A Fuzzy Approach to Classification on SARS-CoV-1 using BiDirectional Neural Network

WeiXin Fam

Research School of Computer Science,
Australian National University,
Canberra, ACT 0200,
Australia
u6554719@anu.edu.au

Abstract. Severe acute respiratory syndrome (SARS) is an infectious respiratory disease with high case fatality. However, SARS symptoms are closely similar to other viral infections like fever. Without the help of medical staff, it is hard for normal citizens to differentiate between catching a cold or SARS. Therefore, we should leverage machine learning algorithms to predict the disease class for early disease detection. Existing models like Bidirectional Neural Network (BDNN) are not suited for non-invertible data sets. Therefore, we experiment and propose two new technique in making the disease data invertible. The two techniques are data aggregation and fuzzy signature. The experimental result of the paper shows the new techniques with BDNN achieve 100% accuracy on SARS-CoV-1.

Keywords: SARS-Cov-1, Neural Network, BiDirectional Neural Network, Supervised Machine Learning, Data Aggregation, Fuzzy Signature

1 Introduction

In the current era, humanity faces all kinds of new diseases. In November 2002, the 21st century's first new disease, SARS, emerged. It spread globally rapidly, and the overall case fatality rate is 11%.[1] Throughout the pandemic, doctors relied on recognition of features obtained through basic clinical inspection, laboratory and radiological.[2] However, not everyone could go through medical checkups, given the number of people in each country and globally. Regular citizens could only rely on the publicized general SARS symptoms to determine if they were infected. As SARS initial symptoms are similar to fever, the citizens could confuse SARS as normal diseases, and miss the best treatment period.

With the emerging technology in health care, developers develop more algorithms to help earlier detection of such deadly diseases based on the collected clinical data. The supervised Machine Learning area is well-matured and is a good starting point for integrating technology with the health care system. Research has shown that when using one supervised machine learning algorithm for disease prediction, Neural Networks is the top choice.[3] Supervised machine learning refers to labelled data being feed into a model. Artificial Neural Networks (ANN), especially, can carry out predictions of the disease by taking the patients' symptoms data and disease label. The con is that the algorithm only learns to produce the disease label.

Medical experts can deduce the possible symptoms for a given disease and vice versa. However, simple ANN only outputs disease class from patients' data. We want a model that can imitate the medical experts' brain, so the model could diagnose patients better and help improve clinical decision. That model is a Bidirectional Neural Network (BDNN).[4] We would explore the application of this model to predict the disease labels with a real-life dataset, SARS-CoV-1. The BDNN is suitable for data on a one-to-one, invertible relationship [4]. SARS-CoV-1, however, like many other classification medical datasets, is not invertible. Thus, to use models like BDNN, we need to apply a transformation to the data to make it invertible.

In the paper [4], the two techniques described to make non-invertible data are an "Extra Node" column and computing the benchmark feature value for each class. The first technique involves giving a unique value to each dataset row under a new "Extra Node" column. This ensures each row is unique, but the value provides no context or relation to the input data. The second technique calculates the means of each feature for each classification class and uses them with simple equations to help determine the classification of the predicted number. The researchers create those equations and thus decreases the model's explainability, as it involves researcher bias.

The new technique combines the pros of both previous techniques. We would calculate the benchmark value of each class and find the difference between each data row and the corresponding class's benchmark values. This difference would be the substitute of the random unique value, mentioned in the first technique. The benchmark value of each class is calculated via data aggregation techniques. Data aggregation helps to summarize the data, giving a representation value to the overall data or class. One common benefit of data aggregation in health care is to improve clinical decision making.[5]

The other technique is fuzzy signature. SARS-CoV-1 is a fuzzy dataset, where each variable only has values between 0 and 1. A paper has shown the application of the fuzzy signature technique on this dataset [6]. We propose the idea of pre-training some of the data to get the fuzzy signature with optimized parameters and then utilize the optimized fuzzy signature on actual data to get the expected membership of each data row to their respective class. A fuzzy membership could help medical experts better identify which class the patient belongs to. However, we know that there could be some similar data having the same fuzzy membership, hence we will also be adding mean. Thus, the value not only represents the membership but also the overall distribution of data.

Therefore, our objective is to explore the different techniques in making non-invertible data invertible and examine the performance of these techniques with BDNN on classifying the 4 patient classes in SARS-CoV-1. The contribution of this paper is twofold. First, the findings allow us to determine the suitability of applying BDNN to the medical domain. Second, propose novel approaches to transforming non-invertible data invertible.

