

RESEARCH

Open Access



A novel Vector-Symbolic Architecture for graph encoding and its application to viral pangenome-based species classification

Fabio Cumbo¹ , Kabir Dhillon² , Jayadev Joshi¹ , Davide Chicco^{3,4} , Sercan Aygun⁵ and Daniel Blankenberg^{1,6*}

*Correspondence:

Daniel Blankenberg
blanked2@ccf.org

¹Computational Life Sciences,
Cleveland Clinic Research,
Cleveland Clinic, 9500 Euclid
Avenue, NA2, Cleveland, OH
44195, USA

²College of Engineering, Ohio State
University, Columbus, OH
43210, USA

³Dipartimento di Informatica
Sistemistica e Comunicazione,
Università di Milano-Bicocca,
20125 Milan, MI, Italy

⁴Institute of Health Policy
Management and Evaluation,
University of Toronto, Toronto,
ON M5T 3M6, Canada

⁵School of Computing and
Informatics, University of Louisiana
at Lafayette, Lafayette, LA
70504, USA

⁶Department of Molecular
Medicine, Cleveland Clinic Lerner
College of Medicine, Case Western
Reserve University, Cleveland,
OH 44195, USA

Abstract

Viral species classification is crucial for understanding viral evolution, epidemiology, and developing effective diagnostics and treatments. Traditional methods often rely on sequence similarity, which can be challenging for rapidly evolving viruses. Pangenomes, offering a comprehensive representation of species' genomic diversity, provide a richer perspective, but their analysis often requires advanced computational methods. We investigate the use of Hyperdimensional Computing (HDC), also known as Vector-Symbolic Architecture (VSA), an emerging computing paradigm that relies on vectors in high-dimensional spaces to encode a multi-species viral pangenome.

We develop a new method for encoding graph-structured viral pangenomes using high-dimensional vectors. Pangenomes are represented as weighted *de Bruijn* graphs constructed using sequences of consecutive k-mers from the genomes, while information about the genome species (their taxonomic label) is encoded as specific high-dimensional vectors (species hypervectors) that act as weights on the edges of the graph. The weighted *de Bruijn* graph representation is encoded into a single high-dimensional vector. We tested three classification strategies: a flat model at the species level, a flat model at the genus level, and a two-step hierarchical model.

We applied our method to a pangenome comprising 542 viral species from NCBI GenBank. Our results reveal a complex relationship between model architecture and classification accuracy. The flat species-level model achieved the highest accuracy, correctly classifying 87.08% of test genomes. Counter-intuitively, simplifying the problem to the genus level or using a hard-routing hierarchical approach degraded performance, with accuracies dropping to 60.51% and 33.57% respectively. Rather than revealing an inherent flaw in hierarchical modeling, these outcomes highlight critical architectural limitations of our current routing strategy, reflecting the interaction between routing errors and downstream error propagation in multi-step models. The model's reconstruction rate proved to be a measure of model-internal coherence, rather than a direct predictor of correctness.



This novel approach offers a promising new direction for viral classification, not only for its predictive power but its ability to reveal underlying challenges in genomic taxonomy.

Keywords Hyperdimensional computing, Vector-symbolic architectures, Pangenomes, Viruses, Classification

Introduction

Viral species classification is a cornerstone of modern virology, essential for understanding viral evolution, tracking epidemiology, and developing diagnostics and therapeutics [1]. However, classifying viruses presents profound challenges rooted in their biology. Their rapid mutation rates, high frequency of genetic recombination, and vast diversity mean that traditional classification methods, which often depend on sequence similarity of conserved marker genes, are frequently inadequate [2]. Unlike cellular life, viruses lack a universally conserved gene analogous to the 16S rRNA gene, making the construction of a comprehensive viral tree of life a complex, ongoing endeavor [3–6]. This biological reality necessitates the development of advanced, scalable, and accurate classification methods that can handle the ever-expanding volume of viral genomic data.

To better capture the full spectrum of viral genetic diversity, the field has increasingly adopted a pangenomic perspective [7–10]. A pangenome represents the entire gene repertoire of a species, encompassing the core genes (gene shared by all strains) and the accessory genes (genes present in only a subset of strains) [11, 12]. This approach provides a more holistic view of a species' adaptive potential, revealing insights into functional diversity, mechanisms of virulence, and evolutionary trajectories. Pangenomes are often visualized as complex graph structures, where nodes represent genetic sequences and edges denote their adjacency. These pangenome graphs are powerful representations that can capture large-scale structural variations, such as insertions, deletion, and rearrangements, which are often missed by linear sequence alignment [13].

While pangenome graphs offer a rich representation, their analysis introduces significant computational challenges. Constructing pangenomes with established gene-based methods often requires computationally expensive algorithms. For instance, creating a core-gene phylogeny, a key output, depends on multiple sequence alignment, which can become a major bottleneck when dealing with thousands of genomes [14]. As sequencing projects continue to deposit massive datasets into public repositories, the need for more efficient and scalable methods for pangenome analysis has become critical. The challenge lies in developing a framework that can both manage this scale and accurately model the complex relationships inherent in pangenomic data.

In this context, Hyperdimensional Computing (HDC, also known as Vector-Symbolic Architectures (VSA), emerges as a promising computational paradigm inspired by principles of neural computation [15–17]. HDC operates on high-dimensional vectors (often referred to as hypervectors), using a defined set of mathematical operations to represent and manipulate complex data structures and relationships between information. This framework offers several key advantages for bioinformatics applications: its distributed nature makes it inherently robust to noise and errors (such as sequencing errors or minor genomic variations), its reliance on simple, parallelizable operations addresses computational bottlenecks, and its capacity for one-shot learning is well-suited for biological classification tasks [18, 19].

The unique properties of HDC have led to its successful application across a range of bioinformatics problems. In genomics, HDC has been used to develop fast, alignment-free methods for sequence comparison and analysis [20–24]. It has also proven effective in classifying complex biological data, from DNA methylation patterns in cancer [25] to microbial abundance profiles in metagenomics [26], and has been applied to challenges in cheminformatics, such as drug discovery and classification [27–29] among other applications to complex problems in the life sciences [18, 19]. The ability of HDC to create compact yet descriptive representations of complex data makes it an ideal candidate for tackling the challenge of pangenome analysis, an area where its potential has not been explored so far.

This work introduces a novel VSA-based framework for viral species classification that directly leverages the power of pangenome graphs. We model the viral pangenome as a weighted *de Bruijn* graph [30] where nodes represent k-mers and edges are colored with hypervectors corresponding to the species in which they appear. This complex graph is then encoded into a single, holistic hypervector. This approach allows us to efficiently query the pangenome to determine the species of a new viral genome.

Our main contributions are:

1. a novel method for encoding large, weighted/colored *de Bruijn* graphs into a single, high-dimensional vector representation;
2. the application and validation of this method for the complex task of viral species classification;
3. an exploration of a hierarchical framework that investigates the architectural trade-offs and error-propagation vulnerabilities associated with specialized, genus-specific VSA routing;
4. experimental proof of linear relation between graph reconstruction and the corresponding accuracy.

The method leverages the right information contained within pangenomes and the computational efficiency of VSAs to enable accurate viral species classification at scale, offering a powerful new tool for computational virology.

Materials and methods

Here we present the data we used for our analysis in addition to an introduction to the theoretical foundations of the Hyperdimensional Computing paradigm together with our graph-based encoding technique.

Data acquisition and preprocessing

We retrieved all the viral reference genomes from the NCBI GenBank database [31] and randomly selected two genomes for each of the 542 species, for a total of 1,084 viral genomes belonging to 218 genera, 61 families, 37 orders, 24 classes and 14 phyla. Our definition of reference genomes was expanded to include any GenBank assembly not excluded from NCBI RefSeq for the following reasons, which we group into two main categories:

- *Issues with source material or provenance:* (i) derived from single cell – the assembly was generated from single-cell amplified material, which can impact sequence

accuracy; (ii) non-standard type material – the assembly was constructed from sequences designed as various type materials;

- *Issues with metadata or project context:* (i) incomplete taxonomic information – the assembly lacked a specified strain identifier or had an undefined genus in its lineage; (ii) part of a large-scale project – the assembly was flagged as belonging to a large project with over 100 isolates of the same species, which often have different data-release and annotation standards.

Accession numbers of the reference genomes considered in our analysis are provided in Supplementary Table S2 alongside with their taxonomic label as provided by NCBI GenBank.

It is worth noting that we focused on viral species with at least 2 reference genomes in NCBI GenBank, also performing a quality control with CheckV [32] by filtering out low- and medium-quality genomes with a completeness < 90% and a contamination > 5%. Then, we randomly selected 2 genomes per species among those that survived the quality filtering.

Since our method is k-mer-based, we first fragmented all the selected genome sequences into overlapping k-mers of length 9 over a sliding window with a single nucleotide step, by also maintaining their order of occurrence in the original sequence. We determined k-mer length empirically by computing the Cumulative Relative Entropy, the Average Number of Common Features, and the Observed Common Features for each k in the closed interval $[6 - 20]$ with *Kitsune* [33], a Python package for the estimation of the optimal k-mer size for a specific set of genomes.

