Designing Representation Learning Methods For Molecular Sequences Analysis

Sarwan Ali

Committee Chair: Murray Patterson

Committee Members: Alexander Zelikovsky, Esra Akbas, José Bento

Georgia State University
November 25, 2024

Table of Contents

1. Introduction
2. Motivation
3. What We Offer
4. Category 1: Representation Learning
5. Category 2: Kernel Methods
6. Category 3: Hashing-Based Approximate Solutions
7. Category 4: Adversarial Attacks On Molecular Sequences

Introduction

Sequence data analysis :

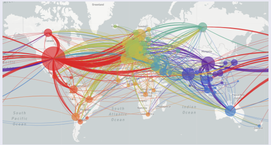
- Studies of Alterations in the protein sequence to classify and predict amino acid changes in SARS-CoV-2 are crucial in
  - Understanding the immune invasion and host-to-host transmission properties of SARS-CoV-2 and its variants
  - Identifying transmission patterns of each variant may help policymakers to prevent the rapid spread
  - May help in vaccine design and efficacy
- Unravel the mysteries of genetic info & its functional implications
- Phylogenetic tree construction-based methods - a Traditional way to trace evolution.
- Later Machine Learning and Deep Learning played major role.

Motivation

- In-depth studies of alterations in the protein sequence to classify and predict amino acid changes in SARS-CoV-2 are crucial in
  - Understanding the immune invasion and host-to-host transmission properties of SARS-CoV-2 and its variants
  - Knowledge of mutations and variants will help identify transmission patterns - facilitate public health measures
  - This will also help in vaccine design and efficacy
- Understanding biological sequence classification can unravel the mysteries of genetic information and its functional implications.
- Improve performance and reduce computational cost.
- Insights into the evolutionary relationships between organisms, helping us understand the origins and diversity of life on Earth.
- Advancements in personalized medicine, identifying genetic variants associated with diseases and predict patient responses to treatments.

Real World Application

- Genomic surveillance: Tracking the spread of pathogens in terms of genomic content
- Real time identification of new and rapidly emerging coronavirus variants
- Track the spread of known coronavirus variants in new municipalities, regions, countries and continents



Challenges

- Mutations happen disproportionately in different regions of genome
- Since new variants (for coronavirus) are emerging, not much information is available about these variants
- Generating fixed-length feature vectors from variable-length sequences
- High dimensionality of generated embeddings (e.g. OHE)
- Challenges:
  - The computation time
  - Predictive Performance
  - Generalizability

What We Offer

- Four categories of solutions
  1. Representation Learning
  2. Kernel Methods
  3. Hashing-Based Approximate Solutions
  4. Adversarial Attacks On Molecular Sequences

What We Offer

Category 1: Representation Learning

Kernel Method

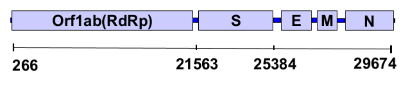
- A method that allows us to apply linear classifiers to non-linear problems by mapping non-linear data into a higher-dimensional space
- Kernel-based methods (e.g., SVM) are proven useful for several machine learning (ML) tasks such as sequence classification
- There are three challenges involved with kernel methods in general:
  - Kernel computation (requires exponential complexity to compute dot product)
  - scalability (storing n x n matrix in memory is not possible when n, the number of data points, is too large)
  - The usage of kernel matrices limited to kernel-based ML methods (difficult to generalize on non-kernel classifiers)
- The computational complexity problem can be solved using approximate methods
- The scalability issue remains for the typical kernel methods in general
- For non-kernel classifiers, we can use kernel PCA (could result in loss of information or computationally expensive)

Methodology (Spike2Vec)

- Use of Spike Sequence
- k-mers Generation from Spike Sequences (Spike2Vec)
- Frequency Vectors Generation
- Low Dimensional Representation of Data
- Classification and Clustering

Spike Sequence

- Since the spike protein is the entry point of the virus to the host cell, it is an important characterizing feature of a coronavirus
- The mRNA vaccines (e.g., Pfizer and Moderna) for COVID-19 are designed to target only the SARS-CoV-2 spike protein (unlike traditional vaccines which comprise an entire virome)
- Since the spike region is sufficient to characterize most of the important features of a viral sample, yet is much smaller in length, we focus on an embedding approach tailored to the spike region of the sequences



k-mers Spectrum

- We design a feature vector that contains the count of each k-mer in its respective spike sequence
- Each sequence A is over an alphabet Σ (amino acids of the spike sequence)
- These fixed length frequency vectors have length $|\Sigma|^k$ (the number of possible k-mers of a spike sequence)
- Since the total number of alphabets in our data are 21 (the number of amino acids), the length of each frequency vector becomes $21^3 = 9261$

Low Dimensional Representation

- For typical supervised and unsupervised classification/clustering tasks, dimensionality reduction methods such as principal component analysis, ridge regression, and lasso regression are used
  - Problem: Not scalable on bigger data
- Solution: User Kernel method with Kernel Trick
- Kernel Trick: It is used to generate features for an algorithm which depends on the inner product between only the pairs of input data points. The main idea is to avoid the need to map the input data (explicitly) to a high-dimensional feature space

Kernel Trick

- Kernel Trick relies on the following observation:
  - Any positive definite function f(a,b), where $a, b \in R^d$, defines an inner product and a lifting ϕ so that we can quickly compute the inner product between the lifted data points

    $$\langle \phi(a), \phi(b) \rangle = f(a, b) \tag{1}$$

    Above expression means that the inner product of the embeddings of two objects, a and b, is equal to a function of the similarity between the two objects.
- Drawback: In case of large training data, the kernel method suffers from large initial computational and storage costs.

Random Fourier Features (RFF)

- To overcome these computational problems, we use an approximate kernel method called random Fourier features (RFF)
- RFF maps the input data to a randomized low dimensional feature space (euclidean inner product space)

$$z : \mathcal{R}^d \rightarrow \mathcal{R}^D \tag{2}$$

- In this way, we can approximate the inner product between a pair of transformed points

$$f(a, b) = \langle \phi(a), \phi(b) \rangle \approx z(a)' z(b) \tag{3}$$

- z is low dimensional (unlike the lifting ϕ)
- In this way, we can transform the original input data with z, which acts as the approximate low dimensional embedding for the original data

Dataset

Pango Lin.	Region	Labels	No. Mut. S/Gen.	No. sequences
B.1.1.7	UK [1]	Alpha	8/17	976077
B.1.351	South Africa [1]	Beta	9/21	20829
B.1.617.2	India [2]	Delta	8/17	242820
P.1	Brazil [3]	Gamma	10/21	56948
B.1.427	California [4]	Epsilon	3/5	17799
AY.4	India [5]	Delta	-	156038
B.1.2	-	-	-	96253
B.1				78741
B.1.177	-	-	-	72298
B.1.1	-		-	44851
B.1.429	-	-	-	38117
AY.12	India [5]	Delta	-	28845
B.1.160	-	-	-	25579
B.1.526	New York [6]	Iota	6/16	25142
B.1.1.519	-	-	-	22509
B.1.1.214	-	-	-	17880
B.1.221	-	-	-	13121
B.1.258	-	-	-	13027
B.1.177.21	-	-	-	13019
D.2	-	-	-	12758
B.1.243	-	-	-	12510
R.1	-	-	-	10034

Table: The SARS-CoV-2 variants which were represented in more than 10,000 sequences (of the ≈2.5 million sequences). The S/Gen. column represents the number of mutations in the Spike (S) region / entire genome. The total number of amino acid sequences in our dataset is 2,519,386. The variants listed in this table comprise 1,995,195 sequences.

Results

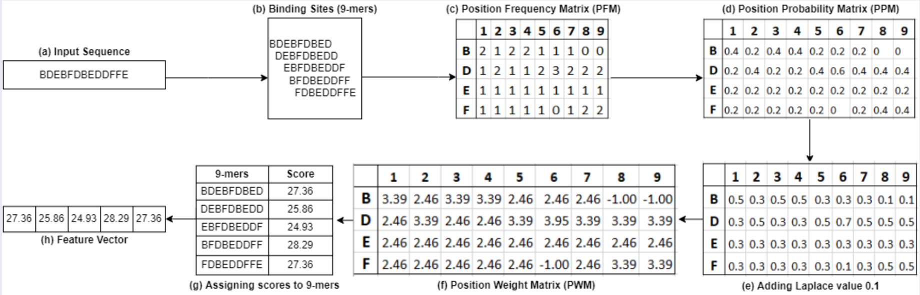
Approach	ML Algo.	Acc.	Prec.	Recall	F_1 (Weig.)	F_1 (Macro)	ROC-AUC	Training time (sec.)
OHE	NB	0.30	0.58	0.30	0.38	0.17	0.59	566.09
	LR	0.56	0.49	0.56	0.49	0.19	0.57	1309.06
	RC	0.56	0.47	0.56	0.48	0.17	0.56	110.76
Spike2Vec	NB	0.42	**0.79**	0.42	0.52	0.39	0.68	457.54
	LR	**0.68**	0.68	**0.68**	**0.64**	**0.49**	**0.69**	830.63
	RC	0.67	0.68	0.67	0.62	0.44	0.67	**95.73**

Table: Variants Classification Results (10% training set and 90% testing set) for the top 22 variants (1995195 spike sequences) listed in Table 19. Best values are shown in bold.

	F_1 Score (Weighted) for Different Variants				
Methods	Alpha	Beta	Delta	Gamma	Epsilon
OHE	0.0410	**0.0479**	0.5942	0.6432	0.0571
Spike2Vec	**0.9997**	0.0300	**0.8531**	**0.9680**	**0.2246**

Table: F_1 scores for five variants from the k-means clustering algorithm on all 1327 variants (2519386 spike sequences) in the GISAID dataset. Best values are in bold.

Methodology (PWM2Vec)



(a) Input Sequence

BDEBFDBEDDFFE

(b) Binding Sites (9-mers)

BDEBFDBED
DEBFDBEDD
EBFDBEDDF
BFDBEDDFF
FDBEDDFFE

(c) Position Frequency Matrix (PFM)

	1	2	3	4	5	6	7	8	9
B	2	1	2	2	1	1	1	0	0
D	1	2	1	1	2	3	2	2	2
E	1	1	1	1	1	1	1	1	1
F	1	1	1	1	1	0	1	2	2

(d) Position Probability Matrix (PPM)

	1	2	3	4	5	6	7	8	9
B	0.4	0.2	0.4	0.4	0.2	0.2	0.2	0	0
D	0.2	0.4	0.2	0.2	0.4	0.6	0.4	0.4	0.4
E	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
F	0.2	0.2	0.2	0.2	0.2	0	0.2	0.4	0.4

	1	2	3	4	5	6	7	8	9
B	0.5	0.3	0.5	0.5	0.3	0.3	0.3	0.1	0.1
D	0.3	0.5	0.3	0.3	0.5	0.7	0.5	0.5	0.5
E	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
F	0.3	0.3	0.3	0.3	0.3	0.1	0.3	0.5	0.5

(e) Adding Laplace value 0.1

	1	2	3	4	5	6	7	8	9
B	3.39	2.46	3.39	3.39	2.46	2.46	2.46	-1.00	-1.00
D	2.46	3.39	2.46	2.46	3.39	3.95	3.39	3.39	3.39
E	2.46	2.46	2.46	2.46	2.46	2.46	2.46	2.46	2.46
F	2.46	2.46	2.46	2.46	2.46	-1.00	2.46	3.39	3.39

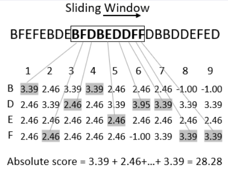
(f) Position Weight Matrix (PWM)

9-mers	Score
BDEBFDBED	27.36
DEBFDBEDD	25.86
EBFDBEDDF	24.93
BFDBEDDFF	28.29
FDBEDDFFE	27.36

(g) Assigning scores to 9-mers

27.36	25.86	24.93	28.29	27.36

(h) Feature Vector



Sliding Window

BFEFEBDE**BFDBEDDFF**DBBDDEFED

	1	2	3	4	5	6	7	8	9
B	3.39	2.46	3.39	3.39	2.46	2.46	2.46	-1.00	-1.00
D	2.46	3.39	2.46	2.46	3.39	3.95	3.39	3.39	3.39
E	2.46	2.46	2.46	2.46	2.46	2.46	2.46	2.46	2.46
F	2.46	2.46	2.46	2.46	2.46	-1.00	3.39	3.39	3.39

Absolute score = 3.39 + 2.46+...+ 3.39 = 28.28

Dataset (PWM2Vec)

Host Name	# of Sequences	Host Name	# of Sequences
Humans	1813	Rats	26
Environment	1034	Pangolins	21
Weasel	994	Hedgehog	15
Swine	558	Dolphin	7
Birds	374	Equine	5
Camels	297	Fish	2
Bats	153	Unknown	2
Cats	123	Python	2
Bovines	88	Monkey	2
Dogs	40	Cattle	1
Turtle	1		

Table: Dataset Statistics for 5558 coronavirus hosts.

