Representation Learning And Sequence Analysis Using A.I

Sarwan Ali

4th Year Ph.D. Candidate

Department of Computer Science,
Georgia State University
Atlanta, GA, USA

Table of Contents

1. String Kernel
2. Kernel Embedding
3. Nanobody-Antigen Binding Prediction
4. t-Cell Sequence Classification
5. Chaos Game Representation

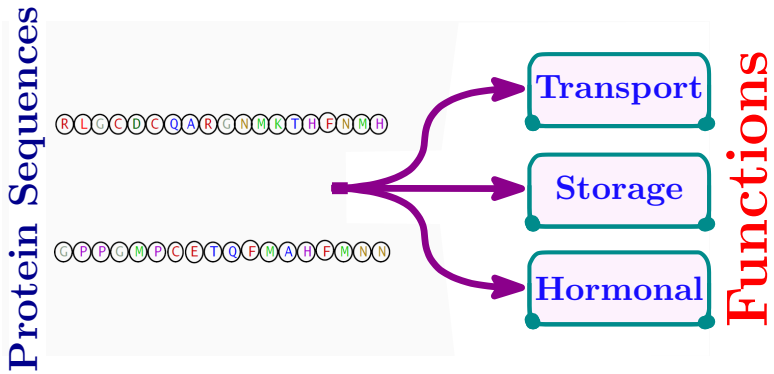
String Kernel

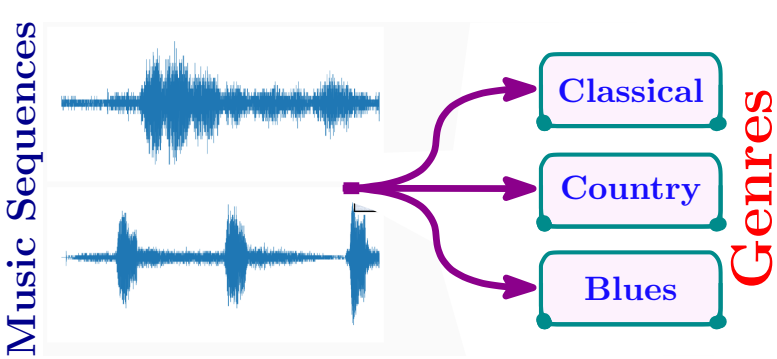
Sequence Classification

Sequence analysis is fundamental in machine learning and data mining

Applications in bioinformatics, text mining, and NLP

- Protein homology detection
- Protein 3d Fold prediction
- Music genre classification
- Music artist identification
- Text categorization





Problem Formulation

Input:

- A set of sequences X
- Alphabet Σ
- k,m

Output:

- Kernel Matrix K

Problem Formulation

Input:

- A set of sequences X
- Alphabet Σ
- k,m

Output:

- Kernel Matrix K

Kernel Value

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,

$$\Phi_{k,m}(X) = (\Phi_{k,m}(X)[\gamma])_{\gamma \in \Sigma^k} = \left(\sum_{\alpha \in X} I_m(\alpha, \gamma) \right)_{\gamma \in \Sigma^k}, \qquad (1)$$

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 Definition

Problem: Efficient computation of k, m-mismatch kernel for sequences X and Y.
Kernel Definition:

$$K(X, Y|k, m) = \sum_{\alpha \in X} \sum_{\beta \in Y} |N_{k,m}(\alpha) \cap N_{k,m}(\beta)| \tag{2}$$

where $N_{k,m}(\alpha)$ and $N_{k,m}(\beta)$ are m-mismatch neighborhoods of α and β
Key Facts:

- $\mathfrak{I}_m(\alpha, \beta) = |N_{k,m}(\alpha) \cap N_{k,m}(\beta)|$ depends on $k, m, |\Sigma|$, and $d(\alpha, \beta)$.
- $\mathfrak{I}_m(\alpha, \beta) = 0$ if $d(\alpha, \beta) > 2m$.

Algorithm for Kernel Computation

Key Steps:

- Compute $M_i(X, Y)$: Number of k-mer pairs (α, β) with $d(\alpha, \beta) = i$.
- Compute $\mathcal{I}_d$ using closed form:

$$n^{ij}(\alpha, \beta) = \sum_{t=0}^{\frac{i+j-d}{2}} \binom{2d - i - j + 2t}{d - (i - t)} \binom{d}{i + j - 2t - d} \times$$

(s -2)$^{i+j-2t-d}\binom{k-d}{t}(s-1)^t(3)$

- Compute $F_i(X, Y)$ efficiently:

$$F_i(X, Y) = \sum_{\theta \in \mathcal{Q}_k(k-i)} f_\theta(X, Y) \tag{4}$$

- Approximate $F_i(X, Y)$ using random sampling.

Approximate Kernel Algorithm

Algorithm Approximate-Kernel(S_X,S_Y,k,m,ϵ, δ)

```
Initialize 𝓘, M̂, F̂ to zero
Populate 𝓘 using closed form
for i = 0 to t do
    μ_F ← 0
    for θ ∈ B_i do
        μ_F ← μ_F + sort-enumerate(S_X, S_Y, k, θ)
    F̂[i] ← μ_F · (1/|B_i|) C(k, k−i)
    M̂[i] ← F̂[i]
    for j = 0 to i − 1 do
        M̂[i] ← M̂[i] − C(k−j, k−i) · M̂[j]
K′ ← sumproduct(M̂, 𝓘)
return K′
```

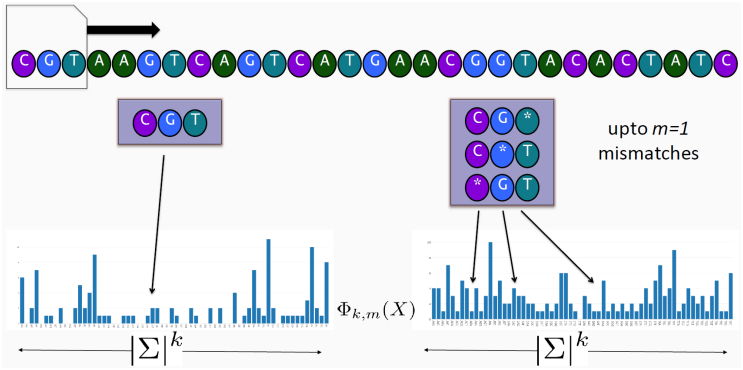
Theoretical Guarantee: Resulting kernel matrix is positive semidefinite.

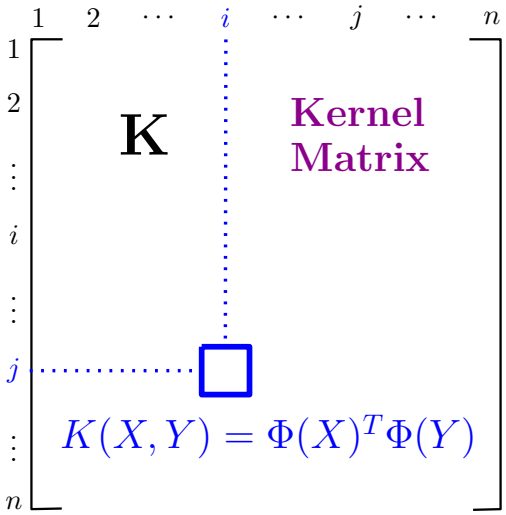
String Kernel based Sequence Classification (recall)

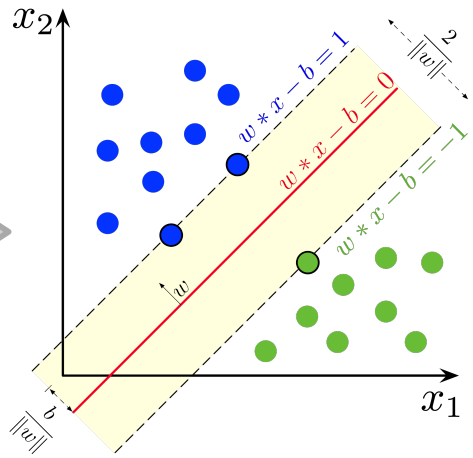
For a sequence X over Σ, the k, m-mismatch spectrum is a $|\Sigma|^k$-d vector

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

where $I_m(\alpha, \gamma) = 1$ **if** $d(\alpha, \gamma) \leq m$







String Kernel based Sequence Classification

We designed a novel method that

1. Efficient Approximate Kernel Based Spike Sequence Classification, *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2022)

- efficiently estimates the pairwise kernel scores $K(X, Y)$
- has approximation guarantees on estimation quality
- and enabling the scalability of kernel methods for larger values of parameters

String Kernel To Embedding Transformation

BioSequence2Vec Computation

- The BioSequence2Vec representation, $\hat{\mathbf{x}}$ for a sequence X represents X by the random projections of $\Phi_k(X)$ on the "discrete approximations" of random directions.
- It allows the application of vector space-based machine learning methods.
- We show that the Euclidean distance between a pair of vectors in BioSequence2Vec representation is closely related to the kernel-based proximity measure between the corresponding sequences.
- We use 4-wise independent hash functions to compute $\Phi'(\cdot)$.

BioSequence2Vec Computation

Definition (4-wise Independent hash function)

A family $\mathcal{H}$ of functions of the form $h : \Sigma^k \mapsto \{-1, 1\}$ is called 4-wise independent, or 4-universal, if a randomly chosen $h \in \mathcal{H}$ has the following properties:

1. for any $\alpha \in \Sigma^k$, $h(\alpha)$ is equally likely to be -1 or 1.
2. for any distinct $\alpha_i \in \Sigma^k$, and $m_i \in \{-1, 1\}$ $(1 \leq i \leq 4)$,

$$Pr[h(\alpha_1) = m_1 \wedge \ldots \wedge h(\alpha_4) = m_4] = 12^4$$

Next, we give a construction of a 4-wise independent family of hash functions due to Carter and Wegman [1]

BioSequence2Vec Computation

Definition

Let p be a large prime number. For integers a_0, a_1, a_2, a_3, such that $0 \leq a_i \leq p-1$, and $\alpha \in \Sigma^k$ (represented as integer base $|\Sigma|$), the hash function $h_{a_0,a_1,a_2,a_3} : \Sigma^k \mapsto \{-1, 1\}$ is defined as

$$h_{a_0,a_1,a_2,a_3}(\alpha) = \begin{cases} -1 & \text{if } g(\alpha) = 0 \\ 1 & \text{if } g(\alpha) = 1 \end{cases} \tag{5}$$

where

$$g(\alpha) = \left(a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3 \mod p\right) \mod 2 \tag{6}$$