Hyperdimensional computing and the MAP model

HDC is a promising computing paradigm inspired by the way the human brain works in encoding information [15, 16, 19]. It uses random binary or bipolar high-dimensional vectors, often called hypervectors, as atomic computing data. In particular, to represent and manipulate symbolic information, HDC excels by leveraging orthogonal representations with vector sizes typically ranging from $D = 100$ to $D = 1000$ bits, with D always much greater than the number of atomic information N to be represented in the high-dimensional space (a D -dimensional space guarantees the existence of up to D orthogonal vectors in the same space; $D \gg N$ to increase the chance of orthogonality in case of randomly generated vectors). Unlike conventional machine learning and neural networks, HDC targets single-pass, error- and backpropagation-free learning methods, building on bio-inspired concepts such as associative and item memory, naturally offering a series of advantages like, noise robustness, efficiency, few shot learning, and hardware efficiency (the details can be found in the Supplementary Material section).

The reason hypervectors are typically large in size is because of their random nature. The higher the dimensionality D , the higher the chances that two randomly generated vectors are orthogonal or quasi-orthogonal in the same space (with a cosine-similarity close to 0).

All these characteristics make HDC well-suited for handling large and complex datasets, particularly in bioinformatics and other data-intensive fields.

HDC operates on hypervectors using a small but efficient set of arithmetic operations, i.e., *multiplication*, *addition*, and *permutation*, that together compose the MAP (Multiply-Add-Permute) model:

- *Multiplication* (a.k.a. binding): this operation binds two hypervectors together to create a new hypervector that represents the association or relationship between the original two. The resulting hypervector is dissimilar to both input vectors. This operation is crucial for representing complex structures and relationships. The specific implementation of multiplication can vary depending on the chosen vector space, i.e., it could be implemented as the component-wise XOR in case of random binary vectors (with 0s and 1s), or component-wise multiplication in case of random bipolar vectors (with -1 s and $+1$ s). Using binary vectors and XOR in hardware simplifies the algebraic operation in a cost-effective manner;
- *Addition* (a.k.a. bundling): this operation bundles multiple hypervectors together to create a new hypervector that represents the set or combination of the input vectors. The resulting hypervector is similar to all input vectors. It is implemented as the component-wise majority rule in case of binary vectors, or component-wise addition in case of bipolar vectors, which is typically followed by a thresholding step (e.g., majority voting) to keep the result within the same vector space. This operation is essential for representing sets, sequences, and other composite structures.
- *Permutation*: it shifts the elements of a hypervector by a certain number. This helps keep positional or temporal features based on their order. It can represent sequences and temporal relationships between concepts (e.g., a permuted hypervector might represent the next item in a sequence, such as consecutive letters in a language processing application).

The combination of these three operations allows for complex computations and manipulations of high-dimensional representations, enabling HDC systems to perform various tasks, including classification, analogy reasoning, and language processing.

Graph encoding

Here, we present a HDC approach to encoding directed, weighted graphs, which we adopt and adapt from a previous study by Prathyush Poduval et al. [34]. To encode a directed weighted graph $G = (V, E, W)$ with V the set of nodes, E the set of edges, and W the set of weights, we assign a random bipolar hypervector $H_i \in \{-1, +1\}^D$ to each node $v_i \in V$. These vectors are quasi-orthogonal due to their random nature as we previously explained.

Encoding directed connections

For each node v_i , we define a node memory vector M_i to capture all the directed outgoing connections from v_i to its neighboring nodes. In directed graphs, edge directionality must be preserved. In order to do so, we introduce a permutation operator ρ , which rotates a hypervector by a fixed number of positions. This is applied to the memory component to break the symmetry in the binding operation. Specifically, the entire set of outgoing edges from node i is encoded as $H_i \otimes \rho(M_i)$, where $\otimes$ denotes binding, and ρ applies a permutation that ensures $H_i \otimes \rho(M_i) \neq H_j \otimes \rho(M_j)$, even if $i = j$ and $M_i = M_j$, thus capturing the directionality.

The global graph memory G is obtained by summing the bound representations of each node and its directed memory: $G = \sum_{i=1}^{|V|} H_i \otimes \rho(M_i)$. This bundled hypervector encodes the entire directed graph in a single, holistic representation.

Encoding edge weights

In our biological context, these weights represent specific taxonomic labels. In order to encode these weights (taxonomic labels), we employ a deterministic transformation to map them $w_{ij} \in [0,1]$ into species hypervectors $V_{w_{ij}}$. This transformation flips a proportion of components in a fixed, randomly generated base hypervector, V_1 , to create a weighted vector such that $f(V_{w_{ij}}) = \delta(V_{w_{ij}}, V_1) + \frac{1}{2} = w_{ij}$, where δ represents the function that evaluates the proportion of differing components between the encoded weight vector and the base vector V_1 . This encoding is deterministic and positionally consistent: the components from index $[w_{ij} \cdot D]$ to D in V_1 are flipped. This results in hypervectors for similar weights being more correlated to each other than to those representing distant weights. For a node i , its outgoing weighted connections are thus encoded into its memory as $M_i = \sum_{j \in \text{out}(i)} V_{w_{ij}} \otimes H_j$, where $\text{out}(i)$ denotes the set of all nodes reachable from i via a directed edge. This ensures that not only the connectivity but also the weight of each edge is holographically embedded. The overall encoding procedure for building the graph model is outlined in Algorithm 1.

Algorithm 1 VSA-based encoding of a weighted, directed graph. The algorithm outlines the procedure for converting a graph into a single high-dimensional vector G . First, a unique, random hypervector H_i is assigned to each node v_i (lines 2–5). The core of the procedure iterates through each node to compute its local memory. For each node v_i , its memory M_i is created by bundling the representations of all its outgoing connections (lines 11–16). Each connection is encoded by binding the hypervector of the target node v_j with a hypervector $V_{w_{ij}}$ representing the edge's weight. To preserve directionality, this memory M_i is permuted using a rotation operation ρ (line 18). The node's own hypervector H_i is then bound to its permuted memory M_i^p (lines 20–21). Finally, this local representation is bundled into the global graph hypervector G (line 23). The final vector holographically represents the entire graph's topology and weights

Input: (i) a directed, weighted graph (V, E, W) , where V is the set of nodes, E is the set of edges, and W is the set of weights; (ii) hypervector dimensionality D .
Output: a single hypervector G representing the entire graph.

```

1.  # Initialize node hypervectors:
2.  Create an item memory  $H$  as a dictionary to store node hypervectors
3.  For each node  $v_i \in V$ :
4.       $H_i \leftarrow$  Generate a random bipolar hypervector of dimension  $D$ 
5.      Store  $H_i$  in  $H$  with key  $v_i$ 
6.  # Initialize global graph hypervector:
7.   $G \leftarrow$  Create a zero vector of dimension  $D$ 
8.  # Encode nodes and edges:
9.  For each node  $v_i \in V$ :
10.     # Compute node memory  $M_i$ :
11.      $M_i \leftarrow$  Create a zero vector of dimension  $D$ 
12.     For each outgoing edge from  $v_i$  to a neighbor  $v_j$  with weight  $w_{ij}$ :
13.          $H_j \leftarrow H[v_j]$  # Retrieve the hypervector for the target node  $v_j$ 
14.          $V_{w_{ij}} \leftarrow \text{EncodeWeight}(w_{ij}, D)$  # Encode the weight into a hypervector
15.         connection  $\leftarrow V_{w_{ij}} \otimes H_j$  # Bind the weight vector with the target node
16.          $M_i \leftarrow M_i \oplus$  connection # Bundle connection into the node's memory
17.     # Apply permutation for directionality:
18.      $M_i^p \leftarrow \rho(M_i)$ 
19.     # Bind node with its permuted memory:
20.      $H_i \leftarrow H[v_i]$ 
21.     node  $\leftarrow H_i \otimes M_i^p$ 
22.     # Bundle into global graph:
23.      $G \leftarrow G \oplus$  node
24. return  $G$ 

```

Reconstruction

Given the global memory G , local structures can be approximately reconstructed by unbinding and reversing the permutation: $M_i \approx \rho^{-1}(H_i \otimes G)$. This estimated M_i may

contain noise due to imperfect orthogonality and vector collisions. To improve reconstruction accuracy, we employ an iterative error mitigation technique called retraining that recomputes each node's memory by implementing a supervised learning rule.

Retraining and inference

The *retraining* process addresses the specific challenge of a multi-label graph where a single edge can be associated with multiple valid taxonomic labels (encoded as edge weights). A naive correction that subtracts a mispredicted species hypervector is problematic, as that weight might be a legitimate label for that edge in another context. To avoid corrupting valid data, we implement a *one-side retraining strategy* that only corrects for false negatives.

This process is as follows: First, we compute the error rate for the graph model, defined as the number of misclassified edges over the total number of training edges. If the error rate is greater than zero, we begin an iterative correction loop. For each misclassified edge, the model has failed to identify the true weight, indicating its signal is too weak. We therefore reinforce this correct signal by additively bundling its vector representation (the target node bound with the true weight) into the source node's memory. This procedure only ever adds correct information, ensuring that no valid signals are ever removed or diminished.

This additive update is performed for all misclassified edges in the training set. Afterward, the error rate is recomputed. The entire process is repeated iteratively until the error rate converges to zero, a maximum number of retraining iterations is reached, or the process stops early if the error rate at the current iteration is greater than the rate at the previous one. This leads to a progressive refinement of the model's memory by strengthening the correct signals until they are reliably reconstructed.

The process of retraining is summarized in Algorithm 2.

The complete methodology, from the initial graph construction using Algorithm 1, through model refinement with Algorithm 2, to the final prediction on new data, is illustrated as a comprehensive workflow in Fig. 1.

