Predicting Influenza A Viral Host Using PSSM and Word Embeddings

Yanhua Xu
Department of Computer Science
University of Liverpool
Liverpool, UK
y.xu137@liverpool.ac.uk

Dominik Wojtczak
Department of Computer Science
University of Liverpool
Liverpool, UK
d.wojtczak@liverpool.ac.uk

***Abstract*—The rapid mutation of influenza virus threatens public health. Reassortment among viruses with different hosts can lead to a fatal pandemic. However, it is difficult to detect the original host of the virus during or after an outbreak as influenza viruses can circulate between different species. Therefore, early and rapid detection of the viral host would help reduce the further spread of the virus. We use various machine learning models with features derived from the position-specific scoring matrix (PSSM) and features learned from word embedding and word encoding to infer the origin host of viruses. The results show that the performance of the PSSM-based model reaches the MCC around 95%, and the F_1 around 96%. The MCC obtained using the model with word embedding is around 96%, and the F_1 is around 97%.**

***Index Terms*—Influenza, Machine Learning, Deep Learning, Position-specific Scoring Matrix, Word Embedding, Support Vector Machine, Ensemble Model, Convolutional Neural Network**

I. Introduction

INFLUENZA is an infectious disease that occurs globally and infects up to 20 percent of the world's population each year, although its prevalence is usually underestimated [1]. Influenza pandemics occur at a lower frequency than seasonal influenza epidemics, but each such crisis can cause millions of deaths. Flu epidemics seriously impact vulnerable people with chronic medical conditions, and flu pandemics put people of all ages at life-threatening risk.

Influenza viruses are divided into four types based on their internal ribonucleoproteins: A, B, C and D. Influenza D viruses are not known to cause human illness. Influenza C viruses only affect humans, but they are less likely to cause large-scale pandemics or seasonal epidemics. Thus, currently, seasonal flu vaccine strains do not inoculate against influenza C and D viruses. Influenza A and B viruses are the major causes of seasonal epidemics. Influenza B viruses also only affect humans, but influenza A viruses affect both humans and animals and can cause global epidemics (i.e., pandemics). Subtypes of influenza A viruses are differentiated by two kinds of glycoproteins under the viral envelope: Hemagglutinin (HA) and Neuraminidase (NA). Within these types, 18 HA subtypes (numbered 1–18) and 11 NA subtypes (numbered 1–11) have been discovered to date [2].

The characteristic of antigenic sites on HA or NA protein that is recognized by the immune system to inhibit flu infectious changes rapidly to escape the recognition of the immune system. This process is also known as antigenic drift, which results in new influenza A, B or C virus strains that are partially recognized by humans' immune systems and contribute to seasonal influenza outbreaks. The HA or NA in influenza A can experience drastic changes on antigenic sites and cause an antigenic shift. Antigenic shifts may result from a re-assortment of different viruses within single or multiple hosts and generate a novel virus [3]. Multiple pandemics have resulted from extreme antigenic shifts, after which most people lack immunity to the novel virus. The origins of four major influenza pandemics that emerged since 1900 may have been caused by a recombination of animal viruses (swine and avian) and human viruses: Spanish flu (1918–1919), Asian flu (1957–1958), Hong Kong flu (1968–1969) and the 2009 flu pandemic (2009–2010).

The Spanish flu was caused by the A/H1N1 virus. It is the deadliest pandemic in the recorded history and killed an estimated 17 – 100 million people [4], [5], [6]. The origin host of the Spanish flu remains a mystery [7], but recent studies suggest it may have sprung from birds or pigs to humans [8], [9], [10]. After the initial pandemic in 1918, the virus adapted to keep playing a major role in flu epidemics until 1957, when the major changes in HA and NA produced the novel virus A/H2N2 and resulted in the Asian flu pandemic. Asian flu has higher morbidity and mortality compared with the subsequent Hong Kong flu in 1968 as Hong Kong flu caused by the A/H3N2 virus which only involved the major changes in the HA antigen [11]. Both Asian flu and Hong Kong flu were caused by reassortment between human viruses and avian viruses. The A/H1N1 virus also caused the 2009 flu pandemic, but that iteration involved a complex triple reassortment between human, avian and swine viruses [12], [13].

Influenza viruses have multiple hosts, such as humans, birds, pigs and horses. Birds are a major natural reservoir of influenza A virus [14], [15], and the virus can infect both human and pigs [16]. Pigs are also considered as an intermediate host of influenza A viruses between humans and birds [17]. Once a virus mutates through reassortment between different hosts, it can produce a life-threatening risk to human populations, as it no longer needs an intermediate host to transmit between

people.

Therefore, the transmission of influenza viruses is not limited to human-to-human contact, but also includes animal-to-human (zoonosis) and animal-to-animal (enzootic) transmission [14]. Zoonotic infections can either be dead-end transmissions or lead to a pandemic in the human population after accumulating enough adaptive mutations to sustain transmissions between people, which then regularly circulates as a seasonal influenza virus [14], [18].

It is difficult to determine the origin of each virus during a virus outbreak because some viruses can cross species barriers. In this case, swine-origin viruses can be isolated from humans. The virus needs enough time to complete the adaptive mutation and accumulation process [14]. Therefore, earlier isolation of the original viral host may effectively control a viral outbreak or give an early warning of the risk of the virus.

Traditional methods are mostly laboratory-based, such as using hemagglutination inhibition (HI) assay to subtype virus or analyze receptor-binding. Laboratory-based methods are time-consuming and labor-intensive. In order to save manpower and time, various machine learning and deep learning algorithms have been used in viral host prediction, such as KNN [19], random forests [19], [20], ANN [21] and decision tree [22]. However, some previous research manually selected balanced data set [19], [23], and some only used relative small data set [21] or encode the sequence as a sparse matrix [21], [25]. Novel deep learning techniques, such as convolutional neural networks (CNN), have also been applied in this field, but thus far only to avian and human viruses [26].

Our study uses both classic machine learning techniques and deep learning techniques to infer the origin host of influenza viruses. Evolutionary features of viral sequences were extracted by the PSI-BLAST program and fed into four traditional machine learning models (sequence alignment-based method). Word encoding and word embedding were used for converting viral sequences to numerical vectors before feeding into a deep learning model (sequence alignment-free method).

The paper is structured as follows: In Section 2, we introduce the data and methods used in the paper. In Section 3, we illustrate the hyperparameters and architecture of models. This is followed by the experimental results in Section 4. Finally, we provide some discussions and conclusions in Section 5.