BioSequence2Vec Computation

- We show that the dot-product between the representations $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ of a pair of sequences X and Y closely approximates the kernel value.
- We are going to show that for any pair of sequences X and Y,

$$\hat{\mathbf{x}} \cdot \hat{\mathbf{y}} \simeq \Phi_k(X) \cdot \Phi_k(Y)$$

- Let $\mathbf{x} = \Phi_k(X)$ and $\mathbf{y} = \Phi_k(Y)$, we show that $\hat{\mathbf{x}} \cdot \hat{\mathbf{y}} \simeq \mathbf{x} \cdot \mathbf{y}$

Theorem

For any $0 < \epsilon, \delta < 1$, *if* $t \geq 2\epsilon^2 \log(1\delta)$, *then*

1. $E\left[\hat{\mathbf{x}} \cdot \hat{\mathbf{y}}\right] = \mathbf{x} \cdot \mathbf{y}$
   - *Average (expected) value of the dot-product representations is equal to the true kernel similarity*
2. $Pr\left[|\hat{\mathbf{x}} \cdot \hat{\mathbf{y}} - \mathbf{x} \cdot \mathbf{y}| \leq \epsilon \|\mathbf{x}\| \|\mathbf{y}\|\right] \geq 1 - \delta$
   - *Probabilistic bound on how close the approximate dot-product is to the true kernel similarity. It guarantees that with high probability (at least 1 - δ, the error in approximation is within a specified tolerance ϵ.*

BioSequence2Vec Computation

Algorithm BioSequence2Vec Computation

```
Input: Set 𝒮 of sequences, integers k, p, Σ, t
Output: Embedding R
function ComputeEmbedding(𝒮, k, p, Σ, t)
    R = []
    for X ∈ 𝒮 do
        x̂ = []
        for i = 1 to t do
            a_0, a_1, a_2, a_3 ← random(0, p − 1)
            for j = 1 to |X| − k + 1 do
                α ← X[j : j + k]
                h ← a_0 + a_1 α_Σ + a_2 α_Σ^2 + a_3 α_Σ^3
                h ← (h mod p) mod 2
                if h = 0 then
                    x̂[i] ← x̂[i] − 1
                else
                    x̂[i] ← x̂[i] + 1
            x̂[i] ← (1/√t) × x̂[i]
        R.append(x̂)
    Return R
```

Dataset Statistics

Dataset	Detail	Source	Total Sequences	Total classes	Sequence Length		
					Min	Max	Average
Spike7k	Aligned spike protein sequences to classify lineages of coronavirus in humans	[2]	7000	22	1274	1274	1274.00
Coronavirus Host	Spike protein sequences to classify hosts effected from coronavirus	[3]	5558	21	9	1584	1272.36
Human DNA	Unaligned nucleotide sequences to classify gene family to which humans belong	[4]	4380	7	5	18921	1263.59

Table: Dataset Statistics.

Embedding Properties

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	[5]	✓	✓	✓	✗
Spaced k-mers			[6]	✓	✓	✓	✗
PWM2Vec			[7]	✗	✓	✓	✓
WDGRL	Neural Network (NN)	Take one-hot representation of biological sequence as input and design NN-based embedding method by minimizing loss	[8]	✗	✗	✓	✓
AutoEncoder			[9]	✗	✗	✓	✓
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	[10]	✓	✓	✗	✓
SeqVec	Pretrained Language Model	Takes biological sequences as input and fine-tunes the weights based on a pre-trained model to get final embedding	[11]	✓	✗	✓	✓
ProteinBERT	Pretrained Transformer	A pretrained protein sequence model to classify the given biological sequence using Transformer/Bert	[12]	✓	✗	✓	✓
BioSequence2Vec (ours)	Hashing	Takes biological sequence as input and design embeddings based on the kernel property of preserving pairwise distance	-	✓	✓	✓	✓

Table: Different methods (ours and SOTA) description.

Results

Embeddings	Algo.	Spike7k							Human DNA						
		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.

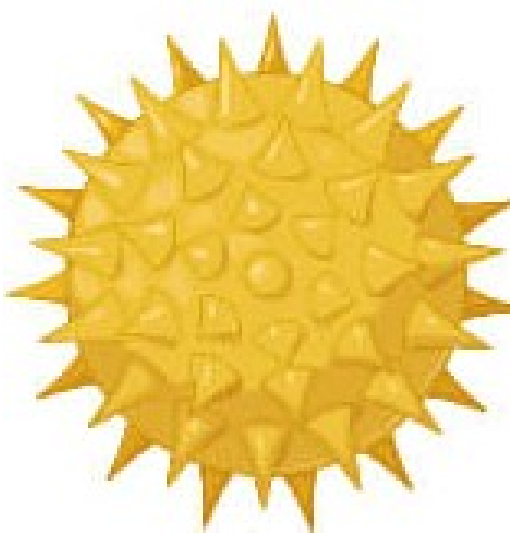
Embeddings	Algo.	Acc.	Prec.	Recall	F1 (Weig.)	F1 (Macro)	ROC AUC	Train Time (Sec.)
BioSequence2Vec (ours)	SVM	0.848	0.860	0.848	0.842	0.736	0.864	10.559
	NB	0.475	0.693	0.475	0.433	0.451	0.719	0.288
	MLP	0.819	0.818	0.819	0.807	0.681	0.836	101.027
	KNN	0.817	0.810	0.817	0.810	0.617	0.832	0.938
	RF	0.844	0.850	0.844	0.836	0.704	0.831	13.519
	LR	**0.856**	**0.853**	**0.856**	**0.847**	**0.781**	**0.888**	62.573
	DT	0.805	0.811	0.805	0.802	0.604	0.809	3.467

Table: Classification results (averaged over 5 runs) for different evaluation metrics for **Coronavirus Host Dataset**. The best values are shown in bold.

Nanobody-Antigen Binding Prediction

Antigen

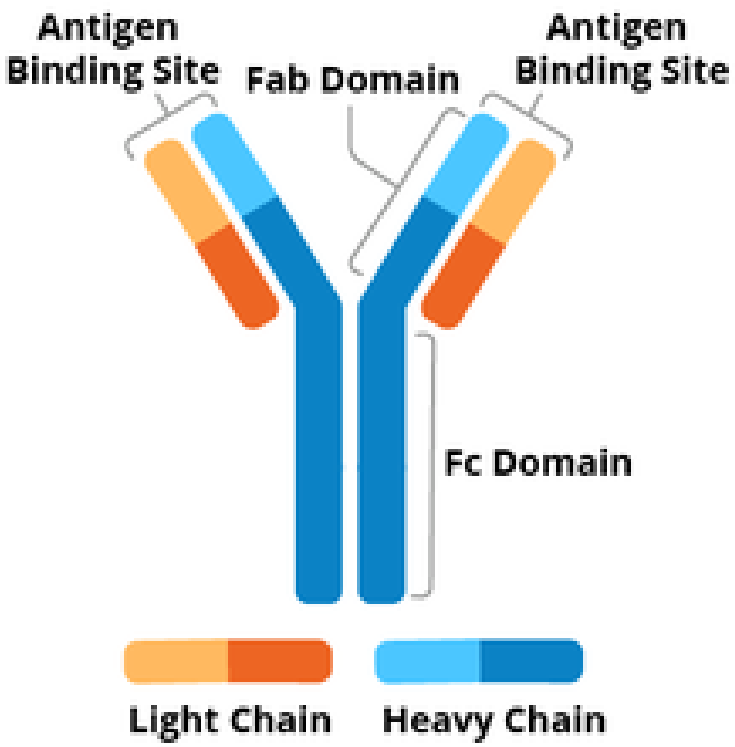
- Toxin, bacteria, or virus
- Induces an immune response
  1. Produces antibodies/nanobodies
- Protein sequence - amino acid residues



Source: Microbe Notes

Antibody

- Large and Y-shaped protein
- Identifies and neutralizes antigens



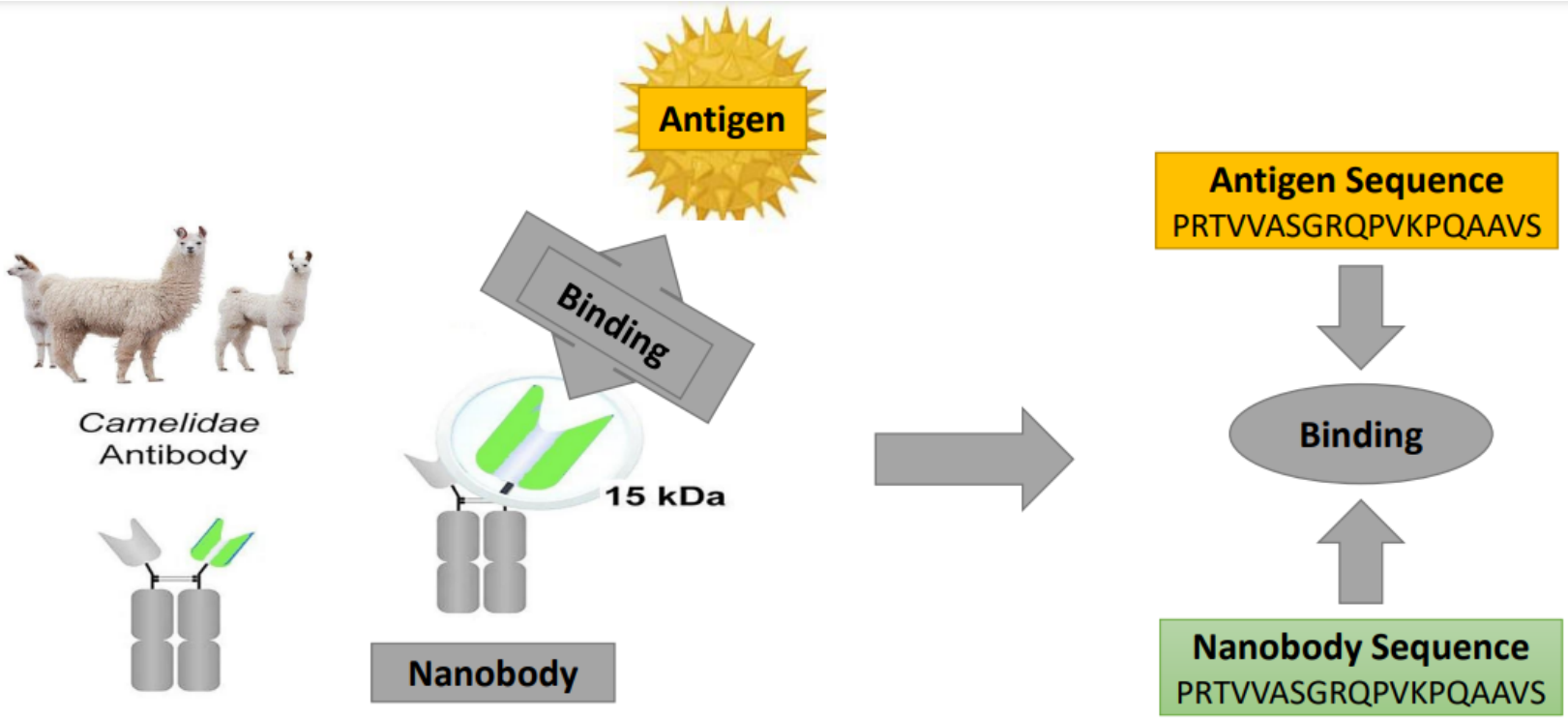
Source: addgene Blog

Nanobody (Nb)