Results

This section presents the findings of our study on using the HDC-based graph encoding for representing viral pangenomes with the aim to assess the efficacy and advantages of this novel approach in capturing the complex relationships and genomic diversity of all the viral species known so far.

Viral pangenome as weighted *de Bruijn* graph

Pangenomes are typically represented as graphs where nodes correspond to genomic elements and edges represent their adjacency or shared sequences [30]. Here, we constructed a *de Bruijn* graph to represent the collective genetic information of the training set of 542 viral genomes. A *de Bruijn* graph is a directed graph where nodes represent sequences of a fixed length, called k-mers, and directed edges connect two k-mers if the suffix of the first k-mer matches the prefix of the second k-mer, indicating an overlap

Algorithm 2 Iterative retraining of the VSA graph model. This algorithm describes the supervised, one-sided learning process used to refine the graph hypervector G . The goal is to minimize classification errors on the training set by selectively reinforcing correct signals without corrupting existing valid information, a crucial step for multi-label graphs. The process begins by calculating the initial error rate, defined as the proportion of misclassified edges (Lines 8–14). An edge (v_i, v_j) is considered misclassified if the model's prediction for its weight w_{pred} does not match the true weight w_{true} . The algorithm then enters an iterative loop that continues until the error rate is zero, stops improving, or a maximum number of iterations is reached. The core of the algorithm is the correction step (Lines 20–27). For every edge identified as a false negative, the algorithm computes a reinforced signal. This signal is the vector representation of the correct connection that was too weak to be reconstructed. This corrective signal is then additively applied to the global graph model. This one-sided approach ensures that only correct information is strengthened, preventing the accidental removal of other valid weights that may exist on the same edge. The process repeats, progressively refining the model's accuracy by amplifying the correct signals until they are reliably reconstructed.

Input: (i) the initial global graph hypervector G from Algorithm 1; (ii) the item memory of node hypervectors H ; (iii) the set of training edges $E_{train} = (v_i, v_j, w_{true})$, where w_{true} is the true weight; (iv) the maximum number of retraining iterations $max_iterations$.

Output: the refined global graph hypervector $G_{retrained}$.

```

1. # Initialize
2.  $G_{retrained} \leftarrow G$ 
3.  $previous\_error\_rate \leftarrow \infty$ 
4.  $iteration \leftarrow 0$ 
5. # Start retraining loop
6. while  $iteration < max\_iterations$ 
7.   # Calculate error rate:
8.    $misclassified\_edges \leftarrow []$ 
9.    $total\_edges \leftarrow length(E_{train})$ 
10.  for each edge  $(v_i, v_j, w_{true})$  in  $E_{train}$ 
11.     $w_{pred} \leftarrow PredictWeight(v_i, v_j, G_{retrained}, H)$  # Predict the weight of the edge
12.    if  $w_{pred} \neq w_{true}$ 
13.      Add  $v_i, v_j, w_{true}$  to  $misclassified\_edges$ 
14.     $current\_error\_rate \leftarrow length(misclassified\_edges)/total\_edges$ 
15.    # Check stopping condition:
16.    if  $current\_error\_rate = 0$  or  $current\_error\_rate \geq previous\_error\_rate$ :
17.      break the loop
18.     $previous\_error\_rate \leftarrow current\_error\_rate$ 
19.    # Apply correction (reinforce false negatives):
20.    for each misclassified edge  $(v_i, v_j, w_{true})$  in  $misclassified\_edges$ 
21.       $V_{w_{true}} \leftarrow EncodeWeight(w_{true}, D)$ 
22.      # Define the connection vector that needs to be reinforced
23.       $\Delta M_i \leftarrow V_{w_{true}} \otimes H[v_j]$ 
24.      # Calculate the corresponding change for the global graph
25.       $\Delta G \leftarrow H[v_i] \otimes \rho(\Delta M_i)$ 
26.      # Apply the correction to the current graph model
27.       $G_{retrained} \leftarrow G_{retrained} \oplus \Delta G$ 
28.    # Increment iteration:
29.     $iteration \leftarrow iteration + 1$ 
30. return  $G_{retrained}$ 

```

of length $k - 1$. For our purposes, k -mers were extracted from the complete genome sequences of the viral species in our dataset. Specifically, for each viral genome, we slid a window of size k along its entire length, extracting every possible subsequence of length k . This process generates a comprehensive set of k -mers that collectively capture the genomic content and its variations. The choice of k is crucial, as a smaller k leads to more connections and potentially a more fragmented graph, while a larger k can resolve more complex regions but may lead to a sparser graph. Here, we chose $k = 9$ as previously described, and k -mers are extracted sequentially and kept in the order of their appearance in the original sequence.

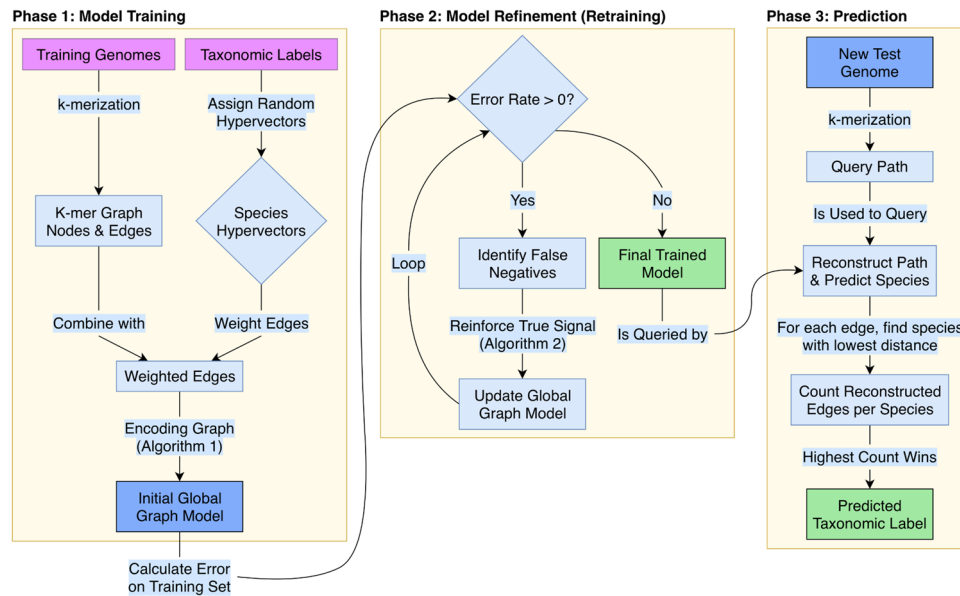


Fig. 1 Overview of the VSA-based pangenome analysis workflow. The flowchart illustrates the three main phases of the methodology. *Phase 1: Model Training* – training genomes and their corresponding taxonomic labels are used to construct an initial weighted *de Bruijn* graph. The labels are converted into unique species hypervectors which color the graph's edges. This entire structure is then encoded into a single global hypervector following the procedure in Algorithm 1; *Phase 2: Model Refinement (Retraining)* – the initial model undergoes an iterative retraining loop where false negatives from the training set are identified. The signals for these correct but missed connections are then reinforced in the model as described in Algorithm 2, producing a final, optimized model; *Phase 3: Prediction* – a new, unseen test genome is decomposed into a query path of k-mers. This path is used to probe the final model, and the system predicts the species label that allows for the highest number of edges to be successfully reconstructed from the graph

The graph structure is defined by k-mer adjacency. A directed edge connects two k-mer nodes if they appear consecutively in a viral genome. Additionally, to encode the biological origin of each connection, we assign a unique species hypervector to each of the 542 viral species. Thus, edges are labeled with these specific hypervectors, which act as edge weights representing different taxonomic labels.

K-mers space reduction with minimizers

To manage the computational complexity associated with handling all possible k-mers, especially in large pangenomes, we employed the concept of minimizers [35]. A minimizer is a subsequence of length k that is chosen as a representative of a set of k-mers within a specific window of length w , with $w > k$. The representative k-mer is established by selecting the first k-mer in the set of alphabetically ordered k-mers of length k within the window of length w . The process is iteratively repeated by scrolling a window of the same length w nucleotide-by-nucleotide over the whole genome. This approach significantly reduces the number of k-mers that need to be processed and stored while still retaining sufficient genomic information to accurately represent the input sequences. Here, we extracted the list of minimizers in the order they appear in the original sequences by choosing a window of length $w = 50$, while maintaining a k-mer size $k = 9$. This specific window size was deliberately selected to balance topological resolution with the strict superposition limits of the Vector-Symbolic Architecture. An excessively small window would generate a dense graph that risks overwhelming the VSA's

memory capacity via signal interference, while a significantly larger window would risk entirely skipping short, discriminative viral sequences.

Problem formulation and classification results

We designed a VSA to encode viral genetic information into a graph data structure with the aim of assigning species labels to given viral genome sequences. Formally, given a test genome g_{test} and a set of $C = 542$ possible species classes encoded in the training set, the goal is to predict the correct label $c_{pred} \in \{S_1, S_2, \dots, S_C\}$. It should be noted that the current algorithm operates as a closed-set classifier, meaning it must assign one of the existing labels in the model. Our approach frames this as a query operation against a holistic, high-dimensional knowledge base. The pangenome, constructed from a training set of genomes, is encoded into a single hypervector G that acts as an associative memory. The classification of a new genome is then treated as a process of querying this memory to retrieve the most closely associated species label. The query itself is a hypervector (as in item memory) derived from the test genome's sequence of minimizers. The success of this approach mostly relies on the ability of the VSA model to (i) encode the complex relational structure of the multi-species pangenome graph into a single vector G , and (ii) faithfully reconstruct the species information (the edge weights) when presented with a query path corresponding to a test genome.

