** pulls surface-similar tokens close, while our **Spin-Repulse** pushes context-interchangeable tokens apart, and the choice of  $\mathcal{H}$  sets the code geometry.

vector of byte  $n$ -gram counts, and  $J$  fixed random hyperplanes  $\mathbf{r}_1, \dots, \mathbf{r}_J \in \mathbb{R}^F$  drawn i.i.d. as  $\mathbf{r}_j \sim \mathcal{N}(\mathbf{0}, I_F)$  with  $I_F$  the  $F \times F$  identity matrix, and writing  $\langle \cdot, \cdot \rangle$  for the inner product and  $\text{sign}(u) = +1$  for  $u \geq 0$  and  $-1$  otherwise, we set

$$c_j(t) = \text{sign}(\langle \psi(t), \mathbf{r}_j \rangle), \quad (3)$$

so  $\Pr[c_j(t) \neq c_j(t')] = \alpha(t, t')/\pi$ , where, with  $\|\cdot\|_2$  the Euclidean norm and  $\arccos$  the inverse cosine,  $\alpha(t, t') = \arccos \frac{\langle \psi(t), \psi(t') \rangle}{\|\psi(t)\|_2 \|\psi(t')\|_2} \in [0, \pi]$  is the angle between  $\psi(t)$  and  $\psi(t')$ , whose cosine is their cosine similarity (Charikar, 2002), so surface-similar tokens at small angle get nearby codes. Here  $\psi(t)$  is a 2048-dimensional byte-trigram count vector, where each 3-byte substring of the token, padded with two spaces per side, is hashed into one of  $F=2048$  buckets and counted, a feature hashing (Weinberger et al., 2009). The  $\mathbf{r}_j$  are fixed once before encoding, with  $J=64$  hyperplanes in total. Thus  $c_j(t) = \uparrow$  when the projection is nonnegative and  $\downarrow$  otherwise.

**Spin-Repulse.** We push *context-interchangeable* tokens apart. Tokens that are substitutable in the same context, such as *red*, *green*, and *yellow* after *traffic*, are the ones a model must not confuse. If their codes are made *similar*, it has to recover a fine distinction from near-identical inputs, a noisy channel, since renaming *green* to *reddy* after *traffic* would garble it. Such confusable tokens therefore want *distant* codes at large Hamming, whereas tokens that never share a context are never confused and may keep *nearby* codes. The code should spend its Hamming distance only where tokens must be told apart. We cast this as an *antiferromagnetic Ising* model run independently for each spin index  $j \in \{1, \dots, J\}$ , where the tokens’ spins at index  $j$  are coupled by a context

similarity  $S(t, t') \in \mathbb{R}$ , the signed pointwise mutual information  $\text{PMI}(t, t') = \log \frac{P(t, t')}{P(t)P(t')}$  of tokens  $t, t'$ , where  $P$  denotes co-occurrence probability. We define an antiferromagnetic Ising energy for spin index  $j$ ,

$$U_j = \sum_{t < t'} S(t, t') s_j(t) s_j(t') + \lambda \left( \sum_t s_j(t) \right)^2, \quad (4)$$

where  $s_j(t) \in \{-1, +1\}$  is the  $\pm 1$  reading of  $c_j(t)$  with  $\uparrow = +1$  and  $\downarrow = -1$ , and  $\lambda > 0$  keeps the two spin states balanced. The spin- $j$  assignment of the *whole* vocabulary, the column  $\{c_j(t)\}_{t \in V}$  that collects spin  $j$  across all tokens, is then drawn jointly, not one token at a time, from the Boltzmann distribution

$$\{c_j(t)\}_{t \in V} \sim \frac{1}{Z_j} e^{-\beta U_j}, \quad Z_j = \sum_{\{c_j(t)\}} e^{-\beta U_j}, \quad (5)$$

where  $Z_j$  is the partition function and  $\beta \geq 0$  is the inverse temperature. The coupling repels context-similar pairs while leaving unrelated pairs free to collide. The first term gives context-similar tokens, those with large similarity  $S(t, t')$ , *opposite* spins and the second keeps the code balanced.  $\beta=0$  recovers Spin-Random, finite  $\beta$  favors separation but never forces it. We sample Spin-Repulse by an *asynchronous* finite-temperature Metropolis MCMC (Metropolis et al., 1953) over Eq. 5, which avoids the consensus oscillation that collapses synchronous updates to a single code.

### 3.4 Multimodal Encoder

Once  $c(t)$  is assigned using Equation 1 with any of the Hash function in Equation 2, 3, and 5, we process it with our multimodal encoder including embedding, aggregation, and compression steps.

---

**Algorithm 1** SPINHASH forward pass

---

**Require:** token sequence  $(x_1, \dots, x_I)$ ,  $x_i \in V$ , with bytes  $t_i = b(x_i)$ ; hash  $\mathcal{H}$ ; one-hot lift  $\phi$ ; shared encoder  $g_\theta$ ; query  $\mathbf{q}$ ; head  $(W_{\text{cls}}, b_{\text{cls}})$  or  $(W_{\text{reg}}, b_{\text{reg}})$

- 1: **for**  $i = 1$  to  $I$  **do**
- 2:   bytes  $t_i \rightarrow$  spin code  $\mathbf{c}(t_i)$  [Eq. 1]
- 3:   spin code  $\mathbf{c}(t_i) \rightarrow$  embedding  $\mathbf{e}(t_i)$  [Eq. 6]
- 4: **end for**
- 5: embeddings  $\mathbf{e}(t_1), \dots, \mathbf{e}(t_I) \rightarrow \mathbf{H}(t_1, \dots, t_I)$  [Eq. 7]
- 6:  $\mathbf{H} \rightarrow \mathbf{v}$  [Eq. 8]
- 7:  $\mathbf{v} \rightarrow$  output  $\hat{y}$  [Eq. 9]
- 8: **return**  $\hat{y}$

---

**Spin embedding.** Given the spin code  $\mathbf{c}(t)$  from §3.3, a frozen one-hot lift  $\phi : \{\uparrow, \downarrow\} \rightarrow \mathbb{R}^2$  with  $\phi(\uparrow) = (1, 0)$  and  $\phi(\downarrow) = (0, 1)$ , which one-hot encodes each spin for the network input and is separate from the  $\pm 1$  Ising reading in Eq. 4, and a trainable shared feed-forward network  $g_\theta$ , the only learned part, decode it into the embedding:

$$\mathbf{e}(t) = g_\theta \left( [\phi(c_1(t)) \parallel \dots \parallel \phi(c_J(t))] \right) \in \mathbb{R}^d. \quad (6)$$

Here  $\parallel$  denotes vector concatenation, so the  $J$  one-hot spins form a  $2J$ -dimensional vector, and  $g_\theta : \mathbb{R}^{2J} \rightarrow \mathbb{R}^d$  is a two-layer FFN with one hidden layer, the only learned part of the embedding. Algorithm 1 summarizes the full forward pass.

Note that every token is embedded by the same shared encoder, regardless of its modality. The tokens are simply concatenated into a single token stream and processed by this shared encoder. There are no modality-specific embedding tables, no explicit alignment objective, and no pre-aligned data. Cross-modal relationships are learned implicitly through the attention body.

