Deep Learning Methodologies to Predict Fluid Responsiveness in Hemodynamically Unstable Patients

By

RAHUL KRISHNAMOORTHY

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Electrical and Computer Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Chen-Nee Chuah, Chair

Vladimir Filkov

Jason Yeates Adams

Committee in Charge

2021

ABSTRACT

Deep Learning Methodologies to Predict Fluid Responsiveness in

Hemodynamically Unstable Patients

Deep Learning is a branch of machine learning with a layered structure where each layer gets its input from the previous layer to analyze the data and has shown promising results to aid healthcare providers in various applications. This thesis presents the development and evaluation of various Deep Learning approaches to predict if a hemodynamically unstable patient would be responsive to infusion of intravenous fluids. This thesis also explores various meticulously designed experiments to thoroughly verify the predictive model’s generalization across various carefully curated datasets to represent shocks resulting from different physiologies in pigs.

Treatment of patients suffering from shock often involves the infusion of intravenous fluids, also known as fluid bolus therapy (FBT). An increase in cardiac output (CO) of 15% or more after a supply of 500 ml of the fluid bolus indicates fluid responsiveness, and the ground truth labels were designed based on this rule. In addition, the arterial blood pressure (ABP) and central venous pressure (CVP) waveforms were recorded before and after the infusion of each bolus. The period before, during, and after the administration of fluid boluses is known as premacrobolus (Premac), macrobolus, post-macrobolus (Postmac), respectively. The deep learning model takes sequences of specific lengths obtained from the ABP and CVP during the premac period and the ground truth labels to classify each bolus to be fluid responsive or not. The models were tuned and trained using nested cross-validation accompanied by grid search algorithms.

The results from our experiments suggest that deep learning can offer a satisfactory framework to classify boluses as fluid responsive or fluid non-responsive. In addition, this thesis presents a comprehensive guide to experimentation on various aspects that could potentially affect the performance of deep learning models while classifying one-dimensional data, including input sequence length, model's architecture, sample weighting in loss functions, normalization, resampling the data, and various methods to sample the data to acquire meaningful inferences from the results. The experiments showcase the restricting nature of small-scale datasets on the deep learning model's performance. The deep learning model fails to generalize when the training and the test sets contain different physiologies but generalizes better when both the training and the test sets contain a comparable mixture of physiologies.

ACKNOWLEDGEMENTS

I would like to thank my exceptional and supportive thesis committee members for reviewing my work and offering your invaluable feedback. Thanks to my graduate advisor and the committee chair, Prof. Chen-Nee Chuah, for mentoring me through this project and guiding me to become a better researcher. You were always available to offer your feedback whenever I required it and constantly nudged me in the right direction to excel in my work. Thank you, Dr. Jason Adams, for supporting me through my thesis and taking time off your busy schedule to guide me through my work. Thanks for letting me shadow you at UC Davis Health and for all your practical lessons. I learned a lot about critical care medicine and data science through our meetings. This thesis would not have been possible without the guidance from Prof. Chen-Nee Chuah and Dr. Jason Adams. Thank you, Prof. Vladimir Filkov, for accepting to be a part of my thesis committee and reviewing my work. I learned a great deal from your data science course, which inspired my work on this thesis.

Thanks to Debraj Basu for all your insights during our meetings and for offering guidance through this thesis. I would also like to thank Gregory Rehm for all those technical paper discussions and for answering my questions. Finally, special thanks to my mom, dad, sister, and friends for always being there for me and supporting me through my Master’s. This wouldn’t have been possible without you all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Various advances in Machine Learning (ML) and Deep Learning (DL) have simplified many complex tasks. ML and DL have many applications in regression, forecasting, and classification problems. For example, they have vast applications in autonomous driving and video surveillance[1], [2]. Furthermore, much research is going on in speech recognition, and as each day progresses, artificial intelligence's (AI) ability to understand human speech is increasing [3]. Automatic stock trading and traffic forecasting also utilize DL methods, reinforcing its versatility [4], [5].

AI has also shown promising results in the healthcare industry. Patient monitoring and recording has resulted in a large influx of data, which helps make robust and reliable ML and DL models to assist patients. For instance, DL has helped the health industry make giant leaps to help diagnose and monitor illnesses like Covid-19 [6]. Shock is another critical condition that can have detrimental effects on the human body. It occurs due to an abrupt contraction of blood flow through the body. Shock could occur due to various reasons, including infection, trauma, blood loss, and fluid loss. For example, septic shock occurs when the blood pressure drops abruptly due to an infection. Likewise, hypovolemic shock occurs due to a large amount of blood or fluid loss from the body. If hypovolemic shock occurs due to blood loss, it is known as a hemorrhagic shock. Finally, neurogenic shock could occur due to trauma or injury in the spine.

Treatment of patients suffering from shock frequently involves the infusion of intravenous fluids, also known as fluid bolus therapy (FBT). However, it is essential to balance the amount of fluid input and output in the body, and an excessive amount of injection of fluids or going against the ‘Fluid Balance’ is associated with high mortality rates [7], [8]. ‘Fluid responsiveness’ or a positive response to the infusion of fluids is defined as a change in Cardiac Output (CO) or Stroke Volume (SV) of 10-15% [9].

Arterial blood pressure (ABP) waveforms are the one-dimensional representation of arterial pressure occurring due to the heart’s left ventricle pumping action and the systemic vascular resistance. The clinically referenced method to measure ABP waveforms involves utilizing invasive Blood Pressure (BP) measurement using an arterial catheter [10]. ABP waveforms contain systolic, diastolic, and mean arterial pressure information (MAPs) for each beat. The utilization of ABP waveforms and their interactions with respiration can help assess patients’ overall cardiovascular and hemodynamic status [11]. The Central venous pressure (CVP) stands for the blood pressure measured in the venae cava next to the heart’s right atrium. CVP waveforms provide vital information about the cardiocirculatory status of the patient and help guide fluid resuscitation irrespective of its shortcomings [12].

There are numerous methods to predict fluid responsiveness in the literature. The passive leg raising technique offers a reliable method to predict fluid responsiveness by creating a reversible increase in venous return [13]. However, the requirement of labor to perform passive leg raising each time hindering automation and the requirement of specialized apparatus are hard to look over despite the method's effectiveness to predict fluid responsiveness. Pulse pressure variation (PPV) is an important metric that has been vastly studied and used to predict fluid responsiveness. PPV is calculated across the respiratory cycles in mechanically ventilated patients by observing the changes in pulse pressure in ABP waveforms. However, the operative performance of PPV is fluctuating across various studies. For example, PPV performed brilliantly to predict fluid responsiveness in critically ill patients but was severely obstructed by the compliance of the respiratory system for patients mechanically ventilated at low tidal volumes [14]. Additionally, Marik et al. pointed out that PPV can offer compelling results but is limited to patients who receive controlled ventilation and those not breathing spontaneously [15]. Finally, Teboul et al. marked that during the various conditions encountered in the ICU, such as spontaneous breathing and cardiac arrhythmias, PPV is often unreliable [16].

Advances in machine learning paved the way for the reliable prediction of hypotension from arterial pressure waveforms [17]. Zhang et al. used XGBoost and logistic regression to predict the volume responsiveness in patients with oliguric acute kidney injury, defined as a rise in urine output in the hours after FBT [18]. Kamaleswaran et al. used various machine learning models, including the random forest and logistic regression algorithms, to predict volume responsiveness among sepsis patients using features such as mean arterial blood pressure and the age identified [19].

In this thesis, we present the development and evaluation of various Deep Learning approaches to predict if a hemodynamically unstable patient would be responsive to infusion of intravenous fluids. We make use of datasets that were carefully curated to represent shock resulting from different physiologies. We also present various sample weighting mechanisms for loss functions and resampling methods, such as the synthetic minority over-sampling technique [20] that deal with class imbalances. We also evaluate the ideal length of the input time-series data to capture the critical information from the waveforms fed into the neural networks for training and making predictions. Additionally, we use various deep learning architectures such as Multivariate Long short-term memory fully convolutional network and DenseNet with various changes to their conventional architectures to estimate the change in performance resulting due to the change. Finally, we experiment with normalization and various sampling of datasets to acquire meaningful interpretation of results.

Chapter 2

BACKGROUND

2.1 Deep Learning

Predictive modeling is the process of finding relationships from structured data to predict the desired outcome [21]. In this thesis, we primarily look at the one-dimensional data of ABP and CVP of pig's waveforms during the premacrobolus phase to classify the sequence to be fluid responsive or not. Supervised learning involves supplying both the dependent and independent variables as an input to a mathematical model. The purpose of the mathematical model is to observe the examples and produce predictions that are close to the ground truth variables as much as possible [22]. The experiments conducted involve using supervised learning techniques to predict whether a pig is fluid responsive or not.

Neural networks are algorithms designed to discern patterns from input data and are modeled based on the human brain. A neural network incorporates input and an output layer. If there are multiple additional layers present in the neural network, they are known as deep neural networks, and the branch of predictive modeling that uses deep neural networks to make predictions is known as deep learning. A deep neural network can approximate very complex functions by increasing the number of units in a layer and the number of layers. Thus, deep learning offers a compelling framework for the task of supervised learning [23]. Our experiments use the deep learning algorithm's capabilities to provide compelling predictions for fluid responsiveness. Specifically, we use two algorithms, namely Multivariate Long Short Term Memory Fully Convolutional Network (MLSTM-FCN) and Densely Connected Convolutional Networks (DenseNet) for binary time series classification [24], [25].

2.1.1 Convolutional Neural Networks

The process of convolution combines two signals to produce a third signal. Consequently, convolution is applying a filter to input data that creates an activation from a deep learning point of view. Equation (22) represents the convolution operation in a convolutional neural network where $\otimes$ represents the convolution operator, and W represents the kernels and b represents the bias vector.

$$y = W \otimes x + b \tag{1}$$

Convolutional neural networks (CNN) can capture the spatial and temporal dependencies in the input data through relevant filters. CNN also eliminate the need for traditional hand-crafted feature extractors required for pattern matching. Due to the ability of CNN to capture the spatial and temporal information from the data, they provide means to create end-to-end models that achieve good performance in time-series classification [26]. Time-series data encourages the usage of one-dimensional convolutions where the kernel slides along one dimension. Apart from the convolution operation, CNN use batch normalization, activation, and pooling layers to make meaningful predictions.

Batch Normalization

Batch normalization (BN) introduces normalization as a part of the model architecture and normalizes the data of each mini-batch during training. Batch normalization allows having aggressive learning rates and thus accelerates the training process. BN also allows models to be less susceptible to weight and bias initializations. In addition, BN tends to have a regularizing effect and could eliminate the requirement for dropout layers for regularization [27].

Activation Functions

Activation functions help to decide whether a neuron should be activated or not. It performs this action by introducing a non-linearity in the neural network. As a result, output from each layer is a non-linear function of the input to that layer. It is essential to introduce non-linearities in the network as it helps different layers learn different features from the input data. Our experiments used 4 different kinds of activation functions: sigmoid, hyperbolic tangent (tanh), Rectified linear unit (ReLU), and softmax.

Sigmoid Function

The sigmoid function is a standard activation function in the deep-learning domain to introduce non-linearities in deep neural networks. Also known as the logistic function, equation (22) is the mathematical representation of it.

$$f(x) = \left(\frac{1}{1 + e^{-x}}\right) \tag{2}$$

It is a bounded differential function and has positive derivatives throughout its domain. However, the sigmoid function has a few drawbacks too. For example, it suffers from gradient saturation, gradient dampening in deeper layers of the neural network, relatively slow convergence compared to other activation functions, and has a non-zero-centered output causing gradients to jump in different directions during training.

Hyperbolic Tangent Function (Tanh)

The hyperbolic tangent function is smoother and zero-centered, and its range lies between -1 and 1. Therefore, equation (22) represents the tanh function.

$$f(x) = \left(\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}\right) \tag{3}$$

The Tanh function had a better training performance when using multi-layer neural networks. However, the tanh function does not solve the vanishing gradient problem prevalent in deeper networks. Another critical aspect of the tanh function is that it can only obtain a gradient of 1 if the input is 0, resulting in the creation of dead neurons while training.

Rectified Linear Unit (ReLU)

Rectified linear unit is one of the most widely used activations functions in the deep learning domain [28]. ReLU computes its output based on the mathematical expression displayed in equation (22).

$$f(x) = \max(0, x) = \begin{cases} x_i, & if\ x_i \geq 0 \\ 0, & if\ x_i \leq 0 \end{cases} \tag{4}$$

ReLU is one of the fastest learning activation functions and offers better performance than sigmoid and tanh. Moreover, since it offers an almost linear function, it is effortless to optimize using gradient descent techniques.

Softmax Function

Softmax computes the probability distribution from an input vector and thus creates output that satisfies the law of total probability. The softmax function computes its output based on the mathematical relationship displayed in equation (5).

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{5}$$

The softmax function has its use case when dealing with multiclass models where it returns the probability of each class with the target class having the highest probability. The primary difference between the sigmoid and softmax function is that the sigmoid function is used for binary classification while the softmax function deals with multiclass classification tasks.

Pooling Layers

The purpose of pooling layers is to downsample the feature vectors by summarizing specific patches in the feature vector. Max pooling and average pooling are the widely used methods to downsample by taking the maximum value from a patch and the average of a patch, respectively.

Instead of working on small patches from the feature vector, global pooling layers downsample the entire feature vector to a single value. Thus, global pooling helps summarize the entire feature vector, and the most common global pooling layer used in modern deep learning networks is the Global average pooling layers (GAP). GAP has replaced the fully connected layers in a plethora of deep learning algorithms, and they have helped reduce overfitting from the fully connected layers.

2.1.2 Densely Connected Convolutional Networks

Convolutional neural networks can get deeper and hence more accurate in their predictions if they have connections between the layers. These connections are known as residual connections, and the networks that utilize these residual connections to exercise better flow of gradient are known as Residual networks [29]. Densely connected convolutional networks (DenseNet) utilize this property by connecting the adjacent layers and every other layer in the neural network [25].

Each layer takes the concatenated feature maps of the preceding layers as its input, and its feature maps are used as inputs by all the subsequent layers in the network.

DenseNet also utilizes a composite function motivated by [30] comprising batch normalization (BN), Rectified linear unit, and convolution operations in conjunction. DenseNet requires a way to downsample the feature vectors to a specific size to be concatenated with the feature vectors from other layers. Transition layers facilitate the concatenation process by having a BN layer with a 1x1 convolution layer followed by an average pooling layer. Since DenseNet concatenates many features in the dense block and each layer takes the concatenated features as input, the size could become huge and slow down the training. Thus, 1x1 convolutions inspired from [31] are introduced before each convolution operation in the dense block to act as bottleneck layers.