II. Materials and Methods

A. Data Collection

Complete influenza A virus protein sequences isolated from avian, swine and human samples were collected from the GISAID [27] database (status 2020-09-25). Only hemagglutinin (HA) protein sequences are used, as HA is the most dominant protein for immunity response and helping the virus bind to target hosts [28]. A protein sequence usually has 20 types of amino acids, which can be represented by a one-letter symbol: A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y and V. However, sometimes a protein sequence may also include other letters to represent uncertain or unknown amino acids, such as X, B and Z [29]. The sequences containing such letters were removed from our data set.

The original data set includes redundant data, and we removed all identical sequences and produced a non-redundant data set used in this research. We did not apply any further sequence reduction steps as the influenza virus can infect across different species and may results in a high similarity between different hosts' sequences, as shown in Fig. 1. Therefore, simply removing similar sequences would reduce sequence diversity and mislead the results, as shown in Fig. 2.

Fig. 1: Example of highly similar sequences with different host. The strain name of each sequence is shown as follows: EPI158898_Avian: *A/chicken/AR/30402/1999*; EPI5571_Human: *A/South Carolina/1/18*; EPI4897_Swine: *A/swine/Wisconsin/1/61*. The percent identity for these sequences is above 98%, but they were isolated in different year and from different hosts.

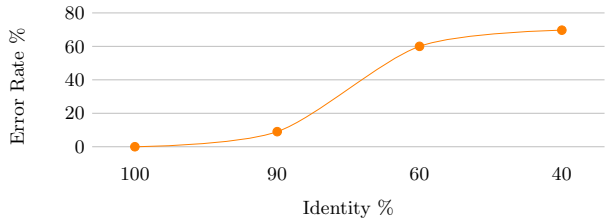


Fig. 2: Percentage of mistakenly deleted sequences based on different identity ratio. Error rate = $\frac{\text{Number of misclassified sequences}}{\text{Total number of sequences}}$

Only sequences belonging to a single host (human, avian and swine) were used in the study, which means each sequence is assigned only one label. Multi-label research is beyond the scope of this paper. Therefore, a total of 60,087 sequences have been used. Table I lists the number and proportion of instances in each class.

TABLE I: Data Set

Class	Instances (#)	Proportion (%)
Human	32,897	54.75
Swine	8,939	14.88
Avian	18,251	30.37
Total	60,087	100

Before filtering: avian (32,299), human: (133,269), swine (15,265) and total (180,833).

B. Features Derived from Position-Specific Scoring Matrix

1) Position-Specific Scoring Matrix: The position-specific scoring matrix (PSSM) [30] is a scoring matrix that contains a highly informative representation of protein sequences and is widely used to extract evolutionary features from sequences. A PSSM can be derived by PSI-BLAST (Position-Specific Iterated BLAST) program [31].

A PSSM is an $L \times 20$ matrix for a query protein sequence with L length, as a protein sequence usually has 20 types of

amino acids. The more intuitive way to represent the PSSM for a sequence $a_1 a_2 \dots a_L$ is as follows:

$$\mathrm{PSSM}_{original} = \begin{pmatrix} & A & R & \dots & V \\ a_1 & p_{1,1} & p_{1,2} & \dots & p_{1,20} \\ a_2 & p_{2,1} & p_{2,2} & \dots & p_{2,20} \\ \dots & \dots & \dots & \dots & \dots \\ a_L & p_{L,1} & p_{L,2} & \dots & p_{L,20} \end{pmatrix}, \quad (1)$$

Each $p_{i,j}$ is the score of amino acid a_i mutated to a_j, it can also represent the probability of mutation by using the sigmoid function to restrict the score into the range [0, 1]:

$$p_{i,j} = \frac{1}{(1 + e^{-p_{i,j}})}, i = 1, 2, \dots, L; j = 1, 2, \dots 20, \quad (2)$$

We apply a standalone version of PSI-BLAST [32] developed by the NCBI to run PSI-BLAST iteratively. The database used for searching is a non-redundant (NR) database, and the parameters of the PSI-BLAST program are set to their default values (E-value = 0.001, number of iterations = 3).

The original PSSMs are variable in size and thus cannot be fed directly into many machine learning models. Therefore, we propose PSSM-based sequence encoding schemes to overcome this hindrance. We first introduce a residue grouping rule to reduce the complexity of proteins and reduce unnecessary computations.

2) Residue Grouping Rule: 20 kinds of amino acids can be grouped into 10 types as they have similar functional or structural characteristics in proteins [33]: G1 (F, **Y**, W), G2 (M, **L**), G3 (I, **V**), G4 (A, T, **S**), G5 (**N**, H), G6 (Q, **E**, D), G7 (R, **K**), G8 (C), G9 (G) and G10 (P). By implementing residue grouping rules to each column of original PSSM, a grouped-PSSM (GPSSM) with $L \times 10$ dimension can be formed:

$$\mathrm{PSSM}_G = \begin{pmatrix} & G_1 & G_2 & \dots & G_{10} \\ a_1 & g_{1,1} & g_{1,2} & \dots & g_{1,10} \\ a_2 & g_{2,1} & g_{2,2} & \dots & g_{2,10} \\ \dots & \dots & \dots & \dots & \dots \\ a_L & g_{L,1} & g_{L,2} & \dots & g_{L,10} \end{pmatrix}, \quad (3)$$

where

$$g_{i,j} = \frac{\sum p_{i,G_j}}{|G_j|}, \quad (4)$$

The GPSSM is produced based on the original PSSM (1), thence $\sum p_{i,G_j}$ means the score of amino acid a_i mutated to an amino acid that belongs to group j. L is the length of sequences; $i = 1, 2, \dots, L$; $|G_j|$ is the number of amino acids types in group j. For instances, if $i = j = 1$, then $|G_1| = 3$, $g_{1,1} = {}^{(p_{1,F} + \ p_{1,Y} + \ p_{1,W})}/_{3}$

The following proposed feature sets (EG-PSSM, GDPC-PSSM and ER-PSSM) are derived from GPSSM (3).