In order to test our model's classification power, we initially partitioned our set of viral genomes into a training and test set with 1 genome per species (i.e., 542 genomes) in each partition. The set of consecutive minimizers in the training set compose the edges that we encoded and collapsed into a single vector representation of the pangenome graph as previously explained. On the other hand, the classification of a test genome proceeds through the following steps:

1. *Query generation*: the test genome is first decomposed into its ordered sequence of minimizers (using the same $k = 9$ and $w = 50$ parameters) to form a graph representation of the test genome. We emphasize that these minimizer parameters, along with the hypervector dimensionality ($D = 50,000$), were kept strictly fixed across all experiments (species-level, genus-level, and hierarchical) to ensure a fair and consistent comparison of the architectural approaches;
2. *Memory probing*: each of the edges that form the graph representation of the test genome is used to probe the main pangenome hypervector G . Given an edge $\langle A, B, Sp \rangle$ where A is the source node, B is the target node, and Sp is a viral species, this is achieved by unbinding the vector representation of the source node A from the hypervector G to retrieve a new vector (the node memory) that contains information about its neighbors in the pangenome graph;
3. *Label prediction*: to predict the species of a test genome g_{test} , we determine which species hypervector (representing the taxonomic label) is most successfully reconstructed along the genome's path in our graph model. This is done by checking for the presence of each edge from the test genome, one species at a time. For a given edge $\langle A, B \rangle$ in the test genome, we test the hypothesis that it belongs to species Sp . To do this, the vector representation of the target node B is bound with the vector representation of the species Sp . We then search for this resulting vector within node A 's memory by computing the cosine distance. It should be noted that a simple "close to 0" cosine distance is insufficient for confirming the presence of

an edge in a node's memory due to varying noise levels across the graph. A node's memory is noisier if it has many bundled connections, making the reconstruction attempts more difficult. Therefore, we apply a node-dependent distance threshold that is estimated automatically for each source node and weight. This is calculated as follows: for each node A in our model, we query its memory for a set of *positive controls* (i.e., nodes we know it connects to) and an equal number of *negative controls* (i.e., nodes we know it does not connect to). This generates a distribution of distances for both true and false connections. We then set the specific threshold for node A as the 5th percentile of this combined distance distribution. An edge $\langle A, B, Sp \rangle$ is considered reconstructed for the species Sp only if the query distance falls below node A 's unique threshold. This entire process is repeated for every possible species $Sp \in \{S_1, S_2, \dots, S_C\}$ and for every edge in the test genome g_{test} . The species that accumulates the highest number of successfully reconstructed edges across the entire test genome is chosen as the final prediction.

Following the encoding strategy previously discussed, we built a graph model comprising 44,183 nodes (i.e., unique minimizers) and 612,665 edges in approximately 3 min using a vector dimensionality of 50,000. The entire pipeline was developed in Python 3, and the computation was performed on a server equipped with a single Intel Xeon E5-2680 v4 Central Processing Unit (CPU) (14 cores/28 threads) running @ 2.40 GHz running CentOS Linux 7. The process utilized a maximum of 24 threads and approximately 90 GB of Random Access Memory (RAM).

We performed a retraining process as previously described, converging to a final error rate of ~5% after 6 iterations.

After training, the final graph model was evaluated on the independent test set, which contained 542 genomes (one for each species). Of these, 472 (87.08%) genomes were correctly assigned to their respective species, resulting in an overall average reconstruction rate of 44.2%.

A key feature of our method is the reconstruction rate. We explicitly define this rate as a model-internal coherence measure, capturing the percentage of a test genome's edges that were successfully reconstructed from the graph model under the predicted species label. Because this rate is calculated as a percentage of the total edges in the query genome's path, it inherently normalizes for varying genome lengths, ensuring that the confidence metric is comparable across exceptionally small and large viral genomes. We observed a strong correlation between this reconstruction rate and the accuracy of the prediction. Among the 472 correctly classified genomes, 162 of them showed a high degree of confidence (reconstruction rate > 70%), with an average reconstruction rate of 85% (with a standard deviation at 8.4%). Conversely, the 70 misclassified genomes had a significantly average reconstruction rate of only 8.6% (with a standard deviation at 11.6%). These results are summarized in Fig. 2.

Figure 2 presents the classification performance for each test genome, showing an overview of the model's behavior. The genomes are sorted by their reconstruction rate, which represents the percentage of the genome's edges successfully reconstructed from the pangenome graph. The results reveal a complex relationship between reconstruction rate, which acts as a confidence score, and classification accuracy.

The plot is dominated by a large number of correctly classified genomes (black bars). Importantly, these correct predictions span a very wide range of reconstruction rates,

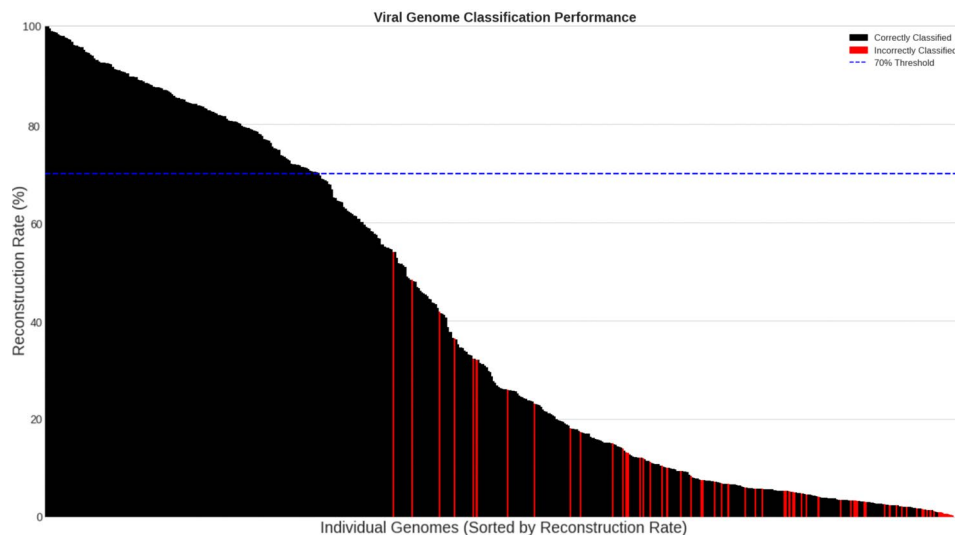


Fig. 2 Viral genome classification performance and reconstruction rate. The figure displays the classification results of each individual test genome, sorted in descending order based on their reconstruction rate. Each vertical bar represents a single genome. Bars are colored black if the species was correctly classified, and red if it was misclassified. The dashed blue line indicates a 70% reconstruction rate threshold

with many falling well below 70%. This demonstrates the model's robustness. Even when a genome's edges are only partially represented in the pangenome graph (due to novel accessory genes or strain-level variation), the signal for the correct species is still significantly stronger than for any incorrect one.

This shows that the reconstruction rate functions effectively as an indicator of model coherence rather than a simple binary measure of success. A prediction with a 65% reconstruction rate can be interpreted as the model was not able to find a perfect match for the entire genome, but a significant portion of its edges point most strongly to its actual taxonomic label.

On the other hand, as expected, the frequency of misclassifications (red bars) begins to increase when getting into the low-confidence region, i.e., when the reconstruction rate drops below 50%.

Further analysis of the misclassifications revealed that they predominantly occurred between taxonomically related species despite their low reconstruction rate. It is the case of the *Gammmapapillomavirus* 14 and 10 species, *Alphapapillomavirus* 9 and 11 species, and other 9 genera that were confused for one another as reported in Table 1, potentially due to their high sequence homology and shared genomic regions.

A detailed breakdown of the species-level classification performance is provided in Supplementary Table S3, which lists the accession number, the ground-truth taxonomic label from NCBI GenBank, the label predicted by our model, and the corresponding reconstruction rate for each test genome.

