Molecular Sequence Analysis And Role of AI

Sarwan Ali

Georgia State University
June 24, 2024

Table of Contents

1. Background
2. Motivation
3. Challenges
4. Methodology
5. Chaos Game Representation (CGR)
6. Dataset
7. Results
8. Conclusion and Future Work

Background

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

Methods :

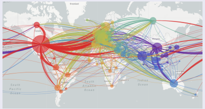
- Phylogenetic tree construction-based methods - a Traditional way to trace evolution.
- Later Machine Learning and Deep Learning played a major role

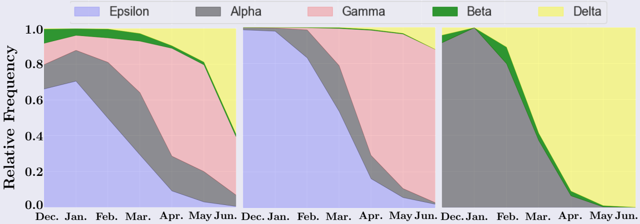
Motivation

- 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 predicting 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





(a) All Countries (b) USA (c) India

Statistical Analysis

We compute Information Gain (IG) between each attribute (amino acid position) and the class (variant). The IG is defined as

$$IG(Class, position) = H(Class) - H(Class|position) \tag{1}$$

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

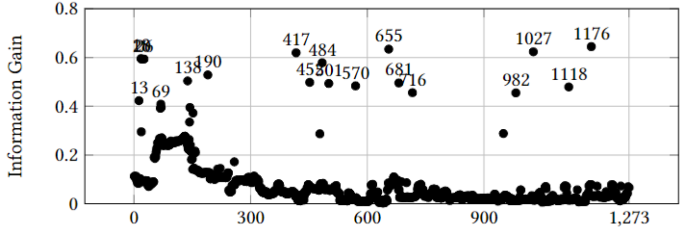


Figure: IG for AA with respect to variants. The x-axis corresponds to AA positions in a spike sequence.

Categories of Solutions

1. Kernel-based Methods
2. Embedding-based methods
3. Sequence-to-Image transformation

Challenges

- For enabling ML/DL-based analysis, biological sequences need to be transformed into numerical representations.
- But usually the numerical feature embedding generation methods undergo sparsity and curse of dimensionality challenges.
- State-of-the-Art DL classifiers perform suboptimal on tabular data compared to tree-based methods due to their interpretability, robustness, efficiency, and feature handling capabilities..

Problems

- Variable lengths of sequences
- Capturing both local and global structures
- Traditional methods (e.g. Phylogenetic Trees) are computationally expensive
- Mutations happen disproportionally

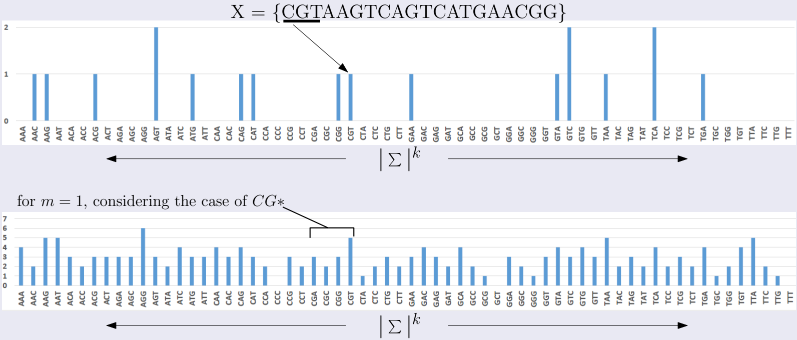
Kernel-based Solution

***k*-spectrum and *k*, *m*-mismatch kernel:** Given a sequence X over alphabet Σ, the k, m-mismatch spectrum of X is a $|\Sigma|^k$-dimensional vector, $\Phi_{k,m}(X)$ of number of times each possible k-mer occurs in X with at most m mismatches. Formally,

$$\Phi_{k,m}(X) = (\Phi_{k,m}(X)[\gamma])_{\gamma \in \Sigma^k} = \left(\sum_{\alpha \in X} I_m(\alpha, \gamma) \right)_{\gamma \in \Sigma^k}, \tag{2}$$

where $I_m(\alpha, \gamma) = 1$, if α belongs to the set of k-mers that differ from γ by at most m mismatches, i.e. the Hamming distance between α and γ, $d(\alpha, \gamma) \leq m$. Note that for $m = 0$, it is known as *k-spectrum* of X.

Kernel-based Solution



$X = \{\underline{CGT}AAGTCAGTCATGAACGG\}$

$|\Sigma|^k$

for $m = 1$, considering the case of $CG*$

$|\Sigma|^k$

Figure: The (k)-spectrum (top) and (k, m)-mismatch spectrum (bottom) for a DNA sequence X with $|X| = 20$, $\Sigma = \{A, C, G, T\}$, $k = 3$ and $m = 1$ are shown. For a selected k-mer $= CGT$, the (k)-spectrum computes the exact occurrences of the k-mer in X. The (k, m)-mismatch spectrum counts the occurrences of the k-mer in X up to Hamming distance of $m = 1$. We show a particular scenario of $CG*$, where $* \in \Sigma$ in this case.

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 [3]	Delta	-	593	516
B.1.2	-	-	-	333	350
B.1				292	276
B.1.177	Spain [4]	-	-	243	281
P.1	Brazil [5]	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 [3]	Delta	-	101	82
B.1.160	-	-	-	92	88
B.1.351	South Africa [1]	Beta	9/21	81	62
B.1.427	California [7]	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

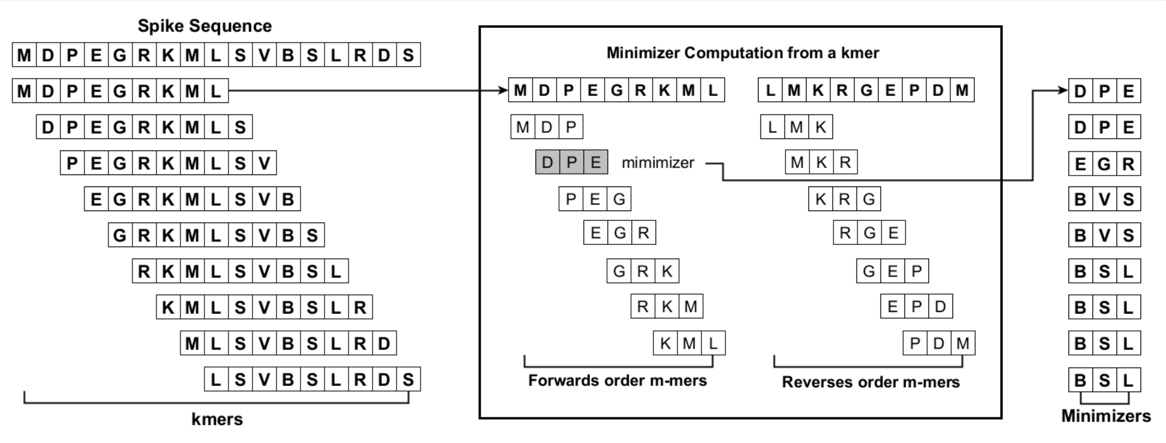
Results (GISAID 1)

