Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Predicting Genome-Wide Approximate Match Frequencies with Hit Frequency Vectors

Nathalie Gocht

nathalie.gocht@b-tu.de

Brandenburg University of Technology Cottbus–Senftenberg

Alexander Schliep

Brandenburg University of Technology Cottbus–Senftenberg

Research Article

Keywords: hit frequencies, approximate string matching, deep learning

Posted Date: February 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8865258/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Predicting Genome-Wide Approximate Match Frequencies with Hit Frequency Vectors

Nathalie Gocht[1*] and Alexander Schliep[1]

[1*]Faculty of Health Sciences Brandenburg, Brandenburg University of Technology Cottbus-Senftenberg, Lipezker Str. 47, Cottbus, 03048, Brandenburg, Germany.

*Corresponding author(s). E-mail(s): nathalie.gocht@b-tu.de;
Contributing authors: alexander.schliep@b-tu.de;

Abstract

Background: Estimating how many approximate matches a query sequence has in a genome is a fundamental task in bioinformatics, with applications including read mapping or genome mappability analysis. While exact string algorithms and specialized index data structures enable low-error matching, counting matches for short k-mers at high dissimilarity (small k/e ratio) becomes computationally intractable as the number of potential matches grow exponentially for Hamming distance, and even more so for other dissimilarity measures including edit distance.

Results: We use machine learning to solve this algorithmic problem. Specifically, we propose deep learning models which in contrast to learned index approaches do not accelerate exact algorithms, but rather directly predict Hit Frequency Vectors (HFVs), which count approximate matches at increasing distance $e = 1, 2, 3, \ldots$. The models are trained on HFVs which are computed using brute-force (for Hamming distance), respectively an efficient sampling approach followed by a search to account for undersampling close matches and correcting counts. Model training and inference are computationally efficient, and enable rapid genome-scale estimation of approximate match frequencies for short query sequences and small k/e. The models capture genome-wide match frequency distributions with high accuracy and allow fast batch prediction even in regimes where exhaustive computation is computationally infeasible for exact algorithms relying on seed-and-extend due to limited seed length leading to exponential effort. We provide loss functions and evaluation measures appropriate for histograms with bin count varying by several orders of magnitude.

Conclusion: Our approach enables practical approximation of HFVs and can be easily extended to complex dissimilarities which are too computationally expensive to evaluate exhaustively across an entire genome. The combination of approximate HFV generation and predictive modeling extends the scope of sequence uniqueness analyses considerably. Our approach can trivially be extended to continuous dissimilarities like oligonucleotide binding affinity via appropriate binning for applications in DNA storage, target depletion, or oligonucleotide therapeutics. Lastly, we believe that our work can provide a blue-print for further applications in bioinformatics where straight-forward ML prediction—possibly followed by confirmation with exact algorithms—could have great utility.

Keywords: hit frequencies, approximate string matching, deep learning

1 Introduction

Identifying all locations in a genome sequence of low edit distance from a typically short query sequence is arguably one of the most fundamental problems in bioinformatics. It is a variant of the well-studied approximate string matching problem routinely solved with sequence alignment tools mapping DNA sequencing reads to genomes. Somewhat less popular is the problem of merely *counting* the number of such locations which has to the best of our knowledge only been considered in greater detail in the context of computing genome mappability. The importance of efficient algorithms for computing genome mappability for resequencing experiments, genome analysis, or the design of CRISPR guide RNAs was highlighted previously [1, 2].

Several bioinformatics problems can be rephrased as variants of the general problems of counting the number of such locations. For example, the frequency f_x of a k-mer x in a genome trivially gives the number of locations at maximal distance 0. Pockrandt et al. [1] defines the (k, e)-frequency of a k-mer x as the number of locations of Hamming distance not greater than e and the (k, e)-mappability as its multiplicative

inverse. Note, that mappability is considered for *all* k-mers in a genome simultaneously; we are interested in typically smaller query sets. The problem we are addressing is defined as follows.

Definition 1 Given a string T and a distance $d(\cdot,\cdot)$ between strings over an alphabet Σ, we define the Hit Frequency Vectors (HFVs) for a given k-character query sequence $x \in \Sigma^k$ as

$$f(x;T) := (f_0(x;T), f_1(x;T), f_2(x;T), \ldots, f_m(x;T)),$$

with $m \leq k$. Here $f_i(x;T)$ is exactly the number of locations $j_1, j_2, \ldots, j_{f_i(x;T)}$ in T for which $d(x, T[j_m, \ldots, j_m + k]) = i$ holds; $1 \leq m \leq f_i(x;T)$. Note that x itself is not required to be a contiguous substring of T. For brevity, we will omit T where it is clear from the context and write $f(x)$ and $f_i(x)$, respectively.

Summing over entries of the hit frequency vectors yields the definitions used by Pockrandt et al. [1], likewise our Hit Frequency Vectors (HFVs) can be recovered from the (k,e)-frequencies defined here.

Definition 2 Given a string T, the (k,e)-frequency of the k-mer $T_i := T[i : i+k-1]$ is the number of occurrences of T_i in T with at most e mismatches or errors, i.e. with Hamming distance $d_H(T_i, x) \leq e$ for all such occurrences x. Clearly, the (k,e)-frequency of T_i is equal to

$$\sum_{i=0}^{e} f_i(T_i;T).$$

The (k,e)-mappability, which is usually computed for all or a majority of positions in T, is its multiplicative inverse.

With k-mer counters such as Turtle [3] or KMC3 [4, 5] $f_0(x;T)$ is obtained very efficiently. For longer k-mers, say on the order of 31 or larger, of a genome T and small number of mismatches, say $e \leq 3$, efficient computational tools such as GenMap [1] and GEM [2] give the (k,e)-mappability, and, with multiple runs with increasing e, also

the $f_i(x;T)$ for $0 \leq i \leq e$ at least for the Hamming distance. However, in particular for modest values of k, e.g. $k < 20$ and large e, computation of the Hit Frequency Vectors (HFVs) is challenging. In this setting seed-and-extend approaches become infeasible as the maximal length of the seed is roughly of length k/e. Moreover, $f_i(x)$ increases exponentially in e, and even efficient *exact* algorithms would need to confirm an exponential number of possible occurrences, leading to an exponential number of confirmed positions.

Additionally, state-of-the-art methods are designed for a fixed distance, predominantly Hamming distance. In contrast, we desire to ultimately use dissimilarity measures between strings other than the Hamming distance.

In the following, we introduce a computational approach to efficiently predict approximate HFVs for large genomes, modest values of k, and moderate to large values of e for, in principle, *arbitrary* distance functions. Our approach combines efficient k-mer counting, fast sampling proportional to k-mer frequencies, and an efficient search for k-mers with small distances to a query sequence with Deep Learning. Our multi-output regression (MOR) models are based on convolutional neural network (CNN) to predict HFVs from query sequences alone. This allows us to predict HFVs in a fraction of the time needed for their computation, as, for example, for edit distance.

Our approach is in contrast to previous approaches using ML in bioinformatics algorithms. Existing fast, exact or heuristic, implementations of approximate matching often use (a) index data structures reducing complexity or queries leading to efficient amortized running times (e.g. suffix arrays in a seed-and-extend approach) [6] and (b) heuristics for further acceleration (e.g. Bowtie 2 [7]) trading speed for accuracy. Learned indexes [8] have been used in conjunction with the same data structures, adapting to the data distribution in the search for exact seed matches within a Suffix arrays for example [9]. For the problem at hand, the exponential number of occurrences of short seeds of $\sim k/e$-characters would easily offset the improvements by the learned index.

In our approach we *directly* predict, for a query x, an approximation to its HFV $f(x;t)$; i.e., what would be the outcome of a potential ML-accelerated computation. To that end, we train a deep learning model on query sequences and exact, respectively approximated, HFVs. Similar to the case of index data structures, the cost for data generation and training the model can be amortized over queries. Clearly, this is a trade-off of accuracy for speed, which nevertheless performs favorably in particular for edit distance as in that case, for lack of efficient algorithms, the only alternative is brute-force computation.

For the prediction we use CNNs, a widely used deep learning models capable of extracting hierarchical features from structured data using local connectivity and shared weights [10]. Their ability to capture spatial patterns and nonlinear relationships has driven major advances in high-dimensional prediction tasks [11, 12]. In the context of genome analysis, CNNs are especially well-suited for modeling sequence data due to their capacity to learn local motifs and patterns relevant to biological functions such as transcription factor binding sites [13, 14].