The rest of the work is organized as follow: Section 2 introduces the dataset, model and evaluation methods. The results and discussion are in Section 3. Finally, Section 4 covers the conclusion of this work and future work.

2 Method

In this section, we describe the dataset we use in this study. Furthermore, explaining the data pre-processing steps for the models, and the evaluation methods.

2.1 Exploratory Data Analysis and Data Preprocessing

We will be using a fuzzy medical dataset SARS-CoV-1[6]. The dataset is separated into different files for each medical condition, such as SARS_Normal.csv for normal patient's data. First, we extract the column names from SARS_data_example.xlsx and then set the column names for each medical condition file, SM_HighBP.csv, SM_Normal.csv, SM_penumonia.csv, SM_Sars.csv. Finally, merge these four files to return the dataset that we will be using, SARS-CoV-1.

SARS-CoV-1 has 4000 rows and 23 columns. Every 1000 data rows correspond to one of the medical conditions, namely high blood pressure patients("HIGHBP"), normal patients("Normal"), SARS patients("SARS"), and pneumonia patients("pneumonia"). The data columns, on the other hand, are related to recorded clinical symptoms. They are temperature, blood pressure, conditions of nausea and abdominal pain. There are several fuzzy sets for each symptom.

For temperature, "slight", "mod", and "high". Blood pressure in levels of "low", "med", and "high". Nausea has "slight", "med", and "high", and "yes", and "no" for abdominal pain. The "no" column of abdominal pain is integer while the other columns are float.

An additional column named "Labels" is added to include the medical condition of each row. As the labels are categorical, we need to encode each label to train ANN. After encoding, "HIGHBP" is 0, "Normal" is 1, "SARS" is 2, and "pneumonia" is 3.

We will now move to explore the features. We create histograms for each symptom, temperature, blood pressure, nausea and abdominal pain to view their data distribution. The histograms had multiple peaks, which indicates different distribution for the different labels. Thus, we create a similar histogram graph for each label. From the created graphs, the data distribution is normal. We also plotted boxplots for each label, and we see no outliers (Fig. 1 shows an example). Hence no data transformation is needed. Additionally, we begin to see a general trend for each class.

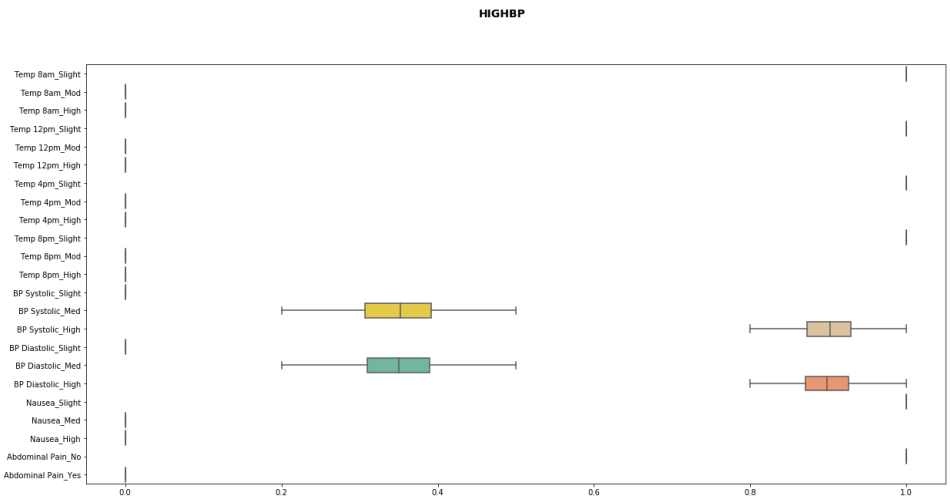

Fig. 1. High Blood Pressure Patient Boxplot Example.

We also calculated the mean of each feature for each class to see the pattern statistically. SARS patients overall experience high temperature across all time measured, high blood pressure and are the only ones encountering high nausea and abdominal pain. A normal person does not experience any extreme or high for each symptom. High blood pressure patients have high blood pressure and slight to moderate or medium values for other symptoms. Similar to high blood pressure patients, pneumonia patients only have high temperature while other symptoms are slight to moderate or medium. From this observation, we can see that the distinction between classes is still significant, thus our model could be learning these patterns quite easily.