DenseNet introduced in [25] utilizes the architecture primarily for image classification and object detection on datasets like CIFAR [32], Street View House Number (SVHN) [33], and ImageNet [34]. However, advances in deep learning and the growing popularity of one-dimensional convolutions enabled DenseNet architecture for time-series classification [35], [36]. This thesis uses DenseNet to perform the binary classification of fluid responsiveness by analyzing the ABP and CVP time-series data.

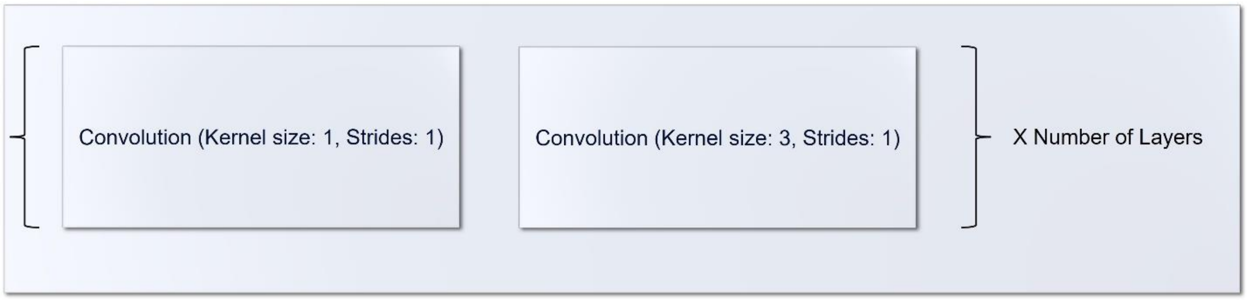
Convolution (Kernel size: 1, Strides: 1)
Convolution (Kernel size: 3, Strides: 1)
X Number of Layers

Figure 2.1: Dense block in a densely connected convolutional network

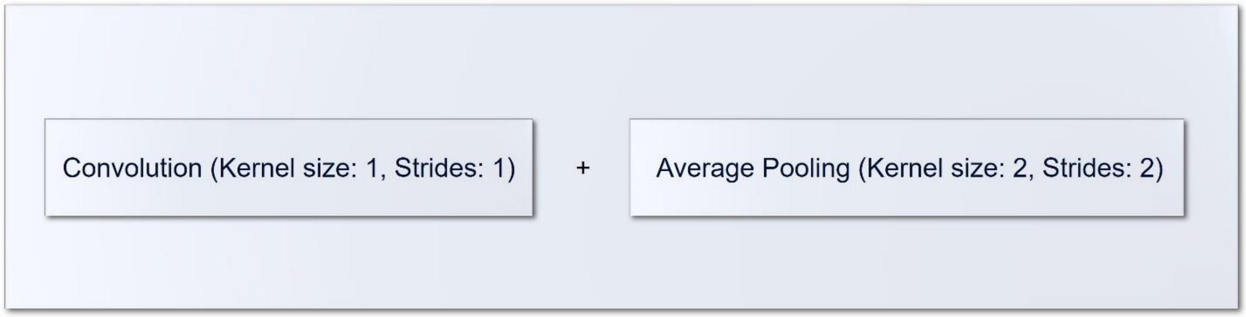
Convolution (Kernel size: 1, Strides: 1)
+
Average Pooling (Kernel size: 2, Strides: 2)

Figure 2.2: Transition layer in a densely connected convolutional network

Figure 2.1 and Figure 2.2 show the dense block and the transition layers present in a densely connected convolutional network. In the DenseNet18 model used in this thesis, The input layer takes the one-dimensional waveforms to the network. The input layer is attached to a convolutional layer with a kernel size equal to 7, a stride length of 2, and a max-pooling layer with a kernel size equal to 3 and a stride length of 2. The architecture then consists of 4 pairs of dense blocks and transition layers, as described in Figure 2.1 and Figure 2.2. Each dense block was cascaded twice immediately, followed by a transition layer. Then, a global average pooling layer with a kernel size of 7 sums up the features in each channel of the convolutional layers. Finally, the global average pooling layer is followed by a 1000 dimensional fully-connected layer with a softmax activation to provide meaningful classifications. This is the basic model architecture of a DenseNet18 model used in the experiments in this thesis.

2.1.3 Multivariate LSTM-FCN for Time-Series Classification

Multivariate Long Short Term Memory Fully Convolutional Network (MLSTM-FCN) [24] combines two of the widely used algorithms in time-series related tasks, namely Long Short Term Memory (LSTM) [37] and one-dimensional Fully Convolutional Networks (FCN) together. Moreover, the algorithm facilitates using the attention mechanism [38] combined with LSTM for multivariate time series classification. Finally, Squeeze-and-excitation [39] blocks augment the fully convolutional block's ability to classify the time-series data better. Furthermore, the MLSTM-FCN model requires minimal preprocessing and feature extraction, making it a robust algorithm for creating an end-to-end model to predict fluid responsiveness using ABP and CVP waveforms.

Long Short Term Memory (LSTM)

Long short term memory is a category of a Recurrent Neural Network (RNN) capable of learning the dependence in the order of terms from a sequence of data. A significant drawback with the previous versions of RNNs are the problems of vanishing and exploding gradients. LSTM tackles the vanishing and exploding gradient problems by integrating gating functions into the state dynamics [37]. There are various computations taking place in the LSTM as depicted by Graves et al. [40] in the following equations:

$$g^u = \sigma(W^u h_{t-1} + I^u x_t) \tag{6}$$

$$g^f = \sigma(W^f h_{t-1} + I^f x_t) \tag{7}$$

$$g^o = \sigma(W^o h_{t-1} + I^o x_t) \tag{8}$$

$$g^c = tanh(W^c h_{t-1} + I^c x_t) \tag{9}$$

$$m_t = g^f \odot m_{t-1} + g^u \odot g^c \tag{10}$$

$$h_t = \tanh\,(g^o \odot m_t) \tag{11}$$

Where g^u, g^f, g^o, g^c are the input, forget, output, and cell state activation functions, respectively, h_t is the LSTM’s hidden state vector, $\odot$ represents element-wise multiplication, σ represents the sigmoid activation function. Projection matrices are represented by I^u, I^f, I^o, I^c while the weight matrices are represented by W^u, W^f, W^o, W^c. Even though LSTMs effectively learn temporal dependencies from the sequential input data, they find it challenging to learn long-term dependencies. To learn long-term dependencies from the input, Bahdanau et al. [38] proposed using the attention mechanism with LSTM.

Attention Mechanism

Bahdanau et al. [38] proposed the attention mechanism that performs a linear combination of encoder and decoder states and is hence known as additive attention. The attention mechanism generates a context vector based on the target sequence y. When an encoder maps an input sequence x, a sequence of annotations $(b_1, \ldots, b_{T_x})$ of length T_x are created where each annotation b_i contains information on the input sequence while still focusing around the i-th word. Based on equation (22), the weighted sum of the annotations b_i is used to compute the context vector.

$$v_i = \sum_{j=1}^{T_x} \alpha_{ij} b_j \tag{12}$$

The weight α_{ij}can be calculated by the following:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{13}$$

In equation (22), e_{ij} is the energy of alignment, computed using a feedforward neural network known as the alignment model. The feedforward neural network is trained parallelly along with the RNN. The gradients of the alignment model and the RNN flow simultaneously.

Squeeze-and-Excitation Block

Channels in a convolutional neural network are independent of each other, and they hardly influence one another. Squeeze-and-excitation blocks are introduced in CNNs to improve channel interdependencies [39]. Squeeze-and-excitation blocks adjust the filter response to improve channel interdependencies in two steps: squeeze and excite.

The global average pooling layer computes the channel-wise global average over the temporal dimension T over the convolved output during the squeeze operation.

$$z_c = F_{sq}(u_c) = \frac{1}{T}\sum_{t=1}^{T} u_c(t) \tag{14}$$

In equation (22), c is the channel and z_c is the channel wise statistics obtained after squeezing the convolved data through a global average pooling layer. Excite operation follows the squeeze operation, whose primary objective is to capture the inter-channel dependencies, represented in equation (22).

$$s = F_{ex}(z, W) = \sigma(W_2\delta(W_1 z)) \tag{15}$$

In equation (22), F_{ex} is a neural network, σ is the sigmoid activation function, δ represents the ReLU activation function. W_1 and W_2 are the learnable parameters from the neural network and are used to limit the model complexity and aid in better generalization. Finally, rescale the output of the block, represented in equation (22).

$$\widetilde{x_c} = s_c . u_c \tag{16}$$

Where $\tilde{x} = [\widetilde{x_1}, \dots, \widetilde{x_c}]$ is obtained by channel-wise multiplication of the scale and the feature map. Thus, the output feature map is scaled and contains information about the channel dependencies. Hence, the squeeze-and-excitation block acts as a powerful augmentation tool for convolutional neural networks.

Model Architecture

The multivariate LSTM fully convolutional network model has the one-dimensional convolutional neural networks, long short-term memory, squeeze and excitation blocks, and global average pooling layers in its architecture, as shown in

Figure 2.3. Apart from the components of the architecture mentioned, the model architecture has a unique entity called the dimensional shuffle, which helps transform the input data to be compatible with the long short-term memory cells. The dimensional shuffle layer swaps the dimensions of the input data. The default format for the input to the MLSTM-FCN model is of the form $(N * L * F)$ where N is

the number of samples, L is the length of more minor sequences (for example, 224), and F is the number of features (F=1 when either ABP or CVP is included and F=2 when both of them are included). The dimensional shuffle layer transforms the last two dimensions of the input so that the LSTM does not have to take L time steps of F features but can take up F time steps of L features.

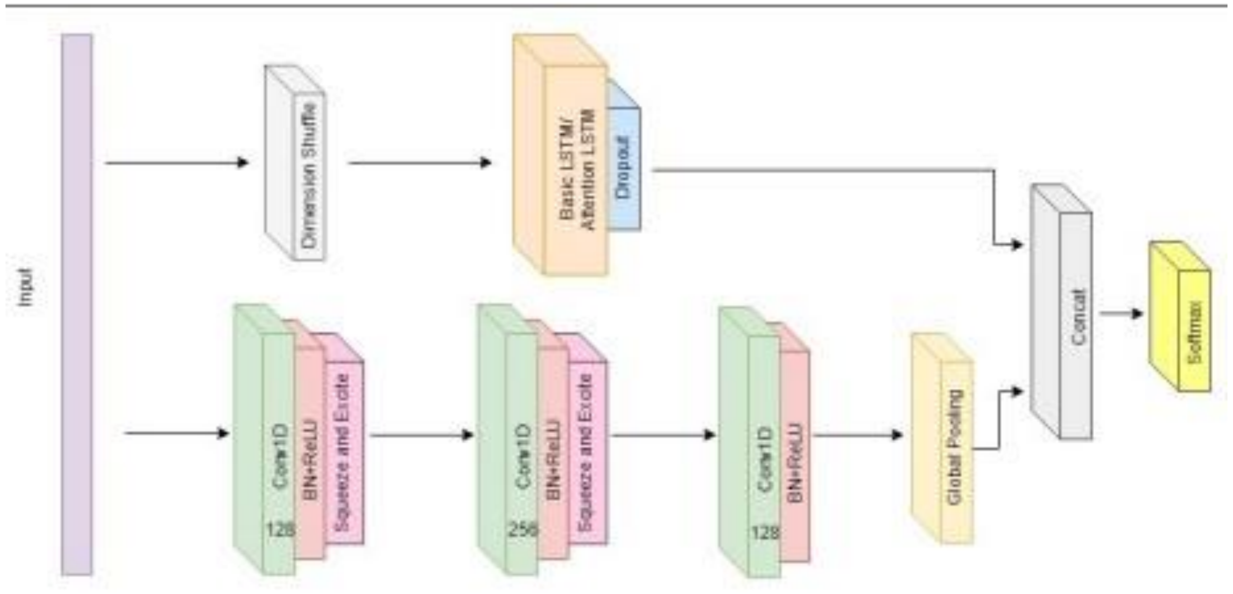

Figure 2.3: MLSTM-FCN architecture. The attention mechanism can be added to the LSTM cells if needed to create the MALSTM-FCN architecture. Reprinted from Neural Networks, Volume 116, Fazle Karim, Somshubra Majumdar, Houshang Darabi, Samuel Harford, Multivariate LSTM-FCNs for time series classification, pages 237-245, Copyright (2019), with permission from Elsevier.

Thus, the dimensional shuffle layer helps the LSTM with the global understanding of the data by providing a single feature's time steps to a single cell.

Thus, the regular input goes through the convolutional side of the architecture, and the transformed input goes through the LSTM side of the architecture. Once the input passes through the subsequent layers, the feature vectors on each side are concatenated together. Finally, this concatenated feature vector is sent to the softmax layer for binary classification.

2.2 Nested Cross-Validation

Hyperparameter optimization and model selection are two of the essential steps in any machine learning project. Any model has a set of hyperparameters which are the variables that can be changed to better fit a model to the training data. Hyperparameter optimization is the process of tuning the hyperparameters to find the best fit of the model to the training data such that the model does not overfit the validation and, in turn, the training set. Comparison of multiple models to find the best model based on the chosen performance metrics is known as model selection. K-fold cross-validation is a procedure used to find the machine learning model's performance on the data not used during training. But the usage of the same cross-validation procedure and the dataset for both hyperparameter tuning and model selection can produce an optimistically biased result.

Unlike K-fold cross-validation, nested cross-validations aim to separate the hyperparameter optimization and model selection procedures into independent steps. Nested cross-validation nests the K-fold cross-validation for hyperparameter tuning in the K-fold cross-validation for model selection. Due to the nesting, the K-fold cross-validation for hyperparameter tuning is not exposed to the entire dataset and hence doesn’t overfit. A downside to nested cross-validation is the significant increase in the number of trained and evaluated models. Another vital aspect of nested cross-validation is to have the same seed for the experiments conducted on the same dataset. This seeding is necessary to ensure that the experiment setup is similar across the experiments to make meaningful comparisons and validate various hypotheses. In this thesis, we make use of nested cross-validation for both hyperparameter optimization and model selection.

Chapter 3

METHODOLOGY

3.1. Predictive Modeling Pipeline

Any predictive modeling, including deep learning pipelines, has to go through a series of steps. These steps may vary depending on the dataset and on the model used in the pipeline. Figure 3.1 explains the steps involved in the predictive modeling pipeline used in this thesis. The steps involve obtaining the dataset, followed by data cleaning and preprocessing necessary to make the data ready to input into the model, followed by model development which involves training and tuning the hyperparameters, followed by nested cross-validation. Once the model has gone through all these steps, the model is ready to make predictions. This section discusses each step involved in the predictive modeling pipeline in detail.