		Acc.	Prec.	Recall	F1 (Weig.)	F1 (Macro)	ROC AUC	Train Time (Sec.)
Kernel Method	SVM	0.84 ± 0.0016	0.83 ± 0.0045	0.84 ± 0.0016	0.82 ± 0.0026	0.63 ± 0.0120	0.81 ± 0.0040	7.35 ± 0.2239
	NB	0.75 ± 0.0073	0.82 ± 0.0072	0.75 ± 0.0082	0.77 ± 0.0076	0.6 ± 0.0133	0.82 ± 0.0088	0.17 ± 0.2408
	MLP	0.83 ± 0.0038	0.82 ± 0.0517	0.83 ± 0.0038	0.82 ± 0.0052	0.62 ± 0.0173	0.81 ± 0.0068	12.65 ± 0.0140
	KNN	0.82 ± 0.0099	0.82 ± 0.0063	0.82 ± 0.0099	0.82 ± 0.0084	0.62 ± 0.0245	0.79 ± 0.0135	0.32 ± 1.2661
	RF	0.84 ± 0.0056	0.84 ± 0.0082	0.84 ± 0.0056	0.83 ± 0.0066	0.66 ± 0.0121	0.82 ± 0.0045	1.46 ± 0.0126
	LR	0.84 ± 0.0041	0.84 ± 0.0042	0.84 ± 0.0041	0.82 ± 0.0055	0.62 ± 0.0294	0.81 ± 0.0148	1.86 ± 0.0378
	DT	0.82 ± 0.0086	0.82 ± 0.0096	0.82 ± 0.0086	0.82 ± 0.0088	0.63 ± 0.0207	0.82 ± 0.0124	0.24 ± 0.0102

Results (GISAID 2)

		Acc.	Prec.	Recall	F1 (Weig.)	F1 (Macro)	ROC AUC	Train Time (Sec.)
Kernel Method	SVM	0.85 ± 0.0023	0.85 ± 0.0043	0.85 ± 0.0021	0.84 ± 0.0030	0.63 ± 0.0132	0.81 ± 0.0040	5.06 ± 0.2591
	NB	0.75 ± 0.0101	0.81 ± 0.0069	0.75 ± 0.0106	0.76 ± 0.0091	0.58 ± 0.0147	0.8 ± 0.0086	0.11 ± 0.2787
	MLP	0.85 ± 0.0053	0.84 ± 0.0491	0.85 ± 0.0049	0.83 ± 0.0061	0.66 ± 0.0191	0.83 ± 0.0067	15.92 ± 0.1644
	KNN	0.82 ± 0.0137	0.82 ± 0.0060	0.82 ± 0.0128	0.82 ± 0.0100	0.62 ± 0.0271	0.79 ± 0.0133	0.29 ± 2.4294
	RF	0.85 ± 0.0078	0.85 ± 0.0078	0.85 ± 0.0073	0.84 ± 0.0078	0.66 ± 0.0134	0.82 ± 0.0044	1.49 ± 0.1017
	LR	0.85 ± 0.0057	0.84 ± 0.0040	0.85 ± 0.0053	0.83 ± 0.0066	0.6 ± 0.0325	0.81 ± 0.0146	1.76 ± 0.1108
	DT	0.83 ± 0.0119	0.83 ± 0.0091	0.83 ± 0.0111	0.82 ± 0.0104	0.63 ± 0.0228	0.81 ± 0.0122	0.25 ± 0.0850

Kernel-based Solution (Using Minimizer)



Spike Sequence

M D P E G R K M L S V B S L R D S

M D P E G R K M L

D P E G R K M L S

P E G R K M L S V

E G R K M L S V B

G R K M L S V B S

R K M L S V B S L

K M L S V B S L R

M L S V B S L R D

L S V B S L R D S

kmers

Minimizer Computation from a kmer

M D P E G R K M L

L M K R G E P D M

M D P

D P E mimimizer

P E G

E G R

G R K

R K M

K M L

Forwards order m-mers

L M K

M K R

K R G

R G E

G E P

E P D

P D M

Reverses order m-mers

D P E

D P E

E G R

B V S

B V S

B S L

B S L

B S L

B S L

Minimizers

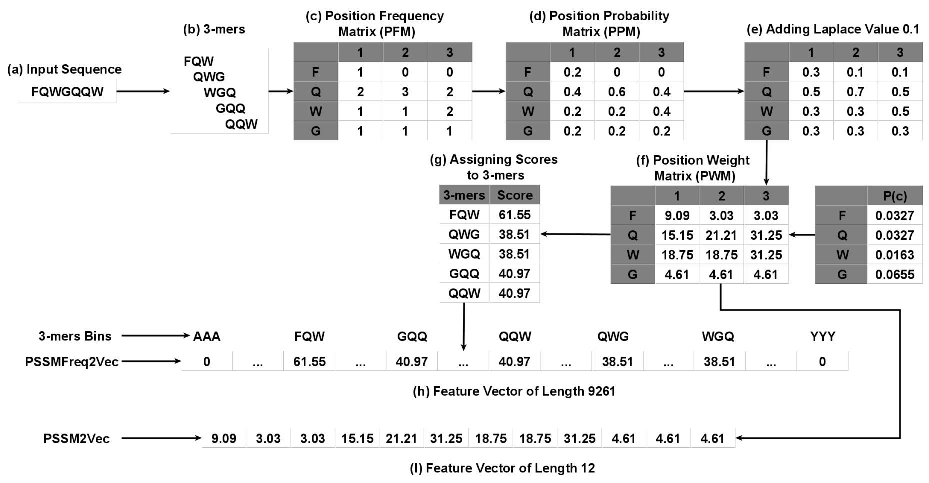
Results (GISAID 1)

		Acc.	Prec.	Recall	F1 (Weig.)	F1 (Macro)	ROC AUC	Train Time (Sec.)
Kernel Method	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

Results (GISAID 2)

		Acc.	Prec.	Recall	F1 (Weig.)	F1 (Macro)	ROC AUC	Train Time (Sec.)
Kernel Method	SVM	0.86 ± 0.0018	0.86 ± 0.0052	0.86 ± 0.0026	0.85 ± 0.0034	0.67 ± 0.0156	0.83 ± 0.0060	46.7 ± 0.4012
	NB	0.71 ± 0.0079	0.79 ± 0.0083	0.71 ± 0.0132	0.73 ± 0.0102	0.49 ± 0.0173	0.75 ± 0.0129	0.12 ± 0.4315
	MLP	0.85 ± 0.0042	0.85 ± 0.0593	0.85 ± 0.0061	0.83 ± 0.0069	0.64 ± 0.0225	0.82 ± 0.0100	30.54 ± 0.1191
	KNN	0.83 ± 0.0108	0.85 ± 0.0073	0.83 ± 0.0159	0.83 ± 0.0112	0.64 ± 0.0319	0.82 ± 0.0199	0.27 ± 3.7619
	RF	0.86 ± 0.0061	0.86 ± 0.0094	0.86 ± 0.0090	0.84 ± 0.0087	0.65 ± 0.0158	0.82 ± 0.0066	1.43 ± 0.1574
	LR	0.87 ± 0.0045	0.87 ± 0.0049	0.87 ± 0.0066	0.86 ± 0.0073	0.69 ± 0.0383	0.84 ± 0.0218	3.1 ± 0.1716
	DT	0.86 ± 0.0093	0.86 ± 0.0110	0.86 ± 0.0137	0.85 ± 0.0117	0.68 ± 0.0269	0.83 ± 0.0182	0.19 ± 0.1317