**Aggregation.** Stacking the per-token embeddings  $\mathbf{e}(t)$  from Eq. 6 over the sequence gives the encoded matrix  $\mathbf{H}$ :

$$\mathbf{H}(t_1, \dots, t_I) = \begin{bmatrix} \mathbf{e}(t_1)^\top \\ \vdots \\ \mathbf{e}(t_I)^\top \end{bmatrix} \in \mathbb{R}^{I \times d} \quad (7)$$

$I$  is the sequence length, i.e., the number of input tokens per sample.

Because the same FFN  $g_\theta$  encodes every token, all modalities share one set of parameters. A direct consequence is that two tokens with similar surface form receive similar embeddings even when they belong to different modalities. For example, a gene symbol that also appears as a substring of the evidence text is encoded by the same network, so the two produce correlated embeddings with no explicit alignment loss and no aligned

training data. We call this parameter sharing rather than demonstrated semantic alignment, since the correlation follows from reusing one network and not from a separately trained alignment objective. For brevity, in the equations below we write  $\mathbf{H}$  for  $\mathbf{H}(t_1, \dots, t_I)$ ,  $\mathbf{v}$  for  $\mathbf{v}(t_1, \dots, t_I)$ , and  $\hat{y}$  for  $\hat{y}(t_1, \dots, t_I)$ .

**Compression.** A single query vector  $\mathbf{q} \in \mathbb{R}^d$ , a free parameter initialized randomly and learned during training rather than computed from the input, attends over the token embeddings in  $\mathbf{H}$  and pools them, then refines the result with a residual MLP that compresses the  $I \times d$  matrix into a single  $d$ -vector,

$$\begin{aligned} \mathbf{a} &= \text{Attn}(\mathbf{q}, \mathbf{H}), \\ \mathbf{v} &= \mathbf{a} + \text{LN}(\text{MLP}(\mathbf{a})) \in \mathbb{R}^d, \end{aligned} \quad (8)$$

where  $\text{LN}(\cdot)$  denotes layer normalization,  $\text{Attn}(\cdot)$  is single-head attention pooling with  $\mathbf{q}$  as the query, and  $\text{MLP}(\cdot)$  is a two-layer feedforward network with ReLU.

**Predictor.** A linear head maps the compressed vector  $\mathbf{v}$  from Eq. 8 to the output:

$$\hat{y} = \mathbf{v} W_{\text{cls}} + b_{\text{cls}}, \quad (9)$$

where  $(W_{\text{cls}}, b_{\text{cls}})$  is the classification head that produces class logits. For regression the head  $(W_{\text{reg}}, b_{\text{reg}})$  gives a scalar  $\hat{y} = \mathbf{v} W_{\text{reg}} + b_{\text{reg}} \in \mathbb{R}$ . We use a cross-attention plus MLP aggregator for the multimodal body.

**Why SPINHASH works.** Two properties, stated and proved in Appendix B, explain the design. First, Spin-SimHash *preserves similarity*. The chance that two tokens differ on a given spin equals the angle between their feature vectors divided by  $\pi$ , shown in Prop. 1, so close tokens land a small Hamming distance apart and the shared encoder treats them as neighbors. For instance *cardiomyopathy* and *cardiomyopathic*, at cosine  $\approx 0.9$  and angle  $\approx 26^\circ$ , agree on  $\approx 86\%$  of spins, whereas unrelated tokens at angle  $\approx 90^\circ$  agree on only  $\approx 50\%$ . A learned table must discover this geometry from data. Spin-SimHash supplies it for free, which is why it is the most stable variant, as Table 9 shows. Second, the encoder cost is *constant* in the vocabulary size and the modality count, proved as Proposition 2 and detailed in §3.5.

### 3.5 Complexity

Because  $g_\theta$  is shared across all  $K$  modalities and all tokens, its parameter count is independent of every

| Encoder         | Embedding cost                         | Constant in $ V $ ? |
|-----------------|----------------------------------------|---------------------|
| Subword table   | $\mathcal{O}( V d)$                    | No                  |
| Byte-level      | $\mathcal{O}(256d)$                    | $ V $ only          |
| SPINHASH (spin) | $\mathcal{O}(2d)$ , $\sim 0.2\text{M}$ | $ V , K$            |

Table 1: Only SPINHASH has embedding cost constant in both the vocabulary size  $|V|$  and the fused-modality count  $K$ . Byte-level encoders remove the vocabulary but lengthen the input sequence.

| $d$  | Subword table | SPINHASH |
|------|---------------|----------|
| 256  | 4.6 GB        | 0.05 GB  |
| 768  | 13.7 GB       | 0.10 GB  |
| 2048 | 36.6 GB OOM   | 0.20 GB  |
| 4096 | 73.4 GB OOM   | 0.47 GB  |

Table 2: Projected peak embedding memory, no model trained. At a  $\approx 892\text{K}$ -symbol medical-code union, the subword table OOMs a 32 GB GPU at  $d \geq 2048$  while SPINHASH stays constant, about  $6900\times$  fewer embedding parameters. Single GPU.

| Vocab | Spin-SimHash | Spin-Random | Spin-Repulse       |
|-------|--------------|-------------|--------------------|
| 1K    | 0.02 s       | 0.09 s      | 0.14 s             |
| 10K   | 0.11 s       | 0.89 s      | 4.2 s              |
| 30K   | 0.39 s       | 2.8 s       | 41.0 s             |
| 100K  | 1.2 s        | 9.0 s       | $\sim 40\text{GB}$ |

Table 3: One-time construction cost (CPU) by assignment rule. Spin-SimHash and Spin-Random are  $\mathcal{O}(|V|)$  and need no corpus. Spin-Repulse builds a dense  $|V| \times |V|$  co-occurrence matrix at cost  $\mathcal{O}(|V|^2)$  and is infeasible at large  $|V|$ . This is a property of the assignment rule, not of the learned encoder.

per-modality vocabulary  $V_k$  and of  $K$ . Adding a modality, extending a terminology, or encountering a previously unseen symbol adds *no* parameters and requires no retraining. The standard  $\sum_k |V_k|d$  table instead grows with each added modality and with vocabulary size. Table 1 contrasts the embedding cost of each paradigm.

**Feasibility at scale.** At a realistic fused medical-code vocabulary of ICD-10-CM, ICD-10-PCS, SNOMED CT, and RxNorm, a union of  $\approx 892\text{K}$  symbols, the subword embedding table needs 13.7 GB at  $d=768$  and 73.4 GB at  $d=4096$ , exceeding a 32 GB GPU at  $d \geq 2048$  before any training begins, whereas SPINHASH’s encoder stays at 0.05 to 0.47 GB regardless of  $d$  or  $|V|$ , about  $6900\times$  fewer embedding parameters as Table 2 reports. Fusing many high-cardinality terminologies on a single GPU is therefore feasible *only* with a vocabulary-independent encoder.