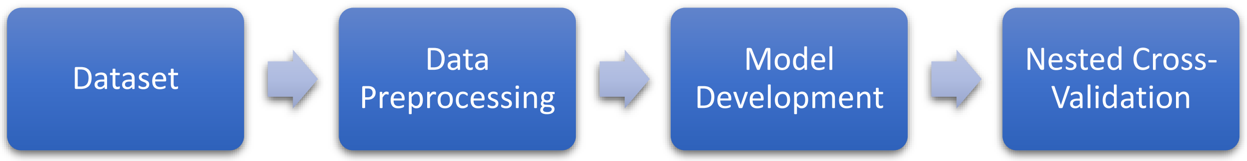

Figure 3.1: Steps involved in the predictive modeling pipeline

3.2. Dataset

Arterial Blood Pressure (ABP) and Central Venous Pressure (CVP) waveform data were collected from 62 pigs obtained from different experiments. Out of the 62 pigs, 13 experienced hemorrhagic shock, 4 experienced septic shock, 32 were from EPACC_Trial1, and 13 were from the IRI_FR experiment. The 4 pigs with the septic shock model were subjected to an intravenous infusion of Pseudomonas aeruginosa bacteria [41]. The animals were supplied with sequential 500 mL boluses. Same micro-bolus protocols were maintained throughout each phase of the experiment while the boluses were delivered. The timing for the bolus delivery was made at the discretion of the veterinary team that treated the pigs.

13 pigs in the hemorrhagic model received a controlled hemorrhage of 25% of the standard estimated blood volume resulting in a hypovolemic shock. The other 13 pigs in the IRI_FR experiment correspond to the ischemia-reperfusion

injury (IRI) model of circulatory shock. A controlled hemorrhage immediately followed by 30 minutes of complete aortic occlusion and restoration of aortic flow to the lower half of the body resulted in an ischemia-reperfusion injury [42]. Hypovolemic, euvolemic, and hypervolemic are the different phases present in the hemorrhagic and IRI shock models. The hypovolemic phase is caused purely by blood loss, while the euvolemic and hypervolemic phases correspond to a transfusion of 25% volume of shed blood and transfusion of an additional 25% blood volume from a donor animal, respectively. The hemorrhagic and IRI shock models animals received four separate 500mL boluses of Vetivex^{TM} Veterinary pHyLyte^{TM} solution. These boluses were provided for 10 minutes at regular intervals throughout each experiment phase, with a 5-minute pause between each bolus. During each bolus sequence, the fluids were administered in micro-boluses of 100mL for 60 seconds, followed by a 60-second pause between each micro-bolus. In addition, all the animals in the experiments were provided with a continuous infusion of norepinephrine. The infusion was adjusted to maintain a baseline mean arterial pressure above 60 mmHg before initiating the experiment. Once the experiments proceeds, the mean arterial blood pressure was maintained at the baseline rate for the remaining time.

The pigs were given fluid boluses at regular intervals during each phase of the experiment conducted. The ABP and CVP waveforms were recorded before and after the delivery of each bolus. The period before the administration of fluid boluses is known as pre-macrobolus (Premac), the period after the administration of boluses is known as post-macrobolus (Postmac), and the time during the administration of fluid boluses is known as macrobolus. Recording of the various physiological waveforms used a 60-second window during each phase of the experiment. In addition, the ABP and CVP waveforms went through steps involving data quality assessment and preprocessing developed by Basu et al. [43].

3.2.1. Ground Truth Label

The primary purpose of this thesis is to model, train and validate the performance of supervised learning models to find whether the pigs are fluid responsive or not by working on the ABP and CVP waveforms. Furthermore, generating ground truth labels for the dataset is necessary to create a supervised learning model for binary classification. Cardiac output was measured using either an intra-cardiac pressure-volume loop catheter or an ultrasound flow probe placed over zone 1 of the descending aorta as a surrogate for cardiac output [44]. An increase in cardiac output of 15% or more after a supply of 500 ml of the fluid bolus indicates fluid responsiveness [45]. The records of cardiac output during the Premac and Postmac periods helped track down the change in cardiac output before and after the administration of fluid boluses. Whenever the change in cardiac output is greater than or equal to 15%, the bolus was considered to be fluid responsive (Ground truth label = 1), and whenever the change in cardiac output is less than 15%, the bolus was considered to be fluid non-responsive (Ground truth label = 0).

3.2.2. Dataset Characteristics

Apart from having different physiologies, the Hemorrhage (Hem), Sepsis (Sep), EPACC_Trial1, and IRI_FR datasets also had different distributions. On the whole, 497 samples corresponded to 497 macroboluses. Out of these 497 maroboluses, 134 came from the hemorrhage pigs, 90 were obtained from Sepsis pigs, 150 macroboluses were derived from the EPACC_Trial1 dataset, and the IRI_FR dataset had the remaining 123 macroboluses. The ratio of positive to negative samples for the Hemorrhage pigs was 55:79, while the ratio for Sepsis pigs was 4:86. The ratios for EPACC_Trial1 and IRI_FR datasets were 109:41 and 39:84, respectively. Figure 3.2 represents the distribution of positive and negative samples in each dataset. Table 3.1 has the various combination of the datasets used in all the experiments and their respective characteristics.

During the experiments in this thesis, the datasets are combined to form the training and test sets. Table 3.1 shows the various combinations and characteristics of the datasets used in the experiments in this thesis. The Hem_Sep dataset combines the boluses from the hemorrhage and the sepsis pigs. If an additional term, 'scaledCVP,' is added to the name, it means that the CVP waveforms of a few boluses were rescaled, as shown in Section4.4.2.2. The Hem_Sep_ScaledCVP_EPACC_Trial1 combines the boluses from hemorrhage, sepsis, and the EPACC_Trial1 pigs. IRI_FR_1_2, IRI_FR_1_2_3, and

IRI_FR_ALL are different samples obtained from the same experiment and similar physiology.

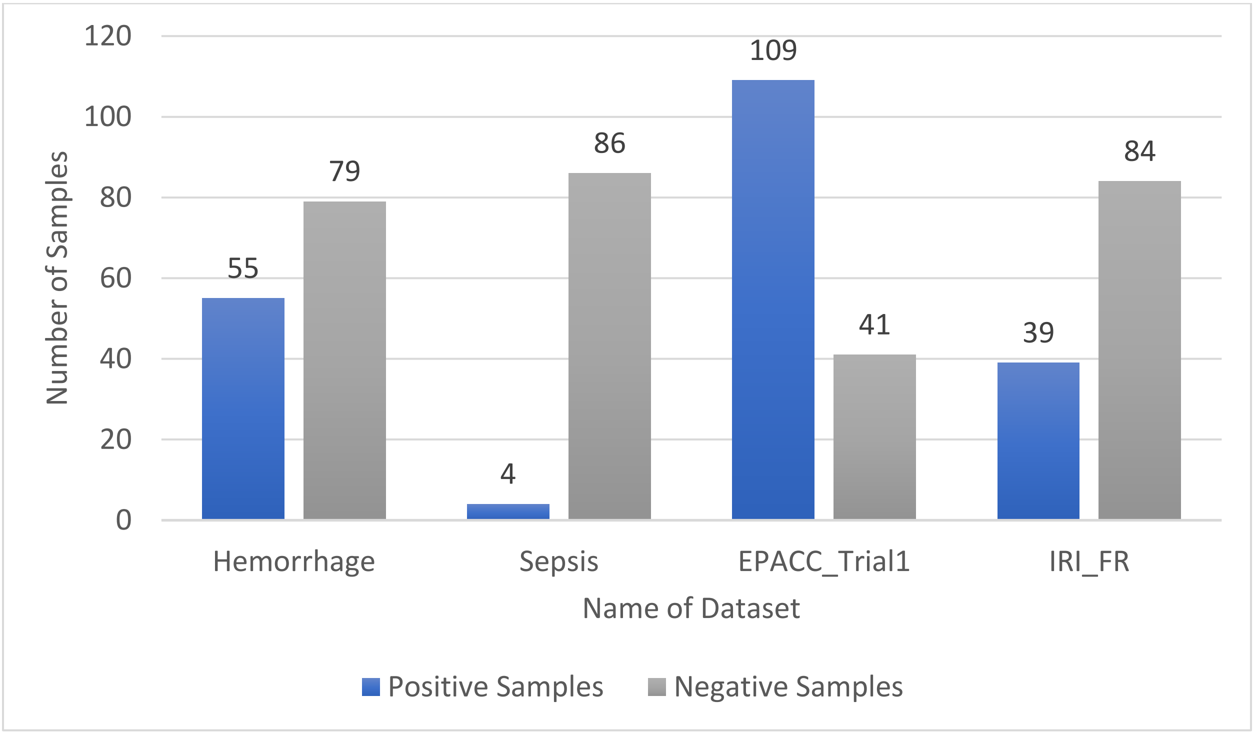

Figure 3.2: Distribution of Positive and Negative Samples for each dataset

Dataset	Number of Pigs	Number of Boluses	Number of Positive Boluses	Number of Negative Boluses
Hem_Sep_scaledCVP	17	224	59	165
Hem_Sep_scaledCVP_EPACC_Trial1	49	374	168	206
Hem	13	134	55	79
EPACC_Trial1	32	150	109	41
IRI_FR_1_2	8	74	23	51
IRI_FR_1_2_3	9	83	24	59
IRI_FR_ALL	13	123	39	84

Table 3.1: Combination of Datasets and their characteristics

3.3. Data Preprocessing

Preprocessing the raw data is an essential step in a Machine/Deep learning model pipeline. Preprocessing the data transforms the input data into a format that the machine can easily interpret. In this thesis, the preprocessing pipeline involved a variety of steps, as displayed in Figure 3.3.

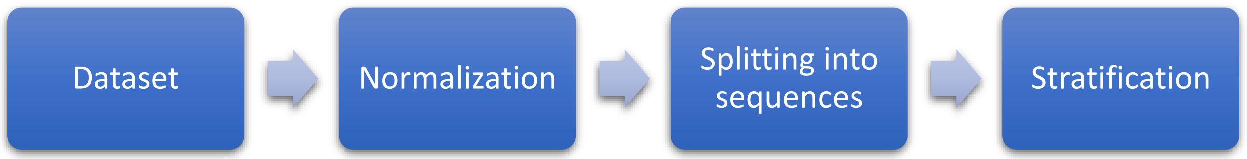

Figure 3.3: Data Preprocessing Pipeline

3.3.1. Normalization

Whenever a deep learning model takes two different variables as input, normalization is an essential preprocessing step to ensure that the two variables have similar distributions and are comparable with one another. Furthermore, it is crucial to perform this preprocessing step to ensure that the deep learning model does not learn that a particular variable may be more or less important than the alternative just because the magnitudes of the variables are different. For example, in the dataset used for this thesis, there was a general trend that the ABP signals tend to have a higher magnitude than their counterpart, the CVP signal. But it is vital to ensure that the ABP signals do not have a higher weightage during training because they have a higher magnitude than the CVP signals. There are a variety of techniques to perform normalization of the raw data.

$$x_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{17}$$

$$x_{norm} = \frac{x - \mu}{\sigma} \tag{18}$$

Equation (17) represents the min-max normalization, while equation (22) shows the Z-score normalization. In equation (17), 'x' indicates the input data to be normalized and '$\min(x)$' and '$\max(x)$' represents the minimum and the maximum value from the input data. In equation (22), 'x' represents the input data while μ and σ represents the population mean and population standard deviation respectively. While both min-max normalization and Z-score normalization are widely used techniques in fields of machine and deep learning, all the experiments in this thesis made use of min-max normalization.

3.3.2. Splitting into Sequences

In general, convolutional neural networks require reshaping of the input data to be of a specific length. For example, one-dimensional convolutional neural networks used in the experiments require the input data to be of the shape [number of samples * timesteps * the number of features]. Thus, the first dimension in the shape vector is the number of samples obtained after splitting the entire data into specific lengths. The second dimension is the sequence length that the time series data has been split into and provided as input to the neural network. Finally, the third dimension is the number of features included, and the value can vary between 1 and 2 depending on whether only one or both ABP and CVP signals were provided as input.

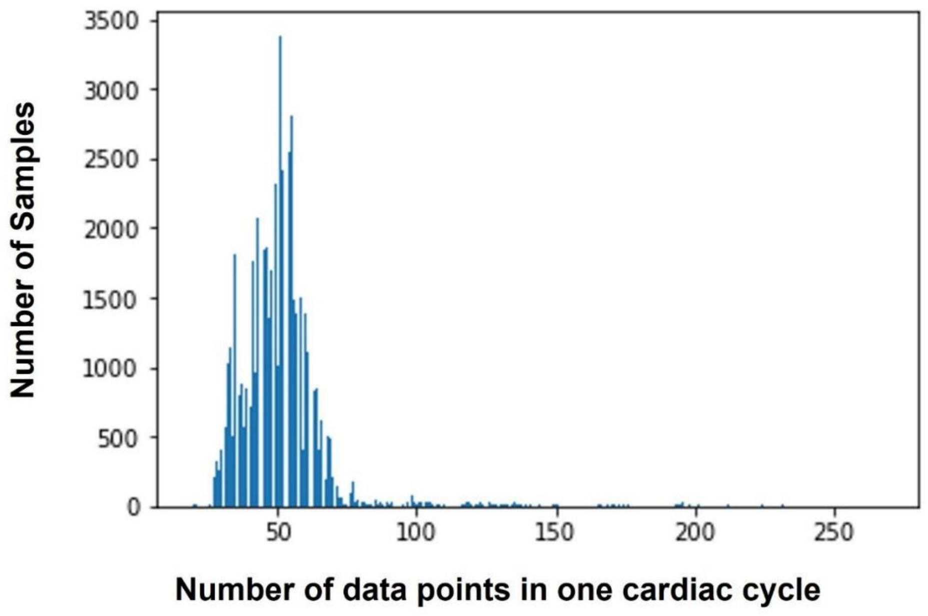

Figure 3.4: Histogram for the number of data points in each cardiac cycle

Figure 3.4 has the histogram for the number of data points in each cardiac cycle. It can be seen that almost every cardiac cycle has the number of data points to be between 25 to 75. For most cases, the sequence length has to be at least of length around 25-75 to cover a single cardiac cycle. While splitting into sequences, each sequence must cover at least a single cardiac cycle so that the sequence has enough information to supply the neural network. For this reason, 56 was the minimum sequence length used in the experiments in this thesis so that the majority of the sequences would have at least one cardiac cycle.

The cardiac cycle has two phases, namely systole and diastole. The systolic phase occurs when the heart pumps out blood by contracting, and the diastolic phase occurs when the heart relaxes. The ABP and CVP waveforms had the systolic and diastolic points marked, and splitting of waveforms into more minor sequences always originated from the diastolic points marked in the waveforms. Figure 3.5 shows the procedure to split the ABP and CVP time-series waveforms into smaller length sequences suitable for input into the deep learning model.