3) EG-PSSM: Because the length of the input sequences can vary, so can original PSSMs (1) and GPSSMs (3). As a result, they cannot be directly used in many machine learning models. One intuitive and simple way to overcome this problem is to also apply the residue grouping rule to each row of GPSSM (3). Therefore, each sequence can produce a 10×10 matrix. Reformatting the matrix in rows from top to bottom and left to right, a 1×100 feature vector is extracted from a GPSSM (3).

$$\mathrm{PSSM}_{EG} = \begin{pmatrix} E_{G_1,G_1} & E_{G_1,G_2} & \cdots & E_{G_{10},G_{10}} \end{pmatrix}^T, \quad (5)$$

where

$$E_{G_i,G_j} = \frac{\sum g_{G_i,G_j}}{|G_i|}, i, j = 1 \dots 10, \quad (6)$$

4) GDPC-PSSM: The traditional dipeptide composition (DPC) [34] captures the composition information of amino acids and partial local-order information in protein sequences. Original DPC acts directly on raw sequence data and gives a 400-dimensional feature vector for each sequence, but it can be further extended to PSSM [35]. Therefore, each $L \times 10$ GPSSM (3) can be further defined as 100-dimensional feature vector by grouped dipeptide composition encoding:

$$\mathrm{PSSM}_{GDPC} = \begin{pmatrix} D_{1,1} & D_{1,2} & \cdots & D_{10,10} \end{pmatrix}^T, \quad (7)$$

where

$$D_{i,j} = \frac{1}{L-1} \sum_{k=1}^{L-1} g_{k,i} \times g_{k+1,j} \ i, j = 1, 2, \dots, 10, \quad (8)$$

Each $g_{k,i}$ is the value of row k and column i in the GPSSM (3).

5) ER-PSSM: The third proposed sequence representation is adapted from RPSSM [36], which computes the pseudo-composition of dipeptide in sequences. Same as GDPC-PSSM, RPSSM also extracts the partial local sequence order information in sequences. Original RPSSMs only compute the pseudo-composition of any two adjacent amino acids. We extend the computation of RPSSM for any two amino acids $a_k a_{k+t}$ with gap t in sequences and extract a 1×910 feature vector per sequence:

$$\mathrm{PSSM}_{ER} = \begin{pmatrix} M_{1,1,1} & M_{1,2,1} & \cdots & M_{10,10,9} & T_1 & \cdots & T_{10} \end{pmatrix}^T, \quad (9)$$

where

$$M_{i,j,t} = \frac{1}{L-t} \sum_{k=1}^{L-t} \frac{\left(g_{k,i} - g_{k+t,j}\right)^2}{2}, \quad i, j = 1, 2, \dots, 10; t = 1, 2, \dots, 9 \quad (10)$$

and

$$T_i = \frac{1}{L} \sum_{k=1}^{L} \left(g_{k,i} - \bar{G}_i\right)^2, \quad i, j = 1, 2, \dots, 10. \quad (11)$$

$\bar{G}_i$ is the average of values of GPSSM (3) in column i, T_i computes the average pseudo-composition of all the amino acids in the protein sequence corresponding to column i in GPSSM (3).

C. Features Learned from N-grams

Traditional machine learning methods require manual data preprocessing and feature extraction to extract representative features of each protein sequence. The feature extraction process requires prior knowledge to select suitable features. In contrast, deep learning methods can directly learn the implicit representation of protein sequences. This subsection introduces two vectorization schemes, word encoding and word embedding, to map protein sequences into numerical vectors.

1) Overlapping N-grams: A protein sequence is morphologically similar to a text sentence, except that the text is composed of words but the protein sequence is formed by amino acid letters. Therefore, we split the sequence into overlapping n-grams (n is ranging from 3 to 5) to transform a protein sequence into a protein "sentence" of n-grams. An n-gram is a protein "word" with successive n amino acids. Fig. 3 is an example of overlapping 3-grams for a protein sequence, and Fig. 4 shows the word clouds of n-grams for all sequences.

M L S I T I L F L ...
overlapping 3-grams: MLS LSI SIT ITI TIL ILF LFL ...

Fig. 3: Example of Overlapping Trigrams.

(a) 3-grams (b) 4-grams (c) 5-grams

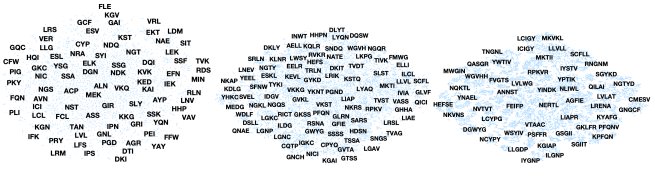
Fig. 4: Word Clouds of N-grams

2) Word Encoding: Word encoding, also called indexed-based encoding, maps words to numbers, as shown in Fig. 5. Compared with commonly used one-hot encoding, word encoding can generate non-sparse vectors and is more efficient. However, word encoding is not trainable and unexplainable for models that need to learn feature weights as it loses the relationship between words.

Index :	1 2 3 4 5 6 7
Vocabulary :	MLS LSI SIT ITI TIL ILF LFL
Words :	LSI ITI ILF INL
Word Encoding :	[2 4 6]
Word Encoding : (left padding)	[0 0 0 2 4 6]

Fig. 5: Example of Word Encoding.

3) Word Embedding: Word embedding compensates for the drawbacks of word encoding and one-hot encoding. It cannot only produce dense vectors but also capture the relationship between similar words. Popular implementations of word embedding include Word2Vec [24], but it lacks domain-specific words. Therefore, we generated a custom word embedding from the dataset only used for training and mapped the n-grams of each sequence to the embedding vectors. Each n-gram is represented as a vector of size N, and a protein sequence is represented as a $L \times N$, where L is the length of the sequence (number of n-grams in the sequence) and N is the embedding dimension. Fig. 6 depicts the visualization of word embedding using 2-d t-SNE.



(a) 3-grams (b) 4-grams (c) 5-grams

Fig. 6: Word Embeddings Visualization Using 2-D t-SNE

We left-pad and truncate the sequence with the most frequent sequence length to unify the dimensionality of matrices, which means that most sequence information will be retained, but more noise will be introduced to the shortest sequence, and the longest sequence information will be discarded.

D. Machine Learning Techniques

1) RUSBoosted Tree: Data sampling with boosting algorithms are applied to tackling the problem of class imbalance. Commonly used data samples methods include oversampling (enriching the minority class) and undersampling (decreasing the majority class). Random undersampling boosting (RUSBoost) algorithm [37], as its name implies, combines undersampling methods with boosting algorithms. Compared with other oversampling methods, e.g., SMOTEBoost, RUSBoost is computationally cheaper and more efficient.

2) Extreme Gradient Boosting: Extreme Gradient Boosting (XGBoost) [38] is a highly efficient and scalable implementation of gradient boosting algorithms. Gradient boosting algorithms are similar to AdaBoost but use gradient descent to optimize the derivable loss function when adding the new models. XGBoost can handle stubborn issues in the data science area, such as it can solve missing values and sparse data in an automatic way. One of the biggest advantages of XGBoost is that it provides parallel training to speed up the training process and can handle large datasets.