Embedding-based Solution (Position Weight Matrix)



(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

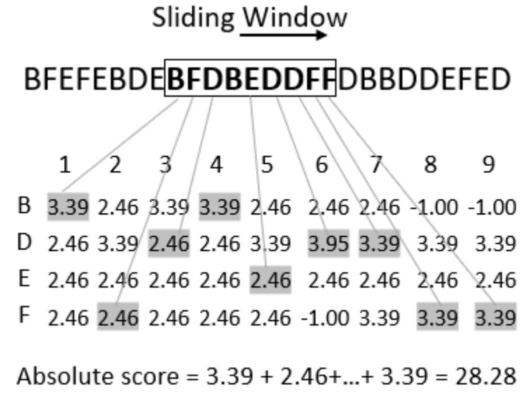
3-mers Bins	AAA		FQW		GQQ		QQW		QWG		WGQ		YYY
PSSMFreq2Vec	0	...	61.55	...	40.97	...	40.97	...	38.51	...	38.51	...	0

(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

Embedding-based Solution (Position Weight Matrix)



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

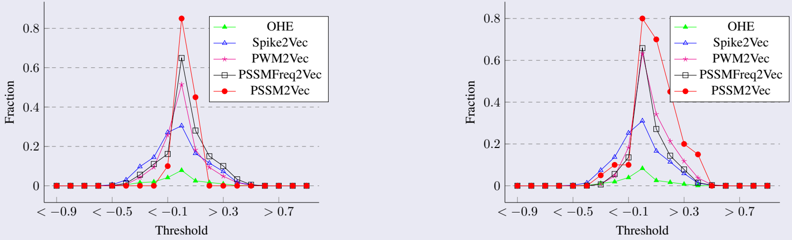
Host Name	# of Sequences	Host Name	# of Sequences
Humans	1813	Rats	26
Environment	1034	Pangolins	21
Weasel	994	Hedgehog	15
Swine	558	Dolphin	7
Birds	374	Equine	5
Camels	297	Fish	2
Bats	153	Unknown	2
Cats	123	Python	2
Bovines	88	Monkey	2
Dogs	40	Cattle	1
Turtle	1		

Table: Dataset Statistics for 5558 coronavirus hosts.

Results

Method	ML. Algo.	Acc.	Prec.	Recall	F1 (Weig.)	ROC AUC	Train Time (Sec.)
PSSMFreq2Vec	SVM	0.83	0.83	0.83	0.82	0.81	50.72
	NB	0.64	0.74	0.64	0.61	0.75	5.90
	MLP	0.83	0.82	0.83	0.83	0.77	33.44
	KNN	0.80	0.80	0.80	0.80	0.75	65.20
	RF	0.84	0.85	0.84	0.83	0.81	11.42
	LR	0.84	0.85	0.84	0.84	0.81	57.55
	DT	0.81	0.82	0.81	0.80	0.79	7.50
PSSM2Vec	SVM	0.78	0.79	0.78	0.76	0.85	1.81
	NB	0.60	0.62	0.60	0.57	0.77	**0.15**
	MLP	0.81	0.81	0.81	0.80	0.89	13.70
	KNN	0.82	0.82	0.82	0.81	0.87	0.66
	RF	**0.86**	**0.86**	**0.86**	**0.85**	**0.91**	1.43
	LR	0.73	0.75	0.73	0.70	0.78	1.91
	DT	0.82	0.82	0.82	0.82	0.89	0.20

Results



(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).

Results

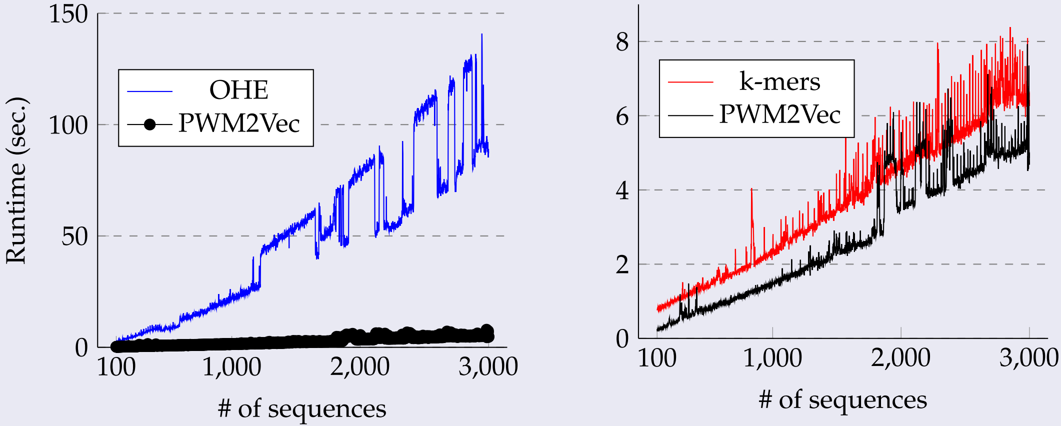
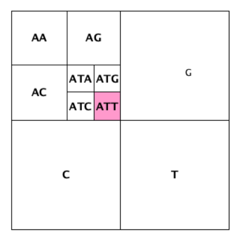


Figure: Runtime comparison for different embedding methods with increasing number of sequences using Random Forest classifier (best performing classifier). The figure is best seen in color.

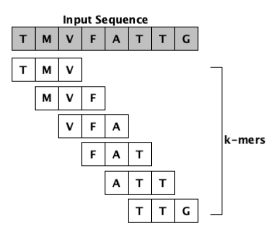
Sequence-to-Image Transformation

- We propose Chaos Game Representation-based method, which is an efficient way to convert sequences into images.
- Our proposed embedding method is alignment-free and could improve the "area of interest" within the image by performing biologically meaningful manipulation of a sequence first and then mapping the manipulated sequence into an image

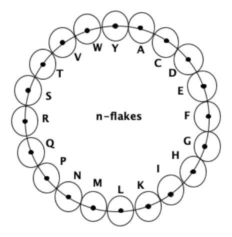
Chaos Game Representation (CGR)



(a) CGR-based allocation.



(b) 3-mers for a protein sequence



(c) 20-flakes for protein sequence.

(a) illustrates the CGR-based space allocation for a given *k*-mer in the respective image.(b) shows an example of 3-mers from a given sequence. (c) shows an example of 20-flakes for protein sequences.

Chaos Game Representation (CGR)

- CGR is used to convert sequences into images. Works well for nucleotide sequences.
- FCGR follows CGR to get images of protein sequences.
  - Get the x and y axis for an amino acid i using the given equations:

$$x[i] = r \cdot sin(\frac{2\pi i}{n} + \theta) \tag{3}$$

    Here, r is a scaling factor that determines the size of the image, i is the position of the amino acid in the sequence, n is the total number of amino acids in the sequence, and θ is an angle parameter that affects the orientation of the image.

$$y[i] = r \cdot cos(\frac{2\pi i}{n} + \theta) \tag{4}$$

- These equations create a positional mapping of amino acids in a protein sequence onto a 2D plane, allowing the visualization of protein sequences as images. The values of r and θ can be adjusted to modify the appearance and characteristics of the resulting images.