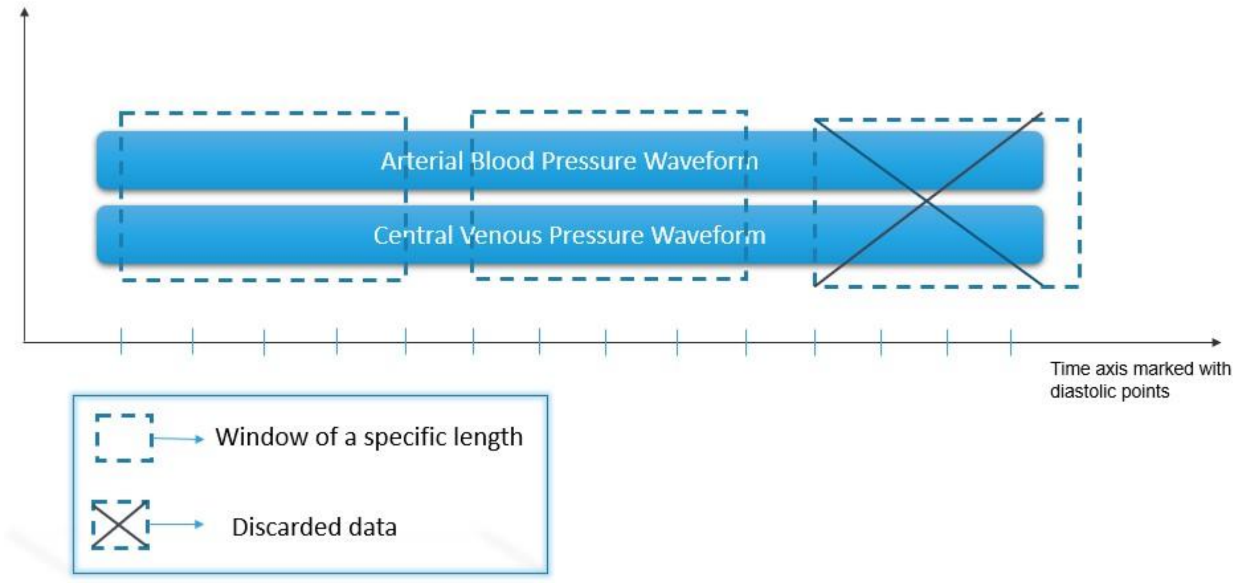
Arterial Blood Pressure Waveform
Central Venous Pressure Waveform
Time axis marked with diastolic points
Window of a specific length
Discarded data

Figure 3.5: Splitting of ABP and CVP waveforms into smaller sequences of a specified length

The splitting of sequences always originated from the first diastolic point in the waveform. Thus, the starting marker to obtain the waveform was placed in the first diastolic point while the ending marker was placed at a previously specified length from the starting marker, and the ending marker may or may not coincide with a diastolic point. Nextly, Obtaining the following sequence involved moving the starting marker to the next diastolic point immediately after the previous marker, and the procedure repeats until the end of the waveforms. If the ending markers go beyond the waveforms, we discarded the sequence starting from the last marker. This splitting the waveforms into more minor one-dimensional sequences resulted in losing about 5% of the raw data. After splitting, each minor sequence belonging to a particular bolus was given the same label as the initial bolus used. The deep learning model takes up these smaller sequences with identical ground truth labels as input.

3.3.3. Stratification

During nested cross-validation, the dataset gets split into several instances of training, test and validation sets. It is crucial to make sure that the process of splitting does not lead to a sampling bias. Sampling bias occurs when the sampling of a stochastic variable does not indicate the distribution of the whole population. Sampling bias can lead to one class of the population being overrepresented or underrepresented and can systematically affect the model's training. During stratification, the population gets split into subgroups, and the data gets randomly sampled so that each subgroup has the same proportion of samples as the original population. Since the current datasets used in the experiments were heavily imbalanced concerning the ground truth labels, it could be possible for the minor class to be underrepresented during random sampling. Stratification solves this problem of underrepresentation of the minor class and indicates an overall outlook of the entire data population in the test set.

3.4. Training and Hyperparameter Tuning

One-dimensional convolutional neural networks have shown encouraging results while working on time-series data. Multivariate LSTM Fully convolutional neural network and DenseNet both exploit this feature of the one-dimensional convolutional neural network to obtain remarkable results in time-series classification, and both of them were used in the experiments. Training the model involves choosing the right set of hyperparameters to maximize the effect of deep neural networks to make good predictions. The following sections explain a systematic approach to building the model by choosing the right set of hyperparameters.

3.4.1. Grid Search

Grid search is one of the traditional methods to tune the hyperparameters for a model. The algorithm makes a complete run over all the values listed for a particular hyperparameter and searches for the best set based on a criterion [46]. During grid search, the hyperparameters were considered independent of each other and were optimized one followed by another [47]. Some hyperparameters are more critical than others, and they require precedence over one another. All the experiments used the grid search algorithm to tune various hyperparameters such as the learning rate, number of epochs, batch size, and the degree of regularization.

3.4.2. Loss Functions

Loss functions are used to find the distances between the ground truth labels and the model's predicted output. There are a variety of loss functions used for various tasks such as classification and regression. One such loss function widely used in the field of multiclass classification is the categorical cross-entropy function. An extension to the default categorical cross-entropy function for an imbalanced dataset is the weighted categorical cross-entropy function. A weighted categorical cross-entropy function introduces sample weighting in the loss function so that the majority and the minority classes get differently weighed while computing the loss. Since the entire dataset used in all the experiments was heavily imbalanced, a weighted categorical cross-entropy function was primarily used as the model’s loss function. Equation (22) represents the weighted categorical cross-entropy function and this is the cost function used in all the experiments.

$$L = -\frac{1}{N}\left[\sum_{i=1}^{N}[W_{c1}t_i \log(p_i) + W_{c2}(1-t_i)\log{(1-p_i)}]\right] \tag{19}$$

In equation (22), W_{c1} and W_{c2} indicate the weights for the classes $c1$ and $c2$, N represents the total number of samples, t_i is the ground truth labels while p_i

represents the softmax probability for the i^{th} sample. There are various ways to compute the sample weight for the weighted loss functions, namely the inverse of the number of samples, the inverse of the square root of number of samples, and the effective number of samples.

Inverse of Number of Samples (INS)

The inverse of the number of samples is a widely used weighting mechanism in the field of deep learning [48]. The inverse of individual class frequencies directs the sample weight, as shown in equation (22).

$$W_{n,c} = \frac{1}{Number\ of\ samples\ in\ class\ c} \tag{20}$$

Inverse of Square Root of Number of Samples (ISNS)

The inverse of the square root of the number of samples was proposed to provide a smoother version of the sample weighting mechanism [49]. As shown in equation (22), it is computed by calculating the inverse of the square root of the individual class frequencies.

$$W_{n,c} = \frac{1}{\sqrt[2]{Number\ of\ samples\ in\ class\ c}} \tag{21}$$

Effective Number of Samples

As a weighting mechanism, the effective number of samples has seen favourable results on datasets like CIFAR and ImageNet [50]. Equation (22) shows how the effective number of samples can be used to provide weights for the loss function.

$$W_{n,c} = \frac{1}{E_{n,c}} \; where, \; E_{n,c} = \frac{1 - \beta^{nc}}{1 - \beta} \tag{22}$$

3.4.3. Optimization

Any deep learning model would require an optimization procedure that aids the model to train by optimizing the cost function. The optimization process would help to minimize or maximize the cost function depending on its use case. Gradient descent is a popular algorithm to optimize dynamic systems like neural networks and a plethora of other machine learning models [51]. One of the disadvantages of gradient descent is that the parameters get updated after an entire run through the data. There were a variety of advances in using smaller batches to run the gradient descent approaches so that there can be multiple runs of the algorithm on the dataset [52]. Momentum upgrades the traditional gradient descent algorithm by using moving averages to update the trainable parameters [53].

Adam improves the stochastic gradient descent algorithm to offer quicker convergence using momentum and adaptive learning rates [54]. Adam was the optimizer primarily used in all the experiments. Learning rate is the hyperparameter that controls the degree to which the network adjusts the weights using the gradient. It is one of the essential hyperparameters to tune in a neural network as having a small value can result in the network learning very slowly, and having a large value can result in the optimization algorithm missing the optima and failing to converge. The learning rate was an important parameter used in the grid search and had values ranging from 1E-6 through 1E-3. Adam also has other parameters called the decay constants (β_1 and β_2), which control the first and second moments of the gradient average, respectively. β_1 and β_2 take up preset values of 0.9 and 0.99, and they were not a part of the grid search. Adam has another parameter called ϵ, which takes up a value of 1E-8 to ensure there is no division by zero and was also not a part of grid search. Decaying learning rates from a considerable initial value to a smaller value has several benefits. A sizeable initial learning rate helps the model look over the noisy data during learning, and decaying the learning rate to smaller values helps the model learn complex patterns and structures from the data [55]. The learning rate decay mechanism requires two important hyperparameters, namely learning rate patience and decay factor. The mechanism decays the learning rate by a decay factor whenever the validation loss starts to increase for a specific number of epochs called learning rate patience. The learning rate patience took a value of 10 epochs while the decay factor took a value of 0.9 during training.

3.4.4. Batch Size

Training in deep learning involves both forward and backward propagation of the entire dataset through the neural network. One complete pass of the entire dataset through the neural network is called an epoch. However, it gets difficult to fit the entire dataset through the neural network as it would consume much memory. Consequently, the data is split into batches of a particular size known as batch size.

It is crucial to tune batch size as having large batch sizes can lead to poor generalization but can guarantee convergence to the global optima. On the other hand, smaller batch sizes can have faster convergence but may not lead to a global optimum and bounce around the global optima [56]. So it is vital to find the optimal batch size to have both fast and guaranteed convergence to the optima. Therefore, the batch size was a critical hyperparameter experimented on and had values varying from 32 to 512 in the grid search.

3.4.5. Number of Training Epochs

Forward and backward propagation of the entire dataset through the neural network is known as an entire epoch. Therefore, having the number of training epochs to be more than 1 indicates that the dataset is run several times through the neural network. Furthermore, having fewer training epochs leads to underfitting, and having a massive number of epochs leads to overfitting. Figure

3.6 displays an example of how the training and validation loss curves would look in the case of overfitting. When the training loss decreases but the validation loss begins to increase, the model is overfitting. However, when the training loss does not begin to reduce, the model is underfitting to the data.

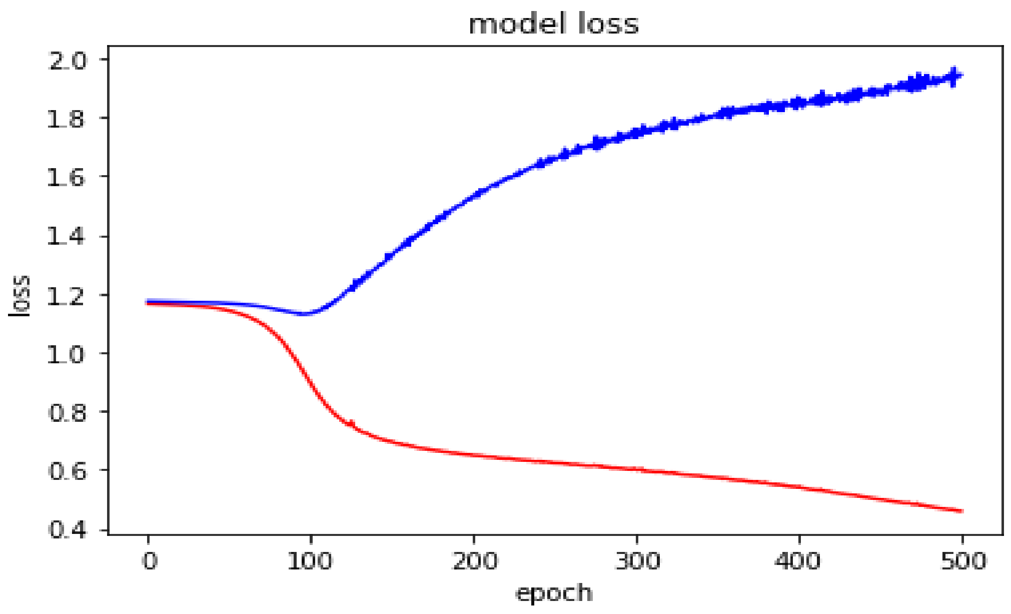

Figure 3.6: Overfitting explained by loss curves. Red: Training loss, Blue: Validation loss

The deep learning models used in the experiments used the early stopping mechanism to ensure the model did not overfit the data. The training process initially takes a considerable random number of epochs and starts training towards it. The validation loss was given as the parameter to monitor for the early stopping mechanism. If the validation loss increases steadily for several epochs, the model immediately stops training, and the current network configuration, including the weights, gets saved. This number is essential to tune as well as having minimal patience for early stopping can make small fluctuations in training to trigger early stopping and having enormous patience would stop the early stopping from triggering. Several values for early stopping patience varying from 100 to 1000 epochs were included in the grid search.

Chapter 4

EXPERIMENTS AND RESULTS

This chapter elaborates on the various experiments conducted and the results associated with those experiments. The experiments were conducted in phases based on the inferences from the subsequent experiment's results. This chapter also explains the inferences made from the results and the rationale for the subsequent experiments. Accuracy, Area under Reciever operating characteristics curve (AUROC), Precision, Recall, and Specificity, along with the respective confidence intervals (CI), were the performance metrics used to evaluate, compare and make inferences about the model's classifying capability.

4.1 Finding Ideal Sequence Length

Since one-dimensional convolutional neural networks take sequences of a specific length as an input, it is crucial to find the right length of the input waveforms so that each waveform input carries enough information to help the neural network make compelling predictions. The Arterial blood pressure and Central venous pressure waveforms from the Hem_Sep dataset, which includes the data obtained from Hemorrhage and Sepsis Pigs, were split into more minor sequences by following Section 3.3.2. Hyperparameters were tuned with a nested cross-validation pipeline along with grid search. 224, 512, and 1024 were the length of sequences used to split the sequences to provide input to the Multivariate LSTM FCN model. Except for the sequence lengths, every aspect of the experiment remained constant to validate the smallest sequence length that could capture the most information.