Results (PWM2Vec)

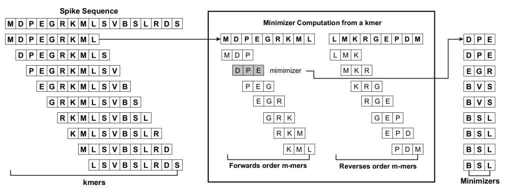
		Acc.	Prec.	Recall	F1	F1	ROC	Train
					(Weig.)	(Macro)	AUC	Time
								(Sec.)
OHE	SVM	0.81	0.82	0.81	0.81	0.69	0.82	389.128
	NB	0.68	0.81	0.68	0.66	0.65	0.80	56.741
	MLP	0.76	0.75	0.76	0.74	0.43	0.70	390.289
	KNN	0.79	0.78	0.79	0.78	0.54	0.77	16.211
	RF	0.83	0.83	0.82	0.82	0.66	0.82	151.911
	LR	0.82	0.83	0.83	0.82	0.70	0.83	48.786
	DT	0.82	0.83	0.82	0.81	0.63	0.80	21.581
k-mers	SVM	0.80	0.81	0.80	0.81	0.64	0.82	52.384
	NB	0.64	0.76	0.66	0.65	0.47	0.73	9.031
	MLP	0.81	0.82	0.81	0.81	0.52	0.77	44.982
	KNN	0.81	0.80	0.81	0.79	0.55	0.75	2.917
	RF	0.82	0.83	0.83	0.81	0.63	0.81	17.252
	LR	0.82	0.84	0.81	0.82	0.68	0.82	48.826
	DT	0.81	0.82	0.81	0.80	0.64	0.81	4.096
PWM2Vec	SVM	0.80	0.81	0.80	0.81	0.71	**0.85**	40.55
	NB	0.46	0.70	0.46	0.40	0.47	0.76	**1.56**
	MLP	0.80	0.81	0.81	0.79	0.57	0.78	17.28
	KNN	0.82	0.81	0.82	0.81	0.58	0.79	2.86
	RF	**0.85**	**0.85**	**0.85**	**0.84**	**0.72**	0.84	5.44
	LR	0.82	0.82	0.82	0.82	0.71	0.84	43.35
	DT	0.81	0.81	0.82	0.81	0.66	0.83	3.46

Methodology (Virus2Vec)

- Virus2Vec is a compact alignment-free embedding approach
- Eliminates the need for the sequence alignment
- Uses a fraction of the information as compared to a more traditional k-mers-based approach.
- Optimizes and reduces efforts in counting k-mers, which can be an expensive — and redundant — task.
- The process involves :
  - Compute minimizer using sliding window on k-mer
  - The lexicographically smallest is selected as the minimizer for that k-mer
  - For each minimizer, we compute a weight using the “Position Weight Matrix” (PWM) method.
  - We use the score of each m-mer (computed using the PWM-based approach) to the corresponding bin to get the final feature vector representation.

Feature Vector Representation

- To convert the sequences into fixed-length numerical representations, we use a recently proposed method called Spike2Vec [7].
- Spike2Vec generates a fixed-length numerical representation using the concept of k-mers (also called n-gram) for a sequence.

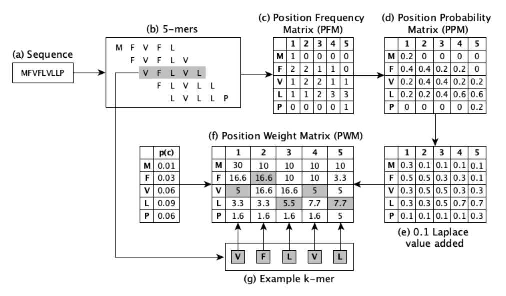


- Uses sliding window to generate k-mers of length k (window size).
- For a set of k-mers in a sequence, the feature vector of length $|\Sigma|^k$ (Σ is the set of alphabets "amino acids" or nucleotides), is generated using their count.

Feature Vector Representation

- To compute the minimizer, a sliding window is again used but this time on k-mer in both directions (forward and reverse).
- Lexicographically smallest is selected as the minimizer for that k-mer.
- Minimizers ignore many amino acids in each k-mer, only preserving a fraction of the m-mers, for which binning of these m-mers becomes much more efficient.
- For each minimizer, we compute a weight using the "Position Weight Matrix" (PWM) method.

Virus2Vec Workflow


(a) Sequence
MFVFLVLLP
(b) 5-mers
M F V F L
F V F L V
V F L V L
F L V L L
L V L L P
(c) Position Frequency Matrix (PFM)
(d) Position Probability Matrix (PPM)
(e) 0.1 Laplace value added
(f) Position Weight Matrix (PWM)
(g) Example k-mer
V F L V L

(c) Position Frequency Matrix (PFM)

	1	2	3	4	5
M	1	0	0	0	0
F	2	2	1	1	0
V	1	2	2	1	1
L	1	1	2	3	3
P	0	0	0	0	1

(d) Position Probability Matrix (PPM)

	1	2	3	4	5
M	0.2	0	0	0	0
F	0.4	0.4	0.2	0.2	0
V	0.2	0.4	0.4	0.2	0.2
L	0.2	0.2	0.4	0.6	0.6
P	0	0	0	0	0.2

(e) 0.1 Laplace value added

	1	2	3	4	5
M	0.3	0.1	0.1	0.1	0.1
F	0.5	0.5	0.3	0.3	0.1
V	0.3	0.5	0.5	0.3	0.3
L	0.3	0.3	0.5	0.7	0.7
P	0.1	0.1	0.1	0.1	0.3

	p(c)
M	0.01
F	0.03
V	0.06
L	0.09
P	0.06

(f) Position Weight Matrix (PWM)

	1	2	3	4	5
M	30	10	10	10	10
F	16.6	16.6	10	10	3.3
V	5	16.6	16.6	5	5
L	3.3	3.3	5.5	7.7	7.7
P	1.6	1.6	1.6	1.6	5

- After computing the Minimizers, a Position Frequency Matrix (PFM) is generated which contains the frequency count for each character at each position.
- We have 20 unique amino acids in the spike protein sequence dataset, our PFMs have 20 rows and $m = 3$ columns

Virus2Vec Workflow

- Normalize the PFM matrix to create a Position Probability Matrix (PPM) containing the probability of each amino acid at each position
- A position weight matrix (PWM)is then computed from the adjusted probability matrix, by computing the log-likelihood of each amino acid character c, i.e., $c \in A, C, \dots, Y$ for spike sequences or $c \in A, C, G, T$ for rabies virus sequences.
- PWM is used to compute the absolute scores for each individual minimizer generated from the sequence. It is the sum of the score of bases for the index.
- After getting the score for each m-mer, we generate a vector of length $|\Sigma|^m$. Using the score of each m-mer (computed using the PWM-based approach) to the corresponding bin to get the final feature vector representation.

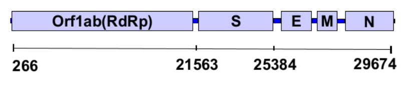
Dataset

- Spike Sequence from the SARS-CoV-2 virus
- Rabies sequences data

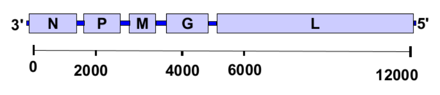
Name	Type	Source	Sequence Count	Classes	Sequence Length			
					Min	Max	Avg	Mode
Coronavirus Host Data	Spike protein sequences for COVID-19 hosts	GISAID, ViPR	5558	22	9	1584	1272.4	1273
Rabies Virus Data	Nucleotide genome sequences for rabies virus hosts	RABV-GLUE	20051	12	90	11930	1948.4	1353

Table: Data Statistics.

Dataset - Spike Sequences Structure



- The SARS-CoV-2 genome, of roughly 30K bps in length
- The structural protein further consists of the spike (or S) protein along with Envelope (E), Membrane (M) and Nucleocapsid (N) proteins.

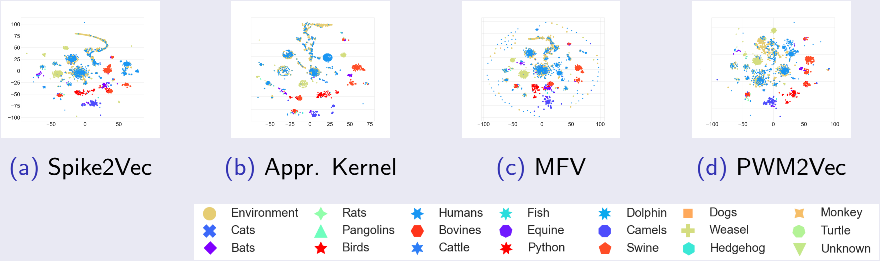


- The rabies genome is 12kb in length and encodes five proteins Nucleoprotein (N), Phosphoprotein (P), Matrix Protein (M), Glycoprotein (G), and Polymerase (L).