Next, we want to know how relatable each feature is to the final label. With heatmaps similar to Fig. 2, observe that "slight" category symptoms are negatively correlated to the labels. While both "mod" and "high" category are highly positively correlated to the labels and one another. The abdominal pain "no" category is negatively related to the labels and vice versa for the "yes" category. In general, the heatmaps show a high negative correlation between both "mod" and "high" category with "slight" categories. Therefore, we can remove the "mod" category as the "high" category provides more value in classifying the prediction of labels, as explained in the observation above. We would keep the "slight" category as the dataset does not show that if a patient has possibilities of a "high" temperature, then there is no possibility of a "slight" temperature. The abdominal "no" category only has 0 and 1, which is the opposite of abdominal "yes". Therefore, we remove abdominal "no", as it is redundant.

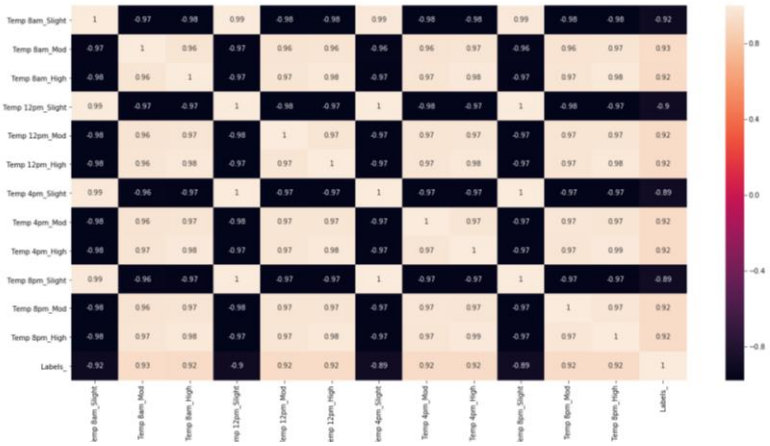

Fig. 2. Temperature Data Heatmap example.

2.2 Models and Techniques

BiDirectional Neural Network (BDNN) and Data Aggregation. BDNN [4] works the same way as normal ANN, with the only difference is that the error back-propagation is applied in the reverse direction. Regarding Fig.3, we denote the leftmost layer as the input layer, the middle layer as the hidden layer and the rightmost layer as the output layer. BDNN training starts with the usual forward training. In the forward training, the weights between the input and hidden and hidden and output layer are used. The bias in the hidden and output layer is also used. In the reverse training, the weights between the layers are used again, but this time, the bias of the input and hidden layer are used instead.

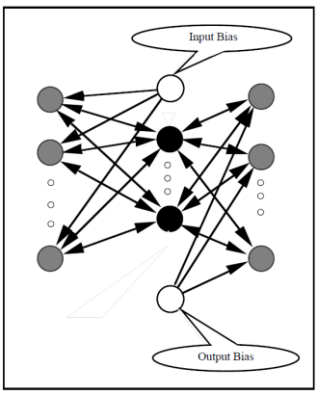

Fig. 3. BDNN Model (with no hidden bias).[4]

The paper that proposes BDNN [4] has explained that training forward and reverse direction requires the dataset to have a one-to-one relationship mapping. If the dataset is not invertible, then BDNN could not learn properly from the reverse direction. Ideally, we would want to input a value in the reverse direction, and the model would work out the parameter values to get the input value. Sars-Cov-1 data are not unique as we map many rows to one particular label. Nevertheless, we would want to see the application of BDNN in this supervised task and find the benefits of using it in the context of non-invertible data. To make the dataset have a one-to-one relationship, the paper [4] suggest the use of meaningless mnemonics on top of the classification labels, in a column named "Extra Node".

In this paper, we suggest the use of data aggregation to create a value that can associate to both the labels and each row. The three main data aggregation technique are minimum, maximum, and average. We will use the data aggregation technique to find the representative of each class. For instance, using the average technique. First, locate the rows for one of the labels such as 0, then calculate the means of each feature. Repeat this process for the other labels and create a data frame with each column corresponding to the label. With this data frame, calculate the difference between each row of data with the mean representation of the row's class label. The differences are then summed to return a single value. The value would then represent the possibility of the patient belonging to that specific label. The values would be stored under a new column called "Difference".

With this new column "Difference", we create a BDNN based on the XOR example in the technique paper [4]. As we have another column "Difference" to predict, in a new ANN module, create an extra layer that will output a value for the "Difference" column. In our forward training, we expect to return the predicted classification and predicted regression value. The challenging implementation is the reverse training.