**Learned cost versus construction cost.** The parameter counts above are the *learned* cost, constant in  $|V|$  and  $K$  for every assignment rule. The one-time *construction* of the code differs by rule, as Table 3 shows. Spin-SimHash and

Spin-Random both build the code directly from each token in linear time  $\mathcal{O}(|V|)$ , need no corpus statistics, and handle unseen or streaming tokens on the fly. Spin-Repulse instead builds a dense  $|V| \times |V|$  co-occurrence matrix and runs an Ising sampler, a one-time  $\mathcal{O}(|V|^2)$  cost that is infeasible for very large or rapidly changing vocabularies. Spin-SimHash is our recommended default because it is the most stable variant while still constructing in linear time, giving constant learned cost and linear construction cost. Spin-Repulse is intended for moderate, static vocabularies and can be made scalable with a sparse top- $k$  co-occurrence graph.

## 4 Experiments

We evaluate SPINHASH on both accuracy and efficiency. Single-modality classification spans text in Table 4, DNA in Table 6, and protein, chemistry, movie, and music in Table 8, while multimodal clinical fusion is in Table 7. Alongside accuracy we report embedding memory and training time in Tables 5 and 6. We then ablate the spin-assignment hash and encoder width in §4.3, in Tables 9 and 10.

### 4.1 Settings

**Datasets.** *Single-modality text:* sentiment classification with two classes on IMDB, SST-2, Yelp polarity, and Amazon polarity, and topic classification with four classes on AG-News. *Single-modality DNA:* promoter classification with two classes on the Genomic Benchmarks human-promoter set (Grešová et al., 2023). *Multimodality, clinical diagnosis:* two datasets, each fusing  $K$  heterogeneous modalities through one shared encoder. ClinVar (Landrum et al., 2018) has  $K=5$ , namely condition, gene, variant, variant type, and review status. PharmGKB (Whirl-Carrillo et al., 2012) has  $K=6$ , namely phenotype, drug, gene, variant, specialty, and evidence level. Train, validation, and test splits follow each dataset’s standard protocol, with 70/10/20 for the clinical sets.

**Baselines.** Our primary baseline is *Subword*, the standard `nn.Embedding( $V, d$ )` table, using bodies identical to SPINHASH across all tasks. We compare it with the vocabulary-reduction families. CharFormer (Tay et al., 2022) removes the vocabulary at the cost of longer inputs. Hash Embeddings (Svenstrup et al., 2017) and Bloom embeddings fix table size while retaining vocabulary dependence, both on text in Table 5. EmByte (Li et al., 2025), a byte-code encoder with a shared FFN, is

evaluated on text and with self-attention in Appendix H. T-Free (Deiseroth et al., 2024) and HashFormers (Xue and Aletras, 2022) provide learned-bucket hashing baselines in Table 9. For DNA we compare with Nucleotide Transformer v2-500M (Dalla-Torre et al., 2025), a same-size pretrained HyenaDNA-tiny (Nguyen et al., 2023), the Genomic Benchmarks CNN (Grešová et al., 2023), and Caduceus-Ph/PS (Schiff et al., 2024) in Table 6, plus a from-scratch 117M DNABERT-2 (Zhou et al., 2024) at Transformer scale in Appendix H.

**Parameters.** Text-classification runs use an MLP-512 spin encoder, a BiLSTM body with hidden 300, 2 layers, and bidirectional, 10 epochs, Adam at  $5 \times 10^{-4}$ , batch size 16, and 3 seeds. The DNA encoder is paired with a Hyena-tiny body of 436K params and  $k=22$  k-mer tokens, fine-tuned on human promoters for 50 epochs, about 16 min on one RTX 5090 with 32 GB. The published-baseline training cost for NT-v2-500M is its full-fine-tune cost of  $\sim 120$  GPU-hr on  $8 \times$  A100 (Dalla-Torre et al., 2025). Clinical fusion uses a cross-attention+MLP body,  $d=64$ , 25 epochs and 10 seeds, with significance from a two-sided paired bootstrap of 10,000 resamples.

## 4.2 Results

**Single-modality text classification.** Table 4 sorts five text datasets by vocabulary size and shows SPINHASH holds near-parity with the subword table, within about one point on every set, while its encoder stays constant at 0.2M parameters and the subword table grows from 3.4M to 55.8M rows. The compression gain increases with vocabulary size, from  $17\times$  on AG-News to  $283\times$  on Amazon, and is unbounded in  $V$ . Table 5 gives a per-method view on SST-2 and IMDB, where at a matched small BiLSTM body Spin-SimHash matches or beats the subword table at 7 to  $39\times$  fewer embedding parameters, is the most stable across seeds, and outperforms the vocabulary-reduction baselines. Per-step inference latency stays within 0.1 s of the subword baseline.

**Single-modality DNA promoter classification.** Table 6 reports human-promoter classification. At  $k=22$  SPINHASH reaches  $98.33 \pm 0.14$  from random initialization, above every param-matched reference we train from scratch under the same protocol, a same-size HyenaDNA-tiny at 90.9, Caduceus-Ph and Caduceus-PS at 94.7 and 95.0, and the  $1000\times$  larger NT-v2-500M at 95.7 while using 0.1% of its parameters. Our

| Dataset              | $V$    | $V \times d$ | Comp        | Subword | SPINHASH |
|----------------------|--------|--------------|-------------|---------|----------|
| AG-News              | 13.4K  | 3.4M         | $17\times$  | 0.878   | 0.872    |
| Yelp <sub>25K</sub>  | 16.2K  | 4.1M         | $21\times$  | 0.918   | 0.918    |
| IMDB                 | 30.1K  | 7.7M         | $39\times$  | 0.867   | 0.874    |
| Yelp <sub>560K</sub> | 83.3K  | 21.3M        | $108\times$ | 0.960   | 0.957    |
| Amazon               | 217.9K | 55.8M        | $283\times$ | 0.951   | 0.941    |
| mean                 |        |              |             | 0.915   | 0.912    |

Table 4: Text classification across five datasets sorted by vocabulary size, the learned subword table versus a constant 0.197M-parameter SPINHASH encoder. Accuracies are 3-seed means at  $d=256$  on a matched BiLSTM body, on AG-News and Yelp (Zhang et al., 2015), IMDB (Maas et al., 2011), and Amazon Polarity.

from-scratch HyenaDNA-tiny falls below the 94.4 its authors report with genome pretraining (Nguyen et al., 2023), so SPINHASH exceeds both our reproduction and the stronger published number, while our Caduceus runs (Schiff et al., 2024) match their published accuracies. At  $k=6$ , where a native table is still feasible, SPINHASH and a learned table are on par,  $92.1 \pm 1.5$  vs  $91.7 \pm 0.4$  over 5 seeds, and both trail the  $k=22$  result by roughly six points. What SPINHASH adds is access to the high- $k$  regime at constant cost, since a native  $4^{22}$ -row table cannot be built. The same encoder holds at Transformer scale. On a from-scratch 117M DNABERT-2 it improves promoter accuracy by 15 points over the native BPE embedding, and it composes with a two-layer self-attention body without collapse, as Appendix H details.

**Multimodal clinical fusion.** Table 7 fuses  $K \in \{5, 6\}$  heterogeneous clinical modalities such as condition, gene, variant, drug, and structured codes through one shared SPINHASH encoder and a cross-attention body. With matched training of 25 epochs, 10 seeds, and a two-sided paired bootstrap on final-epoch per-example correctness, Spin-SimHash reaches *parity* with the learned subword table on ClinVar at  $+1.3$ ,  $p=0.09$ , and PharmGKB at  $+0.9$ ,  $p=0.13$ , while keeping embedding parameters *constant* as modalities are added, whereas the per-modality table grows as  $\sum_k V_k d$ . The contribution here is constant-cost fusion of many high-cardinality modalities, not a clinical accuracy gain. On a third fused dataset, TCGA pan-cancer cancer-type classification, all encoders saturate near 100%, confirming parity at constant embedding cost.