Properties of Different Embedding Methods

Embedding	Alignment Free	Low Dim Vectors	| Vector | (Spike/Rabies)	Space Efficient	Runtime Efficient	Details
One-Hot Encoding	✗	✗	69960 / 5600	✗	✗	length of OHE for a spike sequence 3498
Spike2Vec	✓	✓	8000 / 125	✓	✗	$\Sigma = 20$ and $k = 3$ (for Spike data)
Approx. Kernel	✓	✓	500 / 500	✗	✗	Dimensionality depends on Num of sequences
PWM2Vec	✗	✓	3490 / 125	✓	✓	Length of Spike Seq after alignment 3498 and $k = 9$
LSTM	✓	_	_	✗	✗	End-to-End DL architectures
GRU	✓	_	_	✗	✗	
CNN	✓	_	_	✗	✗	
ProteinBert	✓	_	_	✗	✗	Pretrained Protein language model using Transformer
MFV	✓	✓	8000 / 125	✓	✓	$k = 9$ and $m = 3$
Virus2Vec (ours)	✓	✓	8000 / 125	✓	✓	Proposed method $m = 3$

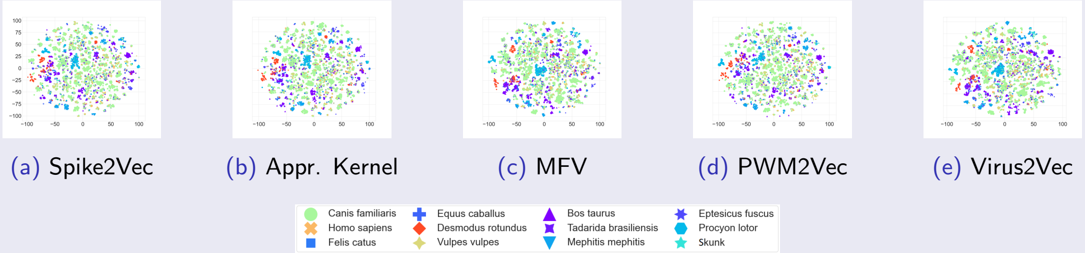
tSne plots - Host Data



(a) Spike2Vec (b) Appr. Kernel (c) MFV (d) PWM2Vec (e) Virus2Vec

- t-SNE plots for **Coronavirus Host** dataset.
- In all of them Environment & Human displays unambiguous grouping.
- Virus2Vec is able to preserve the structure of data in the same way as with the other existing embedding methods.

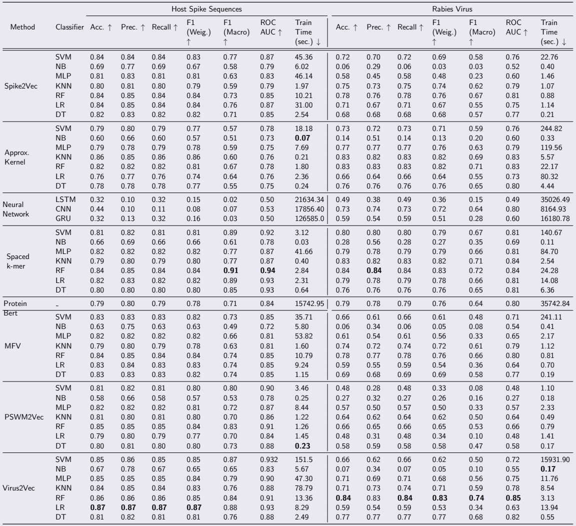
tSne plots - Rabies Data



(a) Spike2Vec (b) Appr. Kernel (c) MFV (d) PWM2Vec (e) Virus2Vec

- t-SNE plots for **Rabiesn Virus** dataset.
- Virus2Vec does not disturb the structure and even provides better clusters as compared to baseline embeddings.

Results



Method	Classifier	Host Spike Sequences							Rabies Virus						
		Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (sec.) ↓	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (sec.) ↓
Spike2Vec	SVM	0.84	0.84	0.84	0.83	0.77	0.87	45.36	0.72	0.70	0.72	0.69	0.58	0.76	22.76
	NB	0.69	0.77	0.69	0.67	0.58	0.79	6.02	0.06	0.29	0.06	0.03	0.03	0.52	0.40
	MLP	0.81	0.83	0.81	0.81	0.63	0.83	46.14	0.58	0.45	0.58	0.48	0.23	0.60	1.46
	KNN	0.80	0.81	0.80	0.79	0.59	0.79	1.97	0.75	0.73	0.75	0.74	0.62	0.79	1.07
	RF	0.84	0.85	0.84	0.84	0.73	0.85	10.21	0.78	0.76	0.78	0.76	0.67	0.81	0.88
	LR	0.84	0.85	0.84	0.84	0.76	0.87	31.00	0.71	0.67	0.71	0.67	0.55	0.75	1.14
	DT	0.82	0.83	0.82	0.82	0.71	0.85	2.54	0.68	0.68	0.68	0.68	0.57	0.77	0.21
Approx. Kernel	SVM	0.79	0.80	0.79	0.77	0.57	0.78	18.18	0.73	0.72	0.73	0.71	0.59	0.76	244.82
	NB	0.60	0.66	0.60	0.57	0.51	0.73	**0.07**	0.14	0.51	0.14	0.13	0.20	0.60	0.33
	MLP	0.79	0.78	0.79	0.78	0.59	0.75	7.69	0.77	0.77	0.77	0.76	0.63	0.79	119.56
	KNN	0.86	0.85	0.86	0.86	0.60	0.76	0.21	0.83	0.82	0.83	0.82	0.69	0.83	5.57
	RF	0.82	0.82	0.82	0.81	0.67	0.78	1.80	0.83	0.83	0.83	0.82	0.71	0.83	22.17
	LR	0.76	0.77	0.76	0.74	0.64	0.76	2.36	0.66	0.64	0.66	0.64	0.55	0.73	80.32
	DT	0.78	0.78	0.78	0.77	0.55	0.75	0.24	0.76	0.76	0.76	0.76	0.65	0.80	4.44
Neural Network	LSTM	0.32	0.10	0.32	0.15	0.02	0.50	21634.34	0.49	0.38	0.49	0.36	0.15	0.49	35026.49
	CNN	0.44	0.10	0.11	0.08	0.07	0.53	17856.40	0.73	0.74	0.73	0.72	0.64	0.80	8164.93
	GRU	0.32	0.13	0.32	0.16	0.03	0.50	126585.0	0.59	0.54	0.59	0.51	0.28	0.60	16180.78
Spaced k-mer	SVM	0.81	0.82	0.81	0.81	0.89	0.92	3.12	0.80	0.80	0.80	0.79	0.67	0.81	140.67
	NB	0.66	0.69	0.66	0.66	0.61	0.78	0.03	0.28	0.56	0.28	0.27	0.35	0.69	0.11
	MLP	0.82	0.82	0.82	0.82	0.77	0.87	41.66	0.79	0.78	0.79	0.79	0.66	0.81	84.70
	KNN	0.79	0.80	0.79	0.80	0.77	0.87	0.40	0.83	0.82	0.83	0.82	0.71	0.84	2.54
	RF	0.84	0.85	0.84	0.84	**0.91**	**0.94**	2.84	0.84	**0.84**	0.84	0.83	0.72	0.84	24.28
	LR	0.82	0.83	0.82	0.82	0.89	0.93	2.31	0.79	0.78	0.79	0.78	0.66	0.81	14.08
	DT	0.80	0.80	0.80	0.80	0.85	0.93	0.64	0.76	0.76	0.76	0.76	0.65	0.81	6.36
Protein Bert	-	0.79	0.80	0.79	0.78	0.71	0.84	15742.95	0.79	0.78	0.79	0.76	0.64	0.80	35742.84
MFV	SVM	0.83	0.83	0.83	0.82	0.73	0.85	35.71	0.66	0.61	0.66	0.61	0.48	0.71	241.11
	NB	0.63	0.75	0.63	0.63	0.49	0.72	5.80	0.06	0.34	0.06	0.05	0.08	0.54	0.41
	MLP	0.82	0.82	0.82	0.82	0.66	0.81	53.82	0.61	0.54	0.61	0.56	0.33	0.65	2.17
	KNN	0.79	0.80	0.79	0.78	0.63	0.81	1.60	0.74	0.72	0.74	0.72	0.61	0.79	1.12
	RF	0.84	0.85	0.84	0.84	0.74	0.85	10.79	0.78	0.77	0.78	0.76	0.66	0.80	0.81
	LR	0.83	0.84	0.83	0.83	0.74	0.85	9.24	0.59	0.55	0.59	0.54	0.36	0.64	0.70
	DT	0.83	0.83	0.83	0.82	0.74	0.85	1.15	0.69	0.68	0.69	0.69	0.58	0.77	0.19
PSWM2Vec	SVM	0.81	0.82	0.81	0.80	0.80	0.90	3.46	0.48	0.28	0.48	0.33	0.08	0.48	1.10
	NB	0.58	0.66	0.58	0.57	0.53	0.78	0.25	0.27	0.32	0.27	0.26	0.16	0.27	0.18
	MLP	0.82	0.82	0.82	0.81	0.72	0.87	8.44	0.57	0.50	0.57	0.50	0.33	0.57	2.33
	KNN	0.81	0.80	0.81	0.80	0.70	0.86	1.22	0.64	0.62	0.64	0.62	0.50	0.64	0.49
	RF	0.85	0.85	0.85	0.84	0.83	0.91	1.26	0.66	0.65	0.66	0.65	0.53	0.66	0.79
	LR	0.79	0.80	0.79	0.77	0.70	0.84	1.45	0.48	0.31	0.48	0.34	0.10	0.48	1.41
	DT	0.80	0.81	0.80	0.80	0.73	0.88	**0.23**	0.58	0.59	0.58	0.58	0.47	0.58	0.17
Virus2Vec	SVM	0.85	0.86	0.85	0.85	0.87	0.932	151.5	0.66	0.62	0.66	0.62	0.50	0.72	15931.90
	NB	0.67	0.78	0.67	0.65	0.65	0.83	5.67	0.07	0.34	0.07	0.05	0.10	0.55	**0.17**
	MLP	0.85	0.85	0.85	0.84	0.79	0.90	47.30	0.71	0.69	0.71	0.68	0.56	0.75	11.76
	KNN	0.84	0.85	0.84	0.83	0.76	0.88	78.79	0.71	0.73	0.74	0.71	0.59	0.78	8.54
	RF	0.86	0.86	0.86	0.85	0.84	0.91	13.36	**0.84**	0.83	**0.84**	**0.83**	**0.74**	**0.85**	3.13
	LR	**0.87**	**0.87**	**0.87**	**0.87**	0.88	0.93	8.29	0.59	0.54	0.59	0.53	0.34	0.63	13.94
	DT	0.81	0.82	0.81	0.81	0.76	0.88	2.49	0.77	0.77	0.77	0.77	0.68	0.82	0.55

Results

- Virus2Vec outperforms the SOTA methods
- The runtime to generate the embeddings makes it a huge factor in considering Virus2Vec over other embeddings.
- Virus2Vec outperforms not only the feature engineering-based baselines but also the neural network-based classifiers.
- The findings are reinforced by its visualization counterpart as well, as we saw in t-SNE plots also for Virus2Vec, it does not disrupt the general structure of the data because t-SNE is able to retain the structure of the data.

Results

Method	Coronavirus data Runtime ↓	Rabies virus data Runtime ↓
OHE	196.31 Sec.	**44.17** Sec.
Spike2Vec	1179.66 Sec.	259.86 Sec.
PWM2Vec	1506.63 Sec.	412.254 Sec.
Approx. Kernel	379.47 Sec.	179.47 Sec.
Virus2Vec	**90.65** Sec.	105.78 Sec.

Table: Runtime for generating feature vectors using different embedding methods for Coronavirus-Host data and Rabies Virus-Host dataset.

- Virus2Vec takes the least time to generate embeddings
- It takes 4 times less as compared to the Approximate Kernel method and 15 times less than PWM2Vec, which are comparable when accuracy is considered.

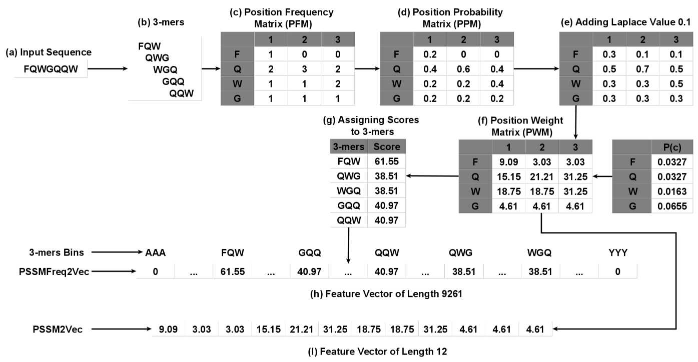
Conclusion and Future Work

- We propose an efficient sequence embedding approach Virus2Vec
- Uses an alignment-free method based on minimizers and PWM to classify genomic sequences.
- Virus2Vec not only performs better but is also an alignment-free approach.
- We show Virus2Vec comparable predictive performance and better runtimes.