Sequence Length	**Accuracy**	**CI**	**AUROC**	**CI**	**Precision**	**CI**	**Recall**	**CI**	**Specificity**	**CI**
224	0.863	0.02	0.875	0.02	0.859	0.07	0.731	0.1	0.925	0.04
512	0.858	0.03	0.859	0.03	0.802	0.1	0.715	0.08	0.925	0.03
1024	0.862	0.01	0.851	0.03	0.792	0.09	0.727	0.08	0.92	0.03

Table 4.2: Performance of MLSTMFCN model using ABP waveform and variable sequence lengths in a nested cross-validation

Sequence Length	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
224	0.814	0.02	0.887	0.01	0.776	0.05	0.774	0.07	0.874	0.04
512	0.827	0.04	0.849	0.02	0.676	0.08	0.761	0.09	0.861	0.05
1024	0.812	0.03	0.818	0.04	0.642	0.06	0.757	0.08	0.851	0.04

Table 4.3: Performance of MLSTMFCN model using CVP waveform and variable sequence lengths in a nested cross-validation

Sequence Length	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
224	0.882	0.01	0.877	0.02	0.858	0.07	0.737	0.12	0.947	0.03
512	0.875	0.02	0.884	0.01	0.784	0.06	0.798	0.05	0.914	0.03
1024	0.863	0.02	0.844	0.03	0.792	0.09	0.724	0.12	0.919	0.04

Table 4.4: Performance of MLSTMFCN model using ABP, CVP waveforms, and variable sequence lengths in a nested cross-validation

From Table 4.2 - Table 4.4, it can be inferred that the models trained with sequences of length 224 consistently performed better or similar to the longer sequence inputs. But before concluding that the 224 sequence length inputs were superior, it is critical to note that the experimental conditions changed involuntarily because of the method used to obtain sequences from the original waveforms. Since all the variable length sequences were obtained from the same set of waveforms, the number of samples having smaller sequence lengths was higher when compared to the longer length sequences. For example, the number of samples with a sequence length of 512 is twice the number of samples with a sequence length of 1024. Since the experimental condition had changed, the difference in performance cannot be attributed just to the change in sequence length but can also be due to the change in the number of input samples given for training.

All experimental conditions except for the condition of interest have to remain constant when testing a hypothesis. The experiment required a unique setup since the number of samples is also an essential criterion while maintaining the experimental conditions while evaluating hypotheses. The altered experiment obtained the most extended length sequence and a smaller chunk from it rather than the entire dataset to get smaller sequences. Instead of splitting the entire dataset into sequences of the required length, split the longest sequence (1024, for example) and get the smaller sequence only from the most extended sequence that was already split instead of obtaining from the original dataset, as shown in Figure 4.1. The number of samples was consistent across different sequence lengths, and the remaining aspects of the experiment were similar to the previous setup.

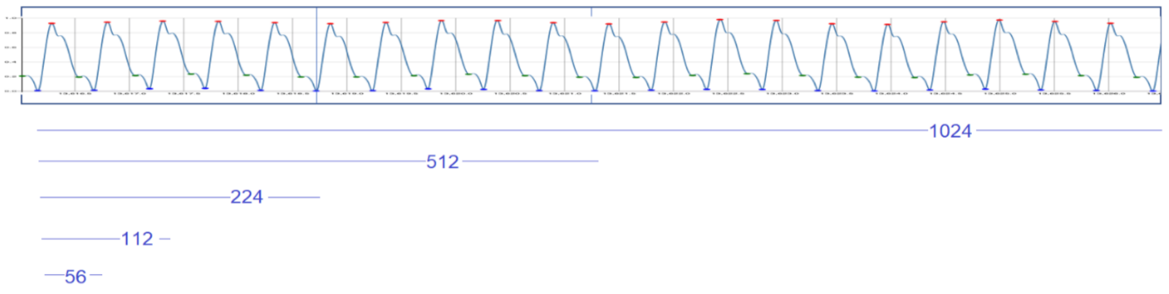

Figure 4.1: Obtaining smaller sequences from the longest sequence for the altered sequence length experiment.

Sequence Length	**Accuracy**	**CI**	**AUROC**	**CI**	**Precision**	**CI**	**Recall**	**CI**	**Specificity**	**CI**
56	0.842	0.02	0.855	0.02	0.768	0.10	0.718	0.07	0.906	0.04
112	0.836	0.04	0.854	0.03	0.745	0.10	0.762	0.09	0.891	0.06
224	0.864	0.02	0.840	0.03	0.775	0.05	0.737	0.09	0.916	0.02
512	0.861	0.01	0.839	0.02	0.775	0.09	0.744	0.09	0.908	0.04
1024	0.862	0.01	0.851	0.03	0.792	0.09	0.727	0.08	0.92	0.03

Table 4.5: Performance of MLSTMFCN model using ABP waveform and altered experiment for variable sequence lengths in a nested cross-validation

Sequence Length	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
56	0.79	0.03	0.851	0.02	0.649	0.07	0.774	0.09	0.838	0.05
112	0.761	0.06	0.839	0.02	0.652	0.11	0.8	0.07	0.798	0.04
224	0.854	0.02	0.843	0.05	0.783	0.1	0.738	0.12	0.91	0.04
512	0.868	0.02	0.835	0.05	0.816	0.09	0.714	0.12	0.928	0.03
1024	0.863	0.02	0.844	0.03	0.792	0.09	0.724	0.12	0.919	0.04

Table 4.6: Performance of MLSTMFCN model using CVP waveform and altered experiment for variable sequence lengths in a nested cross-validation

Sequence Length	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
56	0.867	0.01	0.868	0.05	0.832	0.07	0.746	0.13	0.935	0.03
112	0.825	0.02	0.837	0.04	0.749	0.12	0.71	0.11	0.892	0.05
224	0.854	0.02	0.843	0.05	0.783	0.1	0.738	0.12	0.91	0.04
512	0.868	0.02	0.835	0.05	0.816	0.09	0.714	0.12	0.928	0.03
1024	0.863	0.02	0.844	0.03	0.792	0.09	0.724	0.12	0.919	0.04

Table 4.7: Performance of MLSTMFCN model using ABP and CVP waveforms and altered experiment for variable sequence lengths in a nested cross-validation

The initial hypothesis was that longer-length sequences would perform better than the shorter-length sequences because longer sequences encompass more information than their shorter counterpart. However, from Table 4.5 - Table 4.7, it was evident that the difference in performance while using different length sequences was not statistically different as initially expected. Hence, the hypothesis that a more extended sequence would have better performance can be rejected.

4.2 Model Refinement and Optimization

This series of experiments involves selecting the best model that works well on the test data and generalizes well on the holdout set. Since a nested cross-validation pipeline involves both hyperparameter tuning and model selection, this series of experiments use the nested cross-validation accompanied by a grid search to tune the hyperparameters and select the best model. The experiments included changes in the deep learning architecture, varying the loss functions, and altering the preprocessing pipeline to select the best model. In all these experiments, only one component of the pipeline changes to ensure that these experiments can be compared to find the best model.

4.2.1 Addition of Attention Mechanism to MLSTM-FCN Model

Attention mechanism helps the deep learning model focus on the essential aspects of the data and fade out the less important ones. Therefore, adding an attention mechanism can help the model learn intricate details and make better predictions. In this experiment, the MLSTM-FCN model with and without attention mechanism was used. The hyperparameters were tuned using grid search, and the model was evaluated using nested cross-validation with a 224-length input sequence. The hypothesis is that the addition of an attention mechanism would increase the performance of the model. Apart from having two different models, the remaining aspects of training remained constant throughout the experiment. The experiment utilized all combinations of ABP and CVP as inputs to the model.

With attention	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
No	0.863	0.02	0.875	0.02	0.859	0.07	0.731	0.10	0.925	0.04
Yes	0.853	0.02	0.869	0.03	0.802	0.10	0.694	0.09	0.926	0.03

Table 4.8: Performance of MLSTMFCN model using ABP waveform with and without attention mechanism in a nested cross-validation

With attention	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
No	0.814	0.02	0.887	0.01	0.776	0.05	0.774	0.07	0.874	0.04
Yes	0.83	0.02	0.887	0.02	0.722	0.03	0.792	0.09	0.888	0.01

Table 4.9: Performance of MLSTMFCN model using CVP waveform with and without attention mechanism in a nested cross-validation

With attention	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
No	0.882	0.01	0.877	0.02	0.858	0.07	0.737	0.12	0.947	0.03
Yes	0.842	0.03	0.838	0.02	0.769	0.10	0.668	0.02	0.921	0.04

Table 4.10: Performance of MLSTMFCN model using ABP and CVP waveforms with and without attention mechanism in a nested cross-validation

It can be inferred from Table 4.8 - Table 4.10 that the introduction of the attention mechanism to the MLSTM-FCN model adds no extra value in terms of performance. When ABP and CVP waveforms were used individually, the performance of both the models was comparable, but when both ABP and CVP were used together, the model with attention mechanism performed worse than the model without attention mechanism. Hence, the hypothesis that the attention mechanism would add value in performance can be safely rejected.

4.2.2 Rescaling Central Venous Pressure Waveforms

The Central venous pressure waveforms have been an excellent input source for the neural network to classify boluses to be fluid responsive in all the previous experiments. However, upon examining the CVP waveforms in detail, 2 of the sepsis pigs, namely P2192 and P2187, had a deviation in magnitude when compared to the CVP waveforms of other pigs. This deviation was due to the difference in scaling while obtaining the data for the two pigs. The CVP waveforms of P2192 and P2187 were scaled 5 times and 10 times, respectively. Therefore, the CVP waveforms of both the pigs had to be scaled down to ensure the deviations in scales did not alter the performance. Experimental conditions involved using pig-level splitting to compare the results before and after the CVP waveforms were rescaled. The experiment used the Hem_Sep dataset before and after rescaling the CVP waveforms, and the hyperparameters were tuned using grid search. Both the experiments involved using a sequence length of 224. It was also essential to compare the interaction with both ABP and CVP waveforms and determine if the model trained on the combination of waveforms gets affected upon rescaling.

CVP Rescaling	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
Yes	0.637	0.16	0.79	0.04	0.549	0.13	0.784	0.14	0.694	0.16
No	0.69	0.13	0.817	0.08	0.55	0.11	0.781	0.09	0.742	0.15

Table 4.11: Performance of MLSTMFCN model using CVP waveform with 224 sequence length and pig-level splitting with and without CVP Rescaling in a nested cross-validation

CVP Rescaling	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
Yes	0.577	0.165	0.651	0.092	0.409	0.165	0.708	0.044	0.574	0.207
No	0.569	0.174	0.654	0.102	0.426	0.199	0.738	0.068	0.556	0.232

Table 4.12: Performance of MLSTMFCN model using ABP & CVP waveforms with 224 sequence length and pig-level splitting with and without CVP Rescaling in a nested cross-validation

Table 4.11 and Table 4.12 compare the models' performance using the dataset with and without rescaled CVP waveforms. But the introduction of CVP rescaling in the two sepsis pigs did not alter the model’s performance to be very statistically significant. Hence, whenever the Sepsis pigs were used in future experiments, they were used with the rescaled CVP waveforms.

4.2.3 Removing LSTM layer from MLSTM-FCN Architecture

Long short-term memory usually cause problems when the input space is tiny that they start to memorize the sequences and start overfitting. Even though there were no signs of overfitting during the training of MLSTM-FCN architecture, it was essential to check whether the LSTM layers add value to the model's performance. In this experiment, The LSTM and dropout layers were removed from the MLSTM-FCN architecture to inspect its contribution. This experiment utilized the Hem_Sep_scaledCVP dataset to judge the impact of performance due to the removal of LSTM layers. The setup also had nested cross-validation paired with grid search to tune the right set of hyperparameters. The sequence length used in this experiment was 224. Figure 4.2 represents the model architecture for the MLSTM-FCN model without the LSTM layers.

Features	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
CVP	0.824	0.04	0.901	0.02	0.808	0.07	0.861	0.03	0.826	0.07
ABP	0.842	0.03	0.897	0.03	0.849	0.04	0.825	0.08	0.879	0.04
ABP & CVP	0.861	0.01	0.912	0.01	0.859	0.04	0.86	0.03	0.882	0.03

Table 4.13: Nested Cross-Validation Performance of MLSTM-FCN model without LSTM trained on Hem_Sep_scaledCVP Data

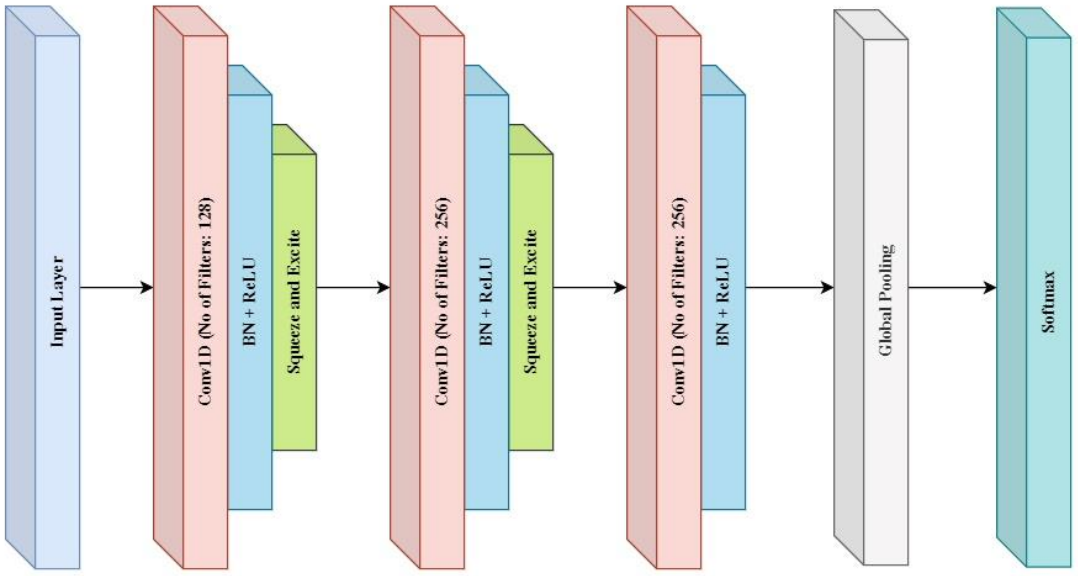

Figure 4.2: MLSTM-FCN without the LSTM layers. Derived from F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate LSTM-FCNs for Time Series Classification," Neural Netw., vol. 116, pp. 237–245, Aug. 2019, doi: 10.1016/j.neunet.2019.04.014.

Table 4.13 showed the nested cross-validation performance when the MLSTM-FCN without LSTM model was trained on Hem_Sep_scaledCVP data. Again, it can be seen that the model's performance is promising, having favourable metric scores during nested cross-validation. This experiment explains that removing LSTM layers improves the performance by a minute amount but not enough to be statistically significant compared to the model that uses LSTM. So, LSTM layers was utilized in all future experiments.