**Breadth across modalities.** Beyond text, DNA, and clinical fusion, we run the same encoder on four further symbolic modalities at a matched budget, reported in Table 8. SPINHASH wins where the learned table bottlenecks, on large-vocabulary

| Method                                 | Accuracy           |                    | Emb params         | Mem MB     | Train s |
|----------------------------------------|--------------------|--------------------|--------------------|------------|---------|
|                                        | SST-2              | IMDB               |                    |            |         |
| Subword table                          | 0.808±0.007        | 0.869±0.002        | 1.41/7.71M         | 5.4        | 51      |
| <i>Vocabulary-reduction baselines:</i> |                    |                    |                    |            |         |
| Hash Emb.                              | 0.783              | 0.840              | 0.70M              | 2.8        | 45      |
| CharFormer                             | 0.704              | 0.777              | 0.45M <sup>†</sup> | 1.8        | 33      |
| <b>Spin-SimHash</b>                    | <b>0.810±0.003</b> | <b>0.875±0.001</b> | <b>0.20M</b>       | <b>0.8</b> | 52      |
| Spin-Random                            | 0.804±0.005        | 0.838±0.040        | 0.20M              | 0.8        | 52      |
| Spin-Repulse                           | 0.786±0.000        | 0.869±0.004        | 0.20M              | 0.8        | 51      |

Table 5: Per-method text classification on SST-2 and IMDB against the subword table and the vocabulary-reduction baselines Hash Embeddings (Svenstrup et al., 2017) and CharFormer (Tay et al., 2022). Accuracies are 3-seed means with standard deviation on a matched BiLSTM body, where Spin-Random is high variance on IMDB. For the Subword table, Emb lists the SST-2 and IMDB table sizes, and Mem and Train are the embedding footprint and one-seed wall clock on SST-2. The constant 0.20M encoder is  $7\times$  smaller than the 1.41M SST-2 subword table and  $39\times$  smaller than the 7.71M IMDB table, dropping the footprint from 5.4 to 0.8MB. <sup>†</sup> CharFormer reports total parameters with no vocabulary table.

| Model                                                        | Params      | Pre-train<br>(size) | Mem<br>(GB) | Tr.<br>(hr) | Acc          |
|--------------------------------------------------------------|-------------|---------------------|-------------|-------------|--------------|
| <i>Matched-resolution control, <math>k=6</math>, 5 seeds</i> |             |                     |             |             |              |
| Learned table                                                | 959K        | none                | 0.7         | 0.11        | 91.7         |
| SPINHASH                                                     | 500K        | none                | 0.8         | 0.11        | 92.1         |
| <i>Param-matched references, ours, from scratch, 50 ep</i>   |             |                     |             |             |              |
| CNN                                                          | 264K        | none                | 0.14        | 0.012       | 85.4         |
| HyenaDNA-tiny                                                | 436K        | none                | 0.68        | 0.128       | 90.9         |
| Caduceus-Ph                                                  | 471K        | none                | 0.59        | 0.093       | 94.7         |
| Caduceus-PS                                                  | 471K        | none                | 0.90        | 0.148       | 95.0         |
| <i>Published large model</i>                                 |             |                     |             |             |              |
| NT-v2-500M                                                   | 500M        | MS (~174G)          | 20.0        | 120.0       | 95.7         |
| <i>SPINHASH, <math>k=22</math>, 11 seeds</i>                 |             |                     |             |             |              |
| <b>SPINHASH (R)</b>                                          | <b>500K</b> | <b>none</b>         | <b>1.0</b>  | <b>0.26</b> | <b>98.33</b> |
| SPINHASH (W)                                                 | 500K        | Wiki (~0.5G)        | 1.0         | 0.26        | 98.52        |

Table 6: Human-promoter classification on Genomic Benchmarks (Grešová et al., 2023), one RTX 5090. Reference rows are our own runs, param-matched to SPINHASH and trained from scratch for 50 epochs on the same split, so accuracy, Mem, and Tr are measured under one protocol. Our from-scratch HyenaDNA-tiny reaches 90.9 versus the 94.4 its authors report with genome pretraining (Nguyen et al., 2023), while our Caduceus runs (Schiff et al., 2024) reproduce their published accuracies. NT-v2-500M is cited from its paper (Dalla-Torre et al., 2025). W = WikiText-103 pretraining (Merity et al., 2017), R = random init, and SPINHASH uses the Spin-Random hash over 11 seeds, with the  $k=6$  control over 5. Mem is peak GPU training memory.

| Dataset (# mod.) | Subword | Spin-SimHash | $\Delta$ | $p$  |
|------------------|---------|--------------|----------|------|
| ClinVar (5)      | 0.648   | <b>0.661</b> | +1.3     | 0.09 |
| PharmGKB (6)     | 0.703   | <b>0.713</b> | +0.9     | 0.13 |

Table 7: Multimodal clinical fusion of  $K \in \{5, 6\}$  high-cardinality modalities through one shared SPINHASH encoder. ClinVar (Landrum et al., 2018) and PharmGKB (Whirl-Carrillo et al., 2012), with the Spin-SimHash hash, a cross-attention body, 25 epochs, 10-seed final-epoch means, and a constant 0.27M encoder.  $\Delta$  is versus subword with a two-sided paired-bootstrap  $p$ .

DNA and high-cardinality recommendation, and sits at parity or slightly below on small-vocabulary

| Task                   | Subword      | SPINHASH     | $\Delta$ |
|------------------------|--------------|--------------|----------|
| Text (IMDB)            | 0.867        | <b>0.874</b> | +0.7     |
| DNA (promoter $k=22$ ) | 0.920        | <b>0.983</b> | +6.3     |
| Protein (DeepLoc)      | <b>0.852</b> | <b>0.852</b> | 0.0      |
| Protein (MetalIon)     | <b>0.619</b> | 0.596        | -2.3     |
| Chem (BBBP)            | <b>0.882</b> | 0.875        | -0.7     |
| Chem (BACE)            | <b>0.690</b> | 0.661        | -2.9     |
| Rec (MovieLens HR@10)  | 0.597        | <b>0.634</b> | +3.7     |
| Music (MIDI genre)     | 0.406        | <b>0.416</b> | +1.0     |

Table 8: SPINHASH against the learned subword table across six modalities at a matched budget, one representative task per modality. Embedding cost is constant  $\sim 0.2$ M in every row.

symbolic tasks such as the protein amino-acid alphabet and the  $\sim 40$ -symbol SMILES character vocabulary, both below the  $V \approx 4,200$  crossover. This is consistent with the scope of the method, vocabulary-independence and constant cost everywhere, with accuracy gains concentrated where a learned table cannot fit or does not generalize.