Future Work

- Try on larger data to evaluate the scalability of Virus2Vec.
- Such an approach could also work even on *unassembled* (short read) data (not just unaligned), in a similar way that it works for metagenomics.

Methodology (PSSM2Vec)



(a) Input Sequence

FQWGQQW

(b) 3-mers

FQW
QWG
WGQ
GQQ
QQW

(c) Position Frequency Matrix (PFM)

	1	2	3
F	1	0	0
Q	2	3	2
W	1	1	2
G	1	1	1

(d) Position Probability Matrix (PPM)

	1	2	3
F	0.2	0	0
Q	0.4	0.6	0.4
W	0.2	0.2	0.4
G	0.2	0.2	0.2

(e) Adding Laplace Value 0.1

	1	2	3
F	0.3	0.1	0.1
Q	0.5	0.7	0.5
W	0.3	0.3	0.5
G	0.3	0.3	0.3

	P(c)
F	0.0327
Q	0.0327
W	0.0163
G	0.0655

(f) Position Weight Matrix (PWM)

	1	2	3
F	9.09	3.03	3.03
Q	15.15	21.21	31.25
W	18.75	18.75	31.25
G	4.61	4.61	4.61

(g) Assigning Scores to 3-mers

3-mers	Score
FQW	61.55
QWG	38.51
WGQ	38.51
GQQ	40.97
QQW	40.97

3-mers Bins	AAA		FQW		GQQ		QQW		QWG		WGQ		YYY
PSSMFreq2Vec	0	...	61.55	...	40.97	...	40.97	...	38.51	...	38.51	...	0

(h) Feature Vector of Length 9261

PSSM2Vec	9.09	3.03	3.03	15.15	21.21	31.25	18.75	18.75	31.25	4.61	4.61	4.61

(I) Feature Vector of Length 12

Figure: PSSMFreq2Vec and PSSM2Vec flow chart. For PSSMFreq2Vec, we build a feature vector from a sequence by computing PWM from k-mers and creating a zero feature vector of length $|\Sigma|^k$ and updating its values accordingly. For PSSM2Vec, we build the vector by flattening the PWM matrix in step (f).

Results (PSSM2Vec)

Method	ML. Algo.	Acc.	Prec.	Recall	F1 (Weig.)	ROC AUC	Train Time (Sec.)
OHE [8]	NB	0.31	0.58	0.31	0.38	0.60	6576.10
	LR	0.57	0.51	0.57	0.50	0.58	191296.4
	RC	0.56	0.49	0.56	0.49	0.57	8725.96
	KC	0.59	0.55	0.59	0.54	0.60	120316.7
Spike2Vec [7]	NB	0.59	0.79	0.59	0.60	0.78	4410.27
	LR	0.88	0.89	0.88	0.87	0.86	140245.19
	RC	0.85	0.83	0.85	0.82	0.82	2985.94
	KC	0.88	0.901	0.88	0.87	0.86	53000.61
PWM2Vec [9]	NB	0.46	0.80	0.46	0.56	0.71	590.13
	LR	0.72	0.71	0.72	0.69	0.72	858.06
	RC	0.70	0.71	0.70	0.67	0.70	138.74
	KC	0.81	0.79	0.81	0.79	0.74	2287.41
PSSMFreq2Vec	NB	0.14	0.73	0.14	0.14	0.71	4605.95
	LR	0.88	0.89	0.88	0.87	0.86	281995.3
	RC	0.86	0.88	0.86	0.84	0.83	7659.69
	KC	**0.89**	**0.905**	**0.89**	**0.88**	**0.87**	90316.71
PSSM2Vec	NB	0.09	0.55	0.09	0.11	0.53	**42.56**
	LR	0.81	0.77	0.81	0.77	0.75	363.13
	RC	0.76	0.70	0.76	0.70	0.64	106.60
	KC	0.82	0.81	0.82	0.81	0.79	695.107

Table: Variants Classification Results on the SARS-CoV-2 dataset for the top 22 variants (1995195 sequences). Best values are shown in bold.

Results (PSSM2Vec)

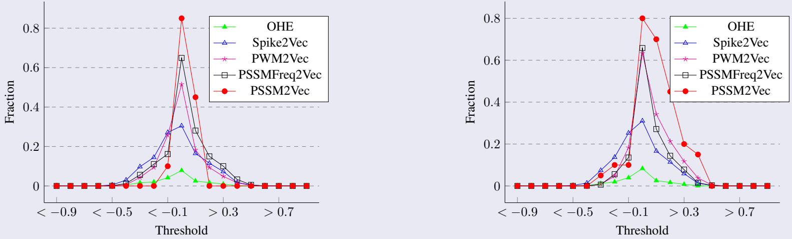
Dataset	Method	AUC
Coronavirus Host	OHE	0.3914
	Spike2Vec	0.4054
	PWM2Vec	0.4169
	PSSMFreq2Vec	0.4029
	PSSM2Vec	**0.4417**
SARS-CoV-2	OHE	0.2248
	Spike2Vec	0.2549
	PWM2Vec	**0.2850**
	PSSMFreq2Vec	0.2554
	PSSM2Vec	0.2819

Table: k-ary neighborhood agreement for $k = 1$ to 99.

Dataset	No. of Seq.	Method	Runtime
Coronavirus Host	5558	OHE	196.31 Sec.
		Spike2Vec	1179.66 Sec.
		PWM2Vec	1506.63 Sec.
		Approx. Kernel	379.47 Sec.
		PSSMFreq2Vec	908.12 Sec.
		PSSM2Vec	48.25 Sec.
SARS-CoV-2	2519386	OHE	> 3 days
		Spike2Vec	> 3 days
		PWM2Vec	> 3 days
		PSSMFreq2Vec	> 3 days
		PSSM2Vec	> 4 Hours

Table: Runtime for generating feature vectors using different methods.

Methodology (PSSM2Vec)



(a) Pearson Correlation

(b) Spearman Correlation

Figure: Correlation values for Coronavirus Host data. (a) and (b) show the fraction of features having correlation values greater than or less than the thresholds (on x-axis). The fractions are computed by taking denominator as the size of embeddings (69960 for OHE, 8000 for Spike2Vec, 3490 for PWM2Vec, 8000 for PSSMFreq2Vec, and 60 for PSSM2Vec).

What We Offer

Category 2: Kernel Methods

Methodology (String Kernel)

- After computing the feature vector, a kernel function is defined that measures the pairwise similarity between pairs of feature vectors
- Problem: Huge dimensionality of the feature vector
- Solution: Use kernel trick
- kernel values are directly evaluated instead of comparing indices.
- Given two feature vectors A and B, the kernel value for these vectors is simply the dot product of A and B
- Example: Given a k-mer, if the frequency of that k-mer in A is 2 and B is 3, its contribution towards the kernel value of A and B is simply $2 \cdot 3$
- The process of kernel value computation is repeated for each pair of sequences and hence we get a (symmetric) matrix (kernel matrix) containing a similarity score between each pair of sequences
- In this study, we use k=9 for the k-mers

Low Dimensional Representation

- Due to a high-dimensional kernel matrix, we use Kernel PCA (K-PCA) to select a subset of principal components
- These extracted principal components corresponding to each spike sequence act as the feature vector representations for the spike sequences

Dataset

Pango	Region	Labels	Num mutations	Num sequences in		
Lineage			S-gene/Genome	GISAID 1	GISAID 2	GISAID 3
B.1.1.7	UK [1]	Alpha	8/17	5979	5979	2055
B.1.351	South Africa [1]	Beta	9/21	124	124	133
P.1	Brazil [3]	Gamma	10/21	202	202	625
B.1.617.2	India [2]	Delta	8/17	596	596	44
B.1.427	California [4]	Epsilon	3/5	99	99	182

Table: Variants information and distribution in the three datasets. The S/Gen. column represents number of mutations on the S gene / entire genome.

Results

Approach	ML Algo.	Acc.	Prec.	Recall	F1 (weighted)	F1 (Macro)	ROC-AUC
One-Hot [8]	SVM	0.990	0.990	0.990	0.990	0.962	0.973
	NB	0.957	0.964	0.951	0.952	0.803	0.881
	MLP	0.972	0.971	0.975	0.974	0.881	0.923
	KNN	0.978	0.964	0.977	0.965	0.881	0.900
	RF	0.964	0.962	0.961	0.963	0.867	0.878
	LR	0.985	0.981	0.983	0.984	0.935	0.950
	DT	0.941	0.945	0.947	0.944	0.793	0.886
Kernel Approx.	SVM	**0.994**	**0.994**	**0.995**	**0.995**	**0.973**	**0.988**
	NB	0.987	0.985	0.985	0.986	0.901	0.912
	MLP	0.975	0.977	0.976	0.978	0.921	0.935
	KNN	0.979	0.967	0.979	0.967	0.887	0.904
	RF	0.981	0.987	0.988	0.980	0.944	0.945
	LR	0.992	0.990	0.993	0.992	0.991	0.990
	DT	0.985	0.981	0.985	0.987	0.898	0.944

Table: Variants Classification Results for the GISAID 1 Dataset. Best values are shown in bold.

Results

Approach	ML Algo.	Acc.	Prec.	Recall	F1 (weighted)	F1 (Macro)	ROC-AUC
One-Hot [8]	SVM	0.994	0.994	0.993	0.992	0.975	0.983
	NB	0.912	0.936	0.912	0.920	0.794	0.913
	MLP	0.970	0.970	0.970	0.969	0.880	0.921
	KNN	0.960	0.960	0.960	0.958	0.841	0.863
	RF	0.966	0.967	0.966	0.964	0.888	0.885
	LR	0.993	0.993	0.993	0.993	0.968	0.973
	DT	0.956	0.957	0.956	0.956	0.848	0.913
Kernel Approx.	SVM	**0.998**	**0.997**	**0.997**	**0.998**	**0.998**	**0.997**
	NB	0.985	0.988	0.985	0.984	0.946	0.967
	MLP	0.973	0.971	0.972	0.970	0.889	0.925
	KNN	0.965	0.962	0.963	0.967	0.845	0.867
	RF	0.990	0.992	0.991	0.996	0.978	0.977
	LR	0.997	0.994	0.996	0.997	0.991	0.993
	DT	0.991	0.990	0.994	0.996	0.952	0.963

Table: Variants Classification Results for the GISAID 2 Dataset. Best values are shown in bold.

Results

Approach	ML Algo.	Acc.	Prec.	Recall	F1 (weighted)	F1 (Macro)	ROC-AUC
One-Hot [8]	SVM	0.988	0.986	0.987	0.982	0.924	0.961
	NB	0.764	0.782	0.761	0.754	0.583	0.747
	MLP	0.947	0.941	0.944	0.942	0.813	0.898
	KNN	0.920	0.901	0.924	0.901	0.632	0.773
	RF	0.928	0.935	0.922	0.913	0.741	0.804
	LR	0.982	0.981	0.983	0.984	0.862	0.921
	DT	0.891	0.891	0.890	0.895	0.679	0.807
Kernel Approx.	SVM	**0.991**	**0.993**	**0.995**	**0.991**	**0.989**	**0.997**
	NB	0.864	0.922	0.861	0.884	0.783	0.887
	MLP	0.926	0.922	0.921	0.923	0.805	0.909
	KNN	0.947	0.921	0.942	0.934	0.701	0.826
	RF	0.975	0.971	0.971	0.972	0.904	0.918
	LR	0.991	0.990	0.994	0.990	0.983	0.992
	DT	0.960	0.969	0.964	0.967	0.812	0.891

Table: Variants Classification Results for the GISAID 3 Dataset. Best values are shown in bold.