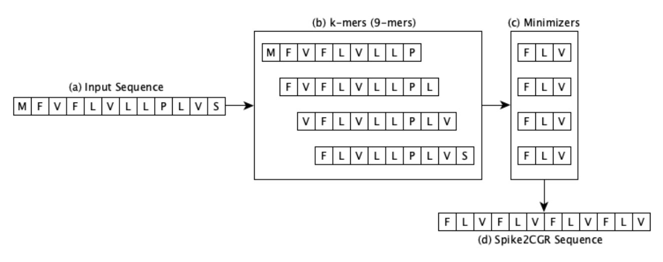
Chaos Game Representation (CGR)

- Sine and cosine are periodic functions with a period of 2π. This means they repeat their values in a regular interval, which is useful for creating repeating patterns in fractals.
- The periodic nature ensures that as i (the index of the current amino acid) changes, the points cycle through positions around the circle, leading to a coherent and continuous pattern.
- Angle Variation: The angle inside the sin and cos functions $(\frac{2\pi i}{n} + \theta)$ controls the variation of positions along the circular pattern. Here: $\frac{2\pi i}{n}$ divides the circle into n equal parts based on the position of the amino acid i in the sequence. θ introduces an additional angle parameter that can rotate or shift the circular pattern, allowing for variations in the resulting image orientation.

Chaos Game Representation (CGR)

- Spatial Distribution: By combining sin and cos with the angle parameters, the equations generate a spatial distribution of points that covers the 2D space effectively. The use of trigonometric functions helps distribute the points evenly along the circular or spiral path, ensuring a balanced representation of the sequence.
- Scaling and Orientation: The scaling factor r in front of sin and cos determines the size of the circular pattern or spiral. A larger r value results in a larger pattern, while a smaller r value creates a tighter and more condensed pattern. The angle parameter θ allows for the adjustment of the image's orientation. By changing θ, we can rotate or shift the circular/spiral pattern, providing flexibility in the visual representation of the sequence.

Spike2CGR



(a) Input Sequence

M F V F L V L L P L V S

(b) k-mers (9-mers)

M F V F L V L L P

F V F L V L L P L

V F L V L L P L V

F L V L L P L V S

(c) Minimizers

F L V

F L V

F L V

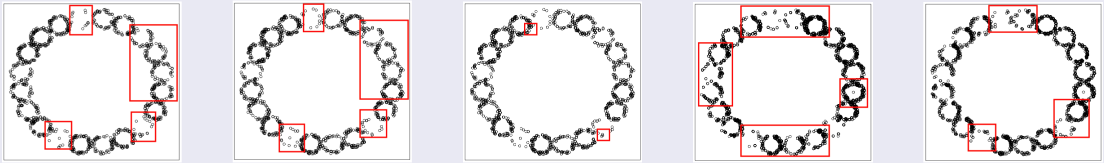
F L V

F L V F L V F L V F L V

(d) Spike2CGR Sequence

Figure: Workflow of Spike2CGR for a given sequence. For a given spike sequence, steps from (a) to (d) are followed to generate the corresponding Spike2CGR sequence.

Spike2CGR (Image Transformation)

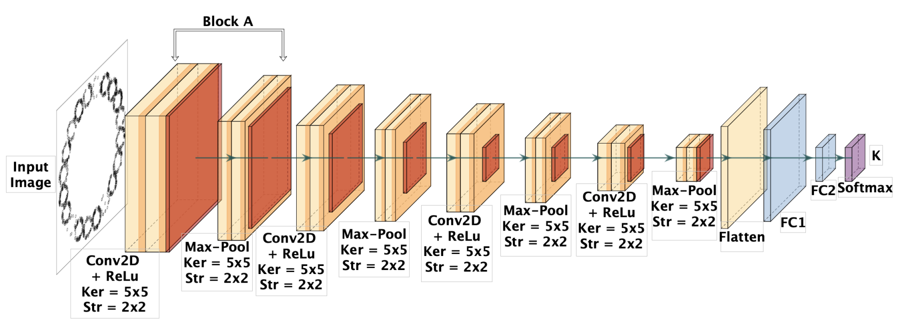


(a) Chaos (b) Spike2Vec (c) PWM2Vec (d) Minimizer (e) Spike2CGR

Figure: Graphical representation of a spike sequence of B.1.351 variant (from SARS-CoV-2 dataset) using different methods. Some of the major changes in the images (area of interest) are highlighted using the red boxes.

Classification Models

- Two types of classification models are used:
  - Tabular Models: 3-layer Tab CNN & 4-layer Tab CNN
  - Vision Models: CNN, RESNET (pre-trained), VGG-19 (pre-trained).



Block A
Input Image
Conv2D + ReLu Ker = 5x5 Str = 2x2
Max−Pool Ker = 5x5 Str = 2x2
Conv2D + ReLu Ker = 5x5 Str = 2x2
Max−Pool Ker = 5x5 Str = 2x2
Conv2D + ReLu Ker = 5x5 Str = 2x2
Max−Pool Ker = 5x5 Str = 2x2
Conv2D + ReLu Ker = 5x5 Str = 2x2
Max−Pool Ker = 5x5 Str = 2x2
Flatten
FC1
FC2
Softmax
K

Figure: The architectures of the 4-layer CNN model. Here ker represents kernel and str represents stride filter size.

Dataset

Lineage	Region	Labels	No. Mut. S/Gen.	No. of sequences		
				Training	Validation	Testing
B.1.1.7	UK [1]	Alpha	8/17	9930	2527	3146
B.1.617.2	India [2]	Delta	8/17	1877	450	456
P.2	Brazil [8]	Zeta	3/7	1780	432	533
B.1.429	California	Epsilon	3/5	1079	256	326
P.1	Brazil [5]	Gamma	10/21	994	245	306
B.1.526	New York [6]	Iota	6/16	847	219	255
B.1.351	South Africa [1]	Beta	9/21	837	221	258
B.1.427	California [7]	Epsilon	3/5	835	218	268
B.1.1.529	South Africa	Omicron	34/53	747	178	253
C.37	Peru [8]	Lambda	8/21	732	169	228
B.1.621	Colombia [8]	Mu	9/21	717	168	219
B.1.525	UK and Nigeria	Eta	8/16	714	187	224
P.3	Philippines [8]	Theta	8/17	111	30	34
Total	_	_	_	21200	5300	6238

Results

DL Model	Method	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (hrs.) ↓
3-Layer Tab CNN	OHE [9]	0.472	0.301	0.472	0.368	0.060	0.552	0.594
	WDGRL [10]	0.636	0.457	0.636	0.523	0.263	0.594	**0.380**
4-Layer Tab CNN	OHE [9]	0.637	0.469	0.637	0.528	0.157	0.511	0.977
	WDGRL [10]	0.688	0.517	0.688	0.582	0.227	0.637	0.866
1-Layer CNN	Chaos [11]	0.700	0.680	0.696	0.651	0.563	0.673	8.195
	Spike2Vec [12]	0.733	0.690	0.733	0.679	0.679	0.850	7.779
	PWM2Vec [13]	0.734	0.676	0.734	0.691	0.697	0.844	5.744
	Minimizer	0.743	0.707	0.743	0.709	0.709	0.832	6.171
	Spike2CGR	0.719	0.730	0.766	**0.739**	0.717	0.840	4.992
% improv. of Spike2CGR from SOTA Chaos [11]		1.9	5	7	8.8	15.8	16.7	39.08