- single-domain and heavy-chain antibodies (sdAb)
- natural occurrence

Nanobody-Antigen Binding

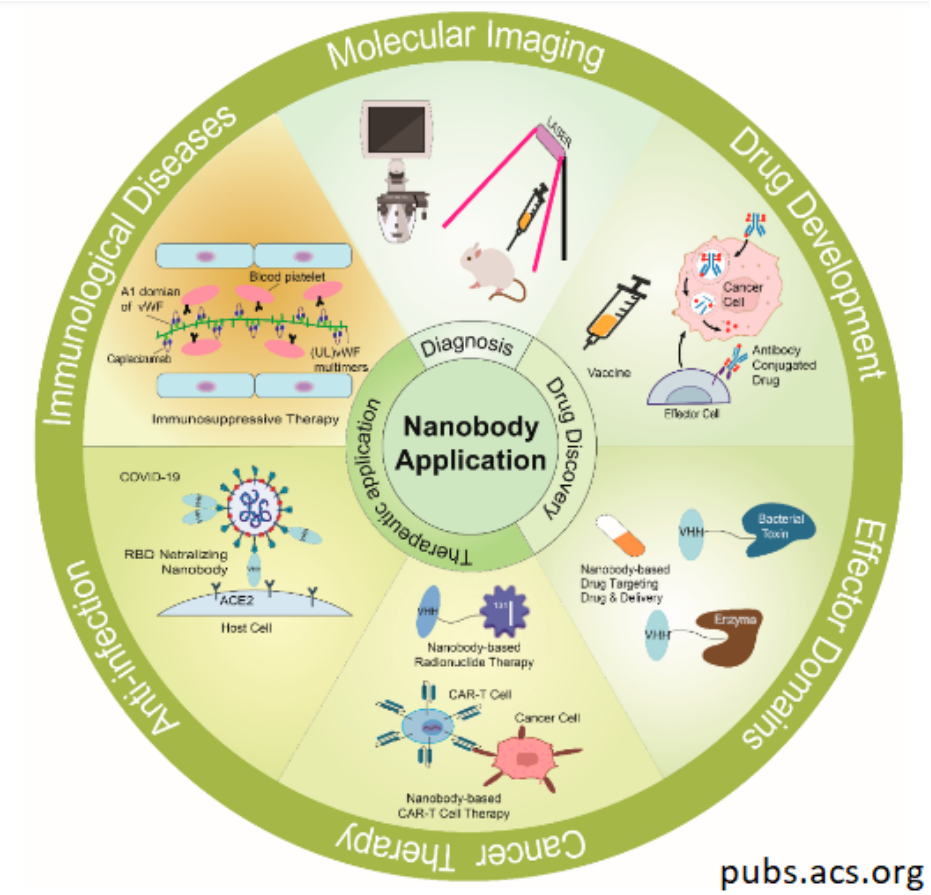
Nanobody bind selectively to a specific antigen



Source: Bioss Antibodies

Nanobody - Applications

- Biotechnology and Medicine
- Therapeutics - treatment of SARS-CoV-2
- Diagnostics



Source: Microbe Notes

Nanobody Design (*in vivo*)

- Antigen injected in animals
- Blood sample collected after a few months
- Nanobody extracted, cloned, and developed

Limitations:

- Costly [13]
- Time-consuming [14]
- Animal sacrifice [15]
- Batch-to-batch variation

Nanobody Design (*in silico*) - Motivation

- Vast protein databases (PDB, etc.) [16, 17]
- Known examples of antigen-nanobody pairs
- Advanced machine learning and deep learning techniques
  1. Transformers, neural networks

Nanobody Design (*in silico*) - Pipeline

1. Antigen-nanobody 3D structure prediction
2. Binding site prediction
3. Molecular docking
4. Binding affinity prediction

Nanobody Design (*in silico*) - Challenges

- Complexity in predicting nanobody 3D structure
- Inaccurate binding site recognition
- Computational costs
- Protein (antigen, nanobody) sequence data >> structural data

Problem Formulation

Before predicting nanobody-antigen interaction

1. 3D structures available
2. Known binding sites
3. Generated antigen-nanobody complex

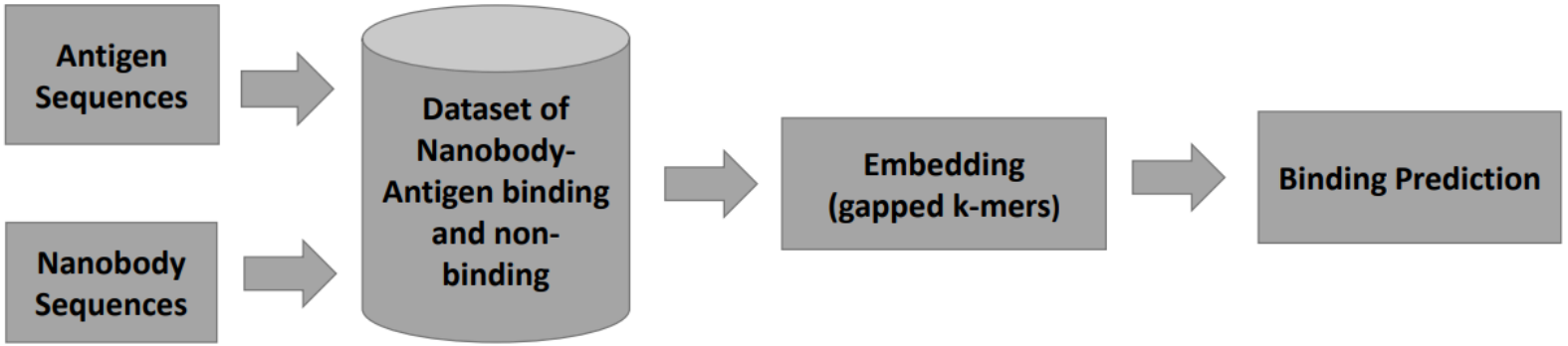
- These steps are:
  - Time-consuming
  - Computationally expensive

Given sequences of antigen-nanobody, predict whether they bind or not

Experimental Setup

A supervised learning problem:

- Known examples of antigen-nanobody sequences - Dataset
- Examples labeled as binding or non-binding - Ground truth
- Learn a formula of features - Classification problem



Dataset

Sequence length statistics for antigen and nanobody sequences

- Collected from UniProt [1] and Single Domain Antibody Database (sdAb) [2]

Type	Count	Sequence Length Statistics				
		Mean	Min	Max	Std. Dev.	Median
Antigens	47	671.51	158	1816	421.24	480
Nanobodies	365	122.84	104	175	8.87	123

Statistics for nanobody sequences binding to each antigen

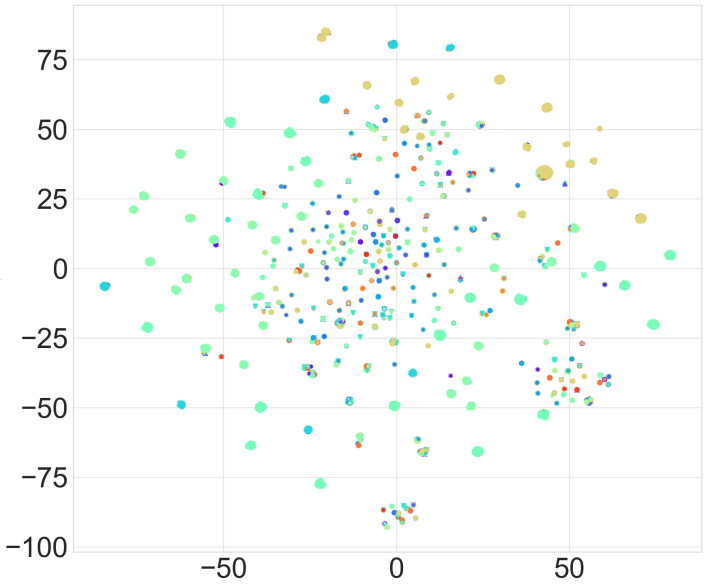
Type	Mean	Min	Max	Std. Dev.	Median
Nanobodies in each antigen	7.77	1	36	9.28	4

[1] https://www.uniprot.org/
[2] http://www.sdab-db.ca/

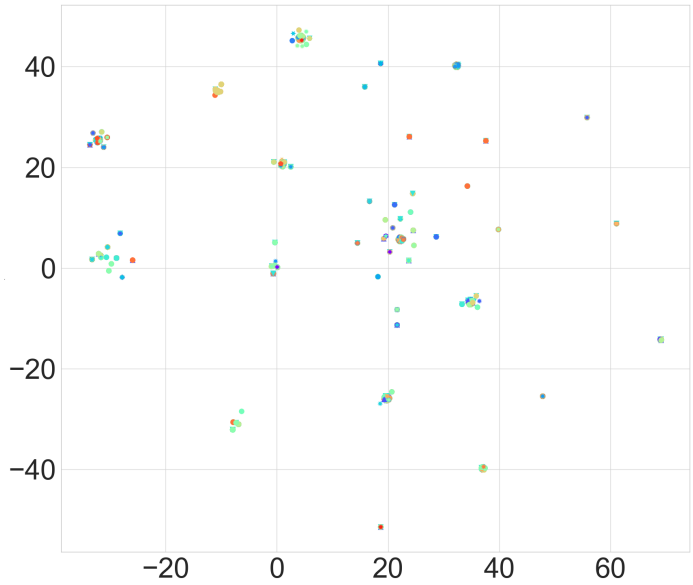
Feature Extraction

- Extracted features of each nanobody and antigen sequence
  1. Charge at pH, Grand Average of Hydropathy (GRAVY) [18], molecular weight, aromaticity
  2. Instability index [19], isoelectric point, secondary structure fraction (helix, turn, and sheet) [20]
  3. Molar extinction coefficient (reduced and oxidized)
- Obtained Non-Binding Nb-Ag Pairs
  1. Pairwise edit distance between antigens and nanobodies - proximity matrix
  2. 1388 additional Nb-Ag binding pairs
  3. 1728 Ng-Ab non-binding pairs in total

Data Visualization - t-SNE



(a) Nanobody



(b) Antigen

The colored data points show the different antigen categories (47 in total)

Sampling Strategy and Evaluation Metrics

- Randomly split:
  - Full data into 70:30 training and testing set
  - Training set into 90:10 training and validation set
- Experiments repeated 10-folds, and average results reported
- Standard classification metrics:
  - Accuracy, Precision, Recall
  - F1-score (weighted and macro) and Area Under Curve (AUC)