Our prediction and regression values are all one-dimensional and not the size of the output neurons. Thus define a new function for reverse function and within this function, we will transform the input, which is the "Labels" and "Difference", to the shape that matches the output neuron nodes, and pass the values in reverse to BDNN. For "Labels", we will transform into one hot encoding and the shape of the values "Difference" is transposed. After the transformation, we need to multiply the output layer weights with the data and add the hidden layer bias respectively. Next, sum these new representations, apply the same activation function in forward training to the sum, multiply the hidden layer weights and add the input bias.

Note that the initial layer in BDNN does not have bias, hence we will initialize the bias the same way as the hidden and output layer is initialized. The bias, b_1 is initialized by randomly sampling from a uniform distribution, U with calculated bounds, bd_1. The bounded value is the reciprocal of the square root of the size of the input neurons.

$$b_1=U(-bd_1,bd_1). \quad \textbf{(1)}$$

With the BDNN module done, we need to train the model with epoch. As both prediction and regression are returned, the only difference to normal forward pass is we sum the losses and apply backward propagation. For training in reverse, we will be passing the labels and "Difference" value and predict the features. The loss for each predicted feature is summed as well and used to perform backward propagation.

An important criterion to check is that the values are unique. After feature selection, only the average technique returns the number of unique values the same as the dataset rows.

Fuzzy Signature and Weighted Relevance Aggregation Operator (WRAO). Fuzzy signature describes complex structures dataset in a hierarchical structure that shows the interdependence of the features. As shown in Fig 5., the SARS fuzzy signature shows the interconnected features across different levels and the possible weights associated with each feature at each level. For instance, the systolic and diastolic feature related to blood pressure and so forth. Therefore, we can carry out different aggregations on a different level to get a value that best represents the membership of a specific data row to a particular class. The simple data aggregation mentioned earlier place all the values equal weighting. With this fuzzy signature technique, we could emphasize more important features such as temperature. This technique is known as Weighted Relevance Aggregation Operator (WRAO). Ultimately, generating a value with membership to the class, and not the overall representation of the data.

WRAO has a few equations.[9] Equation 2 shows the WRAO for an arbitrary branch $a_{q..i}$ in a fuzzy signature, shown in Fig 4. $w_{q..ij}$ is the weight relevance of the leaf $a_{q..i}$, n is the number of leaves in the branch $q..i$, $j\epsilon[1,n]$, $p_{q..i}$ is the aggregation factor of the branch $q..i$. For equation 3, we use sigmoid function to replace $w_{q..ij}$.[9] The λ is any real value. We consider four kinds of aggregation, which are geometric mean, min, max, and normal mean. The p threshold for each is 0, >=99, <=-99, and other integer values. The use of geometric mean and threshold is explained in this paper. [9] To avoid name confusion, we refer λ as lambda parameter. In short, when we construct our graph, a branch needs both p, the aggregation factor, and λ, lambda parameter. The leaves in the branch only need λ, and have p as null. For initialization, we select an arbitrary number 1 for both p and λ.

$$a_{q..i} = \left[\frac{1}{n}\sum_{j=1}^{n}\left(w_{q..ij}a_{q..ij}\right)^{p_{q..i}}\right]^{\frac{1}{p_{q..i}}}. \quad (2)$$

$$w_{q..ij} = \frac{1}{1+e^{-\lambda_{q..ij}}}. \quad (3)$$

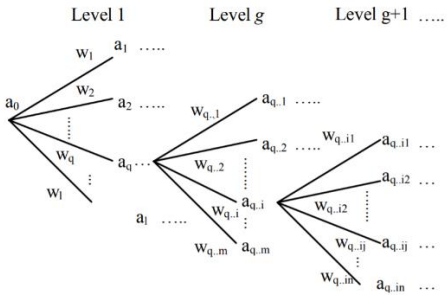

Fig. 4. Fuzzy Signature Structure with two arbitrary levels g and (g+1).[9]

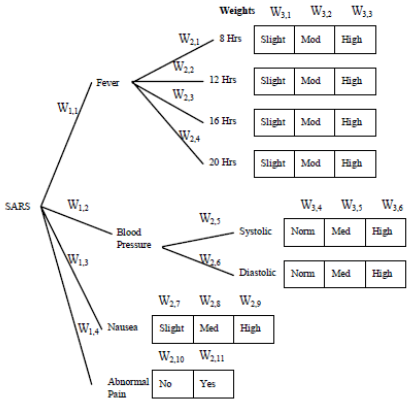

Fig. 5. SARS Fuzzy Signature (levels 1 to 3).[6]