3) Random Forest: Decision trees have higher variance and lower bias, while the bagging algorithm aims to reduce the variance of the model. Therefore, the combination of bagging and decision tree (random forest) improves the overall performance of the model. Random forest [39] considers only a small part of all features in each split. It introduces more randomness to each decision tree. In contrast to boosting-based ensembles, bag-based ensembles are prone to construct deep trees, which means bag-based ensembles are more complex than boosting-based ensembles. Therefore, bag-based ensembles may require more training time than boosting-based ensembles but leave out the validation process to estimate generalization performance.

4) Support Vector Machine: Support vector machine (SVM) is one of the best "off-the-shelf" supervised learning algorithms [40]. SVM can not only classify linearly separable data but also can classify non-linearly separable data by introducing a kernel trick. Kernel tricks help SVMs handle high-dimensional data (even infinite-dimensional data) well by mapping lower-dimensional data into higher dimensional data but without explicitly transforming them. We use the Gaussian kernel as kernel function and the one-vs-all strategy to construct the multi-class SVM.

5) Convolutional Neural Network: Convolutional neural networks (CNN or ConvNets) often appear in areas related to computer vision, such as facial recognition, object recognition, and autonomous vehicles. CNN initially took images as input and expanded to include non-image data such as time series, text, and audio data. Contrary to traditional machine learning, CNN learns features of the data in each hidden layers. A CNN often includes three hidden layers: the convolutional layer for learning certain features, the activation layer for activating features, and the pooling layer for reducing the number of network parameters.

E. Model Parameters and Implementation

Four classic machine learning models (SVM, RF, RUSBoost and XGBoost) were evaluated using optimized parameters (see Table II for the tunning range). We used Bayesian optimization to automatically adjust hyperparameters in 30 iterations and at most 40,000 seconds. The deep learning model (CNN) was evaluated using fixed parameters (see Table III for details).

RF, RUSBoost, SVM and CNN were implemented in MATLAB 2020a, and XGBoost was implemented by XGBoost Python Package.

TABLE II: Hyperparameter Setting for Classic ML Models

ML Classifiers	Hyperparameters	Tuning Range
SVM	BoxConstraint	[0.001, 1000]
	KernelScale	[0.001, 1000]
RUSBoost RF	NumLearningCycles	[10, 500]
	LearnRate	[0.01, 1]
	MinLeafSize	[1, No. of data points]
	MaxNumSplits	[1, $\frac{1}{2}$(No. of data points)]
	NumVariablesToSample	[1, No. of features]
XGBoost	learning_rate	[0.01, 1]
	max_depth	[1, 15]
	colsample_bytree	[0.1, 1]

F. CNN Architecture

We designed a simple CNN for classifying protein n-grams, as shown in Fig. 7. The CNN used in this work contains one input layer (*input*), one convolution layer (*conv*), one batch normalization layer (*bn*), one ReLU activation layer (*relu*), one dropout layer (*dropout*), one max pooling layer (*max-pool*), four fully connected layers (*fc*) and one softmax layer (*softmax*). Each protein feature matrix has (L, N) dimension, where L is the length of the protein sentences after padding or cropping, and N is the embedding dimension of that sequence. Typically, CNN takes the image input with size $(height, width, color\ channels)$. When it is applied to text classification, the height of the input is set as 1. Thence, the output size of the the convolution layer (*conv*) is $(1, L, 256)$. The convolution layer (*conv*) has 256 filters of size $[1\ ngram]$, where $ngram$ is the size of protein words. Two parameters of the convolution layer (*conv*) and max pooling layer (*max-pool*), which are not mentioned in Table III, are step size (stride) and padding value that were set as $[1\ \ 1]$ and 0, respectively. The dropout rate in the dropout layer (*dropout*) is 0.2.

TABLE III: Hyperparameter Setting for CNN

Hyperparameters	Value
MiniBatchSize	128
MaxEpochs	30
solver	adam
InitialLearnRate	0.001
LearnRateSchedule	piecewise
LearnRateDropPeriod	10
LearnRateDropFactor	0.01
ValidationPatience	15

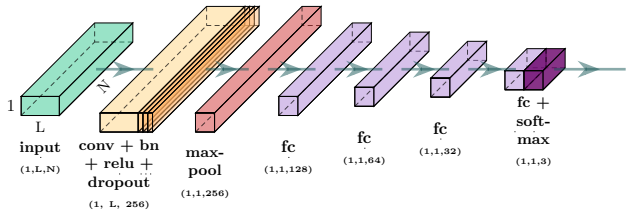


Fig. 7: Example of the CNN Architecture

III. Model Evaluation

A. Cross-Validation

We used stratified K-fold cross-validation (CV) to evaluate models. The class ratio of the training set was almost the same as that of the test set. The generalization performance of models was only evaluated on the test set (unseen data to the model). Nested CV adds the outer K-fold CV for final evaluation to reduce bias when it comes to hyperparameters optimization and model selection [41]. Therefore, nested CV will take advantage of the full diversity of the data set and ensure that all data will be tested. The pseudo-code of stratified nested CV is shown in Fig. 8.

In this study, we chose $k_{outer} = 6$ and $k_{inner} = 5$. Therefore, approximately 68%, 16%, and 16% of the data were used for training, validation, and testing, respectively.

The detailed information on the training set and test set is shown in Table IV.

Algorithm 1: Nested Cross-Validation with Bayesian Optimization

Data: Data set with features X and labels y, $\mathcal{D} = \{X, y\}$
input : Number of inner folds K_{inner}, number of outer folds K_{outer}
Maximum number of steps of Bayesian optimization n_{iter}
Maximum tuning time $MaxTime$
output: Generalization error E_g of the model

Shuffle $\mathcal{D}$;
Stratified split $\mathcal{D}$ into K_{outer} folds;
for $i = 1$ **to** K_{outer} **do**
  Take i^{th} fold of $\mathcal{D}$ as test set $\mathcal{D}^i_{test}$ and remaining as training set $\mathcal{D}^i_{train}$;
  Stratified split $\mathcal{D}^i_{train}$ into K_{inner} folds;
  for $k = 1$ **to** n_{iter} **do**
    for $j = 1$ **to** K_{inner} **do**
      Take j^{th} fold of $\mathcal{D}^i_{train}$ as validation set $\mathcal{D}^j_{val}$ and remaining as training set $\mathcal{D}^j_{train'}$;
      Train the model on $\mathcal{D}^j_{train'}$ with hyperparameter set $\mathcal{P}_k$;
      Compute validation error $E^{k,j}_{val'}$ of the model on $\mathcal{D}^j_{val}$;
    Compute average validation error E^k_{val}, where $E^k_{val} = average(E^k_{val'})$;
    if *elapsed time* $> MaxTime$ **then**
      Stop tuning
  Select optimal hyperparameter set $\mathcal{P}_{opt}$ with the lowest E_{val};
  Fit the model on $\mathcal{D}^i_{test}$ with $\mathcal{P}_{opt}$;
  Compute the test error E^i_{test} of the model on $\mathcal{D}^i_{test}$;
Compute generalization error E_g of the model, $E_g = average(E_{test})$;