Results

Variant	**Alpha**	**Beta**	**Delta**	**Gamma**	**Epsi.**	**Alpha**	**Beta**	**Delta**	**Gamma**	**Epsi.**
Alpha	5373	3	7	0	5	5371	9	5	0	3
Beta	6	110	0	0	0	13	103	0	0	0
Delta	6	0	523	0	0	8	0	521	0	0
Gamma	0	0	0	176	0	0	0	0	176	0
Epsilon	2	0	0	0	89	7	0	3	0	81

Table: Confusion matrices for SVM classifiers using Kernel approach (left) and using One-Hot approach (right) for the GISAID 1 dataset.

Importance of Amino Acids

$IG(Class, position) = H(Class) - H(Class|position)$,
where $H = \sum_{i \in Class} -p_i \log p_i$ is the entropy, and p_i is the probability of the class i.

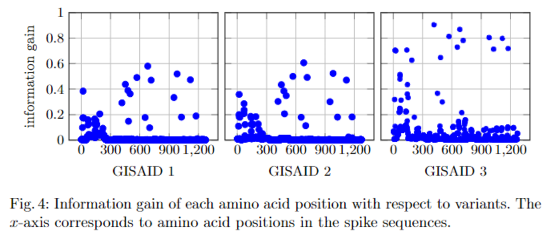


Fig. 4: Information gain of each amino acid position with respect to variants. The x-axis corresponds to amino acid positions in the spike sequences.

Figure: Information gain of each amino acid position with respect to variants. The x-axis corresponds to amino acid positions in the spike sequences.

Methodology (Efficient Kernel)

- k-m mismatch kernel
- Using Minimizers instead of *k*-mers
- Generating Ordered Minimizers-based sequence
- Using IG as pre-proceccong step to reduce dimensionality of input sequence

Dataset

Lineages	Region	Labels	No. Mut. S/Gen.	No. of sequences	
				GISAID-1	GISAID-2
B.1.1.7	UK [1]	Alpha	8/17	3369	3397
B.1.617.2	India [2]	Delta	8/17	875	878
AY.4	India [5]	Delta	-	593	516
B.1.2	-	-	-	333	350
B.1				292	276
B.1.177	Spain	-	-	243	281
P.1	Brazil [3]	Gamma	10/21	194	201
B.1.1	-		-	163	166
B.1.429	California	Epsilon	3/5	107	142
B.1.526	New York [6]	Iota	6/16	104	82
AY.12	India [5]	Delta	-	101	82
B.1.160	-	-	-	92	88
B.1.351	South Africa [1]	Beta	9/21	81	62
B.1.427	California [4]	Epsilon	3/5	65	62
B.1.1.214	-	-	-	64	64
B.1.1.519	-	-	-	56	88
D.2	-	-	-	55	45
B.1.221	-	-	-	52	41
B.1.177.21	-	-	-	47	56
B.1.258	-	-	-	46	42
B.1.243	-	-	-	36	40
R.1	-	-	-	32	41
Total	-	-	-	7000	7000

Results

		Acc.	Prec.	Recall	F1 (Weig.)	F1 (Macro)	ROC AUC	Train Time (Sec.)
OMK	SVM	0.85 ± 0.0015	0.83 ± 0.0041	0.85 ± 0.0015	0.83 ± 0.0023	0.62 ± 0.0110	0.81 ± 0.0037	33.9 ± 0.2053
	NB	0.74 ± 0.0067	0.8 ± 0.0066	0.74 ± 0.0075	0.76 ± 0.0070	0.59 ± 0.0122	0.8 ± 0.0080	0.13 ± 0.2208
	MLP	0.83 ± 0.0035	0.82 ± 0.0474	0.83 ± 0.0035	0.82 ± 0.0047	0.61 ± 0.0158	0.8 ± 0.0062	21.77 ± 0.0128
	KNN	0.81 ± 0.0091	0.81 ± 0.0058	0.81 ± 0.0091	0.8 ± 0.0077	0.63 ± 0.0225	0.8 ± 0.0124	0.31 ± 1.1609
	RF	0.862 ± 0.0052	0.85 ± 0.0075	0.862 ± 0.0052	0.84 ± 0.0060	0.67 ± 0.0111	0.83 ± 0.0041	1.54 ± 0.0116
	LR	0.85 ± 0.0038	0.84 ± 0.0039	0.85 ± 0.0038	0.83 ± 0.0051	0.63 ± 0.0270	0.81 ± 0.0136	2.99 ± 0.0346
	DT	0.83 ± 0.0078	0.83 ± 0.0088	0.83 ± 0.0078	0.82 ± 0.0080	0.63 ± 0.0190	0.81 ± 0.0113	0.23 ± 0.0094
IGK	SVM	0.85 ± 0.0018	0.84 ± 0.0051	0.85 ± 0.0018	0.83 ± 0.0029	0.6 ± 0.0136	0.8 ± 0.0046	3.23 ± 0.2540
	NB	0.74 ± 0.0083	0.82 ± 0.0082	0.74 ± 0.0093	0.76 ± 0.0087	0.58 ± 0.0151	0.8 ± 0.0099	0.1 ± 0.2731
	MLP	0.83 ± 0.0043	0.82 ± 0.0586	0.83 ± 0.0043	0.81 ± 0.0059	0.59 ± 0.0196	0.79 ± 0.0077	9.96 ± 0.0159
	KNN	0.82 ± 0.0113	0.82 ± 0.0072	0.82 ± 0.0113	0.81 ± 0.0096	0.59 ± 0.0278	0.79 ± 0.0153	0.34 ± 1.4364
	RF	0.84 ± 0.0064	0.83 ± 0.0093	0.84 ± 0.0064	0.82 ± 0.0074	0.59 ± 0.0138	0.8 ± 0.0051	1.36 ± 0.0143
	LR	0.85 ± 0.0047	0.84 ± 0.0048	0.85 ± 0.0047	0.83 ± 0.0063	0.61 ± 0.0334	0.8 ± 0.0168	1.7 ± 0.0428
	DT	0.83 ± 0.0097	0.82 ± 0.0109	0.83 ± 0.0097	0.81 ± 0.0100	0.58 ± 0.0234	0.79 ± 0.0140	0.21 ± 0.0116
OMK + IG	SVM	**0.867** ± 0.0016	0.85 ± 0.0045	**0.868** ± 0.0016	**0.85** ± 0.0025	0.66 ± 0.0119	0.83 ± 0.0040	20.83 ± 0.2216
	NB	0.75 ± 0.0072	0.83 ± 0.0072	0.75 ± 0.0082	0.77 ± 0.0076	0.61 ± 0.0131	0.82 ± 0.0087	**0.09** ± 0.2384
	MLP	0.84 ± 0.0038	0.84 ± 0.0511	0.84 ± 0.0038	0.83 ± 0.0051	0.65 ± 0.0171	0.83 ± 0.0067	13.26 ± 0.0138
	KNN	0.83 ± 0.0098	0.84 ± 0.0063	0.83 ± 0.0098	0.83 ± 0.0084	0.65 ± 0.0243	0.81 ± 0.0134	0.31 ± 1.2534
	RF	0.864 ± 0.0056	**0.86** ± 0.0081	0.865 ± 0.0056	0.84 ± 0.0065	**0.69** ± 0.0120	**0.84** ± 0.0045	1.26 ± 0.0125
	LR	0.865 ± 0.0041	0.85 ± 0.0042	0.86 ± 0.0041	0.84 ± 0.0055	0.63 ± 0.0292	0.82 ± 0.0147	2.08 ± 0.0374
	DT	0.84 ± 0.0085	0.84 ± 0.0095	0.84 ± 0.0085	0.84 ± 0.0087	0.65 ± 0.0205	0.83 ± 0.0122	0.19 ± 0.0101

Table: Average ± standard deviation classification results for GISAID-1 dataset. Best average values are shown in bold.

Kernel Computational Runtime

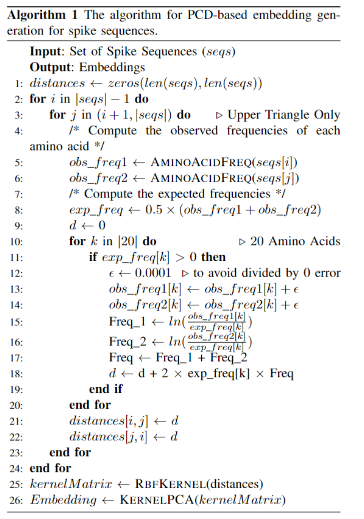
Method	Runtime (sec.)	# of Amino Acids
OMK	2163.02	3798
OMK + IG	1818.05	2184
String Kernel	1510.07	1274
IGK	1048.03	243

Table: Kernel computation runtime for different methods. Note that since both GISAID-1 and GISAID-2 dataset have 7000 sequences each, the kernel computation runtime for both datasets will be similar. The last column shows the number of amino acids in the input data for kernel matrices.

Methodology (PCD2Vec)

$$d = 2 \times exp\_freq \times (\ln(\frac{obs\_freq1}{exp\_freq}) + \ln(\frac{obs\_freq2}{exp\_freq})) \tag{4}$$

Methodology (PCD2Vec)



Algorithm 1 The algorithm for PCD-based embedding generation for spike sequences.

```
Input: Set of Spike Sequences ($seqs$)
Output: Embeddings
$distances \leftarrow zeros(len(seqs), len(seqs))$
for $i$ in $|seqs| - 1$ do
    for $j$ in $(i + 1, |seqs|)$ do    ▷ Upper Triangle Only
        /* Compute the observed frequencies of each amino acid */
        $obs\_freq1 \leftarrow$ AminoAcidFreq($seqs[i]$)
        $obs\_freq2 \leftarrow$ AminoAcidFreq($seqs[j]$)
        /* Compute the expected frequencies */
        $exp\_freq \leftarrow 0.5 \times (obs\_freq1 + obs\_freq2)$
        $d \leftarrow 0$
        for $k$ in $|20|$ do    ▷ 20 Amino Acids
            if $exp\_freq[k] > 0$ then
                $\epsilon \leftarrow 0.0001$    ▷ to avoid divided by 0 error
                $obs\_freq1[k] \leftarrow obs\_freq1[k] + \epsilon$
                $obs\_freq2[k] \leftarrow obs\_freq2[k] + \epsilon$
                Freq_1 $\leftarrow ln(\frac{obs\_freq1[k]}{exp\_freq[k]})$
                Freq_2 $\leftarrow ln(\frac{obs\_freq2[k]}{exp\_freq[k]})$
                Freq $\leftarrow$ Freq_1 + Freq_2
                $d \leftarrow$ d + 2 $\times$ exp_freq[k] $\times$ Freq
            end if
        end for
        $distances[i, j] \leftarrow d$
        $distances[j, i] \leftarrow d$
    end for
end for
$kernelMatrix \leftarrow$ RbfKernel(distances)
$Embedding \leftarrow$ KernelPCA($kernelMatrix$)
```

Dataset

Host	Count	Host	Count
human	957	pangolin	5
swine	785	duck	3
chicken	309	chimpanzee	3
camel	265	goose	2
bat	181	beluga Whale	2
cat	57	falcon	1
civet	5	-	-
Total	2575	-	-

Table: Host (class label) distribution in data.