Results

DL Model	Method	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (hrs.) ↓
2-Layer CNN	Chaos [11]	0.700	0.669	0.697	0.652	0.564	0.645	6.394
	Spike2Vec [12]	0.740	0.730	0.744	0.729	0.736	0.725	7.329
	PWM2Vec [13]	0.740	0.700	0.739	0.688	0.694	0.676	6.615
	Minimizer	0.710	0.710	0.710	0.681	0.581	0.771	6.426
	Spike2CGR	0.633	0.577	0.633	0.559	0.376	0.663	6.193
% improv. of Spike2CGR from SOTA Chaos [11]		-6.7	-9.2	-6.4	-9.3	-18 .8	1.8	3.14

Results

DL Model	Method	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (hrs.) ↓
3-Layer CNN	Chaos [11]	0.740	0.722	0.739	0.717	0.696	0.809	5.658
	Spike2Vec [12]	0.750	0.723	0.750	0.715	0.725	0.838	6.919
	PWM2Vec [13]	0.751	0.715	0.751	0.716	0.732	0.846	7.458
	Minimizer	0.750	0.729	0.750	0.721	0.719	**0.851**	6.332
	Spike2CGR	0.770	0.724	0.767	0.734	0.712	0.845	4.758
% improv. of Spike2CGR from SOTA Chaos [11]		3	0.2	2.8	1.7	1.6	3.6	31.23

Results

DL Model	Method	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (hrs.) ↓
4-Layer CNN	Chaos [11]	0.740	0.686	0.737	0.706	0.678	0.728	7.986
	Spike2Vec [12]	0.750	0.686	0.749	0.712	0.720	0.842	7.447
	PWM2Vec [13]	0.750	**0.733**	0.745	0.736	**0.747**	0.847	7.720
	Minimizer	0.750	0.726	0.750	0.706	0.709	0.846	7.068
	Spike2CGR	**0.7708**	0.731	**0.768**	0.738	0.714	0.843	10.658
% improv. of Spike2CGR from SOTA Chaos [11]		3	4.5	3.1	3.2	3.6	11.5	-33.45

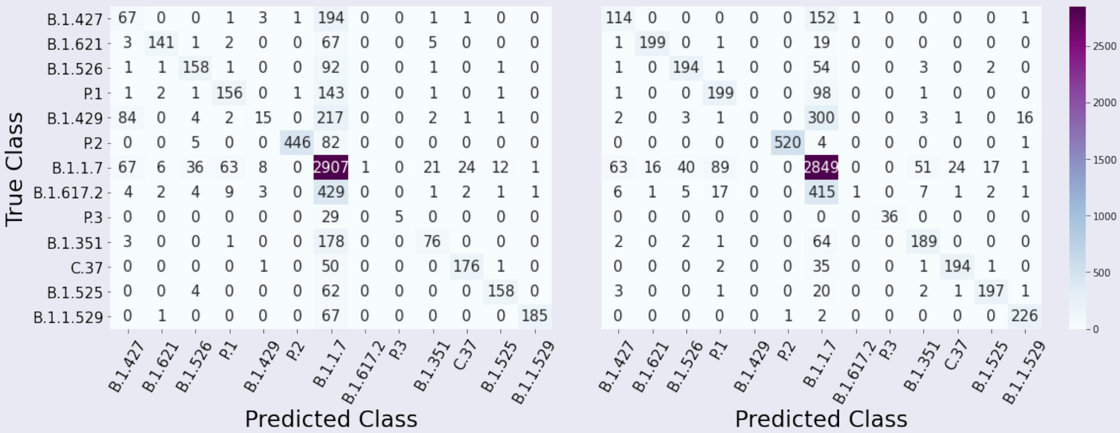
Results

DL Model	Method	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (hrs.) ↓
RESNET50 Pre-Trained Model	Chaos [11]	0.680	0.644	0.676	0.641	0.547	0.743	10.654
	Spike2Vec [12]	0.711	0.657	0.710	0.666	0.644	0.759	10.746
	PWM2Vec [13]	0.680	0.589	0.675	0.606	0.507	0.757	10.264
	Minimizer	0.723	0.665	0.723	0.673	0.647	0.802	11.732
	Spike2CGR	0.740	0.661	0.736	0.683	0.626	0.780	14.299
% improv. of Spike2CGR from SOTA Chaos [11]		6	-1.7	6	4.2	7.9	3.7	-34.21

Results

DL Model	Method	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (hrs.) ↓
VGG-19 Pre-Trained Model	Chaos [11]	0.480	0.233	0.483	0.315	0.050	0.500	27.398
	Spike2Vec [12]	0.470	0.221	0.470	0.301	0.049	0.500	26.599
	PWM2Vec [13]	0.464	0.215	0.464	0.294	0.048	0.500	23.781
	Minimizer	0.480	0.227	0.477	0.308	0.496	0.500	24.459
	Spike2CGR	0.495	0.245	0.495	0.327	0.050	0.500	24.355
% improv. of Spike2CGR from SOTA Chaos [11]		1.5	1.2	1.2	1.2	0	0	8.4

Results



	B.1.427	B.1.621	B.1.526	P.1	B.1.429	P.2	B.1.1.7	B.1.617.2	P.3	B.1.351	C.37	B.1.525	B.1.1.529
B.1.427	67	0	0	1	3	1	194	0	0	1	1	0	0
B.1.621	3	141	1	2	0	0	67	0	0	5	0	0	0
B.1.526	1	1	158	1	0	0	92	0	0	1	0	1	0
P.1	1	2	1	156	0	1	143	0	0	1	0	1	0
B.1.429	84	0	4	2	15	0	217	0	0	2	1	1	0
P.2	0	0	5	0	0	446	82	0	0	0	0	0	0
B.1.1.7	67	6	36	63	8	0	2907	1	0	21	24	12	1
B.1.617.2	4	2	4	9	3	0	429	0	0	1	2	1	1
P.3	0	0	0	0	0	0	29	0	5	0	0	0	0
B.1.351	3	0	0	1	0	0	178	0	0	76	0	0	0
C.37	0	0	0	0	1	0	50	0	0	0	176	1	0
B.1.525	0	0	4	0	0	0	62	0	0	0	0	158	0
B.1.1.529	0	1	0	0	0	0	67	0	0	0	0	0	185

(a) Chaos

	B.1.427	B.1.621	B.1.526	P.1	B.1.429	P.2	B.1.1.7	B.1.617.2	P.3	B.1.351	C.37	B.1.525	B.1.1.529
B.1.427	114	0	0	0	0	0	152	1	0	0	0	0	1
B.1.621	1	199	0	1	0	0	19	0	0	0	0	0	0
B.1.526	1	0	194	1	0	0	54	0	0	3	0	2	0
P.1	1	0	0	199	0	0	98	0	0	1	0	0	0
B.1.429	2	0	3	1	0	0	300	0	0	3	1	0	16
P.2	0	0	0	0	0	520	4	0	0	0	0	0	1
B.1.1.7	63	16	40	89	0	0	2849	0	0	51	24	17	1
B.1.617.2	6	1	5	17	0	0	415	1	0	7	1	2	1
P.3	0	0	0	0	0	0	0	0	36	0	0	0	0
B.1.351	2	0	2	1	0	0	64	0	0	189	0	0	0
C.37	0	0	0	2	0	0	35	0	0	1	194	1	0
B.1.525	3	0	0	1	0	0	20	0	0	2	1	197	1
B.1.1.529	0	0	0	0	0	1	2	0	0	0	0	0	226

(b) Spike2CGR

Molecular Properties (Weights)

- Kyte and Doolittle (KD) Hydropathy Scale
  - Assigns numerical values to amino acids based on their hydrophobicity/hydrophilicity, used in predicting protein structure and function.
- Eisenberg Hydrophobicity Scale
  - Quantifies the hydrophobicity of amino acids, aiding in protein structure prediction and understanding protein interactions with hydrophobic environments.
- Hydrophilicity Scale
  - Measures the propensity of amino acids to interact with water, crucial for understanding protein solubility, folding, and function in aqueous environments.
- Flexibility Of The Characters
  - Evaluates the flexibility or rigidity of amino acids, important for predicting protein dynamics, conformational changes, and flexibility in molecular interactions.
- Hydropathy Scale
  - Ranks amino acids based on their hydrophobic or hydrophilic nature, assisting in studying protein folding, membrane protein structure, and transmembrane domains.