Methods Employed

- Embedding Generation - gapped k-mers
  1. Advantages:
     1.1 Increased sensitivity
     1.2 Enhanced flexibility
     1.3 Comprehensive motif representation
     1.4 Improved specificity
- Comparison with baseline models
  1. Spike2Vec [5], Minimizers [21], and PWM2Vec [7]
- Machine Learning Classifiers
  1. Support Vector Machine (SVM), Naive Bayes (NB), Multi-Layer Perceptron (MLP), K-Nearest Neighbors (KNN), Random Forest (RF), Logistic Regression (LR), and Decision Tree (DT)

Results - Without Sequence Features

Embeddings	Algo.	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (sec.) ↓
Spike2Vec	SVM	0.818	0.824	0.818	0.818	0.818	0.819	5.662
	NB	0.813	0.815	0.813	0.813	0.813	0.813	0.103
	MLP	0.844	0.846	0.844	0.844	0.844	0.844	4.075
	KNN	0.892	0.893	0.892	0.892	0.892	0.892	1.290
	RF	0.906	0.911	0.906	**0.906**	**0.906**	0.906	3.725
	LR	0.813	0.815	0.813	0.813	0.813	0.814	2.417
	DT	0.878	0.878	0.878	0.878	0.877	0.878	1.293
Minimizers	SVM	0.824	0.826	0.824	0.823	0.823	0.823	5.444
	NB	0.791	0.792	0.791	0.790	0.790	0.790	0.091
	MLP	0.844	0.845	0.844	0.844	0.844	0.844	2.997
	KNN	0.880	0.880	0.880	0.880	0.880	0.880	1.257
	RF	0.892	0.898	0.892	0.892	0.892	0.893	4.000
	LR	0.811	0.812	0.811	0.811	0.811	0.811	1.343
	DT	0.851	0.851	0.851	0.850	0.850	0.850	1.677
PWM2Vec	SVM	0.810	0.812	0.810	0.809	0.809	0.809	5.732
	NB	0.792	0.793	0.792	0.792	0.792	0.792	0.095
	MLP	0.820	0.821	0.820	0.820	0.819	0.820	3.730
	KNN	0.875	0.875	0.875	0.875	0.875	0.875	1.232
	RF	0.892	0.899	0.892	0.891	0.891	0.892	3.746
	LR	0.804	0.805	0.804	0.804	0.804	0.804	7.137
	DT	0.866	0.866	0.866	0.866	0.866	0.866	1.692
Gapped *k*-mers	SVM	0.814	0.816	0.814	0.813	0.813	0.812	5.740
	NB	0.798	0.798	0.798	0.797	0.797	0.796	0.087
	MLP	0.824	0.825	0.824	0.824	0.824	0.824	2.886
	KNN	0.885	0.886	0.885	0.885	0.885	0.885	0.995
	RF	**0.907**	**0.912**	**0.907**	0.894	0.894	**0.908**	3.755
	LR	0.812	0.813	0.812	0.812	0.812	0.812	4.395
	DT	0.872	0.872	0.872	0.872	0.871	0.872	1.777

Results - With Sequence Features

Embeddings	Algo.	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (sec.) ↓
Spike2Vec	SVM	0.791	0.796	0.791	0.790	0.790	0.790	8.804
	NB	0.695	0.737	0.695	0.680	0.678	0.691	0.085
	MLP	0.811	0.814	0.811	0.811	0.811	0.811	2.326
	KNN	0.844	0.845	0.844	0.844	0.844	0.844	0.953
	RF	0.897	0.903	0.897	0.896	0.896	0.898	3.890
	LR	0.827	0.827	0.827	0.827	0.826	0.827	1.183
	DT	0.847	0.848	0.847	0.847	0.847	0.847	1.246
Minimizers	SVM	0.778	0.783	0.778	0.777	0.777	0.777	10.938
	NB	0.674	0.736	0.674	0.649	0.647	0.670	0.094
	MLP	0.801	0.806	0.801	0.800	0.800	0.800	3.228
	KNN	0.842	0.842	0.842	0.842	0.842	0.842	0.827
	RF	0.896	0.902	0.896	0.896	0.896	0.897	3.801
	LR	0.823	0.823	0.823	0.823	0.823	0.823	1.167
	DT	0.846	0.846	0.846	0.845	0.845	0.845	1.297
PWM2Vec	SVM	0.766	0.770	0.766	0.765	0.765	0.766	9.569
	NB	0.679	0.726	0.679	0.659	0.657	0.674	0.087
	MLP	0.811	0.813	0.811	0.811	0.811	0.811	2.889
	KNN	0.828	0.828	0.828	0.827	0.827	0.827	0.768
	RF	0.893	0.901	0.893	0.892	0.892	0.894	3.765
	LR	0.819	0.819	0.819	0.819	0.819	0.819	1.495
	DT	0.851	0.851	0.851	0.851	0.851	0.850	1.279
Gapped *k*-mers	SVM	0.785	0.792	0.785	0.784	0.783	0.784	9.270
	NB	0.720	0.745	0.720	0.712	0.711	0.718	0.086
	MLP	0.807	0.810	0.807	0.806	0.806	0.806	2.432
	KNN	0.839	0.839	0.839	0.839	0.838	0.838	0.753
	RF	0.895	0.901	0.895	0.894	0.894	0.895	3.468
	LR	0.823	0.823	0.823	0.823	0.823	0.823	1.123
	DT	0.860	0.861	0.860	0.860	0.860	0.860	0.955

Discussion

- Gapped k-mers spectrum outperforms all other embeddings
  1. Average accuracy, precision, recall, and ROC-AUC - Random forest classifier
- Spike2Vec performs better than other embeddings
  1. Weighted and Macro F1 - Random forest classifier
- Comparison of embeddings with and without sequence features
  1. Multicollinearity and curse of dimensionality
- Student t-test to evaluate the significance of the results
  1. p-values predominantly less than 0.05

t-Cell Sequence Classification

T Cell Receptor (TCR) Sequences

- Located on the surface of T cells
- Responsible for antigen recognition
- Is a core component in the adaptive immune system
  1. Activated in response to specific pathogens or antigens
- Analyzing TCR sequences
  1. Help us to classify cancer types
  2. Very important in early stage detection or immunotherapy

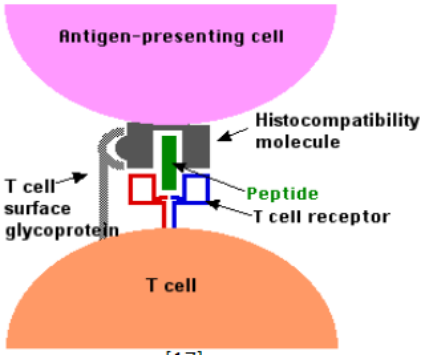


Figure: Source: https://proteinswebteam.github.io/interpro-blog/potm/2005_3/Page1.htm

Pseudo-Amino Acid Composition

- Pseudo-amino acid composition (PseAAC) represents protein sequences considering the frequency and physicochemical properties of amino acids [22].
- It provides a comprehensive representation compared to traditional amino acid composition [23].
- In PseAAC, each amino acid is represented by numerical values describing characteristics like hydrophobicity, polarity, charge, molecular weight, and solvent accessibility [24].

Pseudo-Amino Acid Composition

Hydrophobicity:

- Hydrophobicity in TCR sequences refers to the tendency of certain amino acids to be hydrophobic [25].
- Represented using a scale that assigns numerical values to amino acids based on hydrophobic nature.
- Uses a window size of 3 for calculation and Kyte-Doolittle scale [18].
- Hydrophobicity score $H(S)$ is calculated as:

$$H(S) = \sum_{i=1}^{L} H(S[i]) \tag{7}$$

Polarity:

- Polarity in TCR sequences refers to the distribution of polar and nonpolar amino acids in variable regions [25].

Charge:

- Charge refers to the distribution of charged amino acids in TCR sequences, especially in variable regions [26].

Pseudo-Amino Acid Composition

Molecular Weight:

- Sum of atomic weights of all amino acids in TCR proteins [27].
- Calculated as:

$$\text{Molecular weight} = \sum_{i=1}^{N} M_i \tag{8}$$

Solvent Accessibility:

- Degree to which amino acids in TCR are exposed to solvent [28].
- These properties provide insights into TCR structure, function, antigen recognition, and are useful for targeted immunotherapies and personalized cancer treatments [29, 30, 31, 32, 33].

Algorithm

Algorithm PseAAC2Vec Protein Encoding

```
Input: TCR Sequences S, Output: PseAAC2Vec Embedding
PP ← Dictionary of physicochemical properties
window_size ← 3                                   ▷ Hyperparameter, tunned using validation set
Final_Vectors ← []
for seqs in range(len(S)) do
    protein_sequence ← S[seqs]
    SeqLen ← length(protein_sequence)
    PL ← length(physicochemical_properties)
    VecLen ← PL × window_size
    PseAAC2Vec_feature_vector ← zeros(SeqLen, PL)
    PP ← physicochemical_properties.values()
    for i in range(SeqLen) do
        for k, PEncode in enumerate(PP) do
            AA ← protein_sequence[i]
            PseAAC2Vec_feature_vector[i, k] ← PEncode[AA]
    ExVec ← zeros(SeqLen, VecLen + PL)
    for i in range(SeqLen) do
        for j in range(window_size) do
            if i - j ≥ 0 then
                AA ← protein_sequence[i - j]
                for k, PN in enumerate(PP.keys()) do
                    PEncode = PP[PN]
                    ExVec[i, j ∗ PL + k] = PEncode[AA]
        ExVec[i, VecLen :] ← PseAAC2Vec_feature_vector[i]
    flattened_feature_vector ← FLATTEN(ExVec)
    Final_Vectors.append(flattened_feature_vector)
return PseAAC2VecEmbedding
```

Dataset

Cancer Type	Total Sequences	Unique Sequences	Sequence Length Statistics		
			Min.	Max.	Average
Melanoma	8750	7123	9	24	15
Retroperitoneal	5505	4763	9	20	15
Pancreatic	2887	2883	11	25	15
Ovarian	583	512	10	20	15
Total	17725	15281	-	-	-

Table: Dataset Statistics of TCR Sequences. The table shows the total number and the unique number of sequences for each cancer type, the minimum, the average, and the maximum length of TCR sequences in the dataset used for the experiments in this study.