Traditional single-target regressions map inputs X to a single response $y \in \mathbb{R}$. Numerous complex systems require the prediction of an output vector $\mathbf{Y} = [y_1, y_2, \ldots, y_d] \in \mathbb{R}^d$ [15]. Multi-output regression (MOR), also often referred to as multi-target regression, provides a holistic and unified framework to simultaneously model the potentially intricate relationship between X and the entire output space $\mathbf{Y}$ [16]. MOR is closely related to the well-studied Multi-Task Learning (MTL) paradigm. A key finding in both MOR and MTL literature is that training independent regression models for each target variable y_i is often less effective compared to MOR approaches [17–19]. The independent approach neglects the inherent correlations and shared information structure among the different output variables, which can lead to inconsistent predictions and decreased generalization performance. MOR easily

extends to neural networks since they natively support multiple outputs and facilitate the learning of complex, shared feature representations across targets. Previous research has focused on architectures and optimizers [20, 21].

Note, that the HFV generation pipeline and deep learning methods we propose can be used with other dissimilarity or distance functions, for example those derived from oligonucleotide binding energies which we will explore elsewhere. This is also partly our motivation for considering small k and large e. The proposed prediction model works even for number of errors between query sequence and hits larger than what is feasible to exactly compute for the edit distance. In the following, we will first describe the computational pipeline to extract HFVs for a given set of query sites and a genome followed by the introduction of two CNN-based architectures for predicting HFVs from query sequences. Finally, we will evaluate the accuracy of the HFV generation pipeline as well as the predictive performance of the CNN models.

2 Methods

This section describes the generation of exact HFVs using brute-force calculations for Hamming distances as well as approximate HFVs using the Hybrid Search and Sampling (HSS) approach for Hamming and edit distances. Subsequently, deep learning (DL) models are presented for bulk predictions of HFVs.

2.1 Generating Hit Frequency Vectors

The base for generation of all data sets are the counts of pair-wise distinct 16-mers of the human genome GRCh38 (GCA_000001405.15 [22]) obtained with KMC [4]; k-mers containing ambiguous bases are excluded. There are $\approx 1.1339 \times 10^9$ 16-mers contained in this set S. In order to compute exact HFVs for all sites in the genome, $|S|^2$ distance calculations are needed. For distance measures other than Hamming distance, this is likely to be computationally infeasible. Therefore, a sample X of 5 million query

Table 1: Overview of generated data sets used. Exact computations for Hamming distance (HDE) are possible; the purpose of HDA is to evaluate losses due to approximating HFVs with models trained on the set U of 5 million 16-mers (HDA). For edit distance EDA exact computation is infeasible.

ID	Short Description
S	Set of unique k-mers in genome G and frequencies f_s for $s \in S$
X	a set of query sequences sampled from S
U	set of sampled 16-mers, u, from S according to the k-mer frequency
$V_{x,d\leq e}$	set of searched 16-mers, v, from S for a query sequence x with $d(x,v) \leq e$
HDE : $(x, f_{HE}(x))$	data set of Hamming distance HFVs computed exactly $\forall x \in X$
HDA : $(x, f_{HA}(x))$	data set of Hamming distance HFVs approximated by HSS approach
EDA : $(x, f_{EA}(x))$	data set of edit distance HFVs approximated by HSS approach

sequences were drawn uniformly random from S. The choice of 5 million is to ensure sufficient data for training our DL models and evaluating predictions. The GC content distribution of dataset X is shown in Supplement Figure S4.

For Hamming distance, i.e. the number of mismatches between two equal length sequences, a brute-force calculation between every $x \in X$ and every $s \in S$ is feasible with efficient bit-wise operations on two-bit encoded DNA sequences with the alphabet $\Sigma = \{A, C, G, T\}$. Counting the hits at Hamming distance $0, 1, 2, \ldots$ for a query x yields the HFV $f(x) = f_{\mathbf{HDE}}(x)$. We refer to the data set $\{(x, f_{\mathbf{HDE}}(x)) | \, x \in X\}$ by **HDE**. Despite reducing the number of distance computations to $|X| \cdot |S| \approx 5.6 \cdot 10^{15}$ distance calculations, calculating HFVs for all query sequences in X remains computationally challenging for most other distance measures.

In addition to the set of exact HFVs we introduce a data set of approximate HFV, referred HDA. The two data sets allow us to evaluate losses from approximating HFVs with models trained on the set U of 5 million 16-mers. For edit distance, exact computation is infeasible, thus requiring a data set **EDA**. Both **HDA** and **EDA** are constructed by computing distances between each query and a large number of samples and extrapolate the histogram of these observed counts to an HFVs.

Specifically, we draw $n = 1\,000\,000$ samples from the set S of unique k-mer according to their frequencies f_s and call the set U; see also panel A in Fig. 1. A k-mer

$s \in S$ has the probability $p_s = \frac{f_s}{\sum f_j}$ of being sampled. This is not a fast operation as the number of distinct k-mers is in the billions. We will utilize a two-step sampling process which will greatly accelerate the process. Note that the k-mer frequencies are discrete and from a limited range of values. We partition the set S of k-mers into classes $S_1, S_2, S_3, \ldots$ of k-mers with frequency $1, 2, 3, \ldots$, respectively. Observe that the probability that a k-mer sampled from S has frequency j is equal to

$$q_j = \frac{j \cdot |S_j|}{\sum_i i \cdot |S_i|} \,. \tag{1}$$

As the first step, we sample a frequency j according to $(q_1, q_2, \ldots)$; within the thus determined class S_j we can in the second step simply select any k-mer $s \in S_i$ with uniform probability. The two step procedure is substantially faster as the highest frequency observed, and thus the maximum number of distinct frequency classes, is orders of magnitudes smaller than the number of distinct k-mers.

To accurately represent the genomic environment in the HFV, the absolute abundance of hits per distance bin $f_i(x)$ is needed. The sampled frequencies $\hat{f}_1, \hat{f}_2, \ldots$ sum up to the number of sampled k-mers, that is $\sum \hat{f}_i = n$. The total number of k-mers in S is $N = \sum f_i$. For the sample to represent the same absolute numbers, we assign each sampled k-mer the multiplicity $c = \frac{N}{n}$ yielding scaled frequencies $\sum c\hat{f}_i = N$.

Even for large sample sizes the left tails of the distribution (cf. Fig. 2) will likely remain unsampled. To correct this defect, we utilize a search to explore the left tail, which contains binding sites containing low number of mismatches for Hamming HFV, or low edit distances for edit HFVs, compared to a query sequence x. While the use of alignment tools is commonly preferred for finding exact matches and matches with low number of mismatches, utilizing them for short query sequences and moderate number of mismatches becomes infeasible. Generally, the limit for exact seed length in seed-and-extend approaches is $|q|/(e+1)$, where q is the query length and e is the

number of mismatches [23]. Widely used implementations such as Bowtie2 [7] use a combination of spaced seeds and exact seeds to find candidate locations for alignments. As the number of mismatches increase, the expected number of hits increases (cf. Fig. 2) and the exact match seed length for alignments decreases leading to more false positive seed hits and thus increased running time per reported hit [24]. Therefore using a standard alignment tool with a genome index seemed impractical. Instead we use a recursive search to explore all sequences within a Hamming ball for Hamming HFVs and an edit ball for edit HFVs, defined as following:

Definition 3 Given the Hamming distance $d_H(x, y)$, defined as the number of mismatches between two strings $x, y \in \Sigma^k$ of equal length k, the Hamming ball of radius e around a sequence x is defined as

$$B_H(x, e) = \{y \in \Sigma^k : d_H(x, y) \leq e\}.$$

Definition 4 Given the edit distance $d_E(x, y)$, defined as the number of edit operations – insertions, deletions, substitutions – between two strings $x, y \in \Sigma^k$, the edit ball of radius e around a sequence x is defined as

$$B_E(x, e) = \{y \in \Sigma^k : d_E(x, y) \leq e\}.$$

The search explores all possible sequences $B_H(x, 4)$ for Hamming HFVs and $B_E(x, 3)$ edit HFVs around a given query sequence x. The values were chosen due to computational limitations. A Bloom filter [25] indicates whether a sequence $v' \in B_H(x, 4)$, respectively $v' \in B_E(x, 3)$, is present in S. The result is a set of triples $(x, v, c(v)) \in V_{x,d\leq e}$, where v is the searched site validated to be in S and $c(v)$ denotes the frequency of v in S. For the purpose of retrieving frequency counts for each v occurring in $V_{x,d\leq e}$, two approaches were implemented and evaluated: a linear search (LS) over S and a lookup using minimal perfect hashing (MPH) [26]. The linear search provides an exact result for the search as false positives, a consequence of

Bloom filters, are filtered. The MPH approach cannot guarantee an exact result as it cannot mitigate false positives from the Bloom filter. However, it uses less computational resources than the linear search. We use the linear search for all datasets with approximated HFVs benefiting from the exact search.