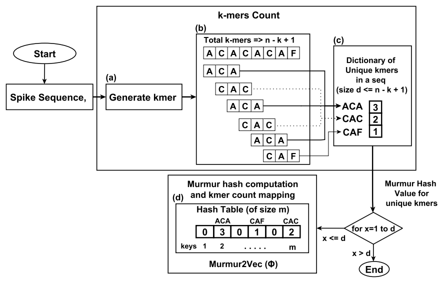
Dataset

Method	Classifier	Accuracy		Precision		Recall		F1 Weigh.		F1 Macro		ROC AUC	
		Avg.	Var.	Avg.	Var.	Avg.	Var.	Avg.	Var.	Avg.	Var.	Avg.	Var.
String Kernel	SVM	0.94	0.0007	0.95	0.0002	0.94	0.0007	0.94	0.0014	0.90	0.0006	0.95	0.0019
	NB	0.69	0.0019	0.86	0.0017	0.69	0.0019	0.72	0.0011	0.70	0.0019	0.86	0.0004
	MLP	0.82	0.0011	0.81	0.0030	0.82	0.0031	0.81	0.0025	0.44	0.0040	0.71	0.0023
	KNN	0.93	0.0007	0.93	0.0055	0.93	0.0097	0.92	0.0092	0.61	0.0023	0.82	0.0030
	RF	0.95	0.0010	0.96	0.0025	0.95	0.0010	0.95	0.0063	0.91	0.0059	0.95	0.0083
	LR	0.94	0.0008	0.95	0.0017	0.94	0.0071	0.94	0.0018	0.90	0.0067	0.95	0.0015
Protein Bert	-	0.92	0.0004	0.93	0.0002	0.92	0.0003	0.91	0.0001	0.86	0.0002	0.92	0.0003
PCD2Vec (Ours)	SVM	0.87	0.0098	0.90	0.0508	0.87	0.0098	0.86	0.0193	0.74	0.1030	0.87	0.0505
	NB	0.68	0.0470	0.87	0.0227	0.68	0.0470	0.71	0.0450	0.75	0.0724	0.90	0.0304
	MLP	0.84	0.0209	0.85	0.0219	0.84	0.0209	0.84	0.0236	0.64	0.0839	0.79	0.0392
	KNN	0.93	0.0107	0.94	0.0153	0.93	0.0107	0.93	0.0136	0.70	0.0663	0.90	0.0430
	RF	**0.97**	0.0085	**0.97**	0.0099	**0.96**	0.0085	**0.96**	0.0090	**0.98**	0.0842	**0.99**	0.0454
	LR	0.86	0.0116	0.87	0.0580	0.86	0.0116	0.84	0.0248	0.62	0.0802	0.80	0.0395

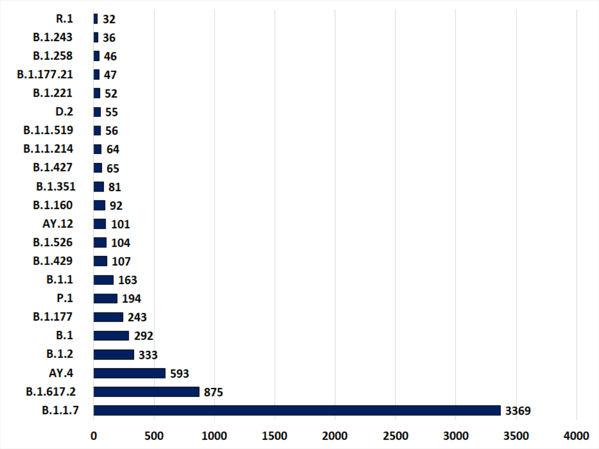
Category 3: Hashing-Based Solutions

Methodology (Murmur2Vec)

Multiply, Rotate, Multiply, Rotate “Murmur” hash, is a non-cryptographic hash function initially developed by Austin Appleby in 2008 [10]. For each 32-bit block, it first initializes a hash variable. Then the hash value is multiplied with a predefined constant, rotated left (every digit is shifted to the left, and the most significant bit becomes the least significant bit), multiplied with another constant, and the XOR operation is applied.



Dataset


R.1 32
B.1.243 36
B.1.258 46
B.1.177.21 47
B.1.221 52
D.2 55
B.1.1.519 56
B.1.1.214 64
B.1.427 65
B.1.351 81
B.1.160 92
AY.12 101
B.1.526 104
B.1.429 107
B.1.1 163
P.1 194
B.1.177 243
B.1 292
B.1.2 333
AY.4 593
B.1.617.2 875
B.1.1.7 3369
0
500
1000
1500
2000
2500
3000
3500
4000

Results

Collision	Method	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (sec.) ↓
40%	SVM	0.848	0.845	0.848	0.835	0.682	0.832	2.888
	NB	0.623	0.749	0.623	0.661	0.525	0.760	**0.274**
	MLP	0.763	0.750	0.763	0.751	0.544	0.772	8.245
	KNN	0.807	0.822	0.807	0.804	0.621	0.798	0.485
	RF	0.832	0.827	0.832	0.817	0.655	0.810	3.763
	LR	0.853	0.850	0.853	0.840	0.694	0.837	13.753
	DT	0.825	0.833	0.825	0.821	0.654	0.819	1.087
30%	SVM	0.850	0.843	0.850	0.835	0.705	0.849	3.141
	NB	0.579	0.718	0.579	0.622	0.468	0.737	0.282
	MLP	0.740	0.742	0.740	0.732	0.535	0.761	10.050
	KNN	0.779	0.800	0.779	0.772	0.605	0.796	0.650
	RF	0.821	0.816	0.821	0.804	0.661	0.806	3.737
	LR	0.856	0.849	0.856	0.842	**0.721**	**0.855**	14.438
	DT	0.818	0.818	0.818	0.813	0.655	0.818	1.116
20%	SVM	0.857	0.856	0.857	0.850	0.688	0.840	4.465
	NB	0.267	0.726	0.267	0.354	0.436	0.703	0.284
	MLP	0.760	0.775	0.760	0.760	0.553	0.780	12.374
	KNN	0.831	0.828	0.831	0.825	0.640	0.807	0.484
	RF	0.845	0.833	0.845	0.822	0.639	0.807	3.659
	LR	0.858	0.849	0.858	0.846	0.673	0.832	14.198
	DT	0.827	0.827	0.827	0.822	0.627	0.809	1.110
10%	SVM	0.845	0.840	0.845	0.835	0.687	0.839	3.709
	NB	0.266	0.730	0.266	0.352	0.404	0.697	0.295
	MLP	0.745	0.752	0.745	0.743	0.538	0.759	11.520
	KNN	0.800	0.803	0.800	0.790	0.603	0.794	0.509
	RF	0.827	0.815	0.827	0.808	0.637	0.804	3.487
	LR	0.842	0.836	0.842	0.830	0.678	0.833	13.791
	DT	0.811	0.815	0.811	0.805	0.606	0.807	1.268
8%	SVM	0.839	0.833	0.839	0.825	0.655	0.826	3.461
	NB	0.286	0.725	0.286	0.374	0.422	0.706	0.279
	MLP	0.741	0.743	0.741	0.735	0.511	0.753	12.437
	KNN	0.782	0.805	0.782	0.775	0.610	0.798	0.493
	RF	0.823	0.803	0.823	0.803	0.622	0.797	3.660
	LR	0.851	0.838	0.851	0.833	0.655	0.826	14.257
	DT	0.809	0.811	0.809	0.803	0.597	0.797	1.177
6%	SVM	0.859	0.857	0.859	0.851	0.696	0.846	3.914
	NB	0.294	0.729	0.294	0.382	0.421	0.709	0.291
	MLP	0.751	0.745	0.751	0.743	0.542	0.767	9.280
	KNN	0.824	0.819	0.824	0.815	0.607	0.793	0.489
	RF	0.841	0.835	0.841	0.818	0.651	0.811	3.645
	LR	**0.864**	**0.859**	**0.864**	**0.854**	0.692	0.845	14.034
	DT	0.830	0.832	0.830	0.826	0.621	0.816	1.262

Results

Collision	Method	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (sec.) ↓
4%	SVM	0.851	0.845	0.851	0.836	0.655	0.829	3.210
	NB	0.595	0.739	0.595	0.632	0.404	0.705	0.334
	MLP	0.758	0.746	0.758	0.746	0.513	0.750	10.174
	KNN	0.820	0.827	0.820	0.812	0.609	0.793	0.455
	RF	0.826	0.815	0.826	0.799	0.604	0.789	3.736
	LR	0.854	**0.855**	0.854	0.836	0.665	0.834	13.436
	DT	0.822	0.819	0.822	0.813	0.610	0.802	1.191
2%	SVM	0.847	0.853	0.847	0.835	0.682	0.841	3.482
	NB	0.585	0.734	0.585	0.634	0.471	0.731	0.281
	MLP	0.751	0.752	0.751	0.743	0.530	0.755	7.015
	KNN	0.810	0.808	0.810	0.799	0.616	0.798	0.462
	RF	0.824	0.821	0.824	0.804	0.628	0.801	3.638
	LR	0.853	0.845	0.853	0.839	0.679	0.837	15.208
	DT	0.820	0.825	0.820	0.814	0.623	0.809	1.225
1%	SVM	0.841	0.843	0.841	0.831	**0.690**	**0.845**	8.054
	NB	0.515	0.696	0.515	0.575	0.423	0.703	0.302
	MLP	0.731	0.735	0.731	0.727	0.504	0.750	9.604
	KNN	0.809	0.806	0.809	0.799	0.623	0.800	0.486
	RF	0.820	0.810	0.820	0.795	0.630	0.808	3.700
	LR	0.841	0.839	0.841	0.827	0.682	0.841	14.081
	DT	0.812	0.815	0.812	0.805	0.632	0.815	1.192
0.5%	SVM	**0.855**	0.842	**0.855**	**0.840**	0.682	0.841	2.803
	NB	0.281	0.725	0.281	0.367	0.405	0.702	0.298
	MLP	0.755	0.761	0.755	0.752	0.532	0.755	7.981
	KNN	0.801	0.798	0.801	0.794	0.582	0.776	0.483
	RF	0.823	0.802	0.823	0.804	0.627	0.795	3.623
	LR	0.853	0.838	0.853	0.837	0.668	0.832	13.777
	DT	0.809	0.804	0.809	0.802	0.586	0.794	1.139
0.25%	SVM	0.832	0.837	0.832	0.824	0.666	0.829	5.052
	NB	0.582	0.719	0.582	0.626	0.472	0.735	**0.271**
	MLP	0.742	0.728	0.742	0.731	0.507	0.747	8.963
	KNN	0.813	0.813	0.813	0.804	0.609	0.788	0.503
	RF	0.817	0.808	0.817	0.799	0.616	0.786	3.593
	LR	0.844	0.846	0.844	0.833	0.681	0.835	14.952
	DT	0.808	0.810	0.808	0.804	0.601	0.792	1.125
0%	SVM	0.843	0.838	0.843	0.831	0.663	0.830	4.563
	NB	0.615	0.740	0.615	0.655	0.485	0.740	0.323
	MLP	0.741	0.733	0.741	0.734	0.516	0.750	6.920
	KNN	0.787	0.797	0.787	0.778	0.599	0.789	0.506
	RF	0.825	0.802	0.825	0.797	0.616	0.801	3.717
	LR	0.849	0.842	0.849	0.834	0.662	0.830	14.764
	DT	0.811	0.800	0.811	0.801	0.597	0.796	1.171

Results

Methods	Embeddings	Runtime (Sec.) ↓	Vector Dimension ↓
SOTA	Spike2Vec	354.061	9261
	PWM2Vec	163.257	1266
	String Kernel	2292.245	500
	WDGRL	438.188	10
	Spaced *k*-mers	12901.808	9261
	AutoEncoder	572.271	500
Murmur2vec	40% Collision	23.352	9261
	30% Collision	24.668	12161
	20% Collision	27.027	19761
	10% Collision	28.957	41761
	8% Collision	28.344	52361
	6% Collision	29.955	68861
	4% Collision	35.068	105761
	2% Collision	39.994	190461
	1% Collision	48.458	319261
	0.5% Collision	103.240	593761
	0.25% Collision	101.827	1045961
	0% Collision	592.371	5421625

Results

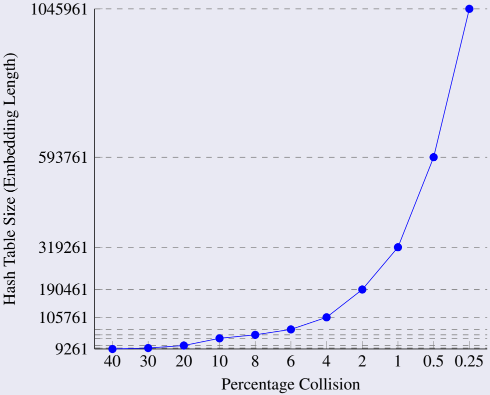


Figure: Relation between hash table size and allowed percentage collision for Murmur2Vec.