Results

Embeddings	Algo.	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (sec.) ↓
PWM2Vec	NB	0.3234	0.4239	0.3234	0.2856	0.2392	0.5242	4.2852
	MLP	0.5199	0.5078	0.5199	0.5128	0.3899	0.5995	77.1383
	KNN	0.5144	0.4674	0.5144	0.4773	0.3105	0.5555	4.6783
	RF	0.5819	0.5805	0.5819	0.5101	0.3776	0.5905	113.3189
	LR	0.4943	0.4885	0.4943	0.4882	0.3662	0.5856	505.1448
	DT	0.5018	0.5006	0.5018	0.5011	0.3847	0.5967	95.9090
Spike2Vec	NB	0.4645	0.4863	0.4645	0.4569	0.3359	0.5755	0.0380
	MLP	0.5382	0.4814	0.5382	0.4910	0.3105	0.5622	24.6133
	KNN	0.5235	0.4840	0.5235	0.4942	0.3304	0.5647	3.1544
	RF	0.6041	0.5650	0.6041	0.5503	0.4120	0.6124	20.9789
	LR	0.5457	0.5000	0.5457	0.4817	0.2931	0.5583	0.6857
	DT	0.5182	0.5165	0.5182	0.5172	0.3957	0.6051	3.2491
String Kernel	NB	0.4464	0.4664	0.4464	0.4477	0.3324	0.5706	0.3123
	MLP	0.5125	0.4939	0.5125	0.5020	0.3642	0.5834	72.9982
	KNN	0.4968	0.4535	0.4968	0.4655	0.3144	0.5533	2.2828
	RF	0.6030	0.5603	0.6030	0.5365	0.4082	0.6111	40.7817
	LR	0.5308	0.4724	0.5308	0.4805	0.3068	0.5604	5.6004
	DT	0.4964	0.4942	0.4964	0.4952	0.3787	0.5934	14.2018
WDGRL	NB	0.4850	0.3820	0.4850	0.4254	0.2522	0.5275	**0.0074**
	MLP	0.5087	0.4247	0.5087	0.4179	0.2416	0.5248	15.9686
	KNN	0.4731	0.4256	0.4731	0.4349	0.2844	0.5316	0.8438
	RF	0.5559	0.5202	0.5559	0.4934	0.3679	0.5798	4.7567
	LR	0.5048	0.4033	0.5048	0.3942	0.2216	0.5169	0.0788
	DT	0.4800	0.4766	0.4800	0.4781	0.3625	0.5800	0.2037
Auto-Encoder	NB	0.4230	0.4219	0.4230	0.4031	0.2767	0.5323	0.0510
	MLP	0.5259	0.4745	0.5259	0.4888	0.3164	0.5613	207.0182
	KNN	0.5244	0.4785	0.5244	0.4867	0.3215	0.5593	6.2745
	RF	0.5836	0.5864	0.5836	0.5102	0.3748	0.5880	35.0135
	LR	0.5389	0.4907	0.5389	0.4751	0.2893	0.5542	2.6055
	DT	0.4866	0.4795	0.4866	0.4826	0.3677	0.5832	7.7863
SeqVec	NB	0.3466	0.4820	0.3466	0.3049	0.2568	0.5438	5.3837
	MLP	0.4996	0.5042	0.4996	0.5016	0.3793	0.5937	109.9893
	KNN	0.5120	0.4626	0.5120	0.4739	0.3121	0.5551	5.4640
	RF	0.5675	0.5905	0.5675	0.4822	0.3438	0.5727	166.7319
	LR	0.5476	0.5269	0.5476	0.5333	0.4059	0.6054	862.3937
	DT	0.4673	0.4691	0.4673	0.4681	0.3481	0.5730	158.9599
Protein Bert	-	0.5344	0.5077	0.5344	0.4724	0.2865	0.5538	301.7492
PseAAC2Vec (ours)	NB	0.3071	0.4824	0.3071	0.1827	0.1632	0.5128	0.3952
	MLP	0.5417	0.4876	0.5417	0.4908	0.3082	0.5646	17.8969
	KNN	0.4985	0.4561	0.4985	0.4625	0.3099	0.5457	1.0735
	RF	**0.6190**	**0.5967**	**0.6190**	**0.5757**	**0.4525**	**0.6300**	5.6696
	LR	0.5401	0.4976	0.5401	0.4729	0.2847	0.5505	130.54
	DT	0.5327	0.5343	0.5327	0.5323	0.4205	0.6219	1.6375

Table

Results

Embedding	Time (sec.)
OHE	39.4524
PWM2Vec	62.1373
String Kernel	1014.61
Auto Encoder	161.623
Bert	257.496
SeqVec	10875.57
Spike2Vec	241.368
WDGRL	20.1513
PseAAC2Vec (ours)	**3.7313**

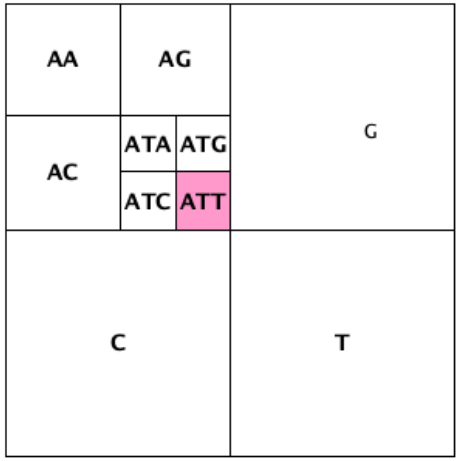
Table: Embedding generation time. The best value is shown in bold.

Image Transformation

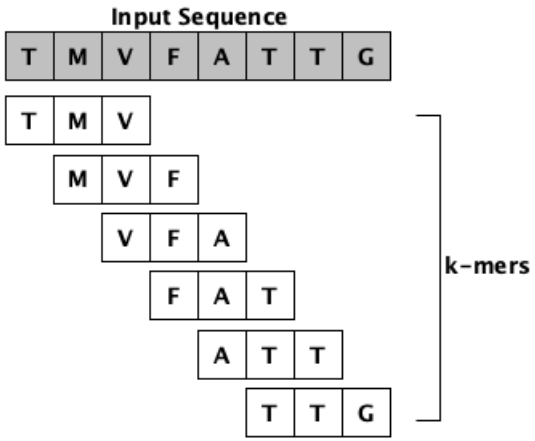
Sequence-to-Image Transformation

- 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

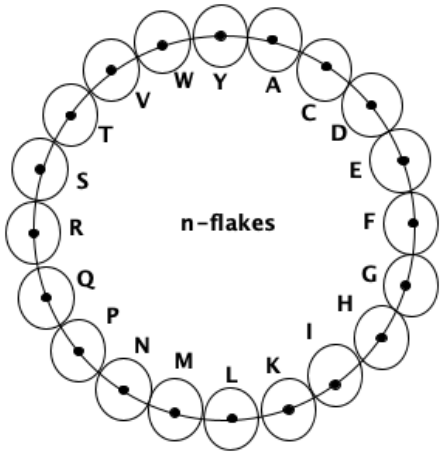
Chaos Game Representation (CGR)



(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:

$$x[i] = r \cdot sin(\frac{2\pi i}{n} + \theta) \tag{9}$$

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.

$$y[i] = r \cdot cos(\frac{2\pi i}{n} + \theta) \tag{10}$$

- 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.

Spike2CGR

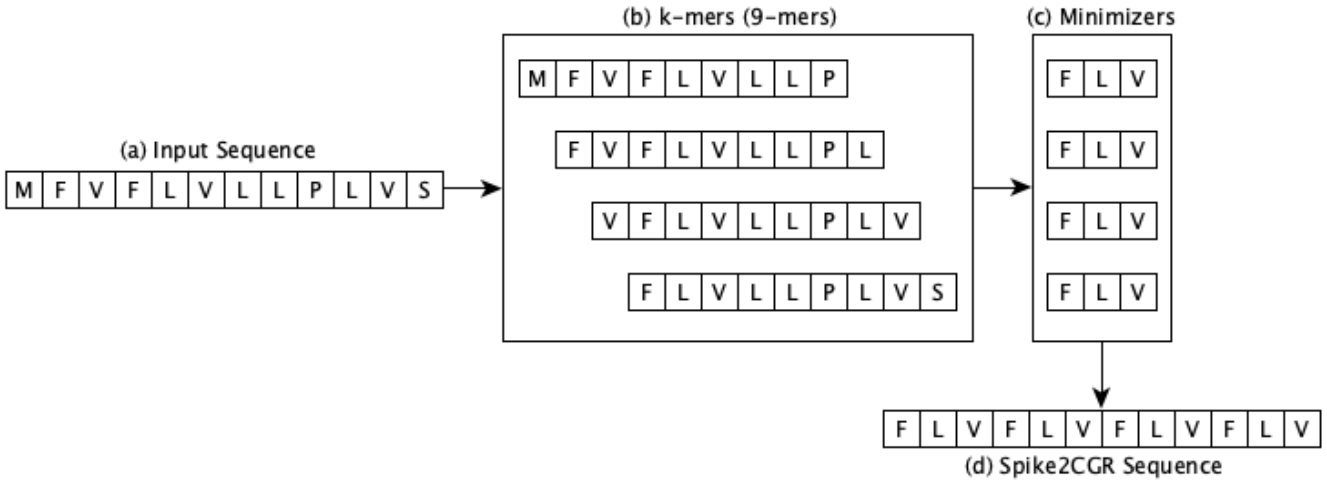
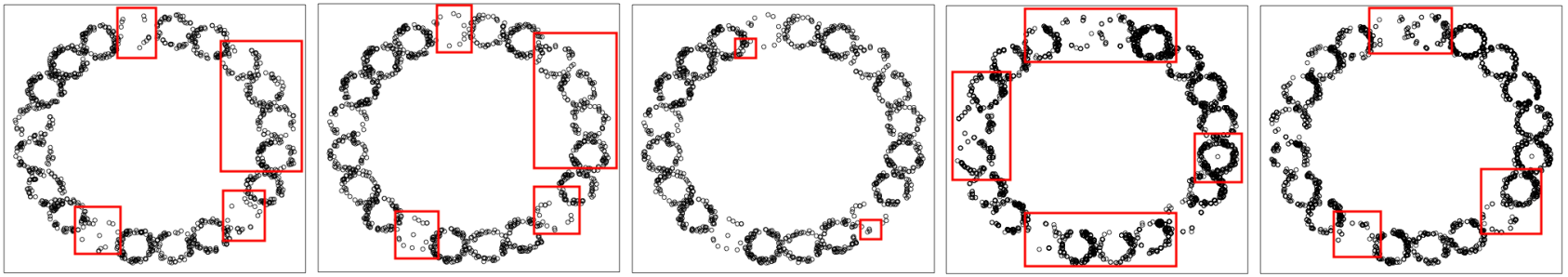


Figure: Workflow of Spike2CGR for a given sequence. For a given spike sequence, steps from (a) to (d) are followed to generate the corresponding Spike2CGR sequence.

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).