Distances are calculated between all pairs (x, u) for $x \in X$ and $u \in U$ as well as all $(x, v) \in V_{x,d\leq e}$, where $e \leq 4$ for Hamming distance calculations and $e \leq 3$ for edit distance calculations. The Hamming distance calculation utilizes 2-bit encoding of sequence strings and a bit-wise Hamming distance calculation enabling a fast calculation of Hamming HFVs. For the calculation of edit HFVs, `quickmers` [27] which provides an efficient C-based implementation of Myers' [28] fast bit-vector algorithm, is used. The resulting data sets **HDA** : $(x, f_{\mathbf{HDA}}(x))$ and **EDA** : $(x, f_{\mathbf{EDA}}(x))$ contain the pairs of query sequences X and their approximate HFVs for Hamming distance and edit distance, respectively. As stated before, Hamming HFVs can be exactly computed efficiently, so the main puropose of **HDA** is to allow evaluation of accuracy differences. An overview over all data sets is given in Table 1 and more technical details on the HSS approach are listed in the Supplement. The HSS approach gives a close approximation with exact hit frequencies for smaller distances using the linear search to retrieve frequencies from the k-mer counts and small errors for distances approximated through sampling (cf. Fig 6). Errors in the sampled distance bins are predominantly in the smallest distance bin not included in the linear search, e.g. for $f_5(x)$ for **HDA**. For genome-scale computations, i.e. computing HFVs for $\approx 1.1339 \times 10^9$ distinct k-mers this approach is not a feasible solution (cf. Table 2). In the following, we propose CNN-based MOR models enabling bulk predictions of HFVs in a fraction of the time compared to the HSS approach.

2.2 Predictive Models

Multi-output regression (MOR) models are used to predict multiple target values simultaneously making use of a shared feature representation rather than training individual models per target. They frequently encounter optimization difficulties because each target may follow a different range, variance, or distribution of values, causing imbalance during training and uneven gradient magnitudes across targets [29]. HFVs show a wide range of possible values which can differ in multiple magnitudes between frequency bins. Such heterogeneity leads models to overemphasize high-scale targets while underfitting low-magnitude ones unless normalization, rescaling, or adjusted loss weighting is applied. This is mitigated by applying the logarithmic transformation $y_i = \log(f_i(x) + 1)$ to each bin i of $f(x)$, yielding a value range that facilitates more effective training and inference.

The input to the models are the query sequences X, transformed into matrices with one-hot encoded columns of dimensions 4×16, where each row corresponds to one letter in the alphabet $\sum = \{A, C, G, T\}$ and each column corresponds to a position in the 16-mer. An extension to other alphabets such as RNA is trivial. Two different CNN-based architectures (cf. Figure 1, panel B) were implemented and evaluated:

- **BUT** is based on Buterez's model [30] using a 2D convolution with a kernel size $4 \times m$, where m is a configurable parameter. The output of this layer is a 1D embedding of the input sequence, which is passed through two 1D convolutional layers applying a sliding window approach followed by two fully-connected layers. In comparison to the model proposed by Buterez [30], which was included for its good performance on a prediction task for k-mers in the context of predicting binding energies, a lighter architecture with smaller kernel sizes was chosen to mitigate overfitting on shorter sequences.
- Based on preliminary analysis of the performance of **BUT** we developed an alternative. **KD** is a more general CNN using two 2D convolutional layers followed by three

fully connected layers. Convolution parameters, including kernel size and padding, are chosen such that the input dimensions, 4×16, are retained through forward propagation of the CNN.

The exact architecture parameter for the evaluated models are listed in Supplementary Table S6 and S7. Commonly used loss functions such as mean squared error (MSE) and mean absolute error (MAE) are well suited for single-target regression and for multi-output regression tasks in which the target variables share comparable value scales. However, in multi-output regression settings where the targets span substantially different numerical ranges, these loss functions can disproportionately penalize errors associated with larger-scale targets. This is partly mitigated by the log-transformation of target values. Relative loss functions, e.g. the symmetric mean absolute percentage error (sMAPE), or weighted loss functions can be used counteract this effect as well. The weighted loss functions follow the formula

$$\mathcal{L}_{\mathrm{w}} = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L} \times w_i, \tag{2}$$

where $\mathcal{L}$ is the loss function, e.g. MSE or MAE, m is the number of target bins, and w_i is the weight for bin i. The weights are chosen such that a larger penalization for bins with smaller indices than bins with larger indices. Kendall et al. [29] proposed a multi-task loss function based on homoscedastic uncertainty for weighting multiple loss functions. Adapting this approach to MOR, the loss function is defined as

$$\mathcal{L}_{\mathrm{uncertaintyMSE}} = \sum_{i=0}^{m} \frac{1}{\sigma_i^2} \cdot \mathcal{L}_i + \log \sigma_i, \tag{3}$$

where σ_i is the learned uncertainty parameter for bin i and $\mathcal{L}_i$ is the MSE loss for bin i. Additionally, additive loss functions were evaluated, specifically an additive loss of sMAPE and KL divergence

$$\mathcal{L}_{\mathrm{add}} = \mathcal{L}_{\mathrm{sMAPE}} + \alpha \times \mathcal{L}_{\mathrm{KL}}, \tag{4}$$

where α is a scaling hyper parameter.

3 Results

In the following, the above outlined pipeline, predictive models and data sets are presented. To evaluate the accuracy of the HFV generated by the HSS approach and predicted by the predictive models, the symmetric absolute percentage error (sAPE) is used. The sAPE is defined as follows

$$\text{sAPE} = \frac{|y_i - \hat{y}_i|}{(y_i + \hat{y}_i)} \times 100\,, \tag{5}$$

in its alternative form to be bound between 0 % and 100 % providing a scale-independent measure of accuracy with intuitive interpretability. The symmetric mean absolute percentage error (sMAPE) is then defined as the mean over all sAPEs for each predicted HFV. It facilitates the comparison of model performance across variables that differ in magnitude where absolute error measures (e.g., MAE or RMSE) would be biased toward frequency bins with larger numerical scales. The symmetric normalization of the denominator aims to mitigate the inflation of error percentages that occurs in traditional MAPE when actual values approach zero, thereby providing a more balanced and comprehensible representation of relative error. Nevertheless, for small true values instability partially remains and it tends to penalize under-predictions more than over-predictions [31].

3.1 Dataset Generation

Depending on the distance function between strings different computational strategies are preferable for generating the HFVs. Generally, Hamming distance can be computed brute force with very high throughput using bit-parallel and SIMD instructions. Linear memory access patterns and branch-free instruction sequences lead to very high performance in particular for modest k. To correct the approximate HFVs obtained

from computing frequencies for sampled k-mers in U, which are likely to underestimate the frequency for small Hamming or edit distance, namely $f_0(x), f_1(x), f_2(x), f_3(x)$, and $f_4(x)$, a search is performed. To prepare for arbitrary distance measures or dissimilarities we evaluated a baseline and an approach utilizing minimal perfect hashing.

We present an overview of CPU times for the generation HFV for batches of 20 000 in Table 2. For Hamming distance brute-force is indeed the best approach to generation of training data; as we show in Section 3.3 even that is not fast enough to brute-force HFVs genome-wide.

The baseline linear search requires substantial computational effort for approximate HFVs (**HDA**, **HDE**). While the MPH approach was significantly faster, it led to a small loss in accuracy for searched Hamming or edit distances (cf. Fig 6), where error outliers could reach up to 100 % for $f_1(x)$, up to 60 % for $f_2(x)$, and up to 12 % for $f_3(x)$. Note, that the sAPE is more sensitive to absolute deviations for smaller true counts than for higher magnitude values. For example, an approximated frequency value $\hat{f}_i(x) = 200$ for its true frequency $f_i(x) = 10$ as a sAPE of 90.5 % according to Equ. 5 with an absolute difference of 190. Meanwhile, for a true count $f_j(x) = 1000$ an approximated value of $\hat{f}_j(x) = 10000$ results in a sAPE of 81.8 % while the absolute difference is 9000. The errors for $f_1(x), \ldots, f_4(x)$ in Figure 6 (b) result from the false positive rate (FPR) of the Bloom filter queried to ascertain presence of queries before querying the MPH. For all data sets, the FPR was set to 1×10^{-5}. Results for Hamming distance 5 to 16, which were retrieved via frequency-based sampling, remain identical.

For edit HFVs, both approaches were substantially slower than generating exact Hamming HFVs, as the distance calculation is computationally more intensive which notably increases the CPU times for sampling step compared to generation of Hamming HFVs (cf. Table 2). The HSS approach for edit HFVs is significantly faster than

Table 2: CPU times for generation of 20 000 HFVs for different generation methods. Times are averaged over 5 million queries, run in batches of 20 000 queries. * indicates an extrapolated value.