Few additional data pre-processing steps are required to carry out the fuzzy signature approach. Firstly, we would need the complete dataset, SARS-CoV-1, with no label column, as we want to create the fuzzy signature that represents the original data structure. Secondly, create a new data column named “class_membership” that holds the fuzzy value of the class labels. Using the trend, we observed in the data, we create our fuzzy rules as such:

1. Normal patient: “slight” temp, “slight” blood pressure, “slight” nausea and “no” abdominal pain
2. SARS patient: “high” temp, “high” blood pressure, “high” nausea and “yes” abdominal pain
3. High blood pressure patient: “slight” temperature, “high” blood pressure, “slight” nausea, “no” abdominal pain
4. Pneumonia patient: “high” temperature, “slight” blood pressure, “slight” nausea, “no” abdominal pain

We did not create rules for every specific feature, such as blood pressure has systolic “slight”, and only generalize them to their overall category, in this case, blood pressure “slight”. The main reason being those detail rules are redundant as they all mean the same thing. Not only that, as the features are already in the fuzzy sets, we only create the fuzzy membership for class labels.

However, we have multiple temperatures, blood pressure data for “slight” and “high”. Here we will group similar categories and find the max value. These max value and other feature values will then input into the rules to get the membership. For instance, blood pressure has systolic “slight” and diastolic “slight”. We would find the max value of these two columns for each data rows to get blood pressure “slight”. The other aggregation method, mean and min, does not result in any different fuzzy membership value. Hence, we will continue using max. It was found that the membership values are mostly 1, which could post a problem when we predict the membership values.

Once done with the additional preprocessing, first, split the complete dataset, with the fuzzy class membership as the label, into a 50:50 ratio training testing set, the random state is 42. Similar to normal training ANN, we want the model to represent the general trend. In this case, ratio of 50 training is sufficient. Note that, the fuzzy signature could only

aggregate the membership value for one class. Thus, we need to create and train fuzzy signature for each 4 class, which are SARS, high blood pressure, normal, pneumonia. For this purpose, when training a class, we will set the "class_membership" of the other class in the training dataset as 0.

We will use the modified training set to construct the fuzzy signature in a tree format, similar to Fig. 5. In this case, the tree would have SARS as the main parent. SARS would then have the 4 medical conditions, fever, blood pressure, nausea, abdominal pain, as children, and each of them would have different weights and aggregation factor. Repeat this process for the other children, like "systolic" for blood pressure.

Next, we want to apply Levenberg-Marquardt Learning to WRAO to find the optimal λ and p. Levenberg-Marquardt Learning minimizes the Sum of Squared Errors (SSE). We will use the minimize function from the SciPy optimize library to help run Levenberg-Marquardt Learning and get the optimal value of the weights and aggregation factor for the tree. The method we use in this minimize function is the Sequential Least Squares Programming (SLSQP). According to the paper [6], it is best to have constraints when finding the optimal fuzzy signature. Therefore, in the package, SLSQ fits unto our purpose as we can specify the scalar function, which is SSE, but also the combination of bounds. The details of the parameter of this function are detailed in the Section 2.4. We will then update the tree with those new parameters' values.

At the end, we would have one optimized fuzzy signature for each of the 4 class. For testing, instead of removing the "mod" category and "no" category from the complete dataset, make those columns have a value of 0. The reason being the fuzzy signature is structured to account for those categories. We will pass the dataset without label information into the respective fuzzy signature and compare the aggregated output with the original fuzzy membership class.

However, we notice a key issue. First is that other than SARS patient, the other fuzzy signature is performing badly, as shown in Figure 6. We define good aggregation when the most of membership for a particular class is at least above 0.5, and the other classes below 0.5. Lastly, we add the mean representation of the data row to the SARS membership. As the end value would not be larger than 2, we would not scale it.

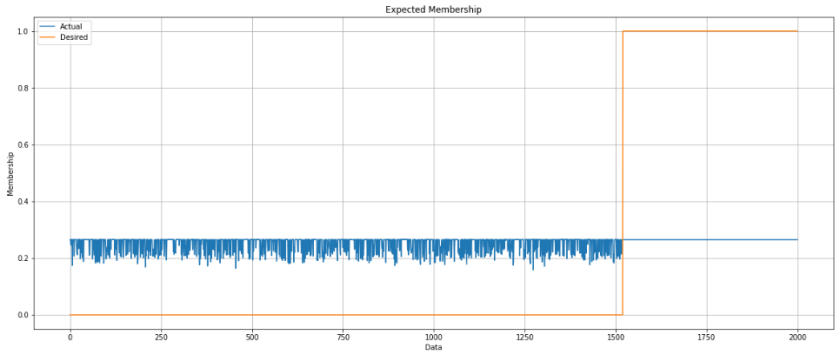

Fig. 6. Fuzzy Signature of High Blood Pressure example.