Workflow

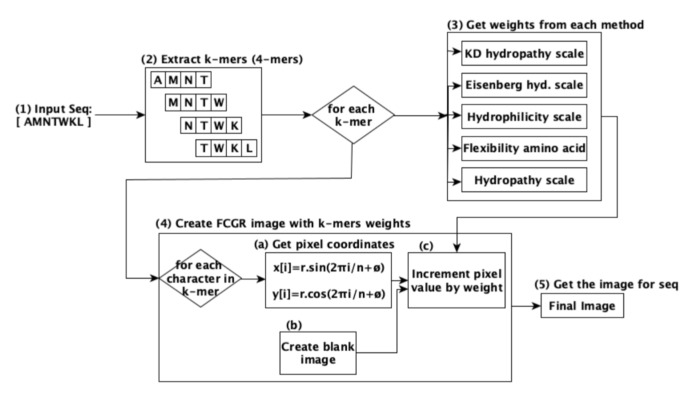

(1) Input Seq: [AMNTWKL]
(2) Extract k-mers (4-mers)
A M N T
M N T W
N T W K
T W K L
for each k-mer
(3) Get weights from each method
KD hydropathy scale
Eisenberg hyd. scale
Hydrophilicity scale
Flexibility amino acid
Hydropathy scale
(4) Create FCGR image with k-mers weights
for each character in k-mer
(a) Get pixel coordinates
x[i]=r.sin(2πi/n+ø)
y[i]=r.cos(2πi/n+ø)
(b)
Create blank image
(c)
Increment pixel value by weight
(5) Get the image for seq
Final Image

Figure: Workflow of the proposed method for creating an image of a sequence.

Dataset

		Rabies Sequence Length			Number of Sequences		
Host Name	Count	Min.	Max.	Average	Training	Validation	Testing
Canis Familiaris	9065	90	11928	1600.50	5802	1450	1813
Bos Taurus	2497	117	11928	995.29	1599	399	499
Vulpes Vulpes	2221	133	11930	2923.77	1422	355	444
Felis Catus	1125	90	11928	1634.43	720	180	225
Procyon Lotor	884	291	11926	6763.80	567	141	176
Desmodus Rotundus	875	164	11923	1051.50	560	140	175
Mephitis Mephitis	864	220	11929	1266.59	554	138	172
Homo Sapiens	838	101	11928	1537.85	537	134	167
Eptesicus Fuscus	718	264	11924	1144.35	460	115	143
Skunk	492	211	11928	6183.26	316	78	98
Tadarida Brasiliensis	270	264	11923	1175.67	173	43	54
Equus Caballus	202	163	11924	1376.74	130	32	40
Total	20051	-	-	-	-	-	-

Table: Dataset Statistics for Rabies data.

Baselines

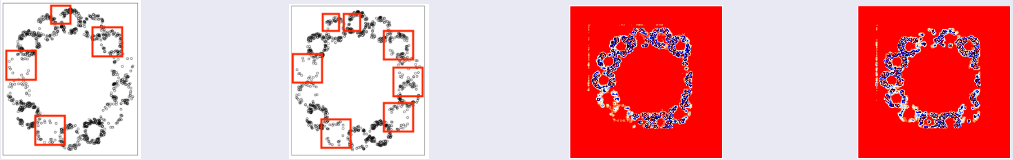
- Feature-engineering-based methods
  - One Hot Encoding (OHE): created embeddings are sparse and face curse of dimensionality challenge.
  - Wasserstein Distance Guided Representation Learning (WDGRL): require large training data for optimal performance.
  - Position Specific Scoring Matrix (PSSM)
- Image-based method
  - Frequency Matrix-based Chaos Game Representation (FCGR): 1-to-1 mapping between the amino acids and pixels.

Results

	Method	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (Sec.) ↓
NB	OHE	0.124	0.447	0.124	0.134	0.195	0.585	979.44
	WDGRL	0.514	0.441	0.514	0.410	0.184	0.575	**0.01**
	PSSM2Vec	0.125	0.296	0.125	0.072	0.105	0.58	**0.04**
3 Layer Tab CNN	OHE	0.451	0.203	0.451	0.280	0.050	0.500	4191.34
	WDGRL	0.450	0.202	0.450	0.279	0.049	0.500	1737.65
	PSSM2Vec	0.452	0.204	0.452	0.281	0.051	0.500	2040.81
4 Layer Tab CNN	OHE	0.452	0.204	0.452	0.281	0.051	0.500	5974.26
	WDGRL	0.535	0.318	0.535	0.395	0.103	0.500	964.97
	PSSM2Vec	0.450	0.204	0.450	0.282	0.052	0.500	3790.09
ViT	Chaos	0.448	0.201	0.448	0.277	0.051	0.500	2943.45
	KD	0.440	0.194	0.440	0.269	0.050	0.500	3593.00
	Eisen.	0.465	0.216	0.465	0.295	0.052	0.500	3474.12
	Flex.	0.441	0.194	0.441	0.270	0.051	0.500	3035.72
	Hydrophil.	0.455	0.207	0.455	0.285	0.052	0.500	2829.95
	Hydropathy	0.449	0.201	0.449	0.278	0.051	0.500	3029.90
CNN	Chaos	0.780	0.763	0.780	0.767	**0.662**	**0.813**	12505.91
	KD	0.771	0.757	0.771	0.756	0.647	0.807	13331.11
	Eisen.	**0.787**	**0.779**	**0.787**	**0.773**	**0.668**	0.810	14127.47
	Flex.	0.775	0.763	0.775	0.758	0.647	0.807	13068.88
	Hydrophil.	**0.785**	**0.770**	**0.785**	**0.774**	0.659	**0.817**	14286.38
	Hydropathy	0.773	0.766	0.773	0.765	0.653	0.809	13115.00
Pretrain	Chaos	0.202	0.365	0.202	0.230	0.081	0.500	146831.05
	KD	0.210	0.370	0.210	0.229	0.079	0.510	147221.45
	Eisen.	0.284	0.451	0.284	0.364	0.095	0.530	161828.01
	Flex.	0.274	0.441	0.274	0.387	0.087	0.500	144477.50
	Hydrophil.	0.283	0.431	0.283	0.363	0.093	0.521	150921.41
	Hydropathy	0.252	0.331	0.252	0.323	0.073	0.500	142441.85

Table: The top 2 best values for each evaluation metric are shown in bold.