HFV generation method	CPU Time Hamming distance search $e \leq 4$	Hamming distance search $e \leq 3$	edit distance search $e \leq 3$
Distance vs. S ($2.26 \cdot 10^{13}$ total)	8 h 15 m 14 s		24 d 22 h 49 m*
Distance vs. U ($2 \cdot 10^{10}$ total)	3 m 45 s		31 m 41 s
Linear search in S	22 h 27 m 2 s	9 h 42 m 48 s	12 h 30 m 16 s
Search with MPH in S	11 h 48 m 18 s	6 m 27 s	20 m 13 s

a brute-force calculation of exact edit HFVs (cf. Table 2) highlighting the efficiency of the HSS method for distance measures more complex to compute than Hamming distance. As the size of the edit ball grows faster than the size of the Hamming ball, as it is a proper superset for same e, we restricted computations to $e \leq 3$ for the edit distance. This is an accuracy-speed-tradeoff, just like the false-positive rate of the Bloom filter above.

Figures 3, 4 and 5 show the distributions of $f_i(x)$ per distance bins for datasets **HDE**, **HDA** and **EDA**, respectively. Across all datasets, a high degree of homogeneity is observed, particularly within the interquartile ranges of each frequency bin. Exploratory data analysis further indicates increased variability at moderate to larger distances in **EDA**. Despite this localized increase in variance, the overall structural consistency and limited dispersion of the data indicate that the relationships across frequency bins remain well structured, supporting the suitability of neural network models for learning these patterns.

3.2 Prediction results

Architecture parameters were found empirically for each model architecture and a grid search was utilized to find the best combination of learning rate and loss function. We used a 80/10/10 (train/validation/test) split. The best performing configurations were tested in a 9-fold cross validation with 10 % of the data set kept as the hold-out

test set. For all splits, stratification over GC content of query sequences was used, such that the GC content distribution of the data set is preserved per split and test data. For exact matches and hits with small numbers of mismatches, common read alignment tools—e.g. Bowtie [7] or StarAligner [32]—can be used with reasonable accuracy and run time. Therefore, the models were trained for prediction of frequency bins $f_3(x), \ldots, f_{11}(x)$ for **HDE**, **HDA** and **EDA**, respectively.

Figure 7 shows the results for the best performing models predicting HFVs of the **HDE** data set. The sMAPE plots (cf. Fig. 7 a, c) show the percentage of cumulative proportion of predictions over the sMAPE (x-axis) over 9 cross folds. Annotated values show the average percentage of predicted HFVs with a sMAPE $\leq 2\,\%$. The box plots (cf. Fig. 7 b, d) show the sAPEs (cf. Eq. 5) per frequency bin for the best performing fold in the cross-fold validation. For the prediction of **HDE**, the **KD** model (MSE loss, learning rate = 0.0004) performed best where an average 68.1 % of predictions have a sMAPE $\leq 2\,\%$. The **BUT** model (sMAPE loss and KL loss, learning rate = 0.0005) achieved 57.1 % of predictions having a sMAPE $\leq 2\,\%$. The model performances for **HDA** are similar in trend, but overall, below the model performances predicting **HDE** data (cf. Fig. 8). The cumulative proportion of predictions with a sMAPE $\leq 2\,\%$ are 4 to 6 % lower compared to the prediction of exact Hamming HFVs. The differences seem to be in the prediction of $f_5(x)$, which show a larger interquartile range (IQR) for sAPE values while the predictions for larger Hamming distances are comparable.

Similarly for predicting edit HFV of **EDA**, the **KD** model (weighted MSE loss, learning rate = 0.0004) performed significantly better than **BUT** (sMAPE + KL loss, learning rate = 0.0005). Figure 9 shows the sMAPE plots (a, c) for all 9 cross folds for the **KD** and **BUT** results. The observations are comparable to the prediction results for **HDE** and **HDA**. While there are no clear differences in the main distributions of errors for edit distances 6 to 11 between the two CNN architectures, differences become

more visible when focusing on the outliers. Additionally, IQRs for edit distances 3 to 5 (cf. Fig. 9 b, d) are larger using the **BUT** model.

The best performing models are summarized in Table 3. While the hyper parameters were the same for predicting **HDE** and **HDA**, the results are significantly better when predicting exact HFV for both model architectures. The prediction accuracy for the **BUT** architecture is comparable between Hamming and edit HFVs, the heterogeneity in the data set of edit distance HFVs (cf. Figure 5) seems to have a positive effect for the **KD** architecture shown in a clear improvement of the prediction results. When investigating how many predictions have a sMAPE of less equals $\leq 5\,\%$, the differences in prediction accuracy become less pronounced with the **KD** models performing 3-4 % better than the **BUT** models.

Table 3: Overview of prediction performance for the best performing models evaluated in 9-fold cross validations. Values are averaged.

Model	Training time per epoch	Prediction time per HFV	% HFVs sMAPE $\leq 2\,\%$	% HFVs sMAPE $\leq 5\,\%$
HDE				
KD	56.70 sec	5.94×10^{-6} sec	68.05 %	95.55 %
BUT	66.87 sec	5.51×10^{-6} sec	57.07 %	92.52 %
HDA				
KD	46.48 sec	5.32×10^{-6} sec	61.77 %	95.11 %
BUT	67.60 sec	5.88×10^{-6} sec	53.54 %	92.38 %
EDA				
KD	66.71 sec	6.44×10^{-6} sec	66.93 %	97.39 %
BUT	80.29 sec	5.98×10^{-6} sec	53.03 %	93.28 %

The prediction results suggest that the driving factor for a high sMAPE are the first three predicted bins for both data sets. Supplementary Figure S8 shows the correlations of sAPEs per distance bin with the resulting sMAPE per predicted HFV, supporting the observation that the distance bins 3 and 4 have the highest correlation with the sMAPEs across all models. Commonly, larger errors can also be a sign of differences in target variable distributions between training and test data. However,

the distributions of frequency bins between training and test sets remain similar for frequency bins $f_3(x), \ldots, f_5(x)$ (cf. Supplementary Figure S9). In order to further understand the relation between the predicted HFVs and the resulting sMAPE, single predicted HFVs are visualized in Supplementary Figure S7 suggesting that the driving factor for a sMAPE between 2% and 5% is a larger error in frequency bin $f_3(x)$.

The comparison of variability across data sets with disparate scales or units of measure necessitates a normalized metric of dispersion. The Coefficient of Variation (CV), calculated as the ratio of the standard deviation, σ, to the mean, μ, $CV = \frac{\sigma}{\mu} \times 100$, serves this function by providing a measure of relative dispersion in percentage. Table 4 shows the coefficients of variation for **HDE**, **HDA** and **EDA** for distances 3-11. **EDA** shows significantly lower coefficient of variation for distance 5-9, but variation increases again in larger distances compared to Hamming HFVs. This indicates that the model's predictions for bins with smaller indices are more variable relative to their mean values, suggesting that these bins are inherently more challenging to predict accurately. This is reflected in the shape of outlier sAPEs observed for these bins in Figures 7, 8, and 9.

Table 4: Coefficient of variation for Hamming HFVs and edit HFVs.

Dataset	$d = 3$	$d = 4$	$d = 5$	$d = 6$	$d = 7$	$d = 8$	$d = 9$	$d = 10$	$d = 11$
HDE	109.41	69.90	53.12	41.47	32.31	24.64	17.95	11.91	6.33
HDA	109.41	69.90	53.44	41.57	32.34	24.65	17.96	11.92	6.34
EDA	92.89	62.10	47.19	37.68	31.04	25.71	19.72	11.38	10.36

While weighted loss functions are expected to counteract imbalances in the target distribution and were in conclusion expected to perform better, the unweighted loss functions performed slightly better for Hamming HFVs. The log transformation already reduces the risk of imbalances by reducing the ranges of values from 7 magnitudes ($10^2 - 10^9$) to 1 magnitude ($10^0 - 10^1$) making the assumed effect of weighted

loss functions negligible. Equal absolute differences of values in log-space can correspond to very different absolute differences of count values. Consequently, weighted loss functions may have a greater effect applied to target values in lower logarithmic space.

Both models perform significantly better than a naive approach predicting the median of the training set for each bin as shown in Supplementary Figure S5 and S6.

3.3 Performance

In the following, the computational resources required for the HSS approach and the predictive models are discussed. We are unable to perform an extensive benchmark analysis as solving the exact version of our problem via brute-force computations at genome scale is computationally infeasible at least for the edit distance and likely for other more dissimilarities of higher complexity than Hamming distance.

In order to put our performance in perspective, we report running times for GenMap [1]. It is important to note, that this is not a fair comparison, as GenMap provides a fast tool to rapidly compute genome mappability, and, moreover, GenMap was designed for longer k-mers and few errors. An additional complication is that GenMap runs on CPU only, and our method generates data on CPU but uses a GPU for training and prediction. We nevertheless believe that the comparison with a tool solving a related, but distinct problem can provide valuable insights into the advantages of the DL prediction.