### 4.3 Ablation Study

**The hash function matters.** Table 9 ablates the hash  $\mathcal{H}$  on IMDB with 3 seeds and an identical MLP-512 encoder. The similarity-preserving **Spin-SimHash** is best and dramatically more stable at  $0.875 \pm 0.001$  than the prior random map Spin-Random at  $0.838 \pm 0.040$ , a  $\sim 40\times$  larger variance with one seed collapsing. Our stochastic context-repulsive **Spin-Repulse** ties the full subword table. Sparse multi-hot codes Spin-MultiHash, Spin-T-FREE, and Spin-HashFormers are markedly less stable. The choice of hash thus matters as much as the encoder itself.

**Encoder width.** Table 10 ablates the shared-FFN compressor width on IMDB: MLP-256/512/768 all reproduce the subword baseline within noise, with

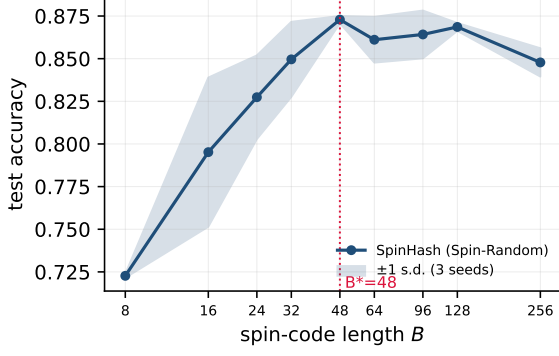


Figure 2: Accuracy versus spin-code length  $J$  on IMDB. 3 seeds, band =  $\pm 1$  s.d.

| SPINHASH hash       | IMDB acc                            | Emb.  |
|---------------------|-------------------------------------|-------|
| Subword             | $0.869 \pm 0.002$                   | 7.71M |
| <b>Spin-SimHash</b> | <b><math>0.875 \pm 0.001</math></b> | 0.20M |
| Spin-Repulse        | $0.869 \pm 0.004$                   | 0.20M |
| Spin-Learned        | $0.849 \pm 0.003$                   | 0.39M |
| Spin-Random         | $0.838 \pm 0.040$                   | 0.20M |
| Spin-MultiHash      | $0.778 \pm 0.056$                   | 0.20M |
| Spin-T-FREE         | $0.701 \pm 0.148$                   | 0.20M |
| Spin-HashFormers    | $0.574 \pm 0.076$                   | 0.20M |

Table 9: Hash ablation on IMDB, 3 seeds, all SPINHASH variants sharing the identical MLP-512 spin encoder at 0.197M. The Spin-MultiHash, Spin-T-FREE, and Spin-HashFormers variants adapt existing hashing rules (Charikar, 2002; Deiseroth et al., 2024; Xue and Aletras, 2022) onto the spin substrate.

| Encoder                | Compressor     | IMDB acc     | Emb. params   |
|------------------------|----------------|--------------|---------------|
| Subword                | none           | 0.867        | 7.71M         |
| EmByte (byte)          | MLP-256        | 0.867        | 0.66M         |
|                        | MLP-512        | 0.860        | 1.25M         |
|                        | MLP-768        | 0.866        | 1.84M         |
| <b>SPINHASH (spin)</b> | MLP-256        | 0.857        | 0.099M        |
|                        | <b>MLP-512</b> | <b>0.874</b> | <b>0.197M</b> |
|                        | MLP-768        | 0.871        | 0.296M        |

Table 10: The byte encoder EmByte and the spin encoder SPINHASH share a decoding FFN, swept over compressor width. IMDB (Maas et al., 2011), single seed, LSTM body, 10 epochs. SPINHASH uses the Spin-Random hash.

MLP-512 best at 0.874 and 0.197M params.

**Code length.** Figure 2 sweeps the spin-code length  $J$  on IMDB with 3 seeds. Accuracy climbs steeply as  $J$  grows from 8 to  $\sim 48$  and then *plateaus*, with the default  $J=64$  squarely on the plateau. Below  $J \approx 24$  codes collide and merge distinct tokens. A small, fixed  $J$  thus suffices regardless of vocabulary size, consistent with SPINHASH’s constant cost.

## 5 Conclusion

We show that the vocabulary boundary between genomic, protein, and clinical models is an architectural artifact. A single shared encoder is reused across DNA, protein, and natural-language bytes, matching or outperforming models orders of magnitude larger. Removing the vocabulary boundary lets one shared encoder serve DNA, protein, and clinical-text modeling at constant cost, enabling multimodal systems that scale by adding modalities rather than retraining from scratch.

## Limitations

SPINHASH has a fixed  $\sim 0.2$ M-parameter cost, so tables can be smaller for small vocabularies. Its advantage grows with vocabulary size and modality count. We do not evaluate autoregressive generation or continuous modalities. The  $k=22$  DNA advantage reflects feasibility rather than embedding quality. We do not claim semantic alignment from parameter sharing. Finally, Spin-Repulse requires  $O(V^2)$  construction, making Spin-SimHash the practical default.

## Ethics Statement

We use public, appropriately licensed datasets. Clinical resources contain no patient identifiers. Clinical results are for research only and should not guide medical decisions. We acknowledge dataset bias and potential performance disparities across languages, naming conventions, and biomedical domains. We used an AI assistant for language editing and experiment scale-up.

## Acknowledgments

We thank collaborators from earlier stages of this research.

## References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. 2022. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Moses S Charikar. 2002. Similarity estimation techniques from rounding algorithms. In

- Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 380–388.
- Richard J Chen, Ming Y Lu, Drew F K Williamson, Tiffany Y Chen, Jana Lipkova, Zahra Noor, Muhammad Shaban, Maha Shady, Mane Williams, Bumjin Joo, and Faisal Mahmood. 2022. Pan-cancer integrative histology-genomic analysis via multimodal deep learning. *Cancer Cell*, 40(8).
- Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics (TACL)*, 10:73–91.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Édouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451.
- Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Bernardo P. de Almeida, Hassan Sirelkhatim, Guillaume Richard, Marcin Skwark, Karim Beguir, Marie Lopez, and Thomas Pierrot. 2025. [Nucleotide transformer: building and evaluating robust foundation models for human genomics](#). *Nature Methods*, 22:287–297.
- Björn Deiseroth, Manuel Brack, Patrick Schramowski, Kristian Kersting, and Samuel Weinbach. 2024. T-FREE: Subword tokenizer-free generative LLMs via sparse representations for memory-efficient embeddings. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186.
- Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. 2023. Imagebind: One embedding space to bind them all. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Katarina Grešová, Vlastimil Martinek, David Čechák, Petr Šimeček, and Panagiotis Alexiou. 2023. Genomic benchmarks: a collection of datasets for genomic sequence classification. *BMC Genomic Data*, 24(1):25.
- Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. *ICLR*.
- Melissa J. Landrum, Jennifer M. Lee, Mark Benson, Garth R. Brown, Chen Chao, Shanmuga Chitipiralla, Baoshan Gu, Jennifer Hart, Douglas Hoffman, Wonhee Jang, Karen Karapetyan, Kenneth Katz, Chunlei Liu, Zenith Maddipatla, Adriana Malheiro, Kurt McDaniel, Michael Ovetsky, George Riley, Guangfeng Zhou, J. Bradley Holmes, Brandi L. Kattman, and Donna R. Maglott. 2018. Clinvar: improving access to variant interpretations and supporting evidence. *Nucleic Acids Research*, 46(D1):D1062–D1067.
- Shenglan Li, Jia Xu, and Mengjiao Zhang. 2025. [Embyte: Decomposition and compression learning for small yet private nlp](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 7182–7201.
- Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. 2023. Unified-io: A unified model for vision, language, and multi-modal tasks. In *International Conference on Learning Representations (ICLR)*.
- Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. [Learning word vectors for sentiment analysis](#). In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. [Pointer sentinel mixture models](#). In *International Conference on Learning Representations (ICLR)*.
- Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092.
- Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Callum Birch-Sykes, Michael Wornow, Aman Patel, Clayton Rabideau, Stefano Massaroli, Yoshua Bengio, Stefano Ermon, Stephen Baccus, and Christopher Ré. 2023. HyenaDNA: Long-range genomic sequence modeling at single nucleotide resolution. In *Advances in Neural Information Processing Systems (NeurIPS)*.

- Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srini Iyer. 2025. Byte latent transformer: Patches scale better than tokens. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. 2022. A generalist agent. *Transactions on Machine Learning Research (TMLR)*.
- Yair Schiff, Chia-Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. 2024. Caduceus: Bi-directional equivariant long-range DNA sequence modeling. In *International Conference on Machine Learning (ICML)*.
- Stefan Schulz, Ann Christin Woerl, Florian Jungmann, Christina Glasner, Philipp Stenzel, Stephanie Strobl, Aurelie Fernandez, Daniel Christoph Wagner, Axel Haferkamp, Peter Mildnerberger, Wilfried Roth, and Sebastian Foersch. 2021. Multimodal deep learning for prognosis prediction in renal cancer. *Frontiers in Oncology*, 11.
- Xiaorui Su, Shvat Messica, Yepeng Huang, Ruth Johnson, Lukas Fesser, Shanghua Gao, Faryad Sahneh, and Marinka Zitnik. 2025. Multimodal medical code tokenizer. In *International Conference on Machine Learning (ICML)*.
- Dan Tito Svenstrup, Jonas Meinertz Hansen, and Ole Winther. 2017. Hash embeddings for efficient word representations. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Yi Tay, Vinh Q Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2022. Charformer: Fast character transformers via gradient-based subword tokenization. In *International Conference on Learning Representations (ICLR)*.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP*, pages 353–355.
- Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *International Conference on Machine Learning (ICML)*.
- Michelle Whirl-Carrillo, Ellen M McDonagh, JM Hebert, Li Gong, Katrin Sangkuhl, Caroline F Thorn, Russ B Altman, and Teri E Klein. 2012. Pharmacogenomics knowledge for personalized medicine. *Clinical Pharmacology & Therapeutics*, 92(4):414–417.
- Michael Wornow, Yizhe Xu, Rahul Thapa, Birju Patel, Ethan Steinberg, Scott Fleming, Michael A. Pfeffer, Jason Fries, and Nigam H. Shah. 2023. [The shaky foundations of large language models and foundation models for electronic health records](#). *npj Digital Medicine*, 6(1):135.
- Huiyin Xue and Nikolaos Aletras. 2022. HashFormers: Towards vocabulary-independent pre-trained transformers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics (TACL)*, 10:291–306.
- Lili Yu, Daniel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. 2023. MegaByte: Predicting million-byte sequences with multiscale transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Zhihan Zhou, Yanrong Ji, Weijian Li, Pratik Dutta, Ramana V. Davuluri, and Han Liu. 2024. DNABERT-2: Efficient foundation model and benchmark for multi-species genome. In *International Conference on Learning Representations (ICLR)*.

## Appendix Overview

The appendix is organized as follows.

- **Appendix A – Notation.** Glossary of every symbol used in the paper.
- **Appendix B – Properties and Proofs.** Similarity-preservation and constant-cost proofs.
- **Appendix C – Datasets and Splits.** Sources, train/validation/test protocol, and per-modality token budgets.
- **Appendix D – Hyperparameters and Training.** All settings per experimental regime, with hardware and compute budget.
- **Appendix E – Hash Algorithms.** Pseudocode for the Spin-SimHash spin assignment.
- **Appendix F – Additional Results.** Collision-free coding analysis.
- **Appendix G – Post-Submission Additional Experiments.** Empirical  $\alpha/\pi$  collision check for the SimHash bound.
- **Appendix H – Transformer-Scale Evaluation.** DNABERT-2 at 117M and a self-attention body.
- **Appendix I – Reproducibility.** Released code and data.

## A Notation

Table 11 lists every symbol used in the paper, its meaning, and where it is introduced.

## B Properties and Proofs

Let  $\psi(t) \in \mathbb{R}^F$  be the feature map of Eq. 3 and  $\alpha(t, t') \in [0, \pi]$  the angle between  $\psi(t)$  and  $\psi(t')$ , whose cosine is their cosine similarity.

**Proposition 1** (Similarity preservation). *For Spin-SimHash in Eq. 3 with hyperplanes  $\mathbf{r}_1, \dots, \mathbf{r}_J$  drawn i.i.d. from  $\mathcal{N}(0, I_F)$ , for any tokens  $t, t'$  and any spin index  $j$ ,*

$$\Pr [c_j(t) \neq c_j(t')] = \frac{\alpha(t, t')}{\pi},$$

so the expected Hamming distance is  $\mathbb{E} d_H(\mathbf{c}(t), \mathbf{c}(t')) = J \alpha(t, t')/\pi$ .

*Proof.* By Eq. 3,  $c_j(t) = \text{sign}(\langle \psi(t), \mathbf{r}_j \rangle)$ , so  $c_j(t) \neq c_j(t')$  exactly when the hyperplane with normal  $\mathbf{r}_j$  separates  $\psi(t)$  and  $\psi(t')$ , i.e.  $\langle \psi(t), \mathbf{r}_j \rangle$  and  $\langle \psi(t'), \mathbf{r}_j \rangle$  have opposite signs. For  $\mathbf{r}_j \sim$

| Symbol                | Meaning (first use)                                                      |
|-----------------------|--------------------------------------------------------------------------|
| $t$                   | a token (subword from any modality)                                      |
| $b(t)$                | byte sequence of token $t$                                               |
| $\mathbf{c}(t)$       | spin code, $\in \{\uparrow, \downarrow\}^J$ (Eq. 1)                      |
| $J$                   | spin-code length ( $J=64$ )                                              |
| $\mathcal{H}$         | spin-assignment hash (§3)                                                |
| $\phi$                | one-hot lift $\{\uparrow, \downarrow\} \rightarrow \mathbb{R}^2$ (Eq. 6) |
| $g_\theta$            | shared FFN encoder; $\theta$ its parameters                              |
| $d$                   | embedding dimension                                                      |
| $H$                   | encoded sequence, $\in \mathbb{R}^{I \times d}$ (Eq. 7)                  |
| $\langle S_k \rangle$ | modality- $k$ separator token                                            |
| $T_k$                 | per-modality token budget                                                |
| $K$                   | number of modalities                                                     |
| $V, V_k$              | vocabulary size (of modality $k$ )                                       |
| $\mathbf{q}$          | learnable cross-attention query                                          |
| $\mathbf{v}$          | aggregated representation (Eq. 8)                                        |
| $\psi(t)$             | feature map (byte $n$ -gram counts), SimHash                             |
| $\mathbf{r}_j$        | random hyperplane $j$ , SimHash (Eq. 3)                                  |
| $S(t, t')$            | signed-PMI context similarity of tokens $t, t'$ , Spin-Repulse           |
| $\beta$               | inverse temperature (Spin-Repulse)                                       |
| $\lambda$             | spin-balance penalty (Spin-Repulse)                                      |
| $I$                   | sequence length, number of input tokens (Eq. 7)                          |
| $F$                   | SimHash feature dimension, $F=2048$                                      |
| $m$                   | shared-FFN hidden width (App. B)                                         |
| $C$                   | number of classes                                                        |
| $\alpha(t, t')$       | angle between $\psi(t)$ and $\psi(t')$ (App. B)                          |

Table 11: Variable glossary.