The challenge of genus-level pangenome signatures

In biological taxonomy, a genus is a higher-level grouping that encompasses one or more closely related species. Moving from species-level to genus-level labels therefore reduces granularity, effectively merging related species into a broader category. Given that a majority of the species-level misclassifications occurred between closely related viruses, we hypothesized that reducing the taxonomic granularity to the genus level

Table 1 Representative classification examples from the test set, illustrating the high reconstruction rates for correct predictions versus the lower rates for incorrect ones

Genome	True Taxonomic Unit	Predicted Taxonomic Unit	Recon- struction Rate
GCA_004288535.1	<i>Gammapapillomavirus_14</i> (species)	<i>Gammapapillomavirus_10</i> (species)	3.56%
GCA_003177875.1	<i>Alphapapillomavirus_9</i> (species)	<i>Alphapapillomavirus_11</i> (species)	7.57%
GCA_001461525.1	<i>Iotapapillomavirus</i> (genus)	<i>Gammapapillomavirus</i> (genus)	12.13%
GCA_004788075.1	<i>Mischivirus</i> (genus)	<i>Megrivirus</i> (genus)	4.85%
GCA_013087405.1	<i>Potamipivirus</i> (genus)	<i>Cosavirus</i> (genus)	3.31%
GCA_004289455.1	<i>Gammapapillomavirus</i> (genus)	<i>Deltapapillomavirus</i> (genus)	2.08%
GCA_023122835.1	<i>Jeilongvirus</i> (genus)	<i>Paraavulavirus</i> (genus)	8.40%
GCA_003178355.1	<i>Betapapillomavirus</i> (genus)	<i>Alphapapillomavirus</i> (genus)	4.14%
GCA_000892175.1	<i>Betapapillomavirus</i> (genus)	<i>Alphapapillomavirus</i> (genus)	3.41%
GCA_000973295.1	<i>Genomoviridae</i> (genus)	<i>Gemycircularvirus</i> (genus)	0.98%
GCA_002817335.1	<i>Rosavirus</i> (genus)	<i>Megrivirus</i> (genus)	3.38%

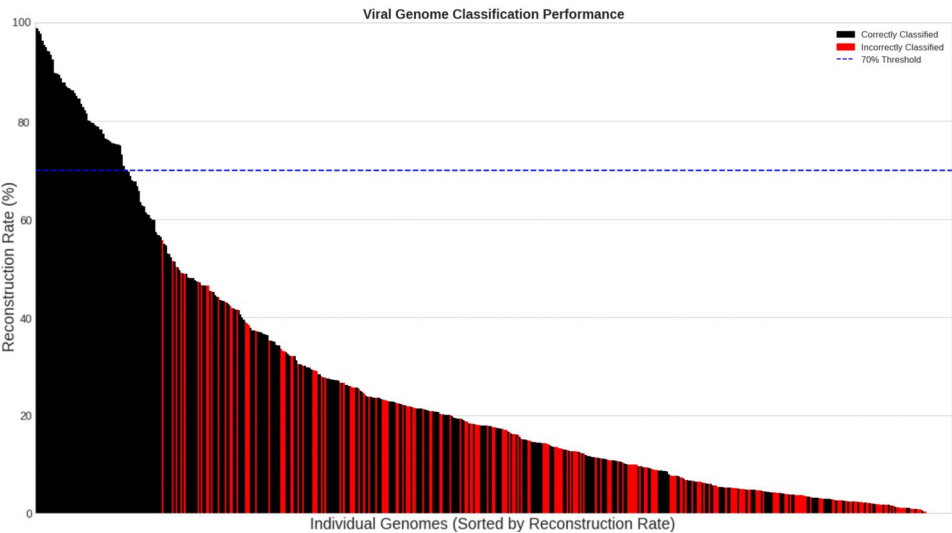


Fig. 3 Genus-level classification performance. The figure shows the classification results aggregated at the genus level. Each bar represents a single test genome, sorted in descending order by reconstruction rate. Bars are colored black if the genome was assigned to the correct genus, and red if it was misclassified. The dashed blue line indicates the 70% reconstruction rate threshold

would resolve these ambiguities and improve overall performance. The expectation was that a genus signature would be a more robust and easier target for the model to learn.

To investigate this, we repeated the analysis by collapsing the taxonomic labels from species to genus, resulting in 218 unique genera for the classification task. Contrary to our initial hypothesis, this approach led to a significant decrease in classification performance. The model correctly classified 328 out of 542 genomes, yielding an overall accuracy of 60.51%, a notable drop from the 87.08% achieved at the species level.

The degradation in performance is visually apparent in Fig. 3. Compared to the species-level results (Fig. 2), the genus-level analysis shows a much larger proportion of misclassified genomes (red bars) and the rate of correct classifications falls off much more steeply. This result suggests that aggregating diverse species into a single genus label can be detrimental to the model’s performance due to signal dilution. While species within a genus are related, they can still possess significant genomic diversity, including distinct

accessory genes and structural variations. By assigning the same genus label to all edges from these varied species, we effectively create a composite signal that is less coherent to find a clear, representative path for a test genome within this pangenome. This indicates that for our VSA-based approach, highly specific labels are more effective.

Detailed results for the genus-level classification, including accession numbers, true and predicted genera, and reconstruction rates for each test genome, are provided in Supplementary Table S4.

Hierarchical classification with genus-level models

The previous analysis revealed a critical trade-off. The global, species-level model struggled with ambiguity between closely related species, while the genus-level model suffered from signal dilution by averaging diverse species into a single noisy label. This suggests that the problem lies not with the taxonomic rank itself, but with the “one-size-fits-all” structure of a single, monolithic pangenome graph.

We therefore hypothesized that a divide-and-conquer strategy would yield better results. By creating specialized, more focused pangenome graphs for each genus, we could resolve ambiguity without losing signal integrity. This led to a two-step hierarchical classification framework. It is important to note that while this architecture ultimately resulted in lower accuracy than the flat model, its inclusion provides critical insights into the vulnerability of VSA models to signal dilution and hard-routing error propagation.

First, we constructed a separate pangenome graph model for each viral genus that contained multiple species. Each genus-specific graph was built exclusively from the genomes belonging to that genus but still retained the individual species labels as edge weights.

The classification of a test genome then proceeds in two stages:

1. *Genus prediction*: the test genome is queried against every genus-specific graph. The genus whose graph yields the highest reconstruction rate for the test genome's path is selected as the predicted genus;
2. *Species prediction*: once the genus is identified, the species prediction is performed only within that genus's specialized graph. This drastically reduces the search space and eliminates the possibility of confusion with species from other genera.

This hierarchical strategy produced a surprising and informative result. While we observed instances of exceptionally high local reconstruction rates during the initial genus prediction step (e.g., >80% as detailed in Table 2), demonstrating that smaller, homologous graphs can yield highly coherent local matches, this localized confidence proved deceptive. The final species-level classification accuracy decreased significantly, dropping to 33.57%. This seemingly paradoxical outcome reveals a critical bottleneck in the hierarchical approach: error propagation from the initial genus prediction step. Specifically, the intermediate classification accuracy for this first genus-routing stage was only 35.42%. Because any misrouting at the genus level guarantees a subsequent species-level failure, the 33.57% terminal accuracy inherently captures the strict penalty introduced by this initial routing step.

Table 2 Intra-genus misclassification in the high-confidence hierarchical model. The table details the eight instances where the final species prediction was incorrect, despite the model first correctly identifying the genus with a high reconstruction rate (> 80%). These errors exclusively occur between similar species, highlighting the model's final challenge in resolving ambiguity at the lowest taxonomic levels

Genome	True genus	True species	Predicted species	Reconstruction rate
GCA_001717415.1	<i>Ranavirus</i>	<i>Frog virus 3</i>	<i>European North Atlantic ranavirus</i>	85.06%
GCA_006401535.1	<i>Mastadenovirus</i>	<i>Human mastadenovirus 8</i>	<i>Human adenovirus sp</i>	96.92%
GCA_009738835.1	<i>Aviadenovirus</i>	<i>Fowl aviadenovirus D</i>	<i>Fowl adenovirus</i>	84.20%
GCA_025023725.1	<i>Aviadenovirus</i>	<i>Fowl adenovirus</i>	<i>Fowl aviadenovirus C</i>	95.05%
GCA_004029355.1	<i>Circovirus</i>	<i>Porcine circovirus 3</i>	<i>Porcine circovirus</i>	96.62%
GCA_004086635.1	<i>Circovirus</i>	<i>Porcine circovirus type 1 2a</i>	<i>European catfish circovirus</i>	97.99%
GCA_004041475.1	<i>Circovirus</i>	<i>Circovirus sp</i>	<i>European catfish circovirus</i>	92.02%
GCA_004033655.1	<i>Circovirus</i>	<i>Porcine circovirus 1</i>	<i>Duck circovirus</i>	96.12%

A second, more subtle challenge emerged even when the first step was successful. Among the genomes where the correct genus was identified with high confidence (reconstruction rate > 80%), there were still eight final species-level misclassifications. As shown in Table 2, these errors exclusively occurred between extremely similar species within the correctly identified genus.

Additionally, the model frequently assigned a test genome to the wrong genus with high confidence (high reconstruction rate). For instance, a genome from the species *Goatpox virus* belonging to the genus *Capripoxvirus* might have a path that is ~78% reconstructed within the *Varicellovirus* genus graph. The model would then confidently, but incorrectly, select *Varicellovirus* as the predicted genus. The subsequent species prediction, now operating within the wrong context, is doomed to fail. This demonstrates that while specialized models are powerful, their effectiveness is entirely dependent on correctly routing the query to the right model in the first place.

This experiment, despite lowering overall accuracy, provides a crucial insight: the primary challenge lies in accurately discriminating genera that share conserved genomic backbones. Results are summarized in Fig. 4, while Table 3 summarizes the performance across the three different classification approaches.

To provide a granular view of the hierarchical model's performance, Supplementary Table S5 documents the outcome for each test genome. It lists the accession number, the true species label, the final predicted species, and the reconstruction rate that informed the initial genus-level assignment.

Discussion and conclusions

This work introduces a novel approach to viral pangenome-based species classification based on a novel graph encoding technique grounded into the Hyperdimensional Computing paradigm. Pangenomes are indeed represented as weighted *de Bruijn* graphs encoded into high-dimensional vectors. We tested our method with all the viral reference genomes in NCBI GenBank, producing a vector representation of the viral pangenome graph comprising 1,084 genomes distributed over 542 species, keeping 1 genome for each species out of the training step. These last genomes have been then used to test for the accuracy of our model in retrieving genome information from the graph hypervector.