Brute-forcing even Hamming distance across the whole genome is infeasible. There are 1.1339×10^9 distinct 16-mers in S. Therefore, computing exact Hamming HFVs for the whole genome would take an estimated $\sim$19 500 d CPU time (that is, $\sim$ 3.3 years on a 16 core machine) extrapolated from Table 2.

The deep learning approach requires a sizable, but completely feasible initial effort for generating data, but training and inference are refreshingly quick thanks to the

Table 5: Runtime approximations for generating and predicting HFVs for all unique 16-mers of the human genome ($\approx 1.1339 \times 10^9$) per data set and predictive model. 4.5 Mio HFVs were used for training and validation. The remaining HFVs need to be predicted using the trained model.

Steps	**HDE**	**EDA**
HSS		
Generation of 4.5 Mio. HFVs (CPU hours)	1857 h	2932 h
Predictive models		
Preprocessing	0:02 h	
Training (run time / single GPU)		
BUT	0:23 h	0:28 h
KD	0:51 h	0:61 h
Prediction of 1.1294×10^9 HFVs (run time / single GPU)		
BUT	1:44 h	1:53 h
KD	1:52 h	2:01 h

modest model sizes required and the scaling of batch inference tasks. Details are given in Table 5, where for the predictive models the average training time measured over nine cross folds is stated. The prediction times for the remaining 1.1294×10^9 distinct 16-mers of the whole genome are extrapolated from Table 3. Training to convergence was obtained for **BUT** on average 21 epochs resulting in a total runtime of ≈ 23 min training time. The **KD** model required on average 54 epochs for training with running times between 51 and 61 min depending on the dataset.

Training and prediction the deep learning models significantly benefited from the use of a single Nvidia A100 GPU with 40GB RAM enabling a training and prediction in ~ 3 h excluding generation of HFVs. Comparing CPU and GPU workloads is non-trivial; in the literature speedups of 40 tot 100× for an Nvidia K20 or TITAN X [33] over CPU can be found; today cost parity between an A100 (which was not even needed memory-wise) and maybe an extra 100 cores in the same server chassis seems somewhat realistic. On the other hand, with GPUs being standard even in laptops one could make an argument that utilizing them in addition to CPU is fair game in a comparison.

Table 6: GenMap CPU times for different (k, e)-mappability. Index building not included. Running times were measured using the `time` Linux terminal command. Note that increasing e by one results in running times roughly an order of magnitude larger. As discussed, this is reflective of the problem and the fact that GenMap was not designed for large e.

(k,e)-mappability	**(36,0)**	**(36,2)**	**(16,0)**	**(16,1)**	**(16,2)**	**(16,3)**
16 threads	40 m	18 h 29 m	1 h 12 m	8 h 31 m	122 h 59 m	—
104 threads	1 h 8 m	33 h 15 m	5 h 13 m	14 h 47 m	207 h 29 m	2410 h

GenMap [1] determines genome mappability by indexing the reference sequence and evaluating all sites to quantify their frequency of occurrence under a user-defined mismatch threshold preserving the location of the sites. Table 6 shows the CPU times required to compute the genome mappability for different (k, e)-mappability and number of threads. The exact command is given in the supplement; for the index also (GCA_000001405.15 [22]) was used.

The CPU times increase significantly as the permissible error increases, but overall we obtained consistent results as reported by Pockrandt et al.'s [1]. Note, that Genmap did not scale well to higher core counts, which was required for computing $(16, 3)$-mappability within 48 hours wall-clock.

Recall that the comparison with GenMap is to provide a frame of reference and not a benchmark because $(16, 3)$ is not in the intended and designed parameter range. However, the 2410 h computation of $(16, 3)$ exceeds the total time of $1857 + 2.75 = 1859.75$ hours generating data, training our model and predicting HFVs per unique k-mer in S, and observing that incrementing e results roughly in an order of magnitude higher running time of GenMap, it seems very unlikely that this won't hold. We are not aware of any method which solved the problem for edit distance.

4 Discussion

We consider the problem of predicting Hit Frequency Vectors (HFVs), that is histograms reflecting the frequencies of approximate matches for distance values up to 15 for 16-mers for Hamming and edit distance, respectively. This is related, and as far as the maximal error is concerned, an extension of prior work on genome mappabilty. Our general approach uses string algorithms followed by deep learning to predict the outcome of exact computations instead of just using ML to accelerate parts of an algorithm, like it is routinely done in the literature using learned indexes [8]. Our approach trivially extends to other distance functions and dissimilarities, as, except of parts of the process for generating training data, no assumptions about the distance functions are made. While the distance measures presented are discrete-valued, our approach can easily be extended to continuous dissimilarities such as binding affinity of oligonucleotides by choosing appropriate bin boundaries. Thus, the approach will likely be useful in exploring oligonucleotide-specific binding partners other than near-perfect matches, and thus be useful in application areas from DNA storage [30], depletion or enrichment of specific DNA/RNA in samples [34] to off-target effects or sequestration of oligonucleotide therapeutics [35, 36]. Results will be reported elsewhere.

Our initial approach combining HSS and MOR provides a scalable, distance-agnostic strategy to obtain HFVs with a favorable trade-off between accuracy and running time in regimes where typical string algorithms using seed-and-extend strategies fail, namely for short k-mers and high error e, in other words small k/e. We see applications also in the study of any fast evolving organisms, e.g. pathogens.

The prediction error across an HFV is highly impacted by the errors in the first few distance bins predicted, here particularly $f_3(x)$ and $f_4(x)$. The relatively larger errors in these bins are partly due to the instability of the sAPE towards small-magnitude counts. On the other hand, multi-output regression (MOR) for predicting values on significantly different magnitudes is challenging. Future research may explore loss

functions focusing on penalizing errors on small-magnitude counts. Additionally, an evaluation framework incorporating additional metrics to the sMAPE, which exhibits a sensitivity to small-magnitude counts, may improve future iterations. Very likely improvements could be made by using improved custom-architectures for histograms with widely varying bin counts.

We expect computational advantages to hold over exact computations as long as k/e is small, so further studies to explore larger k would be interesting. The computational effort for obtaining exact HFV respectively training data is of course a concern. Clearly, our approach offers tunable compute-accuracy tradeoffs in several places. Due to the large effect of data distributions—genomes vary widely in repeat structure—a theoretical analysis might be of lesser utility, but establishing an empirically motivated, but automated approach for tuning those parameters would be beneficial. Ultimately, integrating exact and predictive computations, e.g. using GenMap and our tool together for Hamming distance, would likely be a valuable combination, in particular when the determination which tools to use in which regime and how to adapt parameters to maximize fidelity for a given computational budget could be solved.

To contextualize the current work: heuristics for sequence analysis are a common solution for trading off running time vs accuracy, probabilistic data structures are routinely used if the errors introduced are considered acceptable, and lately learned indexes have found use as part of classical sequence analysis data structures. The current work demonstrates that prediction of the outcome is feasible, and can cover regimes where classical algorithms become too costly. We believe that there are further applications in bioinformatic where straight-foward ML prediction—possibly followed by confirmation with exact algorithms—could have great utility.

Abbreviations

DL deep learning

HFV Hit Frequency Vector

HSS Hybrid Search and Sampling

CNN convolutional neural network

MOR multi-output regression

MPH minimal perfect hashing

MSE mean squared error

MAE mean absolute error

sAPE symmetric absolute percentage error

sMAPE symmetric mean absolute percentage error

IQR interquartile range

FPR false positive rate

MTL Multi-Task Learning

Supplementary information. Supplementary information including additional figures, tables, and details on running software are available in the Supplementary file.

Declarations

Ethics approval and consent to participate

Not applicable

Consent for publication

Not applicable

Availability of data and materials

The computed datasets and prediction results are available in the Zenodo repository (10.5281/zenodo.18497525). The computational pipeline as well as the predictive

models are available in the GitHub repository (https://github.com/Schlieplab/HSS_HitFrequencyVectors).

Competing interests

The authors declare that they have no competing interests.

Funding

Not applicable

Authors' contributions

AS conceptualized the approach and supervised the research. NG and AS conceptualized and developed the methodology. NG performed the data curation, selected respectively designed models, loss functions and evaluation methods and implemented, respectively lead the implementation, of the software. She also analyzed and visualized the data and wrote the original draft. All authors reviewed and edited the submitted manuscript including data visualizations.

Acknowledgements

We thank Kian Jalilian for contributing to the C implementations and implementing components of the exact search scripts.

References

[1] Pockrandt C, Alzamel M, Iliopoulos CS, Reinert K. GenMap: ultra-fast computation of genome mappability. Bioinformatics. 2020 Apr;36(12):3687–3692. https://doi.org/10.1093/bioinformatics/btaa222.