Fig. 8: Pseudo Code of Stratified Nested CV

TABLE IV: Training Set and Test Set

Class	Training[a]		Testing[b]	
	Instances (#)	Proportion (%)	Instances (#)	Proportion (%)
Avian	15,209	30.37	3,042	30.37
Human	27,144	54.75	5,483	54.75
Swine	7,449	14.88	1,490	14.88
Total	50,072	100	10,015	100

[a] *The training set contains 84% of all data, 80% of the training set used for training and 20% of the training set used for validation (5-fold cross-validation).*
[b] *The test set contains 16% of all data and is only used in final evaluation.*

B. Evaluation Metrics

Evaluation measurements used in the study include F_1-score and Matthews's correlation coefficient (*MCC*). The equations of measurements for each class are defined as follows:

$$F_{1_i} = 2 \cdot \frac{\text{Precision}_\text{i} \cdot \text{Sensitivity}_\text{i}}{\text{Precision}_\text{i} + \text{Sensitivity}_\text{i}}, \tag{12}$$

$$MCC_i = \frac{TP_i \times TN_i - FP_i \times FN_i}{\sqrt{(TP_i+FP_i)(TP_i+FN_i)(TN_i+FP_i)(TN_i+FN_i)}}, \tag{13}$$

where $i = 1, 2, \ldots, N$, N is the number of classes; $\text{Sensitivity}_i = {TP_i}/{(TP_i+FN_i)}$ and $\text{Precision}_i = {TP_i}/{(TP_i+FP_i)}$. TP (True Positive) and TN (True Negative) represent the number of data correctly predicted, FP is the number of negative data misclassified as positive, and FN counts the number of positive data incorrectly predicted as negative.

For multi-class classification, the one-vs-all strategy is applied to produce F_1-score for each class. The overall F_1-score and overall *MCC* are defined as follows:

$$\text{Overall } F_1 = \frac{\sum_{i=1}^{N} F_{1_i}}{|C|}, \tag{14}$$

$$\text{Overall } MCC = \frac{c \cdot s - \sum_i^N p_i \cdot t_i}{\sqrt{s^2 - \sum_i^N p_i^2} \cdot \sqrt{s^2 - \sum_i^N t_i^2}} \tag{15}$$

where t_i is the number of times that class i truly occurred, p_i is the number of times class i was predicted, c is the total number of correctly predicted data, s is the total number of data items.

IV. Experimental Results

A. Sequence Alignment-based Methods

Fig. 9 shows the overall performance of the optimized PSSM-based models on the test set. After hyperparameter optimization, the performance of all models is outstanding, but RUSBoost with ER-PSSM has highest variation.

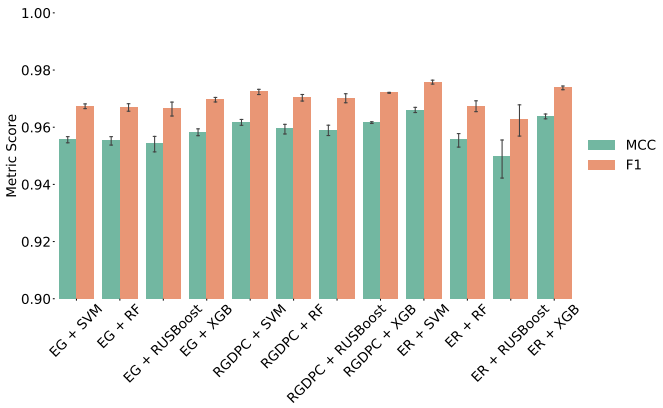


Fig. 9: Overall Performance of PSSM-based Models

The classes in the data set are unbalanced: more than half of the data belongs to human viruses, whereas only around 15% belongs to swine viruses. The class imbalance problem means that the classifier may ignore the minority class, resulting in performance degradation for the minority class. All models yield the worst performance in swine viruses compared with human and avian (see details in Fig. 10). Among all PSSM-based models, the SVM with ER-PSSM performs best in individual classes.

B. Sequence Alignment-free Methods

Regardless of the length of the n-gram, the overall performance of word embedding is better than word encoding in CNN (Fig. 11). Human is the easiest class among all CNN-based models to classify, and swine is the most difficult one (Fig. 12).

Fig. 13 depicts the average performance of all PSSM-based optimized machine learning models (Fig. 13c) and CNN-based models (Fig. 13a, Fig. 13b) in each class based on the confusion matrix on the test set. The human class has above