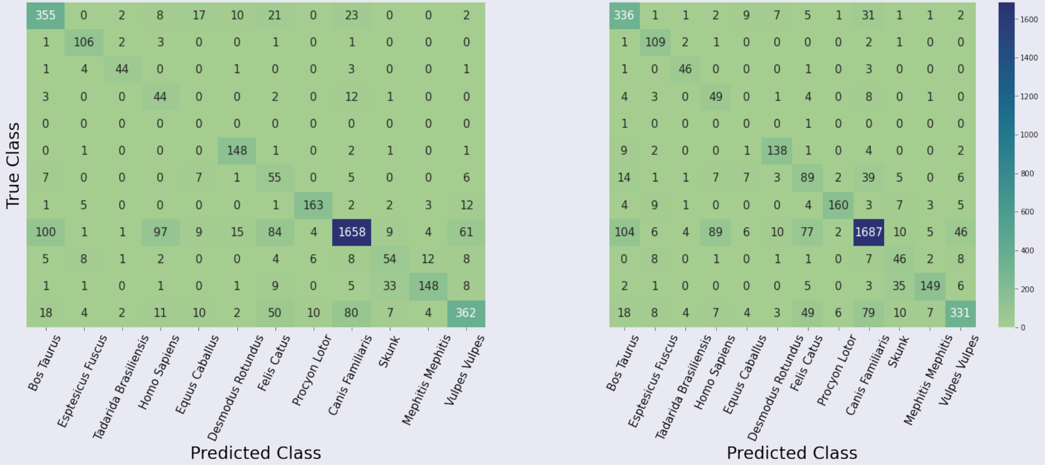
Results



(a) Chaos (b) Eisenberg (c) S.M. Chaos (d) S.M. Eisenberg

Figure: Images generated using Chaos and Eisenberg encoding techniques for a sequence against Cytoplasm location from protein subcellular dataset along with their respective Saliency Maps (S.M.). Some of the major differences between the original images are indicated using the red boxes. The blue color in the saliency maps indicates the most importance. This figure is best seen in colors.

Results



Bos Taurus	Esptesicus Fuscus	Tadarida Brasiliensis	Homo Sapiens	Equus Caballus	Desmodus Rotundus	Felis Catus	Procyon Lotor	Canis Familiaris	Skunk	Mephitis Mephitis	Vulpes Vulpes
355	0	2	8	17	10	21	0	23	0	0	2
1	106	2	3	0	0	1	0	1	0	0	0
1	4	44	0	0	1	0	0	3	0	0	1
3	0	0	44	0	0	2	0	12	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	148	1	0	2	1	0	1
7	0	0	0	7	1	55	0	5	0	0	6
1	5	0	0	0	0	1	163	2	2	3	12
100	1	1	97	9	15	84	4	1658	9	4	61
5	8	1	2	0	0	4	6	8	54	12	8
1	1	0	1	0	1	9	0	5	33	148	8
18	4	2	11	10	2	50	10	80	7	4	362

(a) Chaos

Bos Taurus	Esptesicus Fuscus	Tadarida Brasiliensis	Homo Sapiens	Equus Caballus	Desmodus Rotundus	Felis Catus	Procyon Lotor	Canis Familiaris	Skunk	Mephitis Mephitis	Vulpes Vulpes
336	1	1	2	9	7	5	1	31	1	1	2
1	109	2	1	0	0	0	0	2	1	0	0
1	0	46	0	0	0	1	0	3	0	0	0
4	3	0	49	0	1	4	0	8	0	1	0
1	0	0	0	0	0	1	0	0	0	0	0
9	2	0	0	1	138	1	0	4	0	0	2
14	1	1	7	7	3	89	2	39	5	0	6
4	9	1	0	0	0	4	160	3	7	3	5
104	6	4	89	6	10	77	2	1687	10	5	46
0	8	0	1	0	1	1	0	7	46	2	8
2	1	0	0	0	0	5	0	3	35	149	6
18	8	4	7	4	3	49	6	79	10	7	331

(b) Eisenberg

Bézier curves

The general formula [14] of the Bézier curve is

$$BZ(t) = \Sigma_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} P_i \tag{5}$$

where $0 \leq t \leq 1$, P_i are known as control points and are elements of $\mathbb{R}^k$, and $k \leq n$.
To construct the protein images, we employ a Bézier curve with $n = 3$ and $k = 2$. As images consist of x and y coordinates, therefore $k = 2$ is used. The formulas to determine the coordinates for representing an amino acid in the respective generated image are,

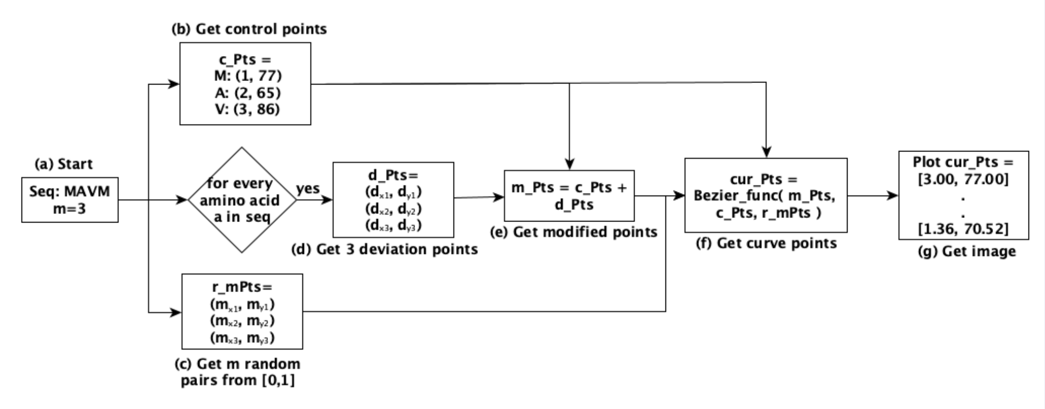
$$x = (1-t)^3 \cdot P_{0_x} + 3 \cdot (1-t)^2 \cdot t \cdot P_{1_x} + 3 \cdot (1-t) \cdot t^2 \cdot P_{2_x} + t^3 \cdot P_{3_x} \tag{6}$$

$$y = (1-t)^3 \cdot P_{0_y} + 3 \cdot (1-t)^2 \cdot t \cdot P_{1_y} + 3 \cdot (1-t) \cdot t^2 \cdot P_{2_y} + t^3 \cdot P_{3_y} \tag{7}$$

Bézier curves

```
Input: Sequence $seq$, No. of Parameters $m$
Output: Image $img$
conPoint = {}                                   ▷ dictionary for control points
for $i, aa \in seq$ do:                         ▷ every unique amino acid aa in seq
    conPoint[aa] = $[i, ASCII(aa)]$             ▷ assign control point the index i and ASCII of aa
$xCord$ = []                                    ▷ list for x coordinates
$yCord$ = []                                    ▷ list for y coordinates
$t\_Val$ = Get $m$ pairs $\in [0, 1]$           ▷ list of m pairs of parameters
$ite = 3$                                       ▷ no. of deviations pair points. It can have any value.
for $a \in seq$ : do                            ▷ every amino acid a in seq
    org_point = conPoint[a]                     ▷ control point of $a$
    points = [org_point]
    for $i \in (ite)$ : do
        dev = Get_Random_Pair                   ▷ get a random pair
        mod_point = org_point + dev             ▷ get a modified control point
        points.append(mod_point)
    curve_point = Get_Bezier_Point(points, $t\_Val$)   ▷ get bezier curve points from bezier func
    $xCord$ = curve_point[:0]                   ▷ get x coords of curve
    $yCord$ = curve_point[:1]                   ▷ get y coords of curve
$img$ = plot($xCord$, $yCord$)                  ▷ get image by plotting x & y coords
return($img$)
```