[2] Derrien T, Estellé J, Marco Sola S, Knowles DG, Raineri E, Guigó R, et al. Fast Computation and Applications of Genome Mappability. PLoS ONE. 2012 Jan;7(1):e30377. https://doi.org/10.1371/journal.pone.0030377.

[3] Roy RS, Bhattacharya D, Schliep A. Turtle: Identifying frequent k -mers with cache-efficient algorithms. Bioinformatics. 2014 03;30(14):1950–1957. https://doi.org/10.1093/bioinformatics/btu132.

[4] Deorowicz S, Kokot M, Grabowski S, Debudaj-Grabysz A. KMC 2: fast and resource-frugal k-mer counting. Bioinformatics. 2015 Jan;31(10):1569–1576. https://doi.org/10.1093/bioinformatics/btv022.

[5] Kokot M, Długosz M, Deorowicz S. KMC 3: counting and manipulating k-mer statistics. Bioinformatics. 2017 May;33(17):2759–2761. https://doi.org/10.1093/bioinformatics/btx304.

[6] Manber U, Myers G. Suffix Arrays: A New Method for on-Line String Searches. In: Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '90. USA: Society for Industrial and Applied Mathematics; 1990. p. 319–327. Available from: https://dl.acm.org/doi/10.5555/320176.320218.

[7] Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology. 2009 Mar;10(3). https://doi.org/10.1186/gb-2009-10-3-r25.

[8] Kraska T, Beutel A, Chi EH, Dean J, Polyzotis N.: The Case for Learned Index Structures. arXiv. Available from: https://arxiv.org/abs/1712.01208.

[9] Kirsche M, Das A, Schatz MC. Sapling: accelerating suffix array queries with learned data models. Bioinformatics. 2020 Oct;37(6):744–749. https://doi.org/10.1093/bioinformatics/btaa911.

[10] Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998;86(11):2278–2324. https://doi.org/10.1109/5.726791.

[11] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Communications of the ACM. 2017 May;60(6):84–90. https://doi.org/10.1145/3065386.

[12] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2016. p. 770–778. Available from: http://dx.doi.org/10.1109/CVPR.2016.90.

[13] Alipanahi B, Delong A, Weirauch MT, Frey BJ. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. Nature Biotechnology. 2015 Jul;33(8):831–838. https://doi.org/10.1038/nbt.3300.

[14] Zhou J, Troyanskaya OG. Predicting effects of noncoding variants with deep learning–based sequence model. Nature Methods. 2015 Aug;12(10):931–934. https://doi.org/10.1038/nmeth.3547.

[15] Borchani H, Varando G, Bielza C, Larrañaga P. A survey on multi-output regression. WIREs Data Mining and Knowledge Discovery. 2015 Jul;5(5):216–233. https://doi.org/10.1002/widm.1157.

[16] Xu D, Shi Y, Tsang IW, Ong YS, Gong C, Shen X.: A Survey on Multi-output Learning. arXiv. Available from: https://arxiv.org/abs/1901.00248.

[17] Caruana R. Multitask Learning. Machine Learning. 1997 Jul;28(1):41–75. https://doi.org/10.1023/a:1007379606734.

[18] Schmid L, Gerharz A, Groll A, Pauly M.: Machine Learning for Multi-Output Regression: When should a holistic multivariate approach be preferred over separate univariate ones? arXiv. Available from: https://arxiv.org/abs/2201.05340.

[19] Aho T, Ženko B, Džeroski S, Elomaa T. Multi-Target Regression with Rule Ensembles. Journal of Machine Learning Research. 2012;13(78):2367–2407.

[20] Reyes O, Ventura S. Performing Multi-Target Regression via a Parameter Sharing-Based Deep Network. International Journal of Neural Systems. 2019 Oct;29(09):1950014. https://doi.org/10.1142/s012906571950014x.

[21] Emami S, Martínez-Muñoz G. Deep Learning for Multi-Output Regression Using Gradient Boosting. IEEE Access. 2024;12:17760–17772. https://doi.org/10.1109/access.2024.3359115.

[22] Consortium GR.: GCA 000001405.15 GRCh38 Sequences for Alignment Pipelines. Available from: ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/001/405/GCA_000001405.15_GRCh38/seqs_for_alignment_pipelines.ucsc_ids/GCA_000001405.15_GRCh38_no_alt_analysis_set.fna.gz.

[23] Baeza-Yates RA, Perleberg CH. Fast and practical approximate string matching. Information Processing Letters. 1996 Jul;59(1):21–27. https://doi.org/10.1016/0020-0190(96)00083-x.

[24] Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. Journal of Molecular Biology. 1990 Oct;215(3):403–410. https://doi.org/10.1016/s0022-2836(05)80360-2.

[25] Sinha P, Mizgiris V.: Pybloomfiltermmap3. Available from: https://github.com/prashnts/pybloomfiltermmap3.

[26] Limasset A, Rizk G, Chikhi R, Peterlongo P. Fast and Scalable Minimal Perfect Hashing for Massive Key Sets. vol. 75. Schloss Dagstuhl – Leibniz-Zentrum für Informatik; 2017. p. 25:1–25:16. Available from: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.SEA.2017.25.

[27] Jalilian K.: Quickmers. Available from: https://github.com/Schlieplab/quickmers.

[28] Myers G. A fast bit-vector algorithm for approximate string matching based on dynamic programming. Journal of the ACM. 1999 May;46(3):395–415. https://doi.org/10.1145/316542.316550.

[29] Kendall A, Gal Y, Cipolla R. Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE; 2018. p. 7482–7491. Available from: http://dx.doi.org/10.1109/CVPR.2018.00781.

[30] Buterez D. Scaling up DNA digital data storage by efficiently predicting DNA hybridisation using deep learning. Scientific Reports. 2021 Oct;11(1). https://doi.org/10.1038/s41598-021-97238-y.

[31] Makridakis S. Accuracy measures: theoretical and practical concerns. International Journal of Forecasting. 1993 Dec;9(4):527–529. https://doi.org/10.1016/0169-2070(93)90079-3.

[32] Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, et al. STAR: ultrafast universal RNA-seq aligner. Bioinformatics. 2012 Oct;29(1):15–21. https://doi.org/10.1093/bioinformatics/bts635.

[33] Sai Sundar KV, Bonta LR, Reddy B AK, Baruah PK, Sankara SS. Evaluating Training Time of Inception-v3 and Resnet-50, 101 Models using TensorFlow

across CPU and GPU. In: 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE; 2018. p. 1964–1968. Available from: http://dx.doi.org/10.1109/ICECA.2018.8474878.

[34] Jayaraman P, Mosbruger T, Hu T, Tairis NG, Wu C, Clark PM, et al. AnthOligo: automating the design of oligonucleotides for capture/enrichment technologies. Bioinformatics. 2020 Jun;36(15):4353–4356. https://doi.org/10.1093/bioinformatics/btaa552.

[35] Yoshida T, Naito Y, Yasuhara H, Sasaki K, Kawaji H, Kawai J, et al. Evaluation of off-target effects of gapmer antisense oligonucleotides using human cells. Genes to Cells. 2019 Nov;24(12):827–835. https://doi.org/10.1111/gtc.12730.

[36] Hagedorn PH, Pontoppidan M, Bisgaard TS, Berrera M, Dieckmann A, Ebeling M, et al. Identifying and avoiding off-target effects of RNase H-dependent antisense oligonucleotides in mice. Nucleic Acids Research. 2018 May;46(11):5366–5380. https://doi.org/10.1093/nar/gky397.

Illustrations and figures

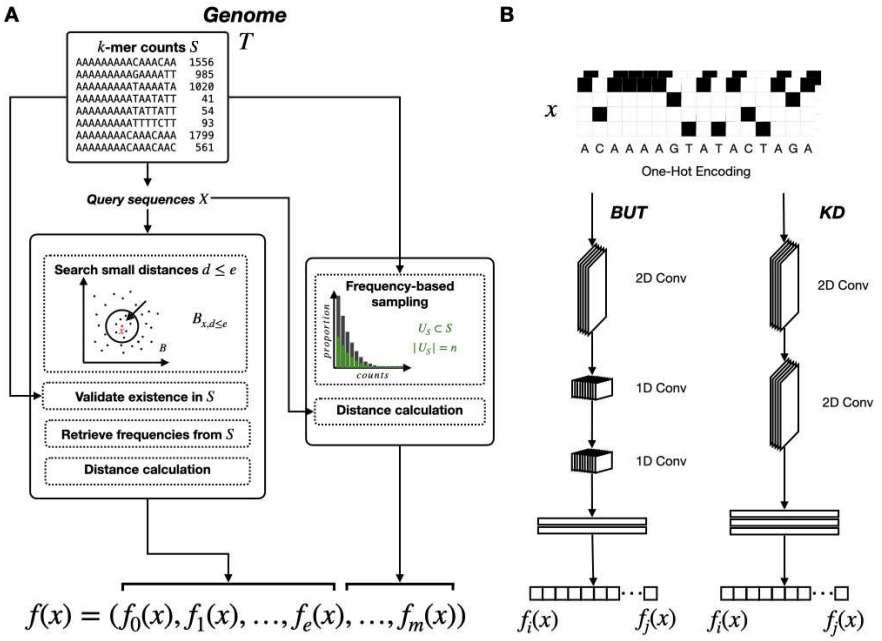


Fig. 1: Overview of HSS pipeline to generate HFV (A) and predictive models (B). HFVs are generated using an exact search for smaller distances and a representative sampled set for moderate to large distances. Calculated distances are binned by a predefined distance measure. Predictive CNN-based models predict partial or complete HFVs reducing the need for computational resources to generate HFVs.