2.3 Model Default Hyperparameters

ANN could be used for supervised and unsupervised machine learning tasks. This study focuses on finding out the capabilities of BDNN for supervised tasks, we will create an ANN that will classify the patients with the given clinical data. The ANN will also be the baseline for accuracy comparison to BDNN. Using PyTorch library, create a custom ANN module named TwoLayerNet and another called BDNN. The implementations of BDNN are described in section 2.2. The fixed hyperparameters are the size of the input layer and output layer. The input layer is the size of our data, 15, while the size of the output is the number of distinct classes we have, which is 4. The default settings for other hyperparameters for both modules are given in Table 1.

Table 1. ANN and BDNN Default Hyperparameter Settings

Hyperparameters	ANN	BDNN
Number of Hidden Neurons	5	5
Learning Rate	0.01	0.01
Number of epochs	150	150
Activation Function	Sigmoid	Sigmoid
Loss Function	Cross Entropy	Cross Entropy, Mean Square Error (MSE)
Optimizer	Stochastic Gradient Descent (SGD)	Stochastic Gradient Descent (SGD)

Together with the dataset, we will tune our hyperparameters accordingly. Hidden neurons are tested within the range of 4 to 15, the sizes of input and output neurons [7]. For the model's learning rate, we will experiment with values from 0.01 to 0.1 first. We can four standard activation function, which are Sigmoid, Softmax, ReLu, and Tanh, to investigate which is best for our models. For ANN, the common loss function for regression is MSE and Cross-Entropy for classification. As for optimizer, we would mainly select SGD [8].

2.4 Fuzzy Signature Best Minimization Hyperparameters

The upper bound and lower bound of the aggregation factor, p, is experimented between (-99, 99) to (1000, 1000). We set (-99,99) as they are the threshold for p in WRAO. As we run the experiment, we find that the aggregation factor never goes over (-99,99). Therefore, we set the threshold to (-99,99). As for λ, we observe as the optimization run, that the weights often get close to 100. Thus, we experiment with 70 to 90, incrementing 10. However, the best performance is when the threshold is 100.

Ftol is the precision goal for the value of the scalar function in the stopping criterion. Our scalar function here is SSE. By default, the parameter has the value 1e-06. Our values would sometimes be even smaller, hence tried value between the range 1e-07 to 1e-10. As we decrease the parameter, more iteration would sometimes be needed as the function still does not get a loss smaller or equal to the specified value. Therefore, we tried 100 to 600 iterations. After a few running, we see that we achieve the best result with ftol as 1e-10, and 100 max iterations.

while eps is the step size used for numerical approximation of the Jacobian. As SLSQP is a method for constrained non-linear optimization, it includes Jacobian for improved efficiency. By default, it has 1.4901161193847656e-08. Same reason with ftol, our values is smaller, so we experience with the exponential power between 8 and 11. There is not much difference between 10 and 11. Hence we choose 1.4901161193847656e-10.

Table 2. Fuzzy Signature Default Hyperparameter Settings.

Hyperparameters	Fuzzy Signature
Upper Bound of Aggregation Factor	99
Lower Bound of Aggregation Factor	-99
Upper Bound of Weight Relevance Factor	100
Lower Bound of Weight Relevance Factor	0
Number of iterations	100
eps	1.4901161193847656e-10
ftol	1e-10

2.5 Evaluation Methods

For the performance evaluation, we will use a confusion matrix and calculate accuracy, precision and recall. The confusion matrix shows the actual and predicted classes, which helps us to view which class is predicted wrongly. Accuracy measures the number of correct predictions out of all the predictions. Thus, reflecting how good our model's predictions are. Precision and recall measure more specific predictions. Precision calculates the total of true positive overall positive predictions. Recall, on the other hand, measures the total positive class over the total true positive and false negative. In our context, precision shows the number of patients that we correctly identify as having SARS out of all the patients having SARS. Whereas recall measures how many patients are correctly identified as having SARS.

As for the testing dataset for evaluation, we use a train test split to split the original dataset into a 70:30 ratio with a random state equal to 42, to ensure the same training set and testing set is created every time. We do not use the cross-validation technique because our data for each class would not be shuffled. Thus, splitting via cross-validation would not ensure the model is training for other classes.