Bézier curves



(a) Start

Seq: MAVM
m=3

for every amino acid a in seq

yes

(b) Get control points

c_Pts =
M: (1, 77)
A: (2, 65)
V: (3, 86)

(c) Get m random pairs from [0,1]

r_mPts=
(m_{x1}, m_{y1})
(m_{x2}, m_{y2})
(m_{x3}, m_{y3})

(d) Get 3 deviation points

d_Pts=
(d_{x1}, d_{y1})
(d_{x2}, d_{y2})
(d_{x3}, d_{y3})

(e) Get modified points

m_Pts = c_Pts + d_Pts

(f) Get curve points

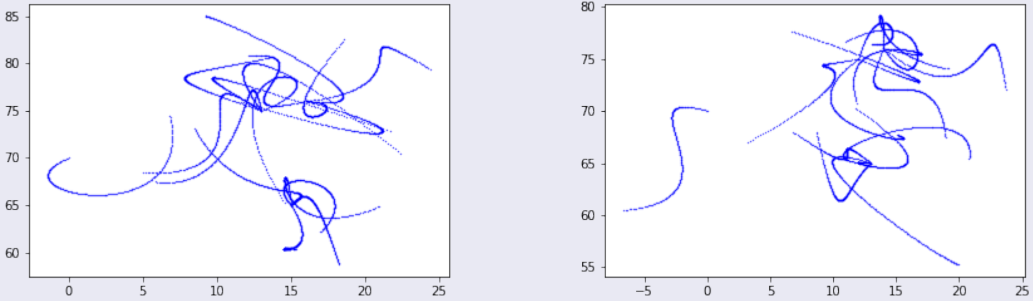
cur_Pts = Bezier_func(m_Pts, c_Pts, r_mPts)

(g) Get image

Plot cur_Pts =
[3.00, 77.00]
.
.
[1.36, 70.52]

Figure: The workflow of our system to create an image from a given sequence and a number of parameters *m*. We have used "MAVM" as an input sequence here. Note that the *cur_Pts* consists of a set of values for x coordinates and y coordinates.

Bézier curves



(a) Active ACP

(b) Inactive ACP

Figure: The Bézier curve method-based images created for two sequences from the ACP dataset. One sequence belongs to the active class of the dataset, while the other is from the inactive class.

Dataset

		Protein Subcellular Sequence Length		
Subcellular Locations	Count	Min.	Max.	Average
Cytoplasm	1411	9	3227	337.32
Plasma Membrane	1238	47	3678	462.21
Extracellular Space	843	22	2820	194.01
Nucleus	837	16	1975	341.35
Mitochondrion	510	21	991	255.78
Chloroplast	449	71	1265	242.03
Endoplasmic Reticulum	198	79	988	314.64
Peroxisome	157	21	906	310.75
Golgi Apparatus	150	116	1060	300.70
Lysosomal	103	101	1744	317.81
Vacuole	63	60	607	297.95
Total	5959	-	-	-

Results

Category	DL Model	Method	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (hrs.) ↓
Vision Transformer	ViT	FCGR	0.226	0.051	0.226	0.083	0.033	0.500	0.180
		RandmCGR	0.222	0.049	0.222	0.080	0.033	0.500	0.154
		Spike2CGR	0.222	0.051	0.222	0.083	0.147	0.500	0.176
		Bézier	0.462	0.254	0.462	0.327	0.147	0.572	0.160
	% improv. of Bézier from FCGR		23.6	20.3	23.6	24.4	11.4	7.2	11.11
	% impro. of Bézier from Spike2CGR		24	20.3	24	24.4	0	7.2	-9.09
Pretrained Vision Models	ResNet-50	FCGR	0.368	0.268	0.368	0.310	0.155	0.556	3.831
		RandmCGR	0.293	0.174	0.293	0.211	0.102	0.527	13.620
		Spike2CGR	0.368	0.175	0.368	0.214	0.105	0.565	10.992
		Bézier	0.964	0.967	0.964	0.961	0.907	0.948	11.415
	% improv. of Bézier from FCGR		59.6	69.9	59.6	65.1	75.2	39.2	-197.96

Results

Category	DL Model	Method	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (hrs.) ↓
Pretrained Vision Models	VGG-19	FCGR	0.316	0.209	0.316	0.241	0.114	0.533	14.058
		RandmCGR	0.288	0.192	0.288	0.218	0.105	0.525	26.136
		Spike2CGR	0.351	0.352	0.351	0.333	0.211	0.550	19.980
		Bézier	0.896	0.879	0.896	0.873	0.680	0.840	18.837
	% improv. of Bézier from FCGR		58	67	58	63.2	56.6	30.7	-33.99
	% impro. of Bézier from Spike2CGR		54.5	52.7	54.5	56.3	46.9	29	5.7
	EfficientNet	FCGR	0.100	0.088	0.100	0.094	0.035	0.532	31.194
		RandmCGR	0.284	0.107	0.284	0.152	0.078	0.500	30.223
		Spike2CGR	0.320	0.230	0.320	0.230	0.200	0.500	25.497
		Bézier	0.834	0.787	0.834	0.797	0.483	0.751	20.312
	% improv. of Bézier from FCGR		73.4	69.9	73.4	70.3	44.8	21.9	34.88

Conclusion and Future Work

- We discuss different methods of molecular sequence analysis.
- Using sequence-to-image transformation, we enable the vision models to be used for sequence classification.

Future Work

- Try on larger data to evaluate the scalability.
- Employ other methods like spaced minimizers to get the images.

Thank You

Feel Free To Contact Me

- Website: https://sarwanpasha.github.io/
- Google Scholar: https://scholar.google.com/citations?user=9dtXSoAAAAAJ&hl=en

References

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.

CDC, "Sars-cov-2 variant classifications and definitions," https://www.cdc.gov/coronavirus/2019-ncov/variants/variant-info.html, 2021, [Online; accessed 29-December-2021].

E. B. Hodcroft, M. Zuber, S. Nadeau, T. G. Vaughan, K. H. Crawford, C. L. Althaus, M. L. Reichmuth, J. E. Bowen, A. C. Walls, D. Corti *et al.*, "Emergence and spread of a sars-cov-2 variant through europe in the summer of 2020," *MedRxiv*, 2020.

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.

- A. West Jr *et al.*, “Detection and characterization of the sars-cov-2 lineage b. 1.526 in new york,” *bioRxiv*, 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.
- WHO Website, https://www.who.int/en/activities/tracking-SARS-CoV-2-variants/.
- K. Kuzmin *et al.*, “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.
- J. Shen, Y. Qu, W. Zhang, and Y. Yu, “Wasserstein distance guided representation learning for domain adaptation,” in *AAAI conference on artificial intelligence*, 2018.
- H. F. Löchel, D. Eger, T. Sperlea, and D. Heider, “Deep learning on chaos game representation for proteins,” *Bioinformatics*, vol. 36, no. 1, pp. 272–279, 2020.

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

- 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.

- S. Baydas and B. Karakas, “Defining a curve as a bezier curve,” *Journal of Taibah University for Science*, vol. 13, no. 1, pp. 522–528, 2019.