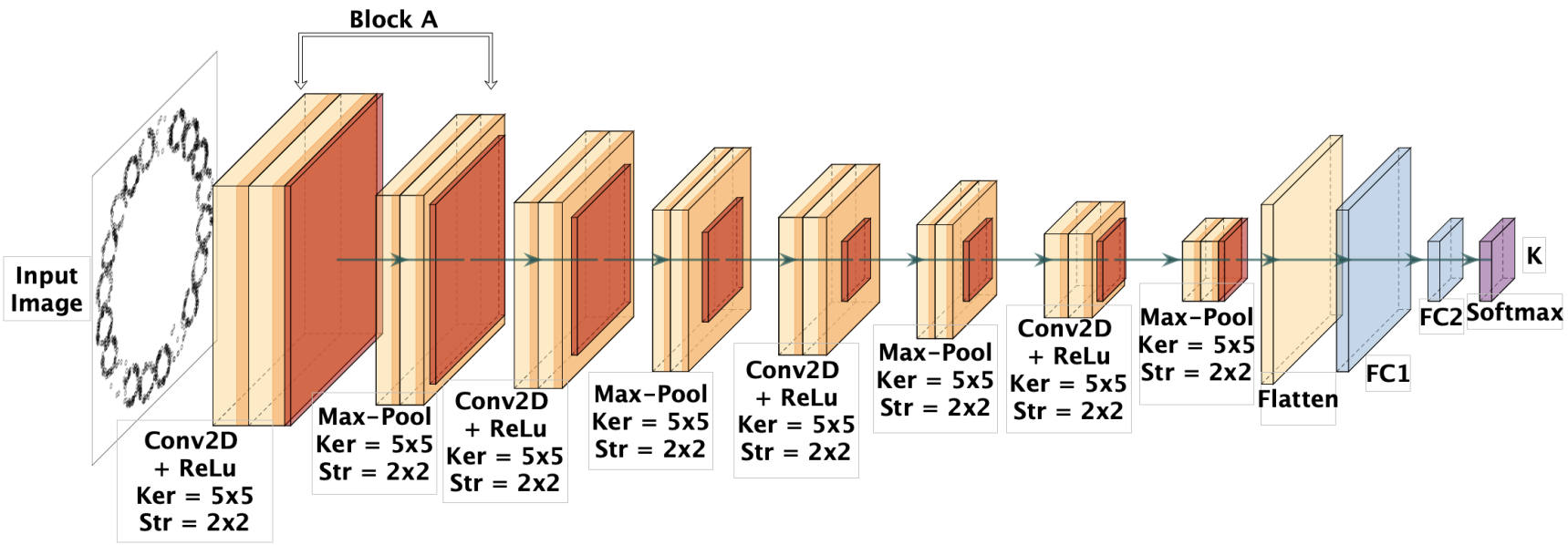


Figure: The architectures of the 4-layer CNN model. Here ker represents kernel and str represents stride filter size.

Dataset

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

Table: Dataset statistics for different coronavirus variants (32738 in total).

Results

DL Model	Method	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (hrs.) ↓
3-Layer Tab CNN	OHE	0.472	0.301	0.472	0.368	0.060	0.552	0.594
	WDGRL	0.636	0.457	0.636	0.523	0.263	0.594	**0.380**
4-Layer Tab CNN	OHE	0.637	0.469	0.637	0.528	0.157	0.511	0.977
	WDGRL	0.688	0.517	0.688	0.582	0.227	0.637	0.866
1-Layer CNN	Chaos	0.700	0.680	0.696	0.651	0.563	0.673	8.195
	Spike2Vec	0.733	0.690	0.733	0.679	0.679	0.850	7.779
	PWM2Vec	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		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	0.700	0.669	0.697	0.652	0.564	0.645	6.394
	Spike2Vec	0.740	0.730	0.744	0.729	0.736	0.725	7.329
	PWM2Vec	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		-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	0.740	0.722	0.739	0.717	0.696	0.809	5.658
	Spike2Vec	0.750	0.723	0.750	0.715	0.725	0.838	6.919
	PWM2Vec	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		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	0.740	0.686	0.737	0.706	0.678	0.728	7.986
	Spike2Vec	0.750	0.686	0.749	0.712	0.720	0.842	7.447
	PWM2Vec	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		3	4.5	3.1	3.2	3.6	11.5	-33.45

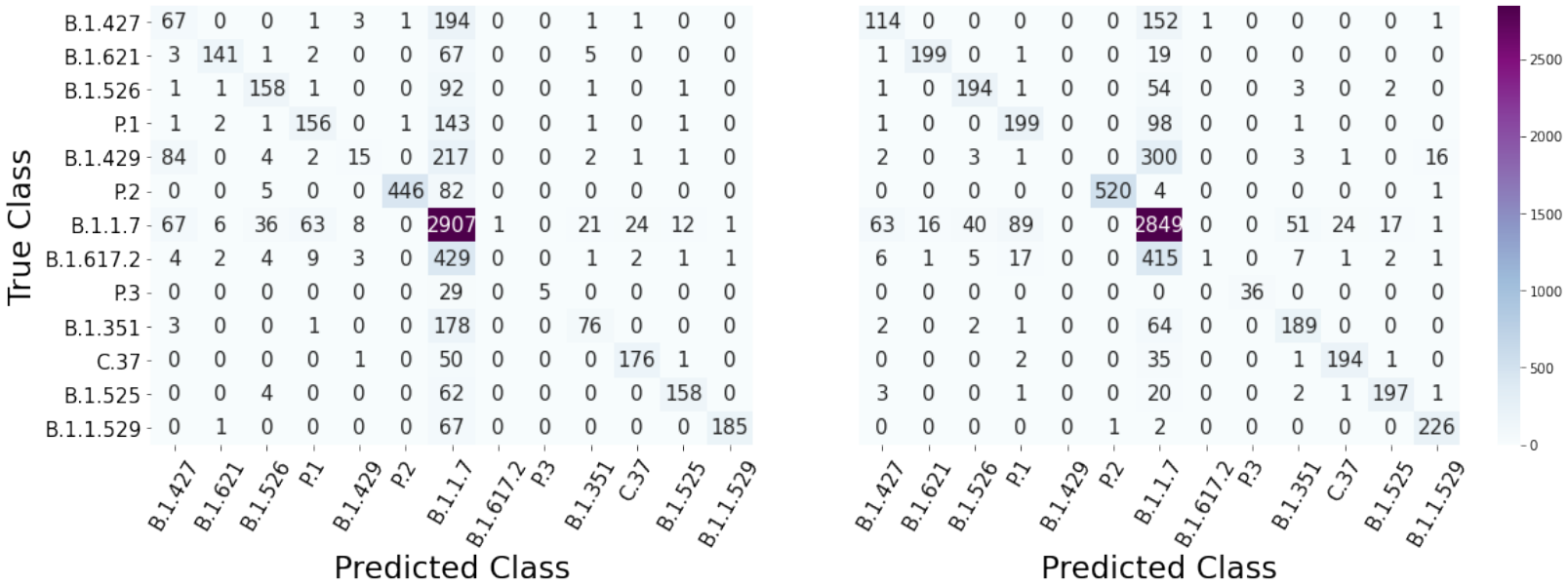
Results

DL Model	Method	Acc. ↑	Prec. ↑	Recall ↑	F1 (Weig.) ↑	F1 (Macro) ↑	ROC AUC ↑	Train Time (hrs.) ↓
RESNET50 Pre-Trained Model	Chaos	0.680	0.644	0.676	0.641	0.547	0.743	10.654
	Spike2Vec	0.711	0.657	0.710	0.666	0.644	0.759	10.746
	PWM2Vec	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		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	0.480	0.233	0.483	0.315	0.050	0.500	27.398
	Spike2Vec	0.470	0.221	0.470	0.301	0.049	0.500	26.599
	PWM2Vec	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		1.5	1.2	1.2	1.2	0	0	8.4

Results



(a) Chaos

(b) Spike2CGR

Molecular Properties (Weights)

- Kyte and Doolittle (KD) Hydropathy Scale
  1. Assigns numerical values to amino acids based on their hydrophobicity/hydrophilicity, used in predicting protein structure and function.
- Eisenberg Hydrophobicity Scale
  1. Quantifies the hydrophobicity of amino acids, aiding in protein structure prediction and understanding protein interactions with hydrophobic environments.
- Hydrophilicity Scale
  1. 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
  1. Evaluates the flexibility or rigidity of amino acids, important for predicting protein dynamics, conformational changes, and flexibility in molecular interactions.
- Hydropathy Scale
  1. Ranks amino acids based on their hydrophobic or hydrophilic nature, assisting in studying protein folding, membrane protein structure, and transmembrane domains.

Workflow

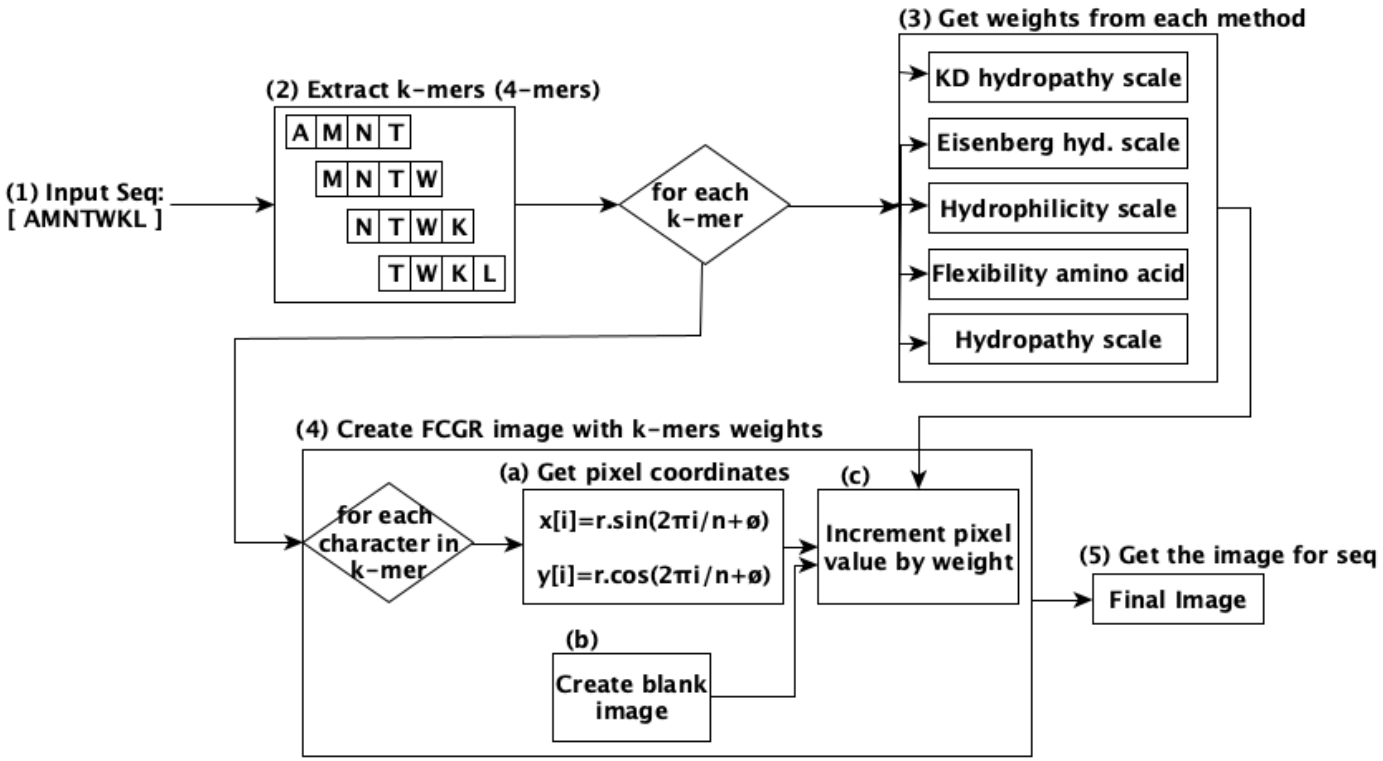


Figure: Workflow of the proposed method for creating an image of a sequence.