4.2.4 Performance of DenseNet model

This experiment used the DenseNet18 model instead of the MLSTM-FCN model to understand changes in model's performance with changes in the model's architecture. In addition, the experiment utilized the Hem_Sep_ScaledCVP data to understand if the model architecture change can help with generalization. Nested cross-validation in combination with grid search helped in the hyperparameter tuning. Additionally, the experiment used 224 length input sequences. During these experiments, combinations of ABP and CVP waveforms were used to find if they improved the model's performance.

Features	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
CVP	0.784	0.06	0.846	0.04	0.773	0.08	0.829	0.06	0.831	0.06
ABP	0.727	0.03	0.824	0.05	0.714	0.01	0.808	0.08	0.719	0.03
ABP & CVP	0.783	0.01	0.848	0.02	0.781	0.04	0.818	0.06	0.797	0.05

Table 4.14: Nested Cross-Validation Performance of DenseNet 18 model trained on Hem_Sep_scaledCVP Data

The DenseNet 18 model performed well in the nested cross-validation pipeline when using Hem_Sep_ScaledCVP data, as represented in Table 4.14. The DenseNet 18 model understood the patterns from each training set in each fold and performed well on their respective validation sets. Thus, DenseNet does provide a promising deep learning framework to predict fluid responsiveness. However, the DenseNet model's performance is not comparable with the MLSTM-FCN model's performance. Hence, the DenseNet model was not used in future experiments regarding model tuning and selection.

4.2.5 Sample Weighting in Cost Function

Weighted categorical cross-entropy was the cost function used to find the correct weights and biases in the deep learning model in all the previous experiments. The weighted categorical cross-entropy function introduces weight in the regular categorical cross-entropy function, where the weights are usually inversely proportional to the number of samples belonging to a particular category. However, there were a variety of ways in which the sample weighting can be introduced to the cross-entropy function, which includes the inverse of the number of samples (INS), the inverse of the square root of the number of samples (ISNS), and the effective number of samples (ENS). This experiment used all these varieties of sample weighting in the loss function while keeping the remaining aspect of the experiment constant.

The experimental setup followed a nested cross-validation procedure accompanied by a grid search to tune the hyperparameters. The sequence length used was 224. ABP and CVP signals were given together as input to the MLSTM-FCN model during training and testing. The experiment utilized the Hem_Sep_scaledCVP dataset during the nested cross-validation procedure. The experiment's purpose was to find whether any change in the sample weighting would affect the model's performance during nested cross-validation.

Features	**Accuracy**	**CI**	**AUROC**	**CI**	**Precision**	**CI**	**Recall**	**CI**	**Specificity**	**CI**
ABP & CVP	0.837	0.02	0.899	0.02	0.844	0.03	0.836	0.03	0.872	0.02

Table 4.15: Nested Cross-Validation Performance of MLSTM-FCN model using INS sample weighting and trained on Hem_Sep_scaledCVP Data

Features	**Accuracy**	**CI**	**AUROC**	**CI**	**Precision**	**CI**	**Recall**	**CI**	**Specificity**	**CI**
ABP & CVP	0.844	0.02	0.895	0.02	0.854	0.02	0.818	0.05	0.886	0.03

Table 4.16: Nested Cross-Validation Performance of MLSTM-FCN model using ISNS sample weighting and trained on Hem_Sep_scaledCVP Data

Features	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
ABP & CVP	0.83	0.05	0.884	0.05	0.819	0.07	0.842	0.02	0.846	0.06

Table 4.17: Nested Cross-Validation Performance of MLSTM-FCN model using ENS sample weighting and trained on Hem_Sep_scaledCVP Data

Table 4.15, Table 4.16, and Table 4.17 explained the nested cross-validation results when INS, ISNS, and ENS sample weighting mechanisms were used, respectively. Again, the performance in the nested cross-validation procedure was favourable for all three sample weighting mechanisms. Based on the results from Table 4.15 - Table 4.17, the INS sample weighting mechanism was superior in comparison to the others though the difference is not large. In all the experiments involving a weighted categorical cross-entropy function in the future, INS was the primary choice of sample weighting.

4.2.6 Resampling the input dataset using SMOTE

Most datasets used in the experiments had a skewed nature where there were more negative samples than positive samples. As a result, the weighted categorical cross-entropy cost function would rate the positive samples to be more important than the negative samples, and hence the models could obtain high recall scores. But without the presence of sample weighting in the cross-entropy function, the model would not learn the basic patterns from the training set as the cross-entropy function expects the ratio of positive to negative samples to be equal. An introduction of oversampling by SMOTE can resample the dataset so that the number of positive and negative samples is the same. If the number of positive and negative samples are equal, there is no need for sample weighting in the categorical cross-entropy cost function. The sequence length used was 224, and all combinations of ABP and CVP waveforms were given as input to the MLSTM-FCN model.

Features	**Accuracy**	**CI**	**AUROC**	**CI**	**Precision**	**CI**	**Recall**	**CI**	**Specificity**	**CI**
CVP	0.814	0.01	0.897	0.01	0.803	0.03	0.835	0.03	0.832	0.03
ABP	0.824	0.04	0.865	0.03	0.835	0.05	0.776	0.08	0.872	0.05
ABP & CVP	0.84	0.03	0.885	0.03	0.838	0.05	0.825	0.03	0.869	0.04

Table 4.18: Nested Cross-Validation Performance of MLSTM-FCN model using SMOTE and trained on Hem_Sep_scaledCVP Data

Table 4.18 describes the nested cross-validation results with SMOTE used to upsample the minority class on Hem_Sep_scaledCVP data. The performance of the model trained on SMOTE was positive. However, the model trained on the resampled dataset did not offer a reasonable improvement in performance compared to the model utilizing a weighted categorical cross-entropy cost function with an INS sample weighting. Hence, resampling the dataset using SMOTE was not used in future experiments.

4.2.7 Removal of Normalization From Preprocessing Pipeline

Normalization helps keep the inputs on the same scale and make them comparable while making predictions. For example, the min-max normalization used in all the experiments brings down the scale of the ABP and CVP waveforms between 0 and 1. Apart from making the inputs comparable, it also helps solve the gradient explosion problem common in deep learning models. In this experiment, the normalization procedure was removed from the preprocessing pipeline to determine if the normalization introduced any dependencies that obstruct the model from generalizing on other physiologies. This experiment followed the nested cross-validation pipeline along with grid search to tune the hyperparameters. The input sequences' length was 224, and all combinations of ABP and CVP waveforms were given as input to the MLSTM-FCN model for training.

Features	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
CVP	0.843	0.04	0.881	0.02	0.773	0.06	0.814	0.04	0.863	0.06
ABP	0.860	0.03	0.865	0.04	0.811	0.11	0.693	0.10	0.928	0.04
ABP & CVP	0.875	0.03	0.881	0.03	0.859	0.05	0.7975	0.08	0.918	0.04

Table 4.19: Nested Cross-Validation Performance of MLSTM-FCN model without normalizing and trained on Hem_Sep_scaledCVP Data

Table 4.19 displayed the nested cross-validation results when the MLSTM-FCN model took in ABP and CVP waveforms without normalizing. The model performed well even with the lack of normalization during preprocessing. However, the model's performance did not improve compared to the model trained on the usual normalization pipeline; hence, normalization was used as part of the preprocessing pipeline in future experiments.

4.2.8 Altering normalization pipeline

During the previous experiment, the normalization pipeline was removed from preprocessing. But when the inputs were normalized, the input ABP and CVP were normalized separately from each other. Additionally, various datasets were kept separate while normalizing. In this experiment, all the datasets were normalized together and then separated into their original cohort. The experiment aimed to see if this altered way of normalization supports the model's performance positively. The experimental setup had nested cross-validation and grid search to aid in hyperparameter tuning. All combinations of ABP and CVP waveforms were given as inputs.

Features	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
CVP	0.814	0.04	0.890	0.02	0.806	0.05	0.841	0.06	0.823	0.06
ABP	0.812	0.04	0.857	0.04	0.823	0.06	0.765	0.05	0.86	0.05
ABP & CVP	0.832	0.05	0.867	0.04	0.841	0.05	0.821	0.05	0.825	0.06

Table 4.20: Nested Cross-Validation Performance of MLSTM-FCN model with everything normalized together and trained on Hem_Sep_scaledCVP_EPACC_Trial1 Data

Table 4.20 clearly explains the nested cross-validation results when the boluses from hemorrhage, sepsis, EPACC_Trial1, and IRI_FR_ALL were normalized together and separated to only include the hemorrhage, sepsis, and the EPACC_Trial1 for training. The model trained with the altered normalization pipeline showed promising results, but the model did not show a statistically significant performance improvement compared to the MLSTM-FCN model trained with the regular normalization pipeline. Hence, the traditional form of normalization was a part of the preprocessing pipeline in all future experiments.

This series of experiments helped identify the best model that is expected to generalize well on the holdout data. In the series of experiments involving identifying the best model, performances of MLSTM-FCN, MLSTM-FCN and DenseNet were all comparable. So in future experiments, MLSTM-FCN was the primary model used since the other models did not significantly improve the model's performance. While using different sample weighting in the cost function, the INS weighting mechanism generated the best results in nested cross-validation, although the difference while using the other mechanisms was not statistically significant. However, future experiments made use of the INS weighting mechanism for the cost function. Additionally, resampling the dataset using SMOTE did not significantly improve the model’s performance compared to the model using the INS weighting mechanism for the cost function. Hence, SMOTE was not used in future experiments to resample the dataset. Finally, removal of normalization and altering the normalization pipeline did not significantly improve the model’s performance; hence, individual normalization as mentioned in the preprocessing pipeline was used in future experiments. Following this series of experiments, the best model with the ideal preprocessing pipeline was chosen, and the setup of this experiment was continued in future experiments.

4.3 Pig-Level and Bolus-Level Experiment

Pig-level and Bolus-level experiment involves doing different splits for the training and the test sets during nested cross-validation. The bolus-level and pig-level experiments utilized the Hem_Sep dataset. During a bolus-level split, the training and the test splits were stratified concerning the number of positive and negative boluses. Therefore, even though the training and the test boluses were chosen at random, they had similar distribution because of stratification.

On the contrary, a pig-level split did not have a stratified split of positive and negative boluses. The pig-level splits had the boluses from entire pigs in each set, which meant that there were no boluses from an individual pig from the training set to the test set or vice versa. The Pig-level split involved choosing all the boluses from one sepsis pig and three hemorrhage pigs for its test set, and the boluses of the remaining pigs were a part of the training set. The inferences from this experiment would help understand whether the model can generalize well to totally unseen data even if the unseen boluses came from unseen pigs. The experiment utilized grid search in a nested cross-validation pipeline to choose the right set of hyperparameters. From section 4.1, it was evident that both 56 and 224 length sequences had pretty good performance concerning the relevant metrics, and hence both the sequence lengths were also tested if they offered any difference in the current experiment.

Sequence Length	BL/PL	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
56	PL	0.541	0.19	0.649	0.14	0.411	0.18	0.668	0.14	0.553	0.25
56	BL	0.842	0.02	0.855	0.02	0.768	0.10	0.718	0.07	0.906	0.04
224	PL	0.773	0.06	0.776	0.05	0.606	0.13	0.65	0.14	0.833	0.08
224	BL	0.863	0.02	0.875	0.02	0.859	0.07	0.731	0.10	0.925	0.04

Table 4.21: Performance of MLSTMFCN model using ABP waveform with 56 and 224 sequence lengths for bolus-level (BL) and pig-level (PL) splitting in a nested cross-validation

Sequence Length	BL/PL	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
56	PL	0.668	0.052	0.801	0.065	0.481	0.099	0.765	0.108	0.712	0.066
56	BL	0.789	0.03	0.851	0.028	0.649	0.073	0.774	0.091	0.838	0.059
224	PL	0.69	0.13	0.817	0.089	0.55	0.119	0.781	0.09	0.742	0.158
224	BL	0.814	0.024	0.887	0.017	0.776	0.053	0.774	0.074	0.874	0.048

Table 4.22: Performance of MLSTMFCN model using CVP waveform with 56 and 224 sequence lengths for bolus-level (BL) and pig-level (PL) splitting in a nested cross-validation

Sequence Length	BL/PL	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
56	PL	0.532	0.166	0.605	0.154	0.382	0.161	0.671	0.154	0.549	0.219
56	BL	0.867	0.007	0.868	0.053	0.832	0.072	0.746	0.131	0.935	0.032
224	PL	0.569	0.174	0.654	0.102	0.426	0.199	0.738	0.068	0.556	0.232
224	BL	0.882	0.014	0.877	0.029	0.858	0.0789	0.737	0.128	0.947	0.031

Table 4.23: Performance of MLSTMFCN model using ABP and CVP waveforms with 56 and 224 sequence lengths for bolus-level (BL) and pig-level (PL) splitting in a nested cross-validation

It can be seen from Table 4.21 and Table 4.23 that there was a steady drop in performance when the experiment switched from bolus-level to pig-level splitting. Additionally, the models trained on sequences of length 56 had a much steeper drop in performance than those trained on sequences of length 224. However, from Table 4.22, there was no significant drop in performance while using just the CVP waveforms when the splitting switched from bolus-level to pig-level splitting. Additionally, the performances with models trained on length 56 were comparable to those trained on sequences of length 224. Therefore, from Table 4.21 - Table 4.23, it can be inferred that whenever ABP signals were supplied as input the neural network, there was a significant drop in the performance in pig-level splits, but the same does not apply for using CVP signals based on results from Table 4.22. Another inference from these experiments was that the sequence length of 56 consistently performed worse when compared to models trained with sequences of length 224; however, not by a significant amount.

Even though there was a significant drop in the performance when shifting from bolus level to pig level splitting, the drop was not huge when the ABP and CVP waveforms were used individually. The results from Table 4.21 - Table 4.23 motivated further experiments on pig-level sampling by including all the datasets, including hemorrhage, sepsis, EPACC_Trial1, and IRI_FR_ALL. The experimental setup was similar to the previous pig-level experiments, and the only difference to the method was to include all the datasets and not just the hemorrhage and sepsis datasets like before.