3 Results and Discussion

In this section, we will discuss the parameters from each of our models and combination with respective techniques. After which, we will compare and reason the results from each of our models.

3.1 Model Best Hyperparameters

We use Epoch for training to train the model to learn and find the patterns faster, and since weight initialization is randomized, we will run the predictions 3 times and average.

Table 3. ANN Hyperparameters.

	Combination 1	Combination 2	**Combination 3**	Combination 4
Number of Hidden Neurons	5	10	5	10
Learning Rate	0.05	0.05	0.05	0.05
Number of epochs	150	150	100	150
Activation Function	ReLu	Sigmoid	Tanh	Softmax
Loss Function	Cross Entropy	Cross Entropy	Cross Entropy	Cross Entropy
Optimizer	SGD	SGD	SGD	SGD
Training Accuracy	100	100	100	74.79
Testing Accuracy	100	100	100	75.70

Table 4. BDNN Hyperparameters

	Combination 1	Combination 2	**Combination 3**	Combination 4
Number of Hidden Neurons	10	10	5	15
Learning Rate	0.05	0.05	0.05	0.05
Number of epochs	150	250	100	150
Activation Function	ReLu	Sigmoid	Tanh	Softmax
Loss Function	Cross Entropy, MSE	Cross Entropy, MSE	Cross Entropy, MSE	Cross Entropy, MSE
Optimizer	SGD	SGD	SGD	SGD
Training Accuracy	100	100	100	74.57
Testing Accuracy	100	100	100	75.33

Table 5. BDNN-Aggregation Hyperparameters.

	Combination 1	Combination 2	**Combination 3**	Combination 4
Number of Hidden Neurons	5	10	10	15
Learning Rate	0.05	0.05	0.05	0.15
Number of epochs	100	250	80	300
Activation Function	ReLu	Sigmoid	Tanh	Softmax
Loss Function	Cross Entropy, MSE	Cross Entropy, MSE	Cross Entropy, MSE	Cross Entropy, MSE
Optimizer	SGD	SGD	SGD	SGD
Training Accuracy	100	100	100	74.50
Testing Accuracy	100	100	100	73.83

Table 6. BDNN-Fuzzy Signature Hyperparameters

	Combination 1	Combination 2	**Combination 3**	Combination 4
Number of Hidden Neurons	10	10	10	15
Learning Rate	0.05	0.05	0.05	0.15
Number of epochs	100	250	80	300
Activation Function	ReLu	Sigmoid	Tanh	Softmax
Loss Function	Cross Entropy, MSE	Cross Entropy, MSE	Cross Entropy, MSE	Cross Entropy, MSE
Optimizer	SGD	SGD	SGD	SGD
Training Accuracy	100	100	100	75.25
Testing Accuracy	100	100	100	74.42

In Table 3 to 5, we see that combination 3 (bolded in the tables) with the tanh activation function performs the best. In Table 3 and 4, we see no change in the hyperparameters in combination 3, and the model was still able to gain 100% on both training and testing accuracy. The combination with tanh performs better as the function maps values of zero to near-zero inputs, which helps in convergence. Our dataset contains many zero values, thus even though it increases the sparsity of the model using ReLu and other activation functions, those features could impact the classification.

In Table 5, we see tanh outperforms the other activation function. The increase in the number of hidden neurons is expected as the new column of values are related to the rows of data, meaning more pattern for the model to learn. Nevertheless, the model could be easily trained within 100 epochs. We also observe that the average aggregation technique also improved the model performance with ReLu, but no changes or even worse performance with Sigmoid

and Softmax. These results show that our dataset is not suitable for activation functions like Sigmoid and Softmax where the features are given a value between 0 and 1 only.

In Table 6, the best combination is still combination 3. The best parameters of the BDNN with fuzzy aggregation technique are exactly the same as the BDNN with aggregation technique. It is expected as the fuzzy signature aggregation adds additional complexity in learning by adding the mean values. We further compare their forward and backward training, and the BDNN with aggregation techniques have lower loss, which means the model is performing better.

Overall, we see that combination 3 hyperparameters that includes tanh as the activation function, is the best condition for our models.

3.2 Discussion on Metric Result

In Table 5, we observe that models with their best-performing hyperparameters always correctly classify all the classes in Sars-Cov-1. This implies that our dataset could be too simple to evaluate the performance of BDNN with the new average technique. Additionally, there is no major problem with the dataset like imbalance of classes or outliers. Hence given such a dataset, it is expected for our models to perform very well.