Methodology (BioSequence2Vec)

- Given a kmer and Alphabet $\Sigma => $ ACDEFGHIKLMNPQRSTVWXY
- For each character in k-mer
  - Find index i of the character in alphabet
  - Sort the k-mer
  - Find position n of character in sorted k-mer
  - The final numerical value v of the character is $i \times |\Sigma|^n$
  - Repeat the above process for all characters in k-mers and concat v to get nk-mer
- Repeat the process for all k-mers

Methodology (BioSequence2Vec)

Algorithm BioSequence2Vec Computation

```
Input: Set $\mathcal{S}$ of sequences, integers $k$, $p$, $\Sigma$, $t$
Output: Embedding $R$
function ComputeEmbedding($\mathcal{S}$, $k$, $p$, $\Sigma$, $t$)
    $R = []$
    for $X \in \mathcal{S}$ do
        $\hat{\mathbf{x}} = []$
        for $i = 1$ to $t$ do
            $a_0, a_1, a_2, a_3 \leftarrow \text{random}(0,\ p-1)$
            for $j = 1$ to $|X| - k + 1$ do
                $\alpha \leftarrow X[j : j + k]$
                $h \leftarrow a_0 + a_1\alpha_\Sigma + a_2\alpha_\Sigma^2 + a_3\alpha_\Sigma^3$
                $h \leftarrow (h \mod p) \mod 2$
                if $h = 0$ then
                    $\hat{\mathbf{x}}[i] \leftarrow \hat{\mathbf{x}}[i] - 1$
                else
                    $\hat{\mathbf{x}}[i] \leftarrow \hat{\mathbf{x}}[i] + 1$
            $\hat{\mathbf{x}}[i] \leftarrow \frac{1}{\sqrt{t}} \times \hat{\mathbf{x}}[i]$
        $R.\text{append}(\hat{\mathbf{x}})$
    Return $R$
```

Dataset

Dataset	Detail	Source	Total Sequences	Total classes	Sequence Length		
					Min	Max	Average
Spike7k	Aligned spike protein sequences to classify lineages of coronavirus in humans	[11]	7000	22	1274	1274	1274.00
Human DNA	Unaligned nucleotide sequences to classify gene family to which humans belong	[12]	4380	7	5	18921	1263.59

Baselines

Method	Category	Detail	Source	Alignment Free	Computationally Efficient	Space Efficient	Low Dim. Embedding
Spike2Vec	Feature Engineering	Take biological sequence as input and design fixed-length numerical embeddings	[7]	✓	✓	✓	✗
Spaced k-mers			[13]	✓	✓	✓	✗
PWM2Vec			[9]	✗	✓	✓	✓
WDGRL	Neural Network (NN)	Take one-hot representation of biological sequence as input and design NN-based embedding method by minimizing loss	[14]	✗	✗	✓	✓
AutoEncoder			[15]	✗	✗	✓	✓
String Kernel	Kernel Matrix	Designs $n \times n$ kernel matrix that can be used with kernel classifiers or with kernel PCA to get feature vector based on principal components	[16]	✓	✗	✗	✓
SeqVec	Pretrained Language Model	Takes biological sequences as input and fine-tunes the weights based on a pre-trained model to get final embedding	[17]	✓	✗	✓	✓
ProteinBERT	Pretrained Transformer	A pretrained protein sequence model to classify the given biological sequence using Transformer/Bert	[18]	✓	✗	✓	✓
BioSequence2Vec (ours)	Hashing	Takes biological sequence as input and design embeddings based on the kernel property of preserving pairwise distance	-	✓	✓	✓	✓

Results

		Spike7k							Human DNA						
Embeddings	Algo.	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (sec.) ↓	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (sec.) ↓
BioSequence2Vec (ours)	SVM	0.848	0.858	0.848	0.841	0.681	0.848	9.801	0.555	0.554	0.555	0.543	0.497	0.700	13.251
	NB	0.732	0.776	0.732	0.741	0.555	0.771	1.440	0.263	0.518	0.263	0.244	0.239	0.572	0.095
	MLP	0.835	0.825	0.835	0.825	0.622	0.819	13.893	0.583	0.598	0.583	0.571	0.541	0.717	70.463
	KNN	0.821	0.818	0.821	0.811	0.616	0.803	1.472	0.613	0.625	0.613	0.615	0.565	0.748	0.313
	RF	**0.863**	**0.867**	**0.863**	**0.854**	**0.703**	**0.851**	2.627	**0.786**	**0.816**	**0.786**	**0.787**	**0.779**	**0.846**	1.544
	LR	0.500	0.264	0.500	0.333	0.031	0.500	11.907	0.527	0.522	0.527	0.501	0.457	0.674	29.029
	DT	0.845	0.856	0.845	0.841	0.683	0.839	0.956	0.663	0.666	0.663	0.664	0.639	0.795	4.064

Table: Classification results (averaged over 5 runs) on **Spike7k** and **Human DNA** datasets for different evaluation metrics. Best values are shown in bold.

Category 4: Adversarial Attacks

Methodology (Becnhmarking ML Robustness)

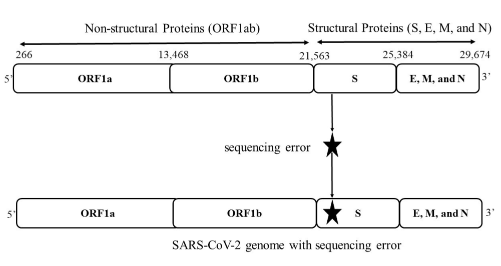

Non-structural Proteins (ORF1ab)
Structural Proteins (S, E, M, and N)
266
13,468
21,563
25,384
29,674
5'
ORF1a
ORF1b
S
E, M, and N
3'
sequencing error
5'
ORF1a
ORF1b
S
E, M, and N
3'
SARS-CoV-2 genome with sequencing error

Figure: The SARS-CoV-2 genome codes for several proteins, including the surface, or spike (S) protein, where mutations happen disproportionately often [5, 8]. Sequencing errors can bias the identification of a variant [19, 20]. The above figure represents the incorporation of a sequencing error that appears as a mutation in the spike region of the SARS-CoV-2 virus. While such a sequence is part of a lineage in the phylogenetic tree, now it will be classed as being part of a different lineage because of the sequencing error.

Methodology (Becnhmarking ML Robustness)

- PBSIM simulated data generation (Long Reads generation)
  - Depths 5, 10, 15, and 20
  - Depth means average number of times that each base in a DNA molecule is sequenced
  - Increasing the depth parameter can improve the accuracy of the sequencing data
- InSilicoSeq simulated data generation (Short Reads generation)
  - Number of reads 5000, 10000, 15000, and 20000
  - More reads means better quality

Dataset

Table: Dataset Statistics for different SARS-CoV-2 lineages in our data. After preprocessing, the total number of sequences (and corresponding lineages) is 8220.

Lineage	No. of sequences	Lineage	No. of sequences
AY.103	2271	AY.121	40
AY.44	1416	AY.75	37
AY.100	717	AY.3.1	30
AY.3	710	AY.3.3	28
AY.25	585	AY.107	27
AY.25.1	382	AY.34.1	25
AY.39	248	AY.46.6	21
AY.119	242	AY.98.1	20
B.1.617.2	175	AY.13	19
AY.20	130	AY.116.1	18
AY.26	107	AY.126	17
AY.4	100	AY.114	15
AY.117	94	AY.125	14
AY.113	94	AY.34	14
AY.118	86	AY.46.1	14
AY.43	85	AY.92	13
AY.122	84	AY.98	12
BA.1	79	AY.46.4	12
AY.119.2	74	AY.127	12
AY.47	73	AY.111	10
AY.39.1	70	-	-

Results

Table: Accuracy Results on 8220 (original) nucleotide sequences (without any error). The best values are shown in bold.

Embed. Method	ML Algo.	Acc.	Prec.	Recall	F1 weigh.	F1 Macro	ROC-AUC	Train. run-time (sec.)
k-mers Vector	SVM	**0.87**	**0.87**	**0.87**	**0.86**	**0.76**	**0.87**	7.43
	NB	0.03	0.05	0.03	0.02	0.05	0.55	0.09
	MLP	0.75	0.74	0.75	0.74	0.36	0.68	18.42
	KNN	0.73	0.73	0.73	0.71	0.48	0.71	2.04
	RF	0.82	0.85	0.82	0.80	0.67	0.78	2.17
	LR	0.86	0.85	0.86	0.85	0.70	0.84	8.67
	DT	0.67	0.67	0.67	0.66	0.42	0.71	0.27
PSSM Vector	SVM	0.28	0.08	0.28	0.12	0.01	0.50	3.14
	NB	0.01	0.01	0.01	0.00	0.01	0.52	**0.03**
	MLP	0.34	0.27	0.34	0.26	0.06	0.53	17.31
	KNN	0.32	0.28	0.32	0.28	0.13	0.55	0.33
	RF	0.33	0.30	0.33	0.31	0.16	0.57	1.60
	LR	0.28	0.08	0.28	0.12	0.01	0.50	0.68
	DT	0.29	0.28	0.29	0.28	0.13	0.56	0.06
Minimizer Vector	SVM	0.60	0.58	0.60	0.56	0.48	0.72	15.19
	NB	0.05	0.12	0.05	0.04	0.12	0.59	0.08
	MLP	0.57	0.52	0.57	0.53	0.30	0.64	26.32
	KNN	0.55	0.56	0.55	0.53	0.37	0.66	1.51
	RF	0.75	0.79	0.75	0.74	0.61	0.76	1.72
	LR	0.58	0.55	0.58	0.54	0.40	0.68	6.36
	DT	0.64	0.64	0.64	0.64	0.48	0.74	0.14

Results

Table: Robustness Results on PBSIM data with 5 and 10 as read depth. The best values are shown in bold.