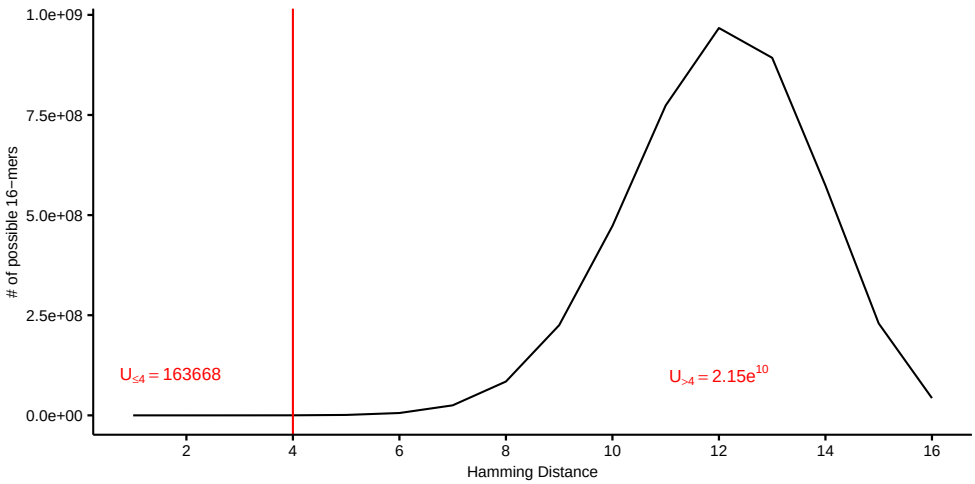


Fig. 2: Distributions of possible k-mers by Hamming distance to a query sequence for a 16-mer. The tail left of the red line $x = 4$ is explored via a search. The bulk of the distribution (to the right of the red line) is represented by a sample.

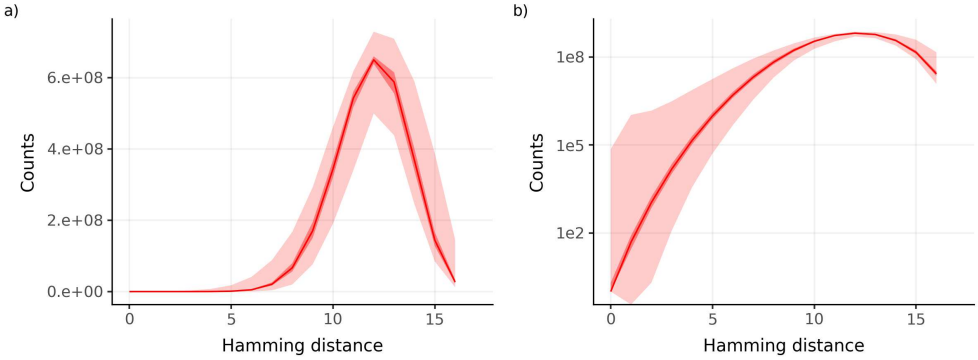


Fig. 3: Dataset distribution for **HDE** in original (a) and log-scale (b). Both graphs represents the range of values for each bin, where the vertical axis denotes the corresponding count. The darker red ribbon indicates the IQR, spanning from the first quartile (Q1) to the third quartile (Q3), capturing the middle 50 % of the data. The lighter red ribbon encompasses the full range from the minimum to the maximum observed values.

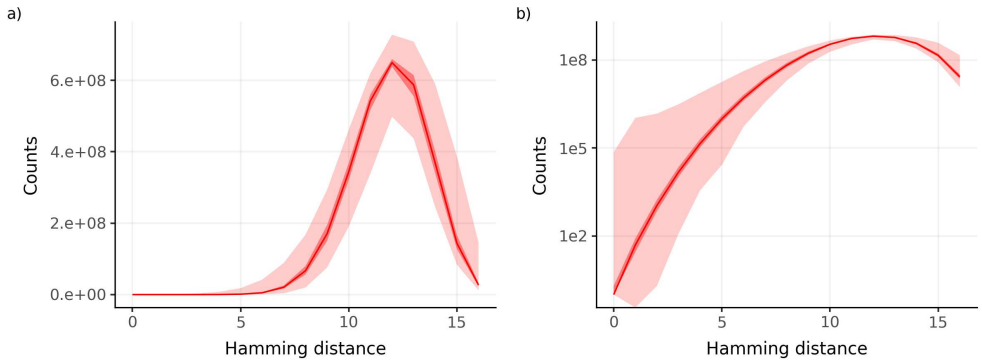


Fig. 4: Dataset distribution for **HDA** (using Linear Search) in original (a) and log-scale (b). Both graphs show the range of values for each bin, where the vertical axis denotes the corresponding count. The darker red ribbon indicates the IQR, spanning from the first quartile (Q1) to the third quartile (Q3), capturing the middle 50 % of the data. The lighter red ribbon encompasses the full range from the minimum to the maximum observed values.

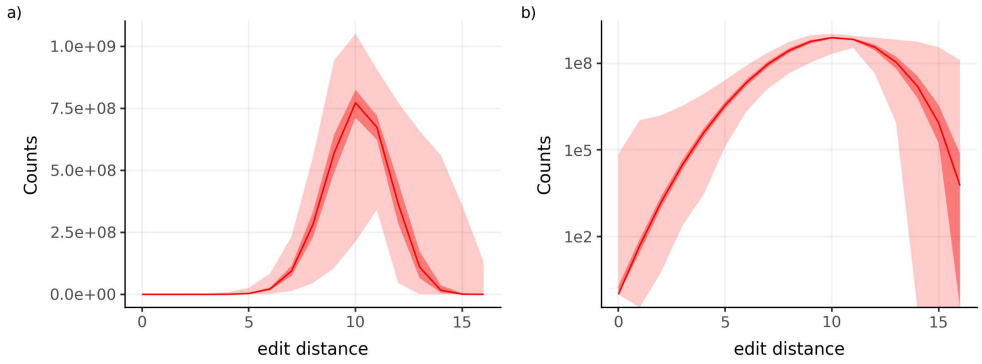


Fig. 5: Dataset distribution for **EDA** (using Linear Search) in original (a) and log-scale (b). Both graphs show the range of values for each bin, where the vertical axis denotes the corresponding count. The darker red ribbon indicates the IQR, spanning from the first quartile (Q1) to the third quartile (Q3), capturing the middle 50 % of the data. The lighter red ribbon encompasses the full range from the minimum to the maximum observed values.

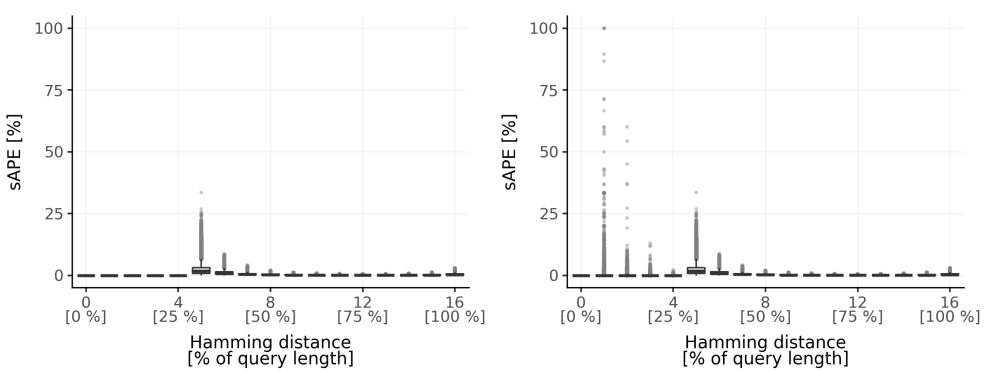


Fig. 6: Distribution of symmetric APEs per bin compared to exact Hamming HFVs for the linear search (a) and MPH (b) approach.