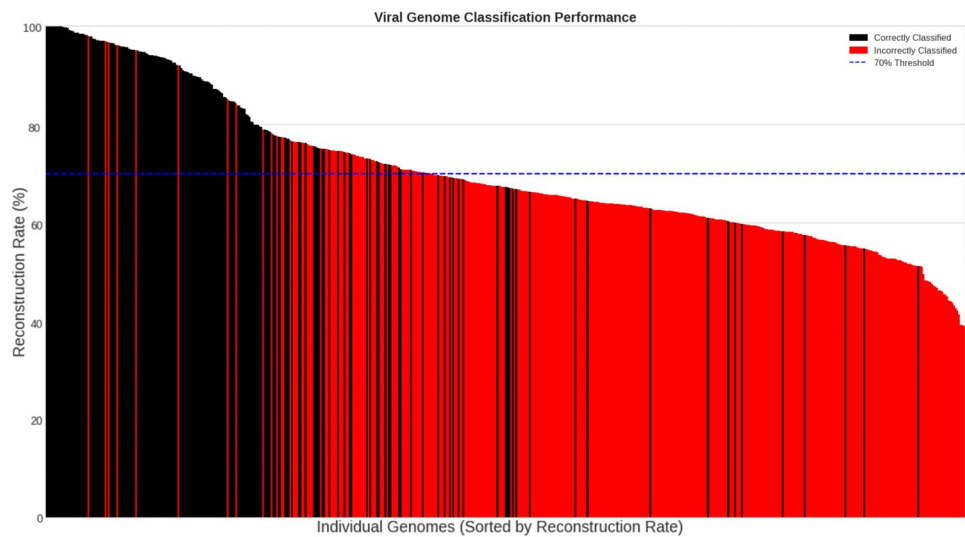


Fig. 4 Performance of the hierarchical classification model. The figure displays the final species-level classification results from the two-step hierarchical model. Each bar represents a single test genome, sorted in descending order by the reconstruction rate achieved during the initial genus-prediction step. Bars are colored red if the final prediction was correct and black if it was misclassified. The dashed blue line indicates a 70% reconstruction rate threshold

Table 3 Comparison of classification accuracy across the three different experimental strategies with the hierarchical approach showing the average species-level accuracy and reconstruction rate

Classification strategy	Target level	Model structure	Overall accuracy	Average reconstruction rate
Flat classification	Species	Single global graph	87.08%	39.60%
Flat classification	Genus	Single global graph	60.51%	25.45%
Hierarchical classification	Species	One graph per genus	33.57%	70.32%

Model performance and the challenge of taxonomic granularity

Our results revealed a complex and counter-intuitive landscape. The most straightforward approach, a single global graph with species-level labels, yielded the highest classification accuracy at 87.08%. While this represents a strong performance for a novel method, it is crucial to place this result in the context of the current methodological landscape. A direct benchmark is challenging since our VSA-based approach represents a new paradigm for using pangenomes as a direct classification engine. While we have provided an empirical baseline comparison with Mash, a traditional alignment-free MinHash method, in the Supplementary Material to contextualize our accuracy, a direct side-by-side comparison is inherently unbalanced due to the differing constraints of these paradigms. The most relevant comparison is with alignment-free distance estimation tools that typically operate on the Jaccard similarity of k-mer sets [36–38]. In this context, our method explicitly encodes the graph of k-mer adjacencies, thereby retaining sequential and structural information that is discarded by set-based approaches. Therefore, our 87.08% accuracy is significant not only for its competitive performance but also for the novel, graph-centric methodology used to achieve it. It demonstrates the viability of using holographic pangenome representations as a fast and accurate alternative to both computationally intensive alignment-based methods and existing set-based sketching techniques.

This strong baseline performance makes the subsequent results even more informative. Surprisingly, attempts to simplify the problem by aggregating labels at the genus level, or to improve it with a hierarchical model, led to significantly worse performance. The flat genus-level model suffered from signal dilution, where the diversity within genera created noisy, incoherent labels, dropping the accuracy to 60.51%.

Most informatively, the hierarchical model failed due to downstream error propagation, achieving the lowest accuracy (33.57%) despite having the highest average reconstruction rate (70.32%). It is important to explicitly frame these hierarchical results as an exploration of architectural limitations under our current hard-routing strategy, rather than a definitive statement about hierarchical classification in general. Because our model forces a strict path, early routing errors permanently trap the model in an incorrect sub-tree, conflating genus-prediction error with intra-genus genomic similarity. This demonstrates a critical finding: the model's reconstruction rate is a measure of its internal coherence or support, not a surrogate for its correctness. A high reconstruction rate can be misleading under certain conditions, such as when a queried genome shares highly conserved genomic backbones or homologous regions with closely related taxa. In such cases, the model may encounter a coherent but incorrect routing (i.e., making a high-reconstruction but incorrect genus prediction in the first step), which inevitably leads to a subsequent incorrect species-level classification. This reveals that the primary challenge for this architecture is not resolving ambiguity between the most closely related species, but accurately discriminating between the genomic backbones of different genera.

Biological context of misclassified patterns

Although our VSA-based approach is primarily methodological, placing the misclassification patterns within their biological context is essential for interpreting the model's true efficacy. Viral genomes are characterized by rapid evolution, frequent recombination, and mosaic architectures, which often combine highly conserved genomic backbones with rapidly evolving accessory regions. These properties naturally blur taxonomic boundaries, even when species-level signals seem otherwise distinct.

A closer qualitative examination of the misclassifications in Tables 1 and 2 reveals that many occur within viral families known for profound taxonomic instability and biological ambiguity. For instance, a significant portion of genus- and species-level confusions involved *Papillomaviridae* (e.g., *Alphapapillomavirus*, *Betapapillomavirus*, and *Gammapapillomavirus*). Papillomaviruses exhibit extensive sequence homology in their core genes (such as L1 and L2) [39, 40] alongside divergent accessory regions [41]. Consequently, their taxonomy has been subject to frequent revisions by the International Committee on Taxonomy of Viruses (ICTV) [42] due to these blurry genetic boundaries [43]. Similarly, our model occasionally confused genera within the *Picornaviridae* family (e.g., *Mischivirus*, *Megrivirus*, and *Cosavirus*), which are RNA viruses notorious for high mutation and recombination rates [44, 45].

Furthermore, intra-genus misclassifications within the hierarchical model (Table 2) predominantly affected groups like *Circovirus* and *Aviadenovirus*. Circoviruses possess exceptionally small, highly recombinogenic genomes where frequent cross-species transmission and mosaicism make discrete species delineation inherently difficult [46,

47]. Adenoviruses are likewise known for frequent homologous recombination that complicates clear species demarcation [48, 49].

Therefore, many of the VSA model's errors likely reflect genuine biological ambiguity, where a test genome truly shares a massive proportion of its k-mer path with multiple reference species due to shared genomic backbones or recent recombination events, rather than a fundamental algorithmic failure. This highlights an inherent challenge in viral classification: hard boundaries imposed by human taxonomy often fail to seamlessly capture the continuous and mosaic nature of viral evolution.

Future directions

A critical next step for this framework is transitioning from closed-set classification to open-set anomaly detection. In its current form, if the model is queried with a novel species not present in the training set, it will assign the genome to its closest encoded relative based on shared k-mers in conserved homologous regions. However, our results demonstrate that the reconstruction rate can effectively serve as an anomaly detection threshold. Because a novel genome would lack a perfectly matching sequence of edge weights, it would yield a severely degraded reconstruction rate. By defining a strict confidence threshold, the system could flag these low-confidence predictions as “unknown” or “novel” entities, allowing the VSA framework to dynamically alert researchers to new emerging species that require inclusion in the pangenome graph.

Similarly, the classification of recombinant viral strains poses an ongoing challenge. Under the current methodology, which selects a single consensus label via an argmax operation across the entire genome path, a recombinant sequence is classified as whichever parent species contributes the strongest aggregate signal. Consequently, such mixed signals also manifest as notably reduced reconstruction rates. However, a major theoretical advantage of Vector-Symbolic Architectures is their capacity for superposition. Because an aggregate query hypervector natively stores the superimposed signals of multiple species, future decoding algorithms could be adjusted to perform multi-label classification, identifying all taxonomic labels that exceed a detection threshold, or modified to employ a sliding-window approach that detects distinct recombination break-points along the sequence path.

While this work presents a framework with significant potential, our results guide future directions toward solving these newly identified challenges. The immediate focus should be on improving the initial genus-prediction step of the hierarchical model, perhaps by incorporating different encoding strategies or features that are more robust to inter-genus homology. Furthermore, investigating methods to counteract signal dilution in global models could make them more effective for broader taxonomic ranks.

A highly promising avenue for resolving the signal dilution observed in our genus-level models is the isolation of core genomic signatures. Currently, our genus-level graph encodes the entirety of the pangenome, inextricably linking conserved core genes with highly variable accessory genes. Grouping diverse accessory elements under a single genus label creates a noisy, composite signal that degrades classification accuracy. Future iterations of this framework will investigate constructing specialized genus-level anchor graphs utilizing exclusively core, anchor-level k-mers derived from an expanded training set of multiple strains per species. By mathematically filtering out accessory noise and mapping only the conserved evolutionary backbones, we anticipate a drastic

improvement in the robustness of flat genus classification and the reliability of initial hierarchical routing.

Additionally, expanding the pangenome to a wider range of viral species, including those present in NCBI for which no reference genomes are currently available and those yet to be discovered [50, 51], is crucial for broadening the applicability of this method. In this regard, we are indeed planning to produce and maintain different models built over genomes belonging to different microbial domains. The use of minimizers to be considered over the classical definition of k-mers could also lead to more succinct vector representations of the pangenome graphs with potentially comparable classification performance. Beyond classification, we will systematically adapt and benchmark our method as a genome-assembly engine with the aim of reconstructing microbial genomes from metagenomic samples.