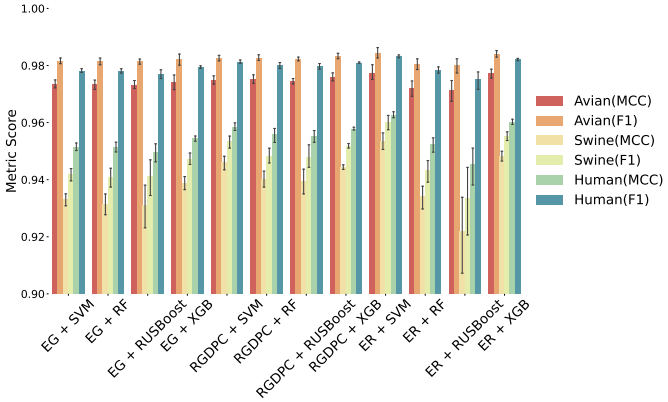


Fig. 10: Performance of PSSM-based Models in Individual Classes

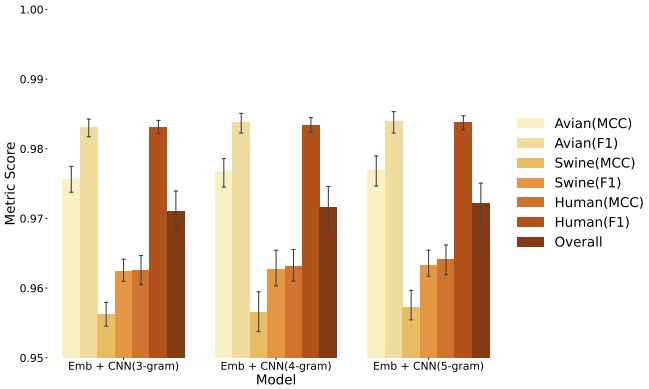


Fig. 11: Performance of Embedding-CNN in Individual Classes

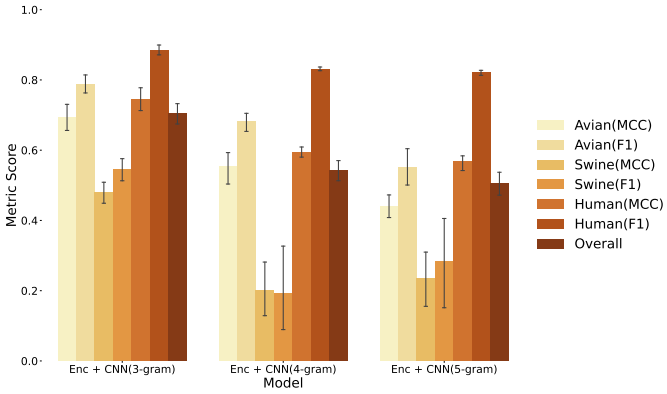


Fig. 12: Performance of Encoding-CNN in Individual Classes

(a) Encoding-CNN (b) Embedding-CNN (c) PSSM-ML

Fig. 13: Confusion Matrix of Different Models: the value on ith row and jth column indicates the probability of predicting the class i as the class j.

92% probability to be correctly classified among all models, as opposed to the swine. The performance between PSSM-ML and Embedding-CNN in human and swine classes is not much different, and Embedding-CNN is better than PSSM-ML in distinguishing swine viruses.

C. Integrated Results from Various models

We looked at the performance of the models and integrated the predicting results from Embedding-CNN and PSSM-ML. For each viral sequence, we only know the infected host of the sequence instead of the original host. Some sequences were collected during the outbreak, which means that the host used to isolate the virus may not be the original host. Previous research [20] only used one model to predict the viral host, which raises a concern about the accuracy of results. Different models may yield different predictions. Therefore, we provided integrated results and investigated the agreement between the model results, see details in Table V.

The models can reach a 100% agreement of approximate 35.50% of the sequences, while 100% disagreement between models occurs in 348 sequences. Among these 348 sequences, 217 belonged to the H1N1 influenza virus, and 196 sequences were similar to or related to the 2009 H1N1 influenza pandemic (pH1N1), including 55 sequences with mixed positive and negative segments. More specifically, 6 out of 13 avian sequences and 190 out of 254 swine sequences, all probably related to the pH1N1. These sequences show the potential ability to break the barrier of species and could be isolated from different species.

From the machine learning perspective, the predicted label mismatch the true label means the model misclassify the data. But this view is open to discussion for Influenza viral host prediction. We listed six sequences that can be traced in literature, as shown in Table VI. These sequences seem to be misclassified by all models, but the literature findings gave us the opposite result.

Previous research speculate that *A/turkey/BC/1529-3/2005* may be a swine-origin virus [42], [43]. *A/Beijing/1/2017* and *A/India/TCM2581/2019* are avian viruses isolated from human [44], [45]. *A/swine/Jangsu/48/2010* is a pH1N1-like swine virus which used for proving the retro-infection from swine to human in China [46]. *A/swine/Jiangsu/1/2008* was isolated from swine but prove to be mostly close relationship to avian [47]. *A/swine/Jiangsu/2/2009* was also isolated from infected swine but also mostly closely related to avian and human [47].

D. Performance at a Lower Taxonomic Level

Above results only represented the performance of proposed models at a higher taxonomic level. This subsection illustrates the performance of models at a lower taxonomic level. Among all PSSM-based models, the performance of ER-PSSM-XGB is outstanding and stable. As for CNN-based models, Embedding-CNN with 5-gram yields the best results. Hence, we also evaluated the performance of ER-PSSM-XGB and Embedding-CNN (5-gram) when it comes to a

TABLE V: Integrated Results

	Disagreement Rate				
	1	[0.5, 1)	(0.1,0.5)	(0, 0.1]	0
Avian	13	112	527	12477	5122
Human	81	468	1241	15048	16059
Swine	254	451	1606	6476	151
Total	348	1031	3374	34001	21332

* *disagreement rate of sequence i equals to E_i/N, where E_i is the number of models that misclassify sequence i, and N is the total number of models. If the disagreement rate of sequence i is in $[0.5, 1)$, it means that sequence i cannot be correctly classified by more than half of the models.*

TABLE VI: Example of Strains Reach Model Disagreement

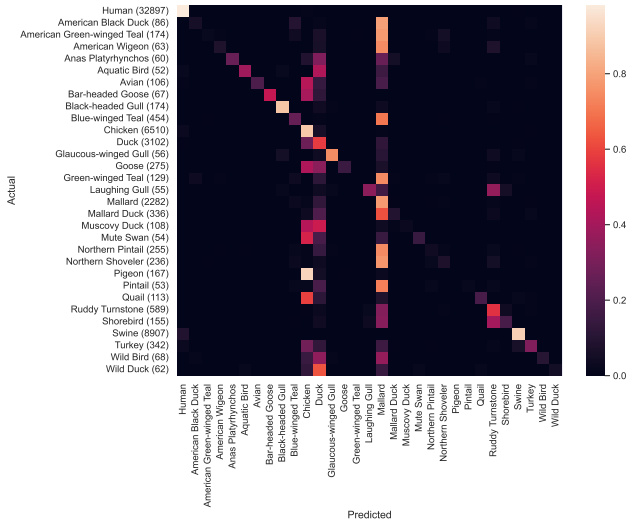
#	Strain Names	Infected Host	Predicted Host
1	A/turkey/BC/1529-3/2005	Turkey	Human (7.4%); Swine (92.6%)
2	A/Beijing/1/2017	Human	Avian (100%)
3	A/India/TCM2581/2019	Human	Avian (100%)
4	A/swine/Jangsu/48/2010	Swine	Human (100%)
5	A/swine/Jiangsu/2/2009	Swine	Avian (66.7%); Human (33.3%)
6	A/swine/Jiangsu/1/2008	Swine	Avian (96.3%); Human (3.7%)

lower taxonomic level. Fig. 14 shows the confusion matrix of Embedding-CNN (5-gram) and ER-PSSM-XGB at a lower taxonomic level.