		Depth: 5							Depth: 10						
Embed. Method	ML Algo.	Acc.	Prec.	Recall	F1 weigh.	F1 Macro	ROC-AUC	Train. run-time (sec.)	Acc.	Prec.	Recall	F1 weigh.	F1 Macro	ROC-AUC	Train. run-time (sec.)
k-mers Vector	SVM	0.01	0.00	0.01	0.00	0.00	0.502	16.48	0.01	0.00	0.01	0.00	0.00	0.500	16.88
	NB	0.00	0.00	0.00	0.00	0.00	0.501	0.68	0.00	0.00	0.00	0.00	0.00	0.501	0.71
	MLP	0.282	0.083	0.285	0.123	0.01	0.505	23.65	0.02	0.00	0.02	0.00	0.00	0.507	16.86
	KNN	0.285	0.081	0.283	0.121	0.01	0.504	1.68	0.28	0.08	0.28	0.12	0.01	0.505	1.78
	RF	**0.289**	**0.085**	**0.289**	**0.124**	0.01	**0.509**	1.78	0.28	0.08	0.28	0.12	0.01	0.502	2.88
	LR	0.01	0.00	0.01	0.00	0.00	0.501	11.30	0.01	0.00	0.01	0.00	0.00	0.501	12.04
	DT	0.01	0.00	0.01	0.00	0.00	0.503	0.34	0.01	0.00	0.01	0.00	0.00	0.505	0.36
PSSM Vector	SVM	0.27	0.07	0.27	0.11	0.01	0.504	8.14	0.30	0.09	0.30	0.13	0.01	0.506	8.32
	NB	0.27	0.07	0.27	0.11	0.01	0.501	0.34	0.30	0.09	0.30	0.13	0.01	0.508	0.36
	MLP	0.27	0.07	0.27	0.11	0.01	0.506	7.47	0.30	0.09	0.30	0.13	0.01	0.503	7.90
	KNN	0.27	0.07	0.27	0.11	0.01	0.502	0.51	0.01	0.05	0.01	0.00	0.00	0.502	0.52
	RF	0.27	0.07	0.27	0.11	0.01	0.507	1.17	0.302	0.096	0.302	0.130	0.012	0.505	0.98
	LR	0.27	0.07	0.27	0.11	0.01	0.503	3.76	0.301	0.095	0.301	0.131	0.016	0.501	3.62
	DT	0.27	0.07	0.27	0.11	0.01	0.501	**0.02**	**0.304**	**0.099**	**0.304**	**0.136**	**0.017**	**0.509**	**0.02**
Minimizer Vector	SVM	0.27	0.07	0.26	0.11	0.01	0.506	5.22	0.27	0.08	0.27	0.12	0.01	0.501	4.91
	NB	0.26	0.07	0.27	0.11	0.265	0.502	0.43	0.27	0.08	0.27	0.12	0.01	0.504	0.34
	MLP	0.26	0.07	0.26	0.11	0.261	0.504	1.63	0.27	0.08	0.27	0.12	0.01	0.506	1.92
	KNN	0.26	0.07	0.26	0.11	0.263	0.506	0.62	0.08	0.01	0.08	0.01	0.00	0.503	0.69
	RF	0.26	0.07	0.26	0.11	**0.268**	0.501	0.67	0.27	0.08	0.27	0.12	0.01	0.502	0.77
	LR	0.26	0.07	0.26	0.11	0.267	0.502	0.69	0.27	0.08	0.27	0.12	0.01	0.504	0.67
	DT	0.26	0.07	0.26	0.11	0.266	0.505	0.17	0.27	0.08	0.27	0.12	0.01	0.501	0.26

Results

Table: Robustness Results on Illumina-based errored sequences with 5000 and 10000 short reads used in the simulation process. The best values are shown in bold.

		# of Short Reads: 5000							# of Short Reads: 10000						
Embed. Method	ML Algo.	Acc.	Prec.	Recall	F1 weigh.	F1 Macro	ROC-AUC	Train. run-time (sec.)	Acc.	Prec.	Recall	F1 weigh.	F1 Macro	ROC-AUC	Train. run-time (sec.)
k-mers Vector	SVM	0.68	0.66	0.68	0.66	0.49	0.73	6.75	0.732	0.72	0.71	0.722	0.55	0.76	10.76
	NB	0.69	0.73	0.69	0.71	0.571	**0.80**	0.31	0.72	0.72	0.72	0.721	0.53	**0.77**	0.32
	MLP	0.68	0.65	0.68	0.66	0.34	0.66	75.93	0.68	0.65	0.68	0.66	0.32	0.65	27.84
	KNN	**0.73**	**0.73**	**0.73**	**0.72**	**0.574**	0.76	0.75	0.731	0.72	**0.733**	**0.727**	**0.56**	0.76	0.68
	RF	0.72	0.72	0.72	0.70	0.51	0.72	2.44	**0.738**	**0.73**	0.731	0.71	0.55	0.74	2.43
	LR	0.72	0.70	0.72	0.70	0.52	0.74	6.71	0.72	0.71	0.72	0.71	0.54	0.75	6.69
	DT	0.51	0.53	0.51	0.52	0.32	0.66	0.24	0.56	0.56	0.56	0.56	0.41	0.70	0.21
PSSM Vector	SVM	0.27	0.07	0.27	0.12	0.01	0.50	8.20	0.28	0.08	0.28	0.12	0.01	0.50	9.64
	NB	0.01	0.00	0.01	0.00	0.01	0.51	0.39	0.02	0.01	0.02	0.01	0.03	0.52	0.25
	MLP	0.32	0.22	0.32	0.24	0.06	0.52	10.30	0.34	0.25	0.34	0.26	0.08	0.53	12.72
	KNN	0.26	0.21	0.26	0.22	0.06	0.52	1.10	0.29	0.26	0.29	0.25	0.09	0.54	0.70
	RF	0.30	0.24	0.30	0.25	0.08	0.52	2.17	0.32	0.25	0.32	0.27	0.08	0.53	1.92
	LR	0.27	0.07	0.27	0.12	0.01	0.50	3.92	0.28	0.08	0.28	0.12	0.01	0.50	3.26
	DT	0.30	0.24	0.30	0.25	0.07	0.52	**0.121**	0.32	0.25	0.32	0.26	0.08	0.53	**0.07**
Minimizer Vector	SVM	0.52	0.47	0.52	0.46	0.30	0.64	11.75	0.54	0.50	0.54	0.49	0.34	0.66	7.45
	NB	0.05	0.27	0.05	0.04	0.09	0.63	0.20	0.07	0.37	0.07	0.08	0.14	0.64	0.19
	MLP	0.52	0.46	0.52	0.46	0.26	0.62	25.0	0.52	0.46	0.52	0.48	0.25	0.62	28.70
	KNN	0.55	0.55	0.55	0.53	0.39	0.67	0.52	0.57	0.57	0.57	0.56	0.47	0.70	0.56
	RF	0.65	0.67	0.65	0.63	0.46	0.70	1.75	0.68	0.69	0.68	0.66	0.56	0.74	1.60
	LR	0.51	0.46	0.51	0.46	0.28	0.63	2.91	0.53	0.49	0.53	0.48	0.34	0.65	2.90
	DT	0.47	0.47	0.47	0.47	0.31	0.65	0.128	0.54	0.54	0.54	0.54	0.42	0.70	0.10

Conclusion

- We discussed four categories of methods
- Some are scalable, some are generalizable, some are more efficient
- Overall, hashing-based method proved to be better in general
- Future work could include working towards privacy preservation (using Federated Learning) and visualization (t-SNE and UMAP)

Thank You

Questions!!

- S. Galloway *et al.*, “Emergence of sars-cov-2 b. 1.1. 7 lineage,” *Morbidity and Mortality Weekly Report*, vol. 70, no. 3, p. 95, 2021.

- P. Yadav *et al.*, “Neutralization potential of covishield vaccinated individuals sera against b. 1.617. 1,” *bioRxiv*, vol. 1, 2021.

- F. Naveca *et al.*, “Phylogenetic relationship of sars-cov-2 sequences from amazonas with emerging brazilian variants harboring mutations e484k and n501y in the spike protein,” *Virological. org*, vol. 1, 2021.

- W. Zhang *et al.*, “Emergence of a novel sars-cov-2 variant in southern california,” *Jama*, vol. 325, no. 13, pp. 1324–1326, 2021.

- SARS-CoV-2 Variant Classifications and Definitions, https://www.cdc.gov/coronavirus/2019-ncov/variants/variant-info.html, 2021, [Online; accessed 29-December-2021].

- A. West Jr *et al.*, “Detection and characterization of the sars-cov-2 lineage b. 1.526 in new york,” *bioRxiv*, 2021.

S. Ali and M. Patterson, “Spike2vec: An efficient and scalable embedding approach for covid-19 spike sequences,” in *IEEE International Conference on Big Data*, 2021, pp. 1533–1540.

K. Kuzmin, A. E. Adeniyi, A. K. DaSouza Jr, D. Lim, H. Nguyen, N. R. Molina, L. Xiong, I. T. Weber, and R. W. Harrison, “Machine learning methods accurately predict host specificity of coronaviruses based on spike sequences alone,” *Biochemical and Biophysical Research Communications*, vol. 533, no. 3, pp. 553–558, 2020.

S. Ali, B. Bello, P. Chourasia, R. T. Punathil, Y. Zhou, and M. Patterson, “Pwm2vec: An efficient embedding approach for viral host specification from coronavirus spike sequences,” *MDPI Biology*, 2022.

A. Appleby, “Murmurhash 2.0,” 2008.

GISAID Website, https://www.gisaid.org/, 2021, [Online; accessed 29-December-2021].

Human DNA, https://www.kaggle.com/code/nageshsingh/demystify-dna-sequencing-with-machine-learning/data, [Online; accessed 10-October-2022].

R. Singh, A. Sekhon *et al.*, “Gakco: a fast gapped k-mer string kernel using counting,” in *Joint ECML and Knowledge Discovery in Databases*, 2017, pp. 356–373.

J. Shen, Y. Qu, W. Zhang, and Y. Yu, “Wasserstein distance guided representation learning for domain adaptation,” in *AAAI conference on artificial intelligence*, 2018.

J. Xie, R. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” in *International conference on machine learning*, 2016, pp. 478–487.

M. Farhan, J. Tariq, A. Zaman, M. Shabbir, and I. Khan, “Efficient approximation algorithms for strings kernel based sequence classification,” in *Advances in neural information processing systems (NeurIPS)*, 2017, pp. 6935–6945.

M. Heinzinger, A. Elnaggar, Y. Wang, C. Dallago, D. Nechaev, F. Matthes, and B. Rost, “Modeling aspects of the language of life through transfer-learning protein sequences,” *BMC bioinformatics*, vol. 20, no. 1, pp. 1–17, 2019.

N. Brandes, D. Ofer, Y. Peleg, N. Rappoport, and M. Linial, “Proteinbert: A universal deep-learning model of protein sequence and func.” *Bioinformatics*, vol. 38, no. 8, 2022.

- Y. Huang, C. Yang, X.-f. Xu, W. Xu, and S.-w. Liu, “Structural and functional properties of sars-cov-2 spike protein: potential antivirus drug development for covid-19,” *Acta Pharmacologica Sinica*, vol. 41, no. 9, pp. 1141–1149, 2020.

- M. M. Arons, K. M. Hatfield, S. C. Reddy, A. Kimball, A. James, J. R. Jacobs, J. Taylor, K. Spicer, A. C. Bardossy, L. P. Oakley *et al.*, “Presymptomatic sars-cov-2 infections and transmission in a skilled nursing facility,” *New England journal of medicine*, vol. 382, no. 22, pp. 2081–2090, 2020.