Ultimately, this work paves the way for targeted hardware acceleration of bioinformatics systems. Leveraging the lightweight and parallelizable arithmetic of HDC, future designs could implement system-on-chip architectures optimized for genomic graph processing. Such low-cost, energy-efficient hardware would enable scalable genomic search analyses and support the long-term vision of continuous, high-throughput genomic surveillance systems powered by dedicated HDC processors.

Broader impact and potential applications

The potential applications of this method extend beyond the simple classification. Integrating this approach with real-time viral surveillance systems could enable rapid identification and characterization of emerging viral threats [52]. Applying this method to outbreak analysis could provide valuable insights into the transmission dynamics and evolutionary trajectories of viruses, aiding in the development of targeted interventions [53]. Furthermore, the scalability of this approach opens up possibilities for analyzing large metagenomic datasets, potentially uncovering novel viral diversity in different environments [54]. In addition to these applications, we are releasing the full implementation of our framework as open-source software (see Availability section below). This includes both the complete pipeline for pangenome-based classification and a general-purpose HDC operation library. We aim to maximize the broader impact across multiple fields by making the source code accessible as open-source. For biomedical research, the code provides a reproducible and extendable foundation for future studies in viral genomics and metagenomics. For the hyperdimensional computing community, it offers a benchmark-ready implementation that bridges theory with practice. This joint contribution creates a “melting pot” between computational biology and emerging computing paradigms, filling an important gap between the two disciplines and opening opportunities for cross-disciplinary innovation.

Conclusions

Here, we introduced and evaluated a novel VSA-based framework for viral species classification. Our findings demonstrate that a global pangenome graph with species-specific labels can achieve high classification accuracy, validating the potential of this alignment-free approach. However, our most significant contribution lies in uncovering the limitations and nuances of this architecture. We show that attempts to simplify or hierarchically structure the classification problem under our current hard-routing

strategy can degrade performance. This degradation reflects the interaction between initial routing errors and downstream error propagation rather than an inherent limitation of hierarchical modeling itself, suggesting that future soft-routing or multi-level consensus VSA approaches could effectively mitigate these specific architectural bottlenecks. This highlights a critical insight: for VSA-based pangenome analysis, the most direct and specific representation is the most effective, and the model's reconstruction rate serves as a measure of internal coherence, rather than an absolute probability of correctness, that must always be interpreted in its biological context. Ultimately, this study provides both a promising new tool for scalable genomics and a clear roadmap for the future research required to overcome the inherent challenges of classifying viruses.

Abbreviations

CPU	Central Processing Unit
HDC	Hyperdimensional Computing
MAP	Multiply-Add-Permute
NCBI	National Center for Biotechnology Information
PyPI	Python Package Index
RAM	Random Access Memory
VSA	Vector-Symbolic Architecture

Supplementary Information

The online version contains supplementary material available at <https://doi.org/10.1186/s13040-026-00561-1>.

Supplementary Material 1

Supplementary Material 2

Supplementary Material 3

Supplementary Material 4

Supplementary Material 5

Acknowledgements

We would like to acknowledge the use of AI in refining the clarity and readability of this manuscript. The AI assistance was primarily used for tasks such as sentence restructuring, word choice suggestions, and identifying potentially unclear phrasing. We emphasize that the AI was used solely for language enhancement and did not contribute to the generation of research ideas, data analysis, code implementation, or the interpretation of results. All conclusions drawn and insights presented in this manuscript are solely the product of the authors' own analysis and expertise.

Author contributions

FC and DB conceived the research; FC and SA designed the encoding and classification model; FC implemented the software and collected the reference genomes; FC and KD built and tested the graph-based model; FC, KD, JJ, and DC performed the analysis and discussed the results; DB supervised the research; FC, KD, JJ, DC, SA, and DB wrote the manuscript and agreed with its final version.

Funding

The authors declare that no funding was received for the conception or writing of this manuscript.

Data availability

The whole set of genomes used for training and testing our graph models are available on NCBI GenBank. The complete list of genomes with their accession numbers and full taxonomic label is available in the Supplementary Tables S2-S5. We integrated our encoding scheme, classification algorithm, and the analysis pipeline into *hdlib* [55]. (<https://paperpile.com/c/AqraL/upN9n>), an open-source Python 3 package available on the Python Package Index (PyPI – *pip install hdlib*) and Conda (*conda install -c conda-forge hdlib*). The entire pipeline is available on GitHub at (<https://github.com/cumbof/hdlib/tree/main/examples/pangenome>).

Declarations

Ethics, consent to participate, and consent to publish

Not applicable.

Competing interests

The authors declare no competing interests.

Received: 2 December 2025 / Accepted: 24 April 2026

Published online: 17 May 2026

References

1. Zuo K, Gao W, Wu Z, Zhang L, Wang J, Yuan X, et al. Evolution of Virology: Science History through Milestones and Technological Advancements. *Viruses*. 2024;16:374. <https://doi.org/10.3390/v16030374>.
2. Toward a global virus genomic surveillance network. *Cell Host Microbe*. 2023;31:861–73. <https://doi.org/10.1016/j.chom.2023.03.003>.
3. Downing T. Approaches to studying virus pangenome variation graphs. 2024. Available: <http://arxiv.org/abs/2412.05096>.
4. Pan-genomics of virus and its applications. Pan-genomics: applications, challenges, and future prospects. Academic Press; 2020. pp. 237–50. <https://doi.org/10.1016/B978-0-12-817076-2.00011-1>.
5. The Computational Pan-Genomics Consortium. Computational pan-genomics: status, promises and challenges. *Brief Bioinform*. 2016;19:118–35. <https://doi.org/10.1093/bib/bbw089>.
6. Bonnici V, Chicco D. Seven quick tips for gene-focused computational pangenomic analysis. *BioData Min*. 2024;17:1–14. <https://doi.org/10.1186/s13040-024-00380-2>.
7. Yang Z, Guarracino A, Biggs PJ, Black MA, Ismail N, Wold JR, et al. Pangenome graphs in infectious disease: a comprehensive genetic variation analysis of *Neisseria meningitidis* leveraging Oxford Nanopore long reads. *Front Genet*. 2023;14:1225248. <https://doi.org/10.3389/fgene.2023.1225248>.
8. Lau BT, Pavlichin D, Hooker AC, Almeda A, Shin G, Chen J, et al. Profiling SARS-CoV-2 mutation fingerprints that range from the viral pangenome to individual infection quasispecies. *Genome Med*. 2021;13:1–23. <https://doi.org/10.1186/s13073-021-00882-2>.
9. Wang Z, Jia L, Li J, Liu H, Liu D. Pan-Genomic Analysis of African Swine Fever Virus. *Virol Sin*. 2019;35:662–5. <https://doi.org/10.1007/s12250-019-00173-6>.
10. Downing T. Approaches to studying virus pangenome variation graphs. *arXiv*. 2025. <https://doi.org/10.48550/arXiv.2412.05096>.
11. Matthews CA, Watson-Haigh NS, Burton RA, Sheppard AE. A gentle introduction to pangenomics. *Brief Bioinform*. 2024;25:bbae588. <https://doi.org/10.1093/bib/bbae588>.
12. Gong Y, Li Y, Liu X, Ma Y, Jiang L. A review of the pangenome: how it affects our understanding of genomic variation, selection and breeding in domestic animals? *J Anim Sci Biotechnol*. 2023;14:1–19. <https://doi.org/10.1186/s40104-023-00860-1>.
13. Cui Y, Peng C, Xia Z, Yang C, Guo Y. A survey of sequence-to-graph mapping algorithms in the pangenome era. *Genome Biol*. 2025;26:1–25. <https://doi.org/10.1186/s13059-025-03606-6>.
14. Morrison DA. Multiple Sequence Alignment is not a Solved Problem. *arXiv*. 2018. <https://doi.org/10.48550/arXiv.1808.07717>.
15. Kanerva P. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognit Comput*. 2009;1:139–59. <https://doi.org/10.1007/s12559-009-9009-8>.
16. Kanerva P. Hyperdimensional computing: an algebra for computing with vectors. *Adv Semiconductor Technol*. Wiley; 2022. pp. 25–42. <https://doi.org/10.1002/9781119869610.ch2>.
17. Karunaratne G, Le Gallo M, Cherubini G, Benini L, Rahimi A, Sebastian A. In-memory hyperdimensional computing. *Nat Electron*. 2020;3:327–37. <https://doi.org/10.1038/s41928-020-0410-3>.
18. Cumbo F, Chicco D. Hyperdimensional computing in biomedical sciences: a brief review. *PeerJ Comput Sci*. 2025;11:e2885. <https://doi.org/10.7717/peerj-cs.2885>.
19. Stock M, Van Criekeing W, Boeckaerts D, Taelman S, Van Haevebeke M, Dewulf P, et al. Hyperdimensional computing: A fast, robust, and interpretable paradigm for biological data. *PLoS Comput Biol*. 2024;20:e1012426. <https://doi.org/10.1371/journal.pcbi.1012426>.
20. Chen H, Imani M. Density-aware parallel hyperdimensional genome sequence matching. [cited 25 Feb 2025]. Available: <https://doi.org/10.1109/FCCM53951.2022.9786145>.
21. Kim Y, Imani M, Moshiri N, Rosing T, GenieHD. Efficient DNA pattern matching accelerator using hyperdimensional computing. [cited 25 Feb 2025]. Available: <https://doi.org/10.23919/DAT48585.2020.9116397>.
22. Zou Z, Chen H, Poduval P, Kim Y, Imani M, Sadredini E, et al. BioHD: an efficient genome sequence search platform using HyperDimensional memorization. *Proceedings of the 49th Annual International Symposium on Computer Architecture*. ACM Digital Library; 2022. pp. 656–669. <https://doi.org/10.1145/3470496.3527422>.
23. Barkam HE, Yun S, Genssler PR, Zou Z, Liu C-K, Amrouh H, et al. HDGIM: hyperdimensional genome sequence matching on unreliable highly scaled FeFET. [cited 25 Feb 2025]. Available: <https://doi.org/10.23919/DAT56975.2023.10137331>.
24. Imani M, Nassar T, Rahimi A, Rosing T. HDNA: Energy-efficient DNA sequencing using hyperdimensional computing. [cited 25 Feb 2025]. Available: <https://doi.org/10.1109/BHI.2018.8333421>.
25. Cumbo F, Cappelli E, Weitschek E. A brain-inspired hyperdimensional computing approach for classifying massive DNA methylation data of cancer algorithms. 2020;13:233. <https://doi.org/10.3390/a13090233>.
26. Cumbo F, Truglia S, Weitschek E, Blankenberg D. Feature selection with vector-symbolic architectures: a case study on microbial profiles of shotgun metagenomic samples of colorectal cancer. *Brief Bioinform*. 2025;26:bbaf177. <https://doi.org/10.1093/bib/bbaf177>.
27. Ma D, Thapa R, Jiao X. MoleHD: efficient drug discovery using brain inspired hyperdimensional computing. [cited 25 Feb 2025]. Available: <https://doi.org/10.1109/BIBM55620.2022.9995708>.
28. Jones D, Zhang X, Bennion BJ, Ping S, Xu W, Kang J, et al. HDBind: encoding of molecular structure with hyperdimensional binary representations. *Sci Rep*. 2024;14:1–16. <https://doi.org/10.1038/s41598-024-80009-w>.
29. Vergés P, Heddes M, Nunes I, Kleyko D, Givargis T, Nicolau A. Classification using hyperdimensional computing: a review with comparative analysis. *Artif Intell Rev*. 2025;58:1–41. <https://doi.org/10.1007/s10462-025-11181-2>.
30. Andreade F, Lechat P, Dufresne Y, Chikhi R. Comparing methods for constructing and representing human pangenome graphs. *Genome Biol*. 2023;24:1–19. <https://doi.org/10.1186/s13059-023-03098-2>.
31. Sayers EW, Beck J, Bolton EE, Brister JR, Chan J, Connor R, et al. Database resources of the National Center for Biotechnology Information in 2025. *Nucleic Acids Res*. 2024;53:D20–9. <https://doi.org/10.1093/nar/gkaf979>.