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

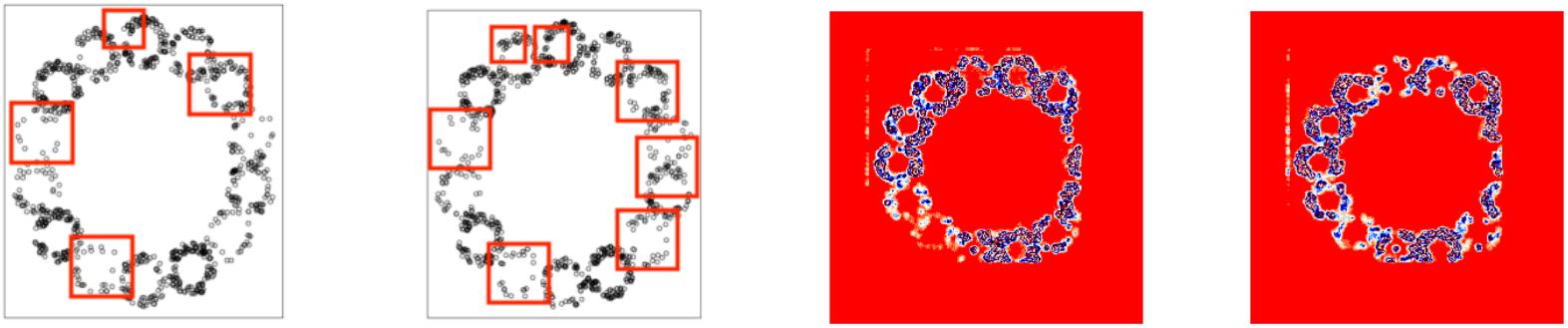
- Feature-engineering-based methods
  - One Hot Encoding (OHE): created embeddings are sparse and face curse of dimensionality challenge.
  - Wasserstein Distance Guided Representation Learning (WDGRL): require large training data for optimal performance.
  - Position Specific Scoring Matrix (PSSM)
- 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.

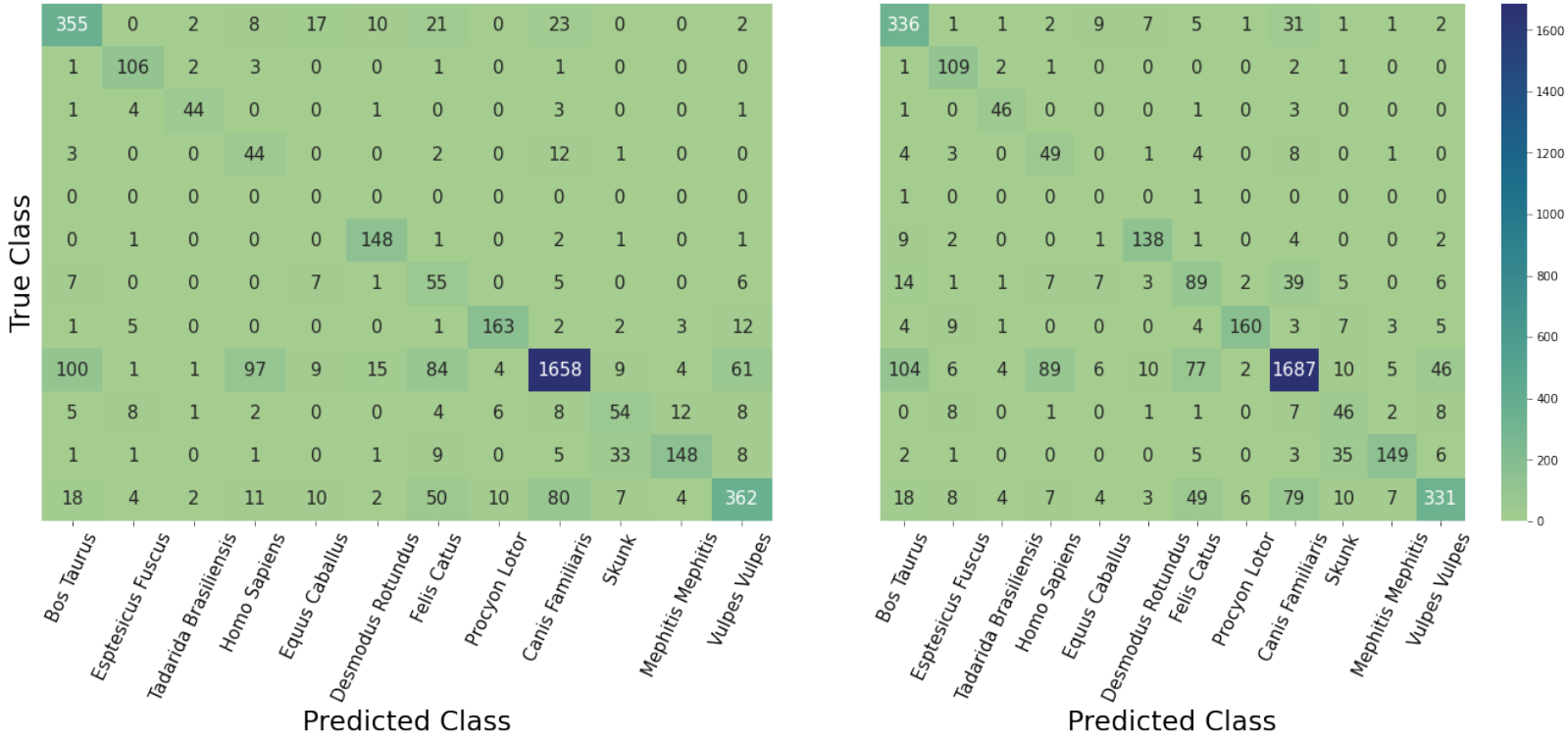
Results



(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



(a) Chaos

(b) Eisenberg

Figure

Bézier curves

The general formula [34] of the Bézier curve is

$$BZ(t) = \Sigma_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} P_i \tag{11}$$

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,

$$x = (1-t)^3 \cdot P_{0_x} + 3 \cdot (1-t)^2 \cdot t \cdot P_{1_x} + 3 \cdot (1-t) \cdot t^2 \cdot P_{2_x} + t^3 \cdot P_{3_x} \tag{12}$$

$$y = (1-t)^3 \cdot P_{0_y} + 3 \cdot (1-t)^2 \cdot t \cdot P_{1_y} + 3 \cdot (1-t) \cdot t^2 \cdot P_{2_y} + t^3 \cdot P_{3_y} \tag{13}$$