Fold	Accuracy	AUROC	Precision	Recall	Specificity
1	0.695	0.755	0.772	0.62	0.792
2	0.778	0.819	0.827	0.735	0.84
3	0.556	0.573	0.567	0.531	0.602
4	0.655	0.711	0.607	0.681	0.677
5	0.704	0.731	0.683	0.773	0.654

Table 4.24: Performance of MLSTMFCN model using ABP waveform with 224 sequence length for pig-level (PL) splitting in a five-fold nested cross-validation

Accuracy	**CI**	**AUROC**	**CI**	**Precision**	**CI**	**Recall**	**CI**	**Specificity**	**CI**
0.678	0.06	0.718	0.07	0.692	0.08	0.668	0.07	0.713	0.07

Table 4.25: Summary of metrics from Table 4.24 with averages and 95% confidence intervals

Fold	**Accuracy**	**AUROC**	**Precision**	**Recall**	**Specificity**
1	0.599	0.673	0.651	0.543	0.673
2	0.684	0.722	0.701	0.692	0.692
3	0.647	0.708	0.630	0.726	0.584
4	0.572	0.628	0.499	0.508	0.63
5	0.698	0.792	0.706	0.753	0.694

Table 4.26: Performance of MLSTMFCN model using CVP waveform with 224 sequence length for pig-level (PL) splitting in a five-fold nested cross-validation

Accuracy	**CI**	**AUROC**	**CI**	**Precision**	**CI**	**Recall**	**CI**	**Specificity**	**CI**
0.639	0.04	0.704	0.04	0.637	0.06	0.644	0.08	0.655	0.03

Table 4.27: Summary of metrics from Table 4.26 with averages and 95% confidence intervals

Fold	Accuracy	AUROC	Precision	Recall	Specificity
1	0.599	0.673	0.651	0.543	0.673
2	0.684	0.722	0.701	0.692	0.692
3	0.647	0.708	0.630	0.726	0.584
4	0.572	0.628	0.499	0.508	0.63
5	0.698	0.792	0.706	0.753	0.694

Table 4.28: Performance of MLSTMFCN model using ABP & CVP waveforms with 224 sequence length for pig-level (PL) splitting in a five-fold nested cross-validation

Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
0.639	0.04	0.704	0.04	0.637	0.06	0.644	0.08	0.655	0.03

Table 4.29: Summary of metrics from Table 4.28 with averages and 95% confidence intervals

Table 4.24 - Table 4.29 indicates that the models developed with the pig-level splitting had established promising results even when all the datasets were included. Comparing the results with pig-level splitting using just the hemorrhage and sepsis pigs and the entire dataset's performance shows a slight drop in the performance metrics. However, the drop in performance is not huge and shows promise that the training set need not necessarily have the boluses from the same pigs to offer compelling performance. The results suggest that if the training and the test set have representations of the same physiologies and not necessarily

from the same pigs, the model can better classify boluses as fluid responsive or not.

4.4 Performance on Combined Physiologies of Models Trained on Individual Physiologies

One of the most critical aspects of these experiments is whether a model trained on individual physiologies can generalize on totally different physiologies. Thus, for example, it would be vital to know whether the individual physiologies play an essential role in the classification or whether these nuances can be learned by the model and can generalize across different physiologies. Therefore, in these experiments, MLSTM-FCN models were trained on datasets possessing individual physiologies such as Hemorrhage, Sepsis, and EPACC_Trial1. Then, the trained models were tested on the remaining physiologies to understand the degree of generalization across physiologies. The experimental setup involved grid search in tuning the right set of hyperparameters. Nested cross-validation was not a part of the setup, but the experiment setup involved using separate training and test sets from individual physiologies instead. The length of the input sequence used was 224. Training and testing the models involved using the ABP and CVP waveforms individually and in combination to understand if these combinations add value to the model's performance.

Features	Accuracy	AUROC	Precision	Recall	Specificity
ABP	0.887	0.814	0.25	0.75	0.894
CVP	0.966	0.744	0.82	0.25	0.921
ABP & CVP	0.854	0.798	0.2	0.75	0.856

Table 4.30: Performance of MLSTMFCN model trained on boluses from EPACC_Trial1 pigs with 224 sequence length and tested on boluses from sepsis pigs

Features	Accuracy	AUROC	Precision	Recall	Specificity
ABP	0.654	0.574	0.9	0.166	0.987
CVP	0.518	0.668	0.483	0.815	0.405
ABP & CVP	0.631	0.559	0.87	0.093	0.953

Table 4.31: Performance of MLSTMFCN model trained on boluses from EPACC_Trial1 pigs with 224 sequence length and tested on boluses from Hemorrhage pigs

Table 4.30 and Table 4.31 explain the performance of the model trained on EPACC_trial1 pigs on boluses of Sepsis and Hemorrhage pigs, respectively. From Table 4.30, it is evident that the model's performance on sepsis pigs was positive concerning the accuracy, AUROC, and specificity. However, the precision and recall metrics tend to be scattered. Table 4.31 had the performance metrics when the same model was tested on boluses from hemorrhage pigs. Collectively all of them had poor accuracy and AUROC scores, and the precision, recall, and specificity metrics were scattered, indicating poor generalization from EPACC_Trial1 to Hemorrhage.

Features	Accuracy	AUROC	Precision	Recall	Specificity
ABP	0.966	0.747	0.926	0.25	0.984
CVP	0.955	0.576	0.487	0.594	0.962
ABP & CVP	0.797	0.758	0.158	0.75	0.811

Table 4.32: Performance of MLSTMFCN model trained on boluses from Hemorrhage pigs with 224 sequence length and tested on boluses from sepsis pigs

Features	Accuracy	AUROC	Precision	Recall	Specificity
ABP	0.725	0.512	0.725	0.948	0.332
CVP	0.517	0.553	0.736	0.62	0.414
ABP & CVP	0.745	0.601	0.767	0.944	0.244

Table 4.33: Performance of MLSTMFCN model trained on boluses from Hemorrhage pigs with 224 sequence length and tested on boluses from EPACC_Trial1 pigs

Table 4.32 and Table 4.33 explain the performance of the model trained on Hemorrhage pigs on boluses of Sepsis and EPACC_Trial1 pigs, respectively. From Table 4.32, it is clear that the model's performance concerning accuracy and specificity was good. However, AUROC was bad when only the CVP waveform was used but was better in the remaining combinations. Additionally, the precision and the recall scores were also scattered, indicating poor generalization on Sepsis data. Table 4.33 had the performance metrics when the same model was tested on boluses from EPACC_Trial1 pigs. The precision and the recall scores were considerably higher, but the remaining metrics had poor scores again, indicating poor generalization on EPACC_Trial1 data.

Features	**Accuracy**	**AUROC**	**Precision**	**Recall**	**Specificity**
ABP	0.406	0.532	0.406	0.978	0.253
CVP	0.594	0.527	0.358	0.297	0.96
ABP & CVP	0.503	0.61	0.46	0.852	0.316

Table 4.34: Performance of MLSTMFCN model trained on boluses from Sepsis pigs with 224 sequence length and tested on boluses from Hemorrhage pigs

Features	Accuracy	AUROC	Precision	Recall	Specificity
ABP	0.664	0.577	0.738	0.889	0.17
CVP	0.678	0.507	0.726	0.907	0.097
ABP & CVP	0.738	0.523	0.734	0.912	0.048

Table 4.35: Performance of MLSTMFCN model trained on boluses from Sepsis pigs with 224 sequence length and tested on boluses from Hemorrhage pigs

Table 4.34 and Table 4.35 explain the performance of the model trained on Sepsis pigs on boluses of Hemorrhage and EPACC_Trial1 pigs, respectively. The model trained on boluses from Sepsis pigs had inferior generalization on the Hemorrhage and EPACC_Trial1 pigs. From these experiments, it can be inferred that the models trained on individual physiologies tend not to generalize well across other physiologies. The model could not generalize to other physiologies by learning the patterns and structure from different physiology.

4.5 Evaluation of Model’s Generalizability

One of the essential requirements of a predictive model is to predict well on the unseen holdout data to ensure the model’s generalizability. In this set of experiments, various combinations of physiologies were used to train the model using nested cross-validation to tune the hyperparameters and test all these models on holdout sets to find out which combination of physiologies helps the model generalize well on the unseen data. Another critical experiment was to reverse the training and the holdout sets to identify if the model behaves similarly if the data gets swapped. Each of these experiments was designed to identify if the model can generalize well on the unseen data.

4.5.1 Holdout Set Performance of Models Trained on Various Combinations of Physiologies

Following the previous experiment where the model utilized the individual physiologies for training and then testing the others, this experiment followed a similar pattern. However, instead of using individual physiologies to train and then test on other physiologies, this experiment made use of a constant holdout set made out of IRI_FR data. Thus, this experiment involved training models on individual and combination of physiologies and then testing out on the same holdout set. The experimental setup involved a bolus-level split for various folds in the nested cross-validation process. It also involved the usage of grid search in searching for the correct hyperparameters. The sequence length used in this experiment was 224. The IRI_FR dataset had three different subsets used in this thesis. However, this experiment made use of two of those subsets, namely

IRI_FR_1_2 and IRI_FR_1_2_3. These datasets had overlaps; however, the IRI_FR_1_2_3 had boluses from 5 more pigs than the IRI_FR_1_2 dataset. The description and the characteristics of each dataset used in this experiment are present in Table 3.1. This experiment also tested various combinations of ABP and CVP waveforms and if they add value to the performance.

Features	**Accuracy**	**CI**	**AUROC**	**CI**	**Precision**	**CI**	**Recall**	**CI**	**Specificity**	**CI**
CVP	0.799	0.07	0.866	0.07	0.815	0.08	0.723	0.07	0.877	0.07
ABP	0.848	0.04	0.873	0.03	0.869	0.07	0.776	0.07	0.906	0.06
ABP & CVP	0.839	0.05	0.88	0.03	0.831	0.08	0.837	0.06	0.865	0.07

Table 4.36: Nested Cross-Validation Performance of MLSTM-FCN model trained on Hem Data

Features	**Accuracy**	**CI**	**AUROC**	**CI**	**Precision**	**CI**	**Recall**	**CI**	**Specificity**	**CI**
CVP	0.568	0.06	0.64	0.04	0.426	0.03	0.64	0.08	0.645	0.04
ABP	0.334	0.01	0.43	0.03	0.26	0.01	0.691	0.06	0.196	0.03
ABP & CVP	0.326	0.03	0.358	0.04	0.201	0.04	0.391	0.08	0.374	0.04

Table 4.37: Performance of Pickled model trained on Hem Data and tested on holdout IRI_FR_1_2 data.

Features	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
CVP	0.591	0.06	0.667	0.04	0.465	0.02	0.654	0.09	0.678	0.03
ABP	0.334	0.02	0.439	0.04	0.279	0.02	0.745	0.07	0.176	0.03
ABP & CVP	0.363	0.03	0.373	0.03	0.216	0.03	0.418	0.08	0.368	0.06

Table 4.38: Performance of Pickled model trained on Hem Data and tested on holdout IRI_FR_1_2_3 data.

Features	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
CVP	0.772	0.05	0.787	0.08	0.856	0.05	0.866	0.04	0.737	0.08
ABP	0.842	0.05	0.842	0.06	0.886	0.05	0.915	0.04	0.721	0.07
ABP & CVP	0.827	0.05	0.845	0.04	0.883	0.04	0.941	0.06	0.742	0.07

Table 4.39: Nested Cross-Validation Performance of MLSTM-FCN model trained on EPACC_Trial1 Data

Features	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
CVP	0.653	0.02	0.604	0.05	0.401	0.04	0.51	0.06	0.767	0.06
ABP	0.395	0.05	0.546	0.04	0.33	0.03	0.783	0.01	0.33	0.08
ABP & CVP	0.736	0.03	0.808	0.01	0.552	0.01	0.883	0.06	0.703	0.03

Table 4.40: Performance of Pickled model trained on EPACC_Trial1 Data and tested on holdout IRI_FR_1_2 data.

Features	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI

CVP	0.657	0.01	0.648	0.02	0.416	0.04	0.522	0.04	0.764	0.06
ABP	0.405	0.05	0.547	0.03	0.339	0.02	0.782	0.01	0.337	0.07
ABP & CVP	0.704	0.04	0.781	0.01	0.527	0.02	0.872	0.07	0.663	0.03

Table 4.41: Performance of Pickled model trained on EPACC_Trial1 Data and tested on holdout IRI_FR_1_2_3 data.

Features	**Accuracy**	**CI**	**AUROC**	**CI**	**Precision**	**CI**	**Recall**	**CI**	**Specificity**	**CI**
CVP	0.83	0.02	0.887	0.02	0.722	0.03	0.792	0.09	0.888	0.01
ABP	0.853	0.02	0.869	0.03	0.802	0.1	0.694	0.09	0.926	0.03
ABP & CVP	0.842	0.03	0.838	0.02	0.769	0.1	0.668	0.02	0.921	0.04

Table 4.42: Nested Cross-Validation Performance of MLSTM-FCN model trained on Hem_Sep_scaledCVP Data

Features	**Accuracy**	**CI**	**AUROC**	**CI**	**Precision**	**CI**	**Recall**	**CI**	**Specificity**	**CI**
CVP	0.419	0.07	0.44	0.07	0.285	0.04	0.5	0.08	0.493	0.06
ABP	0.495	0.03	0.645	0.08	0.396	0.06	0.833	0.08	0.471	0.07
ABP & CVP	0.427	0.05	0.573	0.03	0.345	0.01	0.783	0.09	0.386	0.06

Table 4.43: Performance of Pickled model trained on Hem_Sep_scaledCVP Data and tested on holdout IRI_FR_1_2 data.

Features	**Accuracy**	**CI**	**AUROC**	**CI**	**Precision**	**CI**	**Recall**	**CI**	**Specificity**	**CI**

CVP	0.435	0.08	0.445	0.07	0.299	0.05	0.49	0.08	0.509	0.07
ABP	0.311	0.01	0.519	0.01	0.309	0.01	0.933	0.05	0.402	0.07
ABP & CVP	0.431	0.05	0.562	0.04	0.332	0.01	0.786	0.08	0.355	0.06

Table 4.44: Performance of Pickled model trained on Hem_Sep_scaledCVP Data and tested on holdout IRI_FR_1_2_3 data.

Features	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
CVP	0.806	0.03	0.882	0.02	0.79	0.04	0.837	0.05	0.81	0.05
ABP	0.824	0.04	0.874	0.04	0.829	0.05	0.801	0.06	0.86	0.05
ABP & CVP	0.829	0.04	0.876	0.04	0.826	0.07	0.836	0.05	0.843	0.07

Table 4.45: Nested Cross-Validation Performance of MLSTM-FCN model trained on Hem_Sep_scaledCVP_EPACC_Trial1 Data

Features	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
CVP	0.427	0.03	0.497	0.08	0.3	0.05	0.736	0.08	0.476	0.07
ABP	0.544	0.03	0.681	0.03	0.41	0.01	0.944	0.02	0.483	0.06
ABP & CVP	0.404	0.04	0.569	0.03	0.344	0.02	0.783	0.04	0.379	0.05

Table 4.46: Performance of Pickled model trained on Hem_Sep_scaledCVP_EPACC_Trial1 Data and tested on holdout IRI_FR_1_2 data.