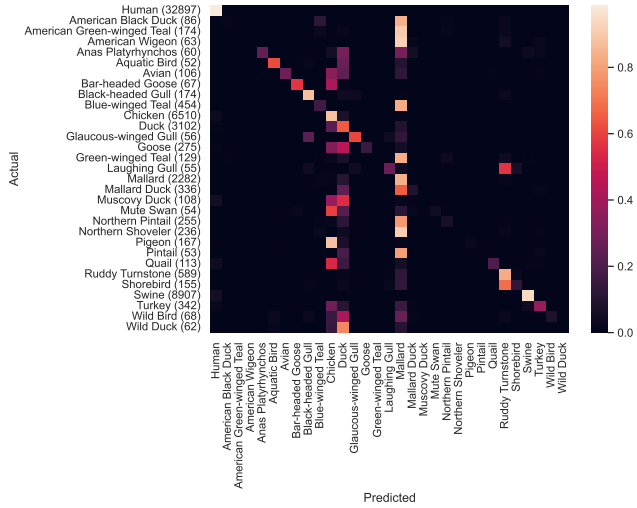
There is not much difference in performance between Embedding-CNN (5-gram) and ER-PSSM-XGB. However, some host can be detected by ER-PSSM-XGB but not Embedding-CNN (5-gram), such as mute swan. The overall performance of Embedding-CNN (5-gram) and ER-PSSM-XGB are both degraded compared with their performance at a higher taxonomic level. Embedding-CNN (5-gram) produces a 89.03% F_1 and a 82.75% MCC, whilst ER-PSSM-XGB has a 87.63% F_1 and a 80.52% MCC. In addition to this, the number of sequences correctly classified by ER-PSSM-XGB (87.63%) is higher than Embedding-CNN (5-gram) (62.18%).

V. Discussion and Conclusion

We used both sequences alignment-based and sequence alignment-free models to predict viral hosts of influenza A viruses. Sequence alignment-based models use the features extracted by PSSM before feeding into classic machine learning models. PSSM-based features cannot only maintain the evolutionary information in the virus sequence but also capture the local information in the sequence. Sequence alignment-free models transfer the virus sequences into numerical vectors through word encoding or word embedding before feeding into CNN and is incapable of capturing the local information in sequences. They also need padding to make the dimension of the vectors uniform. But even so, the overall performance of Embedding-CNN is still slightly better than PSSM-based models at a higher taxonomic level. When it comes to the performance at a lower taxonomic level, the PSSM-based model can classify some viruses that Embedding-CNN cannot do.



(a) Embedding-CNN (5-gram)



(b) ER-PSSM-XGB

Fig. 14: Confusion Matrix of Embedding-CNN (5gram) and ER-PSSM-XGB at a Lower Taxonomic Level: the number of sequences for each class is indicated in the brackets of actual label.

The data set used in the paper is non-redundant and all the multi-label sequences were removed, which means each sequence only belongs to a single host. Multi-label prediction is beyond the scope of this article (only about 3% of the data is multi-label) and is left for future work.

Acknowledgment

This work is supported by the University of Liverpool.

References

[1] N. Cox and K. Subbarao, "Influenza", *The Lancet*, vol. 354, no. 9186, pp. 1277-1282, 1999. I

[2] Lazniewski M, Dawson WK, Szczepinska T, Plewczynski D. "The structural variability of the influenza A hemagglutinin receptor-binding site". *Brief Funct Genomics*, vol 17, no. 6, pp.415-427, 2018. I

[3] C. Brockwell-Staats, R. G. Webster, and R. J. Webby, "Diversity of influenza viruses in swine and the emergence of a novel human pandemic influenza A (H1N1)," *Influenza and Other Respiratory Viruses*, vol. 3, no. 5, pp. 207–213, 2009. I

[4] P. Spreeuwenberg, M. Kroneman, and J. Paget, "Reassessing the Global Mortality Burden of the 1918 Influenza Pandemic," *American Journal of Epidemiology*, vol. 187, no. 12, pp. 2561–2567, 2018. I

[5] D. Morens and A. Fauci, "The 1918 Influenza Pandemic: In-sights for the 21st Century", *The Journal of Infectious Diseases*, vol. 195, no. 7, pp. 1018-1028, 2007. I

[6] N. P. A. S. Johnson and J. Mueller, "Updating the Accounts: Global Mortality of the 1918-1920 'Spanish' Influenza Pandemic," *Bulletin of the History of Medicine*, vol. 76, no. 1, pp. 105–115, 2002. I

[7] J. Antonovics, M. E. Hood, and C. H. Baker, "Was the 1918 flu avian in origin?," *Nature*, vol. 440, no. 7088, 2006. I

[8] J. K. Taubenberger, A. H. Reid, R. M. Lourens, R. Wang, G. Jin, and T. G. Fanning, "Characterization of the 1918 influenza virus polymerase genes," *Nature*, vol. 437, no. 7060, pp. 889–893, 2005. I

[9] M. Worobey, G.-Z. Han, and A. Rambaut, "A synchronized global sweep of the internal genes of modern avian influenza virus," *Nature*, vol. 508, no. 7495, pp. 254–257, 2014. I

[10] G. J. D. Smith, J. Bahl, D. Vijaykrishna, J. Zhang, L. L. M. Poon, H. Chen, R. G. Webster, J. S. M. Peiris, and Y. Guan, "Dating the emergence of pandemic influenza viruses," *Proceedings of the National Academy of Sciences*, vol. 106, no. 28, pp. 11709–11712, 2009. I

[11] E. Kilbourne, "Influenza Pandemics of the 20th Century", *Emerging Infectious Diseases*, vol. 12, no. 1, pp. 9-14, 2006. I

[12] R. J. Garten et al., "Antigenic and genetic characteristics of swine-origin 2009 A(H1N1) influenza viruses circulating in humans," *Science*, vol. 325, no. 5937, pp. 197–201, 2009. I

[13] G. J. D. Smith et al., "Origins and evolutionary genomics of the 2009 swine-origin H1N1 influenza A epidemic," *Nature*, vol. 459, no. 7250, pp. 1122–1125, 2009. I

[14] J. S. Long, B. Mistry, S. M. Haslam, and W. S. Barclay, "Host and viral determinants of influenza A virus species specificity," *Nat. Rev. Microbiol.*, vol. 17, no. 2, pp. 67–81, 2019. I

[15] O. T. Gorman, W. J. Bean, Y. Kawaoka, and R. G. Webster, "Evolution of the nucleoprotein gene of influenza A virus," *J. Virol.*, vol. 64, no. 4, pp. 1487–1497, 1990. I

[16] R. G. Webster, W. J. Bean, O. T. Gorman, T. M. Chambers, and Y. Kawaoka, "Evolution and ecology of influenza A viruses," *Microbiol. Rev.*, vol. 56, no. 1, pp. 152–179, 1992. I