$\mathcal{N}(0, I_F)$  the direction  $\mathbf{r}_j/\|\mathbf{r}_j\|$  is uniform on the unit sphere, and the probability that such a uniformly random hyperplane separates two fixed vectors equals the fraction of directions inside the angular wedge between them, namely  $\alpha(t, t')/\pi$  (Charikar, 2002). The  $J$  spins are i.i.d., so  $\mathbb{E} d_H = \sum_{j=1}^J \Pr[c_j(t) \neq c_j(t')] = J \alpha(t, t')/\pi$ .  $\square$

**Proposition 2** (Constant cost). *The learned parameters of SPINHASH are those of the shared encoder  $g_\theta : \mathbb{R}^{2J} \rightarrow \mathbb{R}^d$  in Eq. 6 and the linear head in Eq. 9. Their number is independent of the vocabulary size  $|V|$  and the number of modalities  $K$ .*

*Proof.* By Eq. 6,  $g_\theta$  is a two-layer FFN with weight matrices of shapes  $2J \times m$  and  $m \times d$  for hidden width  $m$ , so  $|\theta_g| = 2Jm + md + m + d$ , a function of  $J, m, d$  only. The head in Eq. 9 adds  $dC + C$  parameters for  $C$  classes, or  $d + 1$  for regression. None of these depends on  $|V|$  or  $K$ . The spin code  $\mathbf{c}(t)$  is produced by the hash  $\mathcal{H}$  in Eqs. 2–5, whose quantities  $\mathbf{r}_j$  and  $S$  are fixed once and not trained. By contrast,  $K$  per-modality tables  $E^{(k)} \in \mathbb{R}^{V_k \times d}$  contribute  $\sum_{k=1}^K V_k d$  learned parameters, which grows without bound in both  $|V|$  and  $K$ .  $\square$

## C Datasets and Splits

**Sources.** All datasets are public. Single-modality text uses IMDB (Maas et al., 2011), AG-News and Yelp (Zhang et al., 2015), and SST-2 (Wang et al., 2018). The DNA task is Genomic Benchmarks human-promoter classification (Grešová et al., 2023). Multimodal fusion uses ClinVar (Landrum et al., 2018) and PharmGKB (Whirl-Carrillo et al., 2012).

**Splits.** Train/validation/test splits are 70/10/20 throughout, the standard protocol for the clinical sets and the standard train/test split for the text and genomic benchmarks.

**Token budgets.** For multimodal fusion, per-modality token budgets follow  $T_k = \min(\text{literature default}, p_{95} \text{ in data})$  measured on the training split, where free-text fields receive the largest budget and structured fields receive a handful of tokens each.

## D Hyperparameters and Training

Table 12 reports every hyperparameter, grouped by experimental regime, together with how each value was chosen.

| Setting                                                  | Value                     | How chosen     |
|----------------------------------------------------------|---------------------------|----------------|
| <i>Multimodal clinical (classification + regression)</i> |                           |                |
| embedding dim $d$                                        | 64                        | prior work     |
| batch size                                               | 16                        | fixed          |
| optimizer / lr                                           | Adam / $5 \times 10^{-4}$ | fixed          |
| epochs                                                   | 25                        | best val       |
| cross-attn heads                                         | 4                         | fixed          |
| MLP residual width                                       | 128                       | fixed          |
| spin-code length $J$                                     | 64                        | prior work     |
| seed                                                     | 42                        | fixed          |
| hardware                                                 | 1 × H100                  | n/a            |
| <i>Text classification (Tables 5, 9)</i>                 |                           |                |
| compressor                                               | MLP-512                   | Table 10 sweep |
| $J$ / one-hot lift                                       | 64 / dim-2                | fixed          |
| body                                                     | BiLSTM(300,2,bi)          | fixed          |
| epochs / seeds                                           | 10 / 3                    | fixed          |
| <i>DNA promoter classification (Table 6)</i>             |                           |                |
| body                                                     | Hyena-tiny (436K)         | HF checkpoint  |
| $k$ -mer order $k$                                       | 22                        | best val       |
| pre-train corpus                                         | WikiText-103              | n/a            |
| fine-tune epochs                                         | 50                        | fixed          |
| hardware                                                 | 1 × RTX 5090 (32GB)       | n/a            |

Table 12: Hyperparameters and how each was chosen.

**How each value was chosen.** The settings fall into three groups. *First, inherited from prior coded-embedding work (Li et al., 2025),* adopted unchanged so that any difference is attributable to the encoder rather than to re-tuning, namely the embedding dimension  $d=64$ , the

spin-code length  $J=64$ , the frozen one-hot lift, the Adam optimizer at learning rate  $5 \times 10^{-4}$ , batch size 16, and the BiLSTM(300, 2) text body. *Second, selected on the validation split.* The shared-FFN compressor width is chosen by the IMDB sweep in Table 10: MLP-256/512/768 reach 0.857/0.874/0.871, so MLP-512 is best and is used in every regime. The genomic  $k$ -mer order  $k=22$  is the highest-validation-accuracy value among the orders we tried. The clinical fine-tuning budget was raised from the 6-epoch default to 25 epochs after validation curves showed the spin encoder still under-fitting at 6 epochs. At 25 epochs it reaches parity with the subword table, whereas at 6 it trails it, which is why early clinical runs understated SPINHASH. The encoder ablation in Table 9 likewise selects the hash on validation accuracy. *Third, fixed by protocol,* not tuned, namely the 70/10/20 train/validation/test splits, the random seed 42, and the number of seeds, 3 for text, 10 for clinical, and 11 for the DNA study.

**Compute budget.** The clinical multimodal and regression experiments run on a single H100. The DNA experiment runs on a single RTX 5090 with 32 GB, fine-tuning in  $\sim 16$  minutes. For the DNA comparison, the published-baseline cost for NT-v2-500M is its reported full fine-tune of  $\sim 120$  GPU-hr on 8 × A100. HyenaDNA-tiny and our SPINHASH encoder are fine-tuned identically on one consumer GPU.

## E Spin-Assignment Hash Algorithms

Algorithm 2 specifies Spin-SimHash, the best-performing hash in Table 9: Charikar’s SimHash applied to a byte  $n$ -gram feature map, so that tokens with similar surface form receive nearby codes. The complementary Spin-Repulse sampler, which instead pushes context-similar tokens *apart* by minimising the antiferromagnetic Ising energy in Eq. 5 via Metropolis MCMC, is described in §3.3.