32. Nayfach S, Camargo AP, Schulz F, Eloie-Fadrosch E, Roux S, Kyrpides NC. CheckV assesses the quality and completeness of metagenome-assembled viral genomes. *Nat Biotechnol*. 2021;39:578–85. <https://doi.org/10.1038/s41587-020-00774-7>.
33. Pornputtapong N, Acheampong DA, Patumcharoenpol P, Jenjaroenpun P, Wongsurawat T, Jun S-R, et al. KITSUNE: A Tool for Identifying Empirically Optimal K-mer Length for Alignment-Free Phylogenomic Analysis. *Front Bioeng Biotechnol*. 2020;8:556413. <https://doi.org/10.3389/fbioe.2020.556413>.
34. Poduval P, Alimohamadi H, Zakeri A, Imani F, Najafi MH, Givargis T, et al. GraphHD: Graph-Based Hyperdimensional Memorization for Brain-Like Cognitive Learning. *Front Neurosci*. 2022;16:757125. <https://doi.org/10.3389/fnins.2022.757125>.
35. Ndiaye M, Prieto-Baños S, Fitzgerald LM, Yazdizadeh Kharrazi A, Oreshkov S, Dessimoz C, et al. When less is more: sketching with minimizers in genomics. *Genome Biol*. 2024;25:1–35. <https://doi.org/10.1186/s13059-024-03414-4>.
36. Ondov BD, Treangen TJ, Melsted P, Mallonee AB, Bergman NH, Koren S, et al. Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biol*. 2016;17:132. <https://doi.org/10.1186/s13059-016-0997-x>.
37. Xu W, Hsu P-K, Moshiri N, Yu S, Rosing T. HyperGen: compact and efficient genome sketching using hyperdimensional vectors. *Bioinformatics*. 2024;40. <https://doi.org/10.1093/bioinformatics/btae452>.
38. Zhang T, Yin Z, Xu X, Yan L, Zhu F, Duan X, et al. RabbitSketch: a high-performance sketching library for genome analysis. *Bioinformatics*. 2025;41. <https://doi.org/10.1093/bioinformatics/btaf249>.
39. Bernard H-U, Burk RD, Chen Z, van Doorslaer K, zur Hausen H, de Villiers E-M. Classification of papillomaviruses (PVs) based on 189 PV types and proposal of taxonomic amendments. *Virology*. 2010;401:70–9. <https://doi.org/10.1016/j.virol.2010.02.002>.
40. de Villiers EM, Fauquet C, Broker TR, Bernard HU, zur Hausen H. Classification of papillomaviruses. *Virology*. 2004;324:17–27. <https://doi.org/10.1016/j.virol.2004.03.033>.
41. Doorbar J, Egawa N, Griffin H, Kranjec C, Murakami I. Human papillomavirus molecular biology and disease association. *Rev Med Virol*. 2015;25:2–23. <https://doi.org/10.1002/rmv.1822>.
42. Van Doorslaer K, Chen Z, Bernard H-U, Chan PKS, DeSalle R, Dillner J, et al. ICTV Virus Taxonomy Profile: Papillomaviridae. *J Gen Virol*. 2018;99:989–90. <https://doi.org/10.1099/jgv.0.001105>.
43. Bravo IG, de Sanjosé S, Gottschling M. The clinical importance of understanding the evolution of papillomaviruses. *Trends Microbiol*. 2010;18:432–8. <https://doi.org/10.1016/j.tim.2010.07.008>.
44. Zell R, Delwart E, Gorbalenya AE, Hovi T, King AMQ, Knowles NJ, et al. ICTV Virus Taxonomy Profile: Picornaviridae. *J Gen Virol*. 2017;98:2421–2. <https://doi.org/10.1099/jgv.0.000911>.
45. Lukashev AN. Role of recombination in evolution of enteroviruses. *Rev Med Virol*. 2005;15:157–67. <https://doi.org/10.1002/rmv.457>.
46. Rosario K, Breitbart M, Harrach B, Segalés J, Delwart E, Biagini P, et al. Revisiting the taxonomy of the family Circoviridae: establishment of the genus Cyclovirus and removal of the genus Gyrovirus. *Arch Virol*. 2017;162:1447–63. <https://doi.org/10.1007/s00705-017-3247-y>.
47. Lefeuve P, Lett J-M, Varsani A, Martin DP. Widely conserved recombination patterns among single-stranded DNA viruses. *J Virol*. 2009;83:2697–707. <https://doi.org/10.1128/JVI.02152-08>.
48. Benkő M, Aoki K, Arnberg N, Davison AJ, Echavarría M, Hess M, et al. ICTV virus taxonomy profile: 2022. *J Gen Virol*. 2022;103. <https://doi.org/10.1099/jgv.0.001721>.
49. Robinson CM, Singh G, Henquell C, Walsh MP, Peigue-Lafeuille H, Seto D, et al. Computational analysis and identification of an emergent human adenovirus pathogen implicated in a respiratory fatality. *Virology*. 2011;409:141–7. <https://doi.org/10.1016/j.virol.2010.10.020>.
50. Zolfo M, Silverj A, Blanco-Míguez A, Manghi P, Rota-Stabelli O, Heidrich V, et al. Discovering and exploring the hidden diversity of human gut viruses using highly enriched virome samples. *bioRxiv*. 2024. <https://doi.org/10.1101/2024.02.19.580813>.
51. Cumbo F, Blankenberg D. Characterization of microbial dark matter at scale with MetaSBT and taxonomy-aware Sequence Bloom Trees. *bioRxiv*. 2025. <https://doi.org/10.1101/2025.08.25.672238>.
52. Carroll D, Morzaria S, Briand S, Johnson CK, Morens D, Sumption K, et al. Preventing the next pandemic: the power of a global viral surveillance network. *BMJ*. 2021;372. <https://doi.org/10.1136/bmj.n485>.
53. Fang Y, Nie Y, Penny M. Transmission dynamics of the COVID-19 outbreak and effectiveness of government interventions: A data-driven analysis. *J Med Virol*. 2020;92:645–59. <https://doi.org/10.1002/jmv.25750>.
54. Manuel RD, Snyder JC. The Expanding Diversity of Viruses from Extreme Environments. *Int J Mol Sci*. 2024;25:3137. <https://doi.org/10.3390/ijms25063137>.
55. Cumbo F, Weitschek E, Blankenberg D. hdlib: A Python library for designing Vector-Symbolic Architectures. *J Open Source Softw*. 2023;8:5704. <https://doi.org/10.21105/joss.05704>.

Publisher's note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.