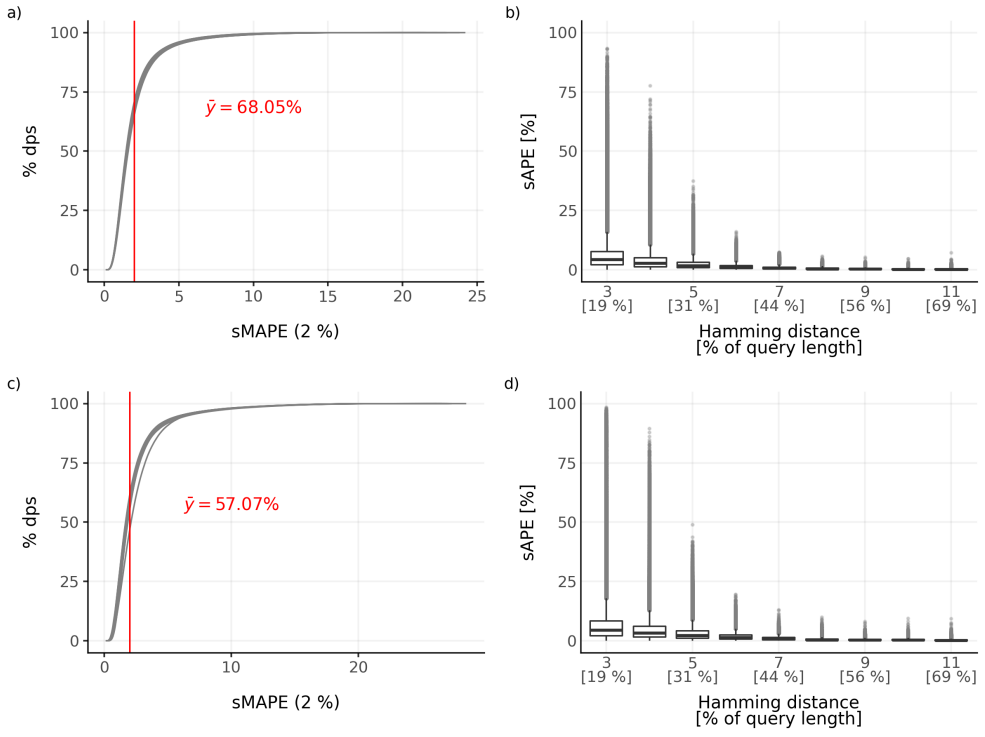


Fig. 7: Accuracy for prediction of HFVs from **HDE** for **KD** (a, b) and **BUT** (c, d). sMAPE plots (a, c) show the percentage of predicted HFVs with a sMAPE less or equals a given threshold (x-axis). Box plots (b, d) show the distributions of symmetric absolute percentage error (sAPE) per distance bins of the best performing fold from cross-fold validation. The x-axis denotes the Hamming distance to the query sequence in absolute values and as a percentage from the query length, $k = 16$.

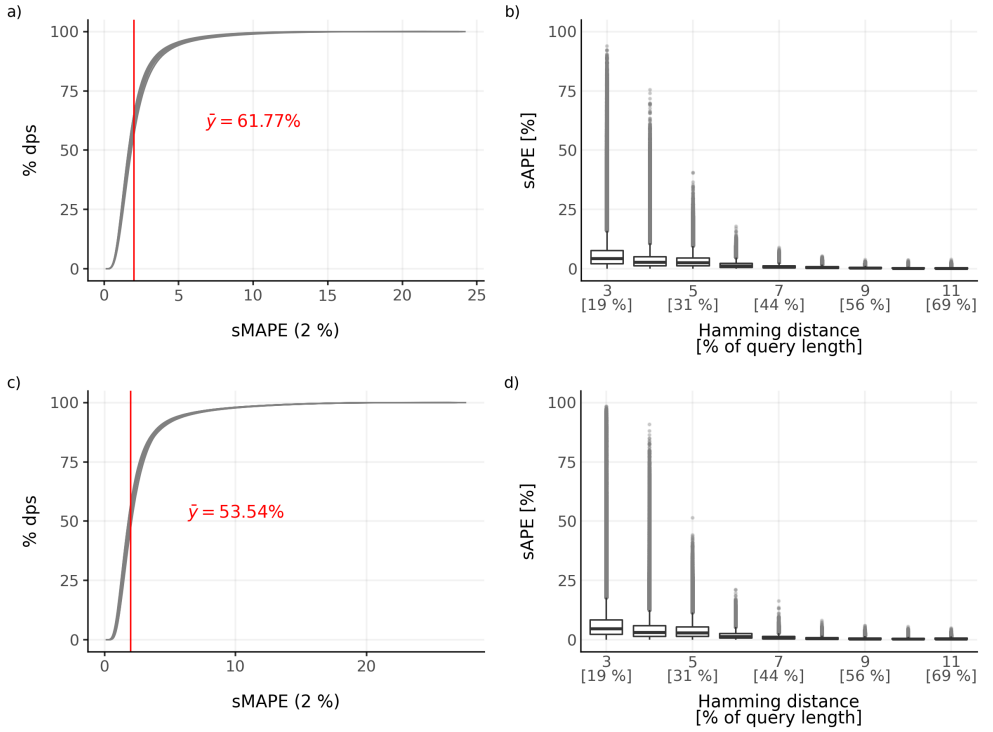


Fig. 8: Accuracy for prediction of HFVs **HDA** using **KD** model (a, b) and **BUT** model (c, d). sMAPE plots (a, c) show the percentage of predicted HFVs with a sMAPE less or equals a given threshold (x-axis). Box plots (b, d) show the distributions of sAPEs per distance bins of the best performing fold from cross-fold validation. The x-axis denotes the Hamming distance to the query sequence in absolute values and as a percentage from the query length, $k = 16$.

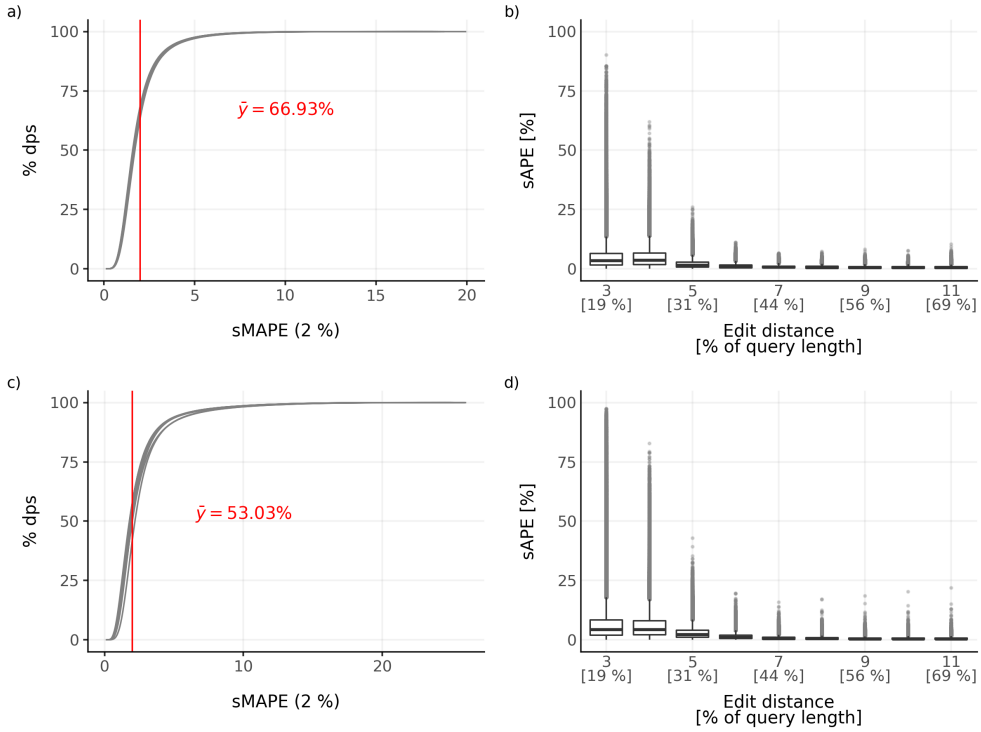


Fig. 9: Accuracy for prediction of HFVs from **EDA** using **KD** model (a, b) and **BUT** model (c, d). sMAPE plots (a, c) show the percentage of predicted HFVs with a sMAPE less or equals a given threshold (x-axis). Box plots (b, d) show the distributions of sAPEs per distance bins of the best performing fold from cross-fold validation. The x-axis denotes the edit distance to the query sequence in absolute values and as a percentage from the query length, $k = 16$.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- DLMappabilityPredictionHFVSupplement.pdf