Bézier curves

```
Input: Sequence seq, No. of Parameters m
Output: Image img
conPoint = {}                                          ▷ dictionary for control points
for i, aa ∈ 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 ∈ [0, 1]                           ▷ list of m pairs of parameters
ite = 3                                                ▷ no. of deviations pair points. It can have any value.
for a ∈ seq : do                                       ▷ every amino acid a in seq
    org_point = conPoint[a]                            ▷ control point of a
    points = [org_point]
    for i ∈ (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

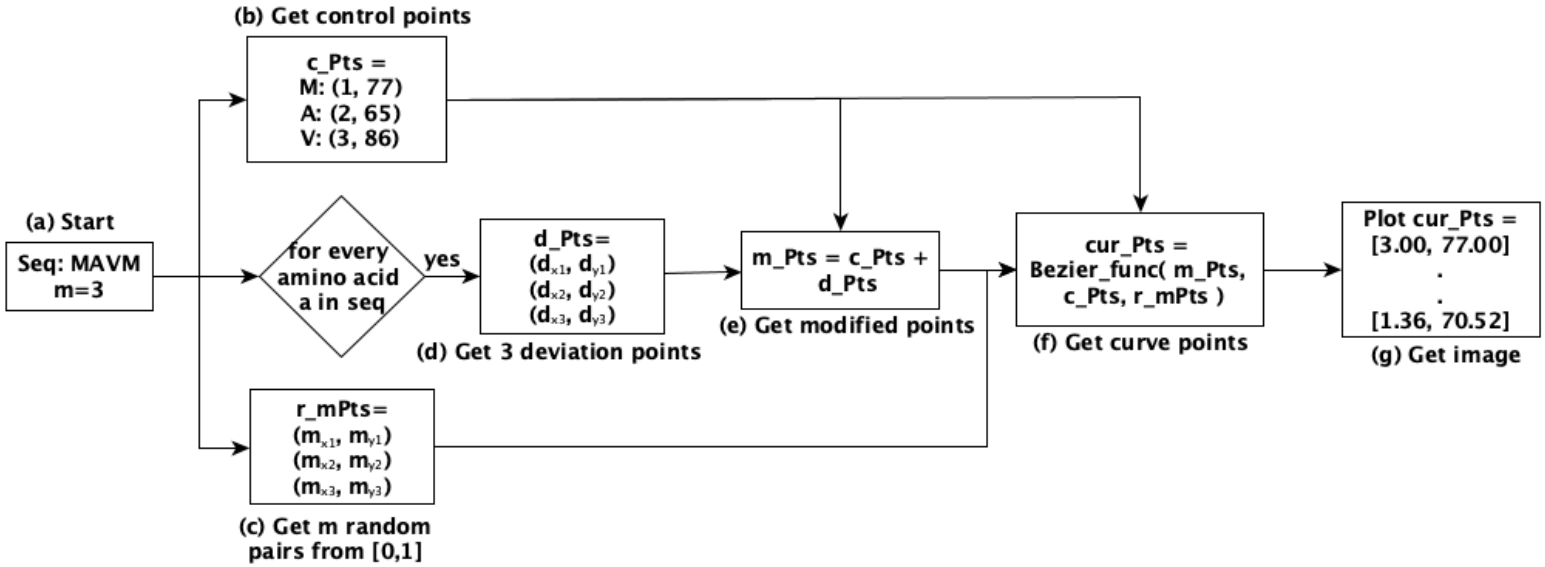
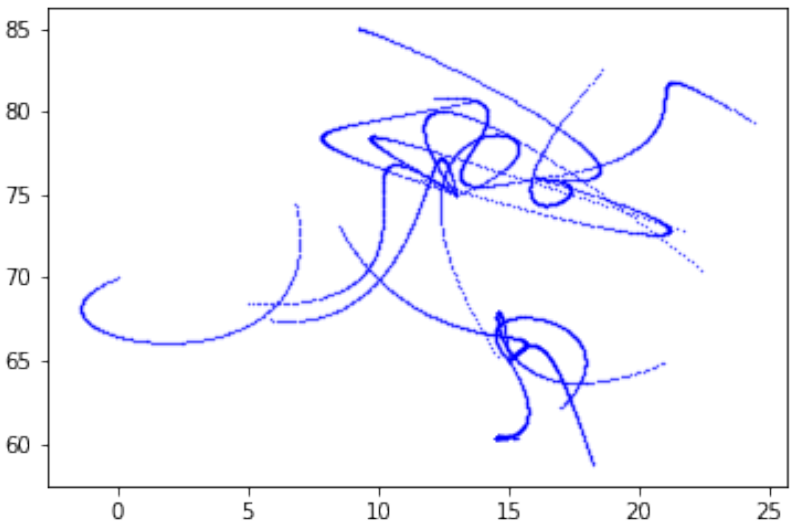
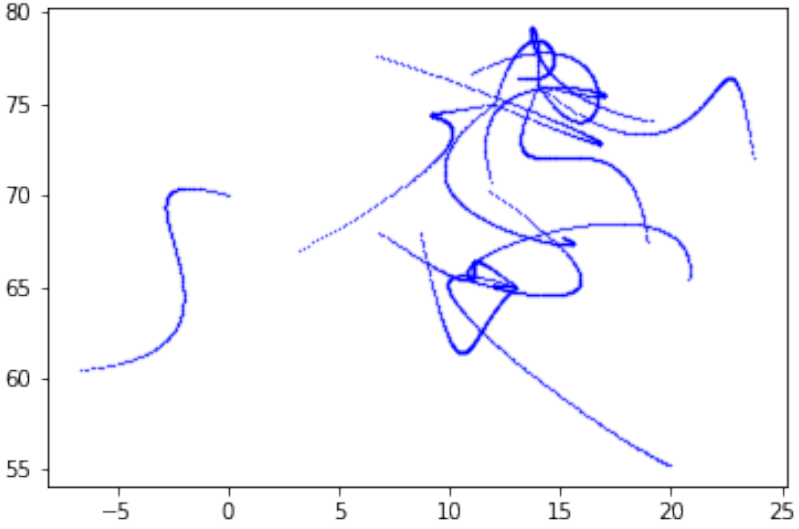


Figure: The workflow of our system to create an image from a given sequence and a number of parameters *m*. We have used "MAVM" as an input sequence here. Note that the *cur_Pts* consists of a set of values for x coordinates and y coordinates.

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

Subcellular Locations	Count	Protein Subcellular Sequence Length		
		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	<u>0.964</u>	<u>0.967</u>	<u>0.964</u>	<u>0.961</u>	<u>0.907</u>	<u>0.948</u>	11.415
	% improv. of Bézier from FCGR		59.6	69.9	59.6	65.1	75.2	39.2	-197.96
	% impro. of Bézier from Spike2CGR		59.6	79.2	59.6	74.7	80.2	38.3	-3.8

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

Thank you for your attention !

Feel Free To Contact Me

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

References

- J. L. Carter and M. N. Wegman, "Universal classes of hash functions," in *ACM symposium on Theory of comp.*, 1979, pp. 106–112.
- GISAID Website, https://www.gisaid.org/, 2021, [Online; accessed 29-December-2021].
- B. E. Pickett, E. L. Sadat, Y. Zhang, J. M. Noronha, R. B. Squires, V. Hunt, M. Liu, S. Kumar, S. Zaremba, Z. Gu *et al.*, "Vipr: an open bioinformatics database and analysis resource for virology research," *Nucleic acids research*, vol. 40, no. D1, pp. D593–D598, 2012.
- Human DNA, https://www.kaggle.com/code/nageshsingh/demystify-dna-sequencing-with-machine-learning/data, 2022, [Online; accessed 10-October-2022].
- 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.

R. Singh, A. Sekhon, K. Kowsari, J. Lanchantin, B. Wang, and Y. Qi, "Gakco: a fast gapped k-mer string kernel using counting," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2017, pp. 356–373.

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.

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.

S. Ali, B. Sahoo, M. A. Khan, A. Zelikovsky, I. U. Khan, and M. Patterson, "Efficient approximate kernel based spike sequence classification," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2022.

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 *et al.*, "Proteinbert: A universal deep-learning model of protein sequence and func." *Bioinformatics*, vol. 38, no. 8, 2022.

A. Ramon, A. Saturnino, K. Didi, M. Greenig, and P. Sormanni, "Abnativ: Vq-vae-based assessment of antibody and nanobody nativeness for engineering, selection, and computational design," *bioRxiv*, pp. 2023–04, 2023.

C. Ye, W. Hu, and B. Gaeta, "Prediction of antibody-antigen binding via machine learning: Development of data sets and evaluation of methods," *JMIR Bioinformatics and Biotechnology*, vol. 3, no. 1, p. e29404, 2022.

N. L. Miller, T. Clark, R. Raman, and R. Sasisekharan, "Learned features of antibody-antigen binding affinity," *Frontiers in Molecular Biosciences*, vol. 10, p. 1112738, 2023.

H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," *Nucleic acids research*, vol. 28, no. 1, pp. 235–242, 2000.

S. K. Burley, H. M. Berman, C. Bhikadiya, C. Bi, L. Chen, L. Di Costanzo, C. Christie, K. Dalenberg, J. M. Duarte, S. Dutta *et al.*, "Rcsb protein data bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy," *Nucleic acids research*, vol. 47, no. D1, pp. D464–D474, 2019.

J. Kyte and R. F. Doolittle, "A simple method for displaying the hydropathic character of a protein," *Journal of molecular biology*, vol. 157, no. 1, pp. 105–132, 1982.

K. Guruprasad, B. B. Reddy, and M. W. Pandit, "Correlation between stability of a protein and its dipeptide composition: a novel approach for predicting in vivo stability of a protein from its primary sequence," *Protein Engineering, Design and Selection*, vol. 4, no. 2, pp. 155–161, 1990.

B. Haimov and S. Srebnik, "A closer look into the α-helix basin," *Scientific reports*, vol. 6, no. 1, p. 38341, 2016.

M. Roberts, W. Hayes, B. R. Hunt, S. M. Mount, and J. A. Yorke, "Reducing storage requirements for biological sequence comparison," *Bioinformatics*, vol. 20, no. 18, pp. 3363–3369, 2004.

K.-C. Chou, "Prediction of protein cellular attributes using pseudo-amino acid composition," *Proteins: Structure, Function, and Bioinformatics*, vol. 43, no. 3, pp. 246–255, 2001.

B. Rozemberczki, A. Gogleva, S. Nilsson, G. Edwards, A. Nikolov, and E. Papa, "Moomin: Deep molecular omics network for anti-cancer drug combination therapy," in *International Conference on Information & Knowledge Management (CIKM)*, 2022, pp. 3472–3483.

C.-H. Tung, C.-H. Chien, C.-W. Chen, L.-Y. Huang, Y.-N. Liu, and Y.-W. Chu, "Quatgo: Protein quaternary structural attributes predicted by two-stage machine learning approaches with heterogeneous feature encoding," *Plos one*, vol. 15, no. 4, p. e0232087, 2020.

D. Chowell, S. Krishna, P. D. Becker, C. Cocita, J. Shu, X. Tan, P. D. Greenberg, L. S. Klavinskis, J. N. Blattman, and K. S. Anderson, "Tcr contact residue hydrophobicity is a hallmark of

immunogenic cd8+ t cell epitopes," *Proceedings of the National Academy of Sciences*, vol. 112, no. 14, pp. E1754–E1762, 2015.

P. F. Robbins, Y. F. Li, M. El-Gamil, Y. Zhao, J. A. Wargo, Z. Zheng, H. Xu, R. A. Morgan, S. A. Feldman, L. A. Johnson *et al.*, "Single and dual amino acid substitutions in tcr cdrs can enhance antigen-specific t cell functions," *The Journal of Immunology*, vol. 180, no. 9, pp. 6116–6131, 2008.

Ø. Molberg, N. Solheim flÆte, T. Jensen, K. E. Lundin, H. Arentz-Hansen, O. D. Anderson, A. K. Uhlen, and L. M. Sollid, "Intestinal t-cell responses to high-molecular-weight glutenins in celiac disease," *Gastroenterology*, vol. 125, no. 2, pp. 337–344, 2003.

W. Zhang, A. Young, M. Imarai, S. G. Nathenson, and J. C. Sacchettini, "Crystal structure of the major histocompatibility complex class i h-2kb molecule containing a single viral peptide: implications for peptide binding and t-cell receptor recognition." *Proceedings of the National Academy of Sciences*, vol. 89, no. 17, pp. 8403–8407, 1992.

M. Smid, F. G. Rodríguez-González, A. M. Sieuwerts, R. Salgado, W. J. Prager-Van der Smissen, M. V. D. Vlugt-Daane, A. Van Galen, S. Nik-Zainal, J. Staaf, A. B. Brinkman *et al.*,

"Breast cancer genome and transcriptome integration implicates specific mutational signatures with immune cell infiltration," *Nature communications*, vol. 7, no. 1, p. 12910, 2016.

R. Wei, H. Liu, C. Li, X. Guan, Z. Zhao, C. Ma, X. Wang, and Z. Jiang, "Computational identification of 29 colon and rectal cancer-associated signatures and their applications in constructing cancer classification and prognostic models," *Frontiers in Genetics*, p. 740, 2020.

I. Alicia Luthy, A. Bruzzone, and C. Perez Pinero, "Adrenergic action in breast cancer," *Current Cancer Therapy Reviews*, vol. 8, no. 2, pp. 90–99, 2012.

V. Pourteimoor, S. Mohammadi-Yeganeh, and M. Paryan, "Breast cancer classification and prognostication through diverse systems along with recent emerging findings in this respect; the dawn of new perspectives in the clinical applications," *Tumor Biology*, vol. 37, pp. 14 479–14 499, 2016.

Y. Sun, W. Du, L. L. Yang, M. Dai, Z. Y. Dou, Y. X. Wang, J. Liu, and G. Zheng, "Computational methods for recognition of cancer protein markers in saliva," *Math. Biosci. Eng*, vol. 17, pp. 2453–2469, 2020.

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.