### Algorithm 2 Spin-SimHash (similarity-preserving)

**Require:** token  $t$ ; feature map  $\psi(t) \in \mathbb{R}^F$  (byte  $n$ -gram counts); length  $J$

- 1: draw  $\mathbf{r}_1, \dots, \mathbf{r}_J \sim \mathcal{N}(0, I_F)$  {fixed seed, shared by all tokens}
- 2: **for**  $j = 1$  to  $J$  **do**
- 3:  $c_j(t) \leftarrow \text{sign}(\langle \mathbf{r}_j, \psi(t) \rangle)$
- 4: **end for**
- 5: **return** spin code  $\mathbf{c}(t) \in \{\uparrow, \downarrow\}^J$

## F Additional Results

**Codes stay collision-free.** With collision resampling, a random length- $J$  spin code is unique across the whole vocabulary whenever  $2^J \geq V$ , and forced collisions occur only when  $J < \log_2 V$ . Every dataset here needs at most  $\lceil \log_2 V \rceil = 18$  spins to be collision-free, with Amazon at  $V=218\text{K}$ , so the default  $J=64$ , which admits  $2^{64} \approx 1.8 \times 10^{19}$  distinct codes, has vast headroom and never collides in practice. The sharp drop at  $J=8$  in Figure 2 is precisely the forced-collision regime: 256 codes must cover IMDB’s 30K types at  $\sim 118$  tokens per code, merging distinct words. Once  $J$  clears this threshold and gives the shared FFN enough input width at  $J \gtrsim 48$ , accuracy plateaus.

## G Post-Submission Additional Experiments

This section reports additional experiments completed after the initial submission. They are included for completeness and do not affect the main claims.

**Empirical  $\alpha/\pi$  collision check for Proposition 1.** Proposition 1 predicts  $\Pr[c_j(t) \neq c_j(t')] = \alpha(t, t')/\pi$ , where  $\alpha(t, t')$  is the angle between  $\psi(t)$  and  $\psi(t')$ . We test this directly on two vocabularies. First,  $|V|=20,000$   $k$ -mers sampled from the  $k=22$  human\_promoters training split. Second, the first 20,000 tokens of the GPT-2 byte-level BPE vocabulary as a text reference. For 4,096 random token pairs in each vocabulary, we compute the byte-trigram feature  $\psi$  with  $F=2048$ , the analytic angle  $\alpha(t, t')$ , the Spin-SimHash code  $\mathbf{c}(t) \in \{\uparrow, \downarrow\}^J$  at  $J=64$ , and the empirical Hamming fraction  $\text{Hamming}(\mathbf{c}(t), \mathbf{c}(t'))/J$ , then compare empirical against analytic per pair.

Empirical Hamming tracks the analytic  $\alpha/\pi$  closely on both vocabularies:

| Vocabulary           | Pearson $r$ | MAE   | RMSE  |
|----------------------|-------------|-------|-------|
| DNA $k=22$ promoters | <b>0.77</b> | 0.043 | 0.053 |
| GPT-2 byte-level BPE | 0.49        | 0.049 | 0.062 |

The correlation is strong on the DNA vocabulary, where every  $k$ -mer has exactly  $L=22$  bytes drawn from  $\{A, C, G, T\}$  so the trigram count  $\psi(t)$  is a near-faithful summary of  $t$ . It is weaker on the GPT-2 vocabulary, where distinct full tokens hash to overlapping trigram directions across  $F=2048$  buckets. On both vocabularies the empirical Hamming hugs the analytic  $\alpha/\pi$  line, confirming Proposition 1 holds quantitatively, not only in the limit.

## H Transformer-Scale Evaluation

The main-text bodies are BiLSTM, Hyena, and cross-attention networks. To test whether SPINHASH holds at real Transformer scale rather than only these compact bodies, we ran two controlled from-scratch experiments that swap only the input representation.

**DNABERT-2 at 117M.** We take a fresh DNABERT-2 (Zhou et al., 2024) of 117M parameters at random initialization and compare its native BPE embedding against the SPINHASH encoder as a drop-in, at a matched budget of 3000 masked-LM steps on a human-genome sample, then fine-tune on human non-TATA promoters. The compact 0.2M SPINHASH encoder is markedly more sample-efficient than the 3.1M BPE embedding, with lower masked-LM perplexity and a +15.2 point downstream gain, as Table 13 reports. Both arms are deliberately under-budget for a controlled comparison, so this is a sample-efficiency result at Transformer scale, not a claim of beating a fully trained DNABERT-2.

| Input representation          | MLM ppl $\downarrow$ | Promoter acc $\uparrow$ |
|-------------------------------|----------------------|-------------------------|
| Native BPE embedding, 3.1M    | 1135.7               | 77.6                    |
| <b>SPINHASH encoder, 0.2M</b> | <b>881.9</b>         | <b>92.8</b>             |

Table 13: On a from-scratch 117M DNABERT-2 (Zhou et al., 2024) at a matched 3000-step budget, the constant 0.2M SPINHASH encoder lowers masked-LM perplexity and improves promoter accuracy by 15.2 points over the native BPE embedding.

**Self-attention body.** To check that SPINHASH composes with a full self-attention Transformer rather than collapsing, we route the spin encoder through a two-layer Transformer-encoder classifier on IMDB. All variants reach 84 to 86% and none collapse to the 50% chance level, as Table 14 shows, so the encoder is not tied to an MLP body.

| Encoder through a self-attention body | IMDB acc |
|---------------------------------------|----------|
| Subword table                         | 84.6     |
| EmByte (Li et al., 2025)              | 85.6     |
| Spin-SimHash                          | 84.2     |
| Spin-Random                           | 83.7     |

Table 14: The spin encoder composes with a two-layer Transformer-encoder classifier on IMDB. All variants reach 84 to 86% with none collapsing to the 50% chance level, so SPINHASH is not tied to an MLP body.

## I Reproducibility

Code and the processed clinical and genomic data are released as supplementary material. The standard public text corpora are available

through the HuggingFace datasets library under the identifiers listed in the data README. All experiments use PyTorch with default library settings, and exact package versions are pinned in the released code.

| Paper table                  | Source code, under software/                      | Repository folder                                 |
|------------------------------|---------------------------------------------------|---------------------------------------------------|
| Table 4, text classification | run_table7_repro.py                               | table04_accuracy/                                 |
| Table 5, SST-2, IMDB, Amazon | run_improvements.py, run_table7_repro.py          | table05_sst2/, table06_imdb/, table07_largescale/ |
| Table 6, text-to-genome      | hyena_wikitext_to_dna.py                          | table09_text2genome/                              |
| Table 7, clinical fusion     | run_clinical_sig_simhash.py                       | table10_clinical/                                 |
| Table 8, breadth             | run_table7_repro.py, bert4rec_exp.py, midi_exp.py | table11_breadth/                                  |
| Table 9, hash ablation       | run_table7_repro.py                               | table12_hash_variants/                            |
| Table 10, encoder width      | run_table7_repro.py                               | table13_coding_repro/                             |
| Table 3, construction cost   | construct_timing.py                               | table01_construct/                                |

Table 15: Each experimental table mapped to its source code under software/ and its folder in the released spinhash-experiments repository, where each folder holds the code, the exact run command, and the result files, with shared helpers in \_shared\_code/. Analytical tables for compression and complexity are computed from vocabulary sizes and  $d$  and need no training.