Our model performance would be similar to the performance reported by the dataset paper [6].

Table 7. Result Comparison Between ANN, BDNN, BDNN-Aggregation Models.

	ANN	BDNN	BDNN-Aggregation	BDNN-Aggregation
Training Accuracy	100	100	100	100
Training Precision	[1. 1. 1. 1.]	[1. 1. 1. 1.]	[1. 1. 1. 1.]	[1. 1. 1. 1.]
Training Recall	[1. 1. 1. 1.]	[1. 1. 1. 1.]	[1. 1. 1. 1.]	[1. 1. 1. 1.]
Testing Accuracy	100	100	100	100
Testing Precision	[1. 1. 1. 1.]	[1. 1. 1. 1.]	[1. 1. 1. 1.]	[1. 1. 1. 1.]
Testing Recall	[1. 1. 1. 1.]	[1. 1. 1. 1.]	[1. 1. 1. 1.]	[1. 1. 1. 1.]

Though all the models achieve high accuracy, with reference to the model hyperparameters table 3 to 6, we see that BDNN with the aggregation technique is performing better as the model could achieve the high accuracy with 10 neuron and train with 80 epochs. Usually, the more hidden neuron, more training data and time is required. Though the hidden neurons are two times the other models, 10 hidden neurons are still acceptable [7]. Hence, BDNN with aggregation technique is the best model.

3.3 Discussion on Techniques

We explored two techniques for converting non-invertible data to invertible, which are simple data aggregation and fuzzy signature. From the implementation description in Section 2.2, only mean aggregation technique could actually compute 4000 unique values. The other data aggregation techniques are unable to produce unique values, with the least number of unique is 1829 values. Fuzzy signature only has 4 unique values, which is even worse than expected.

In the case of our dataset, the membership values for each class are quite distinct, which consist of 1.0, 0.8, 0.6. Thus, when we pre-train our dataset, the optimized fuzzy signature would have weight relevance factor and aggregation factor that would aggregate the value to close to either one of those class membership value. Additionally, we have observed that we only found a somewhat optimal fuzzy signature for SARS patient but the found fuzzy signature on other patient class is bad. The one thing in common in the fuzzy issue appears to be the class membership value. More research could be undertaken to find the different rules that will produce the different membership class value. Additionally, we can try manually setting the aggregation factor, p, for each level of the fuzzy signature [6] and also experiment with other kinds of dataset.

Overall, currently for this dataset, the best technique is the simple mean aggregation technique.

4 Conclusion and Future Works

In this paper, we provided insights on the application of BDNN on a supervised task like medical condition classification. Two new technique was proposed to include more meaning to each data row when converting non-invertible data to invertible. The two techniques are data aggregation and fuzzy signature. We identify the limitations and further improvement of fuzzy signature. Overall, using aggregated data than random values in BDNN proof to be more successful in training the model and classifying the classes. One of our future works is to use a bigger and complicated medical dataset and further investigate the applicability of fuzzy signature in non-invertible data. Finally, given the successful classification of patient classes, we can investigate the BDNN on a non-supervised task like clustering.

References

1. Chan-Yeung, M., Xu, R.H.: SARS: epidemiology. vol. 8, s1, pp. S9-S14 (2003)
2. Hui, D.S.C., Wong, P.C., Wang, C.: SARS: clinical features and diagnosis. vol. 8, s1, pp. S20-S24 (2003)
3. Uddin, S., Khan, A., Hossain, M.E., Moni, M.A.: Comparing different supervised machine learning algorithms for disease prediction. vol. 19, 1(2019)
4. Nejad, A.F., Gedeon, T.D.: Bidirectional neural networks and class prototypes. (1995)
5. Lysaght, T., Lim, H.Y., Xafid, V., Ngiam, K.Y.: AI-Assisted Decision-making in Healthcare. vol.11, 3, pp. 299-314(2019)
6. Mendis, B. S. U., Gedeon, T. D., Kóczy, L. T.: Investigation of aggregation in fuzzy signatures. vol. 406. (2005)
7. Heaton, J.: The Number of Hidden layers, http://www.heatonresearch.com/node/707
8. Keskar, N.S., Socher, R.: Improving Generalization Performance by Switching from Adam to SGD. (2017)
9. Mendis, B.S.U., Gedeon, T.D., Botzheim, J., Kóczy, L.T.: Generalised weighted relevance aggregation operators for hierarchical fuzzy signatures. (2006)