[17] I. H. Brown, "The pig as an intermediate host for influenza A viruses between birds and humans," *Int. Congr. Ser.*, vol. 1219, pp. 173–178, 2001. I

[18] J. K. Taubenberger and J. C. Kash, "Influenza virus evolution, host adaptation, and pandemic formation," *Cell Host Microbe*, vol. 7, no. 6, pp. 440–451, 2010. I

[19] Sherif, FAYROZ F., N. O. U. R. H. A. N. Zayed, and M. A. H. M. O. U. D. Fakhr. "Classification of host origin in influenza A virus by transferring protein sequences into numerical feature vectors." *Int J Biol Biomed Eng 11*, 2017. I

[20] C. L. P. Eng, J. C. Tong, and T. W. Tan, "Predicting host tropism of influenza A virus proteins using random forest," *BMC Med. Genomics*, vol. 7 Suppl 3, no. Suppl 3, p. S1, 2014. I, IV-C

[21] Attaluri, Pavan K., Zhengxin Chen, and Guoqing Lu. "Applying neural networks to classify influenza virus antigenic types and hosts." In *2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pp. 1-6. IEEE, 2010. I

[22] F. Kargarfard, A. Sami, M. Mohammadi-Dehcheshmeh, and E. Ebrahimie, "Novel approach for identification of influenza virus host range and zoonotic transmissible sequences by determination of host-related associative positions in viral genome segments," *BMC Genomics*, vol. 17, no. 1, p. 925, 2016. I

[23] Attaluri, P. K., Zhengxin Chen and Guoqing Lu. "Integrated use of three machine learning techniques for influenza virus classification." 2010. I

[24] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv [cs.CL]*, 2013. II-C3

[25] F. Mock, A. Viehweger, E. Barth, and M. Marz, "VIDHOP, viral host prediction with Deep Learning," *Bioinformatics*, 2020. I

[26] D. Scarafoni, B. A. Telfer, D. O. Ricke, J. R. Thornton, and J. Comolli, "Predicting influenza A tropism with end-to-end learning of deep networks," *Health Secur.*, vol. 17, no. 6, pp. 468–476, 2019. I

[27] "GISAID - Initiative," *Gisaid.org*. [Online]. Available: https://www.gisaid.org. [Accessed: 13-Dec-2020]. II-A

[28] D. J. D. Earn, J. Dushoff, and S. A. Levin, "Ecology and evolution of the flu," *Trends Ecol. Evol.*, vol. 17, no. 7, pp. 334–340, 2002. II-A

[29] "A one-letter notation for amino acid sequences (definitive rules)," *Pure Appl. Chem.*, vol. 31, no. 4, pp. 639–646, 1972. II-A

[30] S. F. Altschul et al., "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," *Nucleic Acids Res.*, vol. 25, no. 17, pp. 3389–3402, 1997. II-B1

[31] S. F. Altschul and E. V. Koonin, "Iterated profile searches with PSI-BLAST–a tool for discovery in protein databases," *Trends Biochem. Sci.*, vol. 23, no. 11, pp. 444–447, 1998. II-B1

[32] "Index of /blast/executables," *Nih.gov*. [Online]. Available: http://ftp.ncbi.nih.gov/blast/executables. [Accessed: 14-Dec-2020]. II-B1

[33] T. Li, K. Fan, J. Wang, and W. Wang, "Reduction of

protein sequence complexity by residue grouping," *Protein Eng.*, vol. 16, no. 5, pp. 323–330, 2003. II-B2

[34] S. Vijayakumar and G. Namasivayam, "Harnessing computational biology for exact linear B-cell Epitope prediction: A novel amino acid composition-based feature descriptor," *OMICS*, vol. 19, no. 10, pp. 648–658, 2015. II-B4

[35] T. Liu, X. Zheng, and J. Wang, "Prediction of protein structural class for low-similarity sequences using support vector machine and PSI-BLAST profile," *Biochimie*, vol. 92, no. 10, pp. 1330–1334, 2010. II-B4

[36] S. Ding, Y. Li, Z. Shi, and S. Yan, "A protein structural classes prediction method based on predicted secondary structure and PSI-BLAST profile," *Biochimie*, vol. 97, pp. 60–65, 2014. II-B5

[37] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: Improving classification performance when training data is skewed," in *2008 19th International Conference on Pattern Recognition*, 2008, pp. 1–4. II-D1

[38] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 2016. II-D2

[39] T. K. Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 1995, pp. 278–282. II-D3

[40] R. Gove and J. Faytong, "Machine learning and event-based software testing: Classifiers for identifying infeasible GUI event sequences," in *Advances in Computers*, Elsevier, 2012, pp. 109–135. II-D4

[41] Gavin C. Cawley and Nicola L. C. Talbot, "On Overfitting in Model Selection and Subsequent Selection Bias in Performance Evaluation," *Journal of Machine Learning Research*, vol. 11, no. 70, pp. 2079–2107, 2010. III-A

[42] C. Nfon, Y. Berhane, S. Zhang, K. Handel, O. Labrecque, and J. Pasick, "Molecular and antigenic characterization of triple-reassortant H3N2 swine influenza viruses isolated from pigs, turkey and quail in Canada: Triple-reassortant swine H3N2 in Canada 2005-2009," *Transbound. Emerg. Dis.*, vol. 58, no. 5, pp. 394–401, 2011. IV-C

[43] Y. Berhane et al., "Molecular and antigenic characterization of reassortant H3N2 viruses from turkeys with a unique constellation of pandemic H1N1 internal genes," *PLoS One*, vol. 7, no. 3, p. e32858, 2012. IV-C

[44] Y. Pan et al., "Human infection with H9N2 avian influenza in northern China," *Clin. Microbiol. Infect.*, vol. 24, no. 3, pp. 321–323, 2018. IV-C

[45] V. Potdar, D. Hinge, A. Satav, E. A. F. Simões, P. D. Yadav, and M. S. Chadha, "Laboratory-confirmed avian influenza A(H9N2) virus infection, India, 2019," *Emerg. Infect. Dis.*, vol. 25, no. 12, pp. 2328–2330, 2019. IV-C

[46] G. Zhao et al., "Isolation and phylogenetic analysis of pandemic H1N1/09 influenza virus from swine in Jiangsu province of China," *Res. Vet. Sci.*, vol. 93, no. 1, pp. 125–132, 2012. IV-C

[47] L. He et al., "Isolation and characterization of two H5N1 influenza viruses from swine in Jiangsu Province of China," *Arch. Virol.*, vol. 158, no. 12, pp. 2531–2541, 2013. IV-C, IV-C