Features	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
CVP	0.46	0.03	0.523	0.08	0.321	0.05	0.77	0.07	0.486	0.08
ABP	0.534	0.02	0.668	0.03	0.408	0.01	0.943	0.02	0.458	0.05
ABP & CVP	0.422	0.05	0.579	0.03	0.361	0.02	0.791	0.05	0.396	0.03

Table 4.47: Performance of Pickled model trained on Hem_Sep_scaledCVP_EPACC_Trial1 Data and tested on holdout IRI_FR_1_2_3 data.

Table 4.36-Table 4.38 explains the nested cross-validation and the pickled model performances on the holdout sets when trained on boluses from hemorrhage pigs. Table 4.39-Table 4.41 elaborates on the nested cross-validation and pickled model performance on the holdout sets when the model was trained on EPACC_Trial1 data. Table 4.42-Table 4.44 elaborated on the nested cross-validation and pickled model performance on holdout sets when trained on boluses from Hem_Sep_scaledCVP data. Finally, Table 4.45-Table 4.47 explained the nested cross-validation and pickled model performance on the holdout sets when trained on Hem_Sep_scaledCVP_EPACC_Trial1 data. From Table 4.36, Table 4.39, Table 4.42, and Table 4.45, it was evident that the MLSTM-FCN model could generalize well on nested cross-validation. An additional inference was that the model tends to perform well when the test set resembles the training set well concerning the physiologies and the distribution. Table 4.37: Performance of Pickled model trained on Hem Data and tested on holdout IRI_FR_1_2 data., Table 4.38, Table 4.40: Performance of Pickled model trained on EPACC_Trial1 Data and tested on holdout IRI_FR_1_2 data., Table 4.41, Table 4.43, Table 4.44, Table 4.46: Performance of Pickled model trained on Hem_Sep_scaledCVP_EPACC_Trial1 Data and tested on holdout IRI_FR_1_2 data.

, and Table 4.47 represent the pickled model performance of the respective models in each category. There was a decrease in performance while testing on the holdout set compared to the nested cross-validation performance. The training set in all the cases did not have even a single bolus from the IRI_FR data, and this explained why the pickled model trained on various physiologies did not perform well on the holdout set but performed well during nested cross-validation.

4.5.2 Reversing the training and holdout sets

Inferences from the previous experiments pointed out that the models trained on any combination of physiologies did not learn the patterns well to generalize well on the holdout set. Therefore, in this experiment, the training and the holdout sets were reversed to check if the model can generalize well when trained on

IRI_FR_1_2_3 data and tested on Hem_Sep_scaledCVP data. The experimental setup had the nested cross-validation pipeline along with the grid search for hyperparameter tuning. The MLSTM-FCN model took in inputs of length 224. All combinations of ABP and CVP waveforms were given as input to the model for training and testing. After the nested cross-validation procedure, the model was pickled and then tested on the Hem_Sep_scaledCVP data to find out if there was any impact on the performance.

Features	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
CVP	0.707	0.06	0.703	0.07	0.574	0.13	0.553	0.15	0.819	0.12
ABP	0.774	0.12	0.739	0.11	0.608	0.14	0.583	0.18	0.848	0.12
ABP & CVP	0.764	0.11	0.748	0.10	0.593	0.14	0.703	0.08	0.828	0.10

Table 4.48: Nested Cross-Validation Performance of MLSTM-FCN model trained on IRI_FR_1_2_3 Data

Features	Accuracy	CI	AUROC	CI	Precision	CI	Recall	CI	Specificity	CI
CVP	0.64	0.02	0.579	0.02	0.326	0.04	0.344	0.06	0.751	0.02
ABP	0.524	0.02	0.471	0.04	0.227	0.03	0.310	0.09	0.630	0.02
ABP & CVP	0.780	0.03	0.626	0.07	0.896	0.09	0.201	0.13	0.987	0.01

Table 4.49: Performance of Pickled model with trained on IRI_FR_1_2_3 Data and tested on holdout Hem_Sep_scaledCVP data.

Observations from Table 4.49 pointed out that the pickled model did not generalize well on the holdout set made from the boluses of Hem_Sep_scaledCVP data. The poor generalization of the pickled model was similar to the results from previous experiments as well. But a significant difference was that the nested cross-validation performance in the previous experiments was better. On the contrary, the nested cross-validation results were subpar when the IRI_FR_1_2_3 was used for training based on results displayed in Table 4.49. The poor performance could be due to the fewer boluses in the IRI_FR_1_2_3 data than the Hem_Sep_scaledCVP data. Due to the decreased number of boluses available for training, the model could not understand the patterns from the training, and the holdout set resulted in poor generalization.

This series of experiments helped in discerning that though the model performed well during nested cross-validation with various combinations of physiologies, the model failed to perform well on the holdout set, validating the lack of generalizability of the model. In the subsequent experiment where the training and the holdout sets were reversed, the model failed to get a comparable score during nested cross-validation and failed to generalize well on the holdout set. But in this experiment, the failure to generalize well on the training set can be explained due to the lack of enough data points for the model to train, explained by the low nested cross-validation score.

4.6 Examination of Overfitting

In the previous set of experiments, the model's performance on the training set was better when compared to the performance on the holdout sets. The lack of generalizability of the model on the holdout sets could result from the overfitting on the training set. As explained in Section 3.4.5, overfitting can be examined with the help of the loss curves during training. The loss curves can be examined multiple times from each fold of the nested cross-validation pipeline since each fold has a different random sample of training and validation sets. The loss curves indicated no signs of overfitting in all the experiments during nested cross-validation. Figure 4.3 shows the training and validation loss curves during one of the folds of nested cross validation for the model trained on Hem_Sep_ScaledCVP_EPACC_Trial1 data with 224 sequence length and on a bolus level.

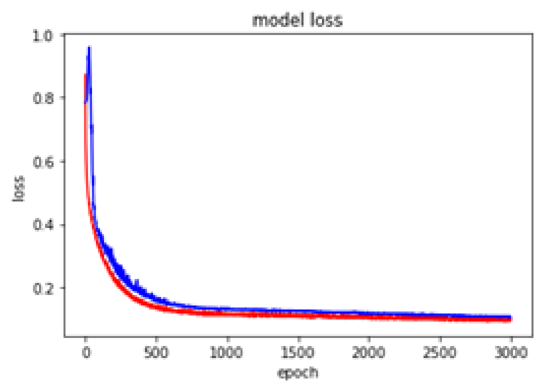

Figure 4.3: Representative training and validation loss curves for one of the folds during nested cross-validation for the model trained on Hem_Sep_ScaledCVP_EPACC_Trial1 data with 224 sequence length and on a bolus level. Red: Training loss, Blue: Validation loss

The examination of loss curves during each nested cross-validation pipeline helped ensure that the model did not overfit on the training data. However, the training and the validation sets in a nested cross-validation pipeline have a similar distribution of samples obtained from a given injury model. Good performance metrics during nested cross-validation with the lack of overfitting suggested that the models were learning generalizable rules within a given injury model; however, the lack of generalization on the holdout set suggest that the models were so good at learning a given physiology that they could not generalize to a different injury model's physiology.

Chapter 5

CONCLUSION

This thesis explored various methodologies involving deep learning to predict fluid responsiveness in hemodynamically unstable patients. The multivariate LSTM fully convolutional neural network and DenseNet showed promise in predicting fluid responsiveness using arterial blood pressure and central venous pressure waveforms. The background information, which includes the fundamentals required for this thesis, was established in Chapter 2. Chapter 3 discusses the meticulously curated datasets used in this thesis and the preprocessing techniques required to transform the data to be compatible with the deep learning models. Normalization, stratification, and splitting of the waveforms into sequences were the preprocessing techniques discussed in detail in chapter 3. The chapter also discusses the training methods adopted to best tune the hyperparameters and pick the best models with the help of nested cross-validation and grid search algorithmic techniques.

The experiments conducted and their associated results were clearly defined and discussed in chapter 4. The chapter discussed experiments involving altering the sequence lengths of waveforms for input to the deep learning models.

The experiments discussed the various improvements to the model training that can help the model make good predictions. These experiments involve altering model architectures from removing LSTM to adding the attention mechanism for LSTM and comparing various deep learning architectures. The experiments also involve explaining various methods to deal with class imbalances using sample weighting in loss function and resampling the dataset using SMOTE. Finally, the chapter discussed altering the preprocessing pipeline for normalization and stratification. These experiments helped find the best architecture and the preprocessing pipeline that can help the model make good predictions. Once the modeling pipeline was optimized, the experiments explored various training and test sets sampling based on bolus-level and pig-level splitting. Finally, the thesis explains various experiments that can be used to infer the model's generalizability on unseen holdout datasets. All the experiments listed in Chapter 4 involved the use of various combinations of the data to obtain meaningful inferences from the experiments.

The DenseNet-18 model obtained an AUROC of 0.824 ± 0.05 while using Hem_Sep_ScaledCVP data using the ABP waveform with a sequence length of 224 in nested cross-validation as explained in Table 4.14. However, the pickled DenseNet-18 model did not perform the same way that it did in the nested cross-validation. In addition, the model secured an AUROC of 0.636 ± 0.03 when tested on the IRI_FR_1_2_3 data indicating poor generalization on physiologies outside the scope of the training set as displayed in **Error! Reference source not found.**.

The multivariate LSTM fully convolutional network obtained promising results during nested cross-validation. For example, the MLSTM-FCN model secured an AUROC of 0.869 ± 0.03 when trained and tested using Hem_Sep_ScaledCVP data using ABP waveform during nested cross-validation on a bolus-level splitting with a sequence length of 224 based on Table 4.42. On the other hand, based on Table 4.44, the pickled model which obtained an AUROC of 0.869 ± 0.03 failed to generalize on IRI_FR_1_2_3 data by securing an AUROC of 0.519 ± 0.01. Finally, using the same data and identical sequence length but with a pig-level splitting, the model secured an AUROC of 0.776 ± 0.05 based on Table 4.21. The results from these experiments indicate that the MLSTM-FCN model could not generalize on the IRI_FR_1_2_3 data when trained on the Hem_Sep_ScaledCVP dataset. Various experiments discussed in chapter 4 indicate the same results. Similarly, when all the datasets were used in a pig-level split, the MLSTM-FCN model secured an AUROC of 0.718 ± 0.07 as explained in Table 4.25.

The failure of the MLSTM-FCN model to generalize on unseen physiology can be because the training data set is tiny for a deep learning model to understand the intricate patterns from one physiology and generalize on totally new physiology. However, the MLSTM-FCN model performed better on the pig-level splitting, sugesting that when both the training and the testing data have a similar mixture of physiology, the model picked up the complex patterns and generalized well on unseen pigs.

5.1 LIMITATIONS AND FUTURE WORK

There are several limitations to the methods discussed in this thesis. Based on the results from this thesis, it is evident that the training set should represent all the physiologies to make a compelling predictive model capable of classifying fluid responsiveness of boluses from unseen pigs. The training set has to be very diverse to cover all the physiologies, and if that's not the case, the model might not be an effective predictor. Another important consideration is that the model is developed on a meticulously curated dataset with strictly maintained experimental conditions. However, it is uncertain how the model would behave in noisy real-world data. Another important consideration is the choice of deep learning algorithms. The deep learning algorithms chosen for the experiments were not exhaustive, and using a new algorithm or a more exhaustive hyperparameter search could potentially improve model performance.

The scope of the work from this thesis can be expanded further in the future. An advance in this field would be to shift from the hemodynamic waveforms of pigs to humans. Though the hemodyanmic waveforms of pigs are similar to humans, they are not identical and can pose difficulties for the model to learn essential patterns from the training data. Another progress would be to apply Auto machine learning (Auto ML) methods to automate the designing of deep learning architecture. An Auto ML pipeline can help find the best architecture to predict fluid responsiveness, and we do not have to worry about the lack of an exhaustive search of the hyperparameter space. Finally, transfer learning offers a compelling framework in various applications by improving the model's performance when a small training dataset restricts the model [57]. Hence, transfer learning could prove to be a valuable approach to predict fluid responsiveness since the experiments in this thesis dealt with a relatively small dataset.

REFERENCES

[1] A. Prakash, K. Chitta, and A. Geiger, “Multi-Modal Fusion Transformer for End-to-End Autonomous Driving,” *ArXiv210409224 Cs*, Apr. 2021, Accessed: Sep. 27, 2021. [Online]. Available: http://arxiv.org/abs/2104.09224

[2] S. Carrell and A. Atapour-Abarghouei, “Identification of Driver Phone Usage Violations via State-of-the-Art Object Detection with Tracking,” *ArXiv210902119 Cs*, Sep. 2021, Accessed: Sep. 27, 2021. [Online]. Available: http://arxiv.org/abs/2109.02119

[3] D. S. Park *et al.*, “Improved Noisy Student Training for Automatic Speech Recognition,” *Interspeech 2020*, pp. 2817–2821, Oct. 2020, doi: 10.21437/Interspeech.2020-1470.

[4] X.-Y. Liu *et al.*, “FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance,” *SSRN Electron. J.*, 2020, doi: 10.2139/ssrn.3737859.

[5] L. Bai, L. Yao, X. Wang, and C. Wang, *Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting*. 2020.

[6] L. Wang and A. Wong, “COVID-Net: A Tailored Deep Convolutional Neural Network Design for Detection of COVID-19 Cases from Chest X-Ray Images,” *ArXiv200309871 Cs Eess*, May 2020, Accessed: Sep. 27, 2021. [Online]. Available: http://arxiv.org/abs/2003.09871

[7] J. Lee *et al.*, “Association between fluid balance and survival in critically ill patients,” *J. Intern. Med.*, vol. 277, no. 4, pp. 468–477, Apr. 2015, doi: 10.1111/joim.12274.

[8] D. Payen *et al.*, “A positive fluid balance is associated with a worse outcome in patients with acute renal failure,” *Crit. Care Lond. Engl.*, vol. 12, no. 3, p. R74, 2008, doi: 10.1186/cc6916.

[9] A. Carsetti, M. Cecconi, and A. Rhodes, “Fluid bolus therapy: monitoring and predicting fluid responsiveness,” *Curr. Opin. Crit. Care*, vol. 21, no. 5, pp. 388–394, Oct. 2015, doi: 10.1097/MCC.0000000000000240.

[10] B. Saugel, K. Kouz, A. S. Meidert, L. Schulte-Uentrop, and S. Romagnoli, “How to measure blood pressure using an arterial catheter: a systematic 5-step approach,” *Crit. Care*, vol. 24, no. 1, p. 172, Apr. 2020, doi: 10.1186/s13054-020-02859-w.

[11] M. Nirmalan and P. M. Dark, “Broader applications of arterial pressure wave form analysis,” *Contin. Educ. Anaesth. Crit. Care Pain*, vol. 14, no. 6, pp. 285–290, Dec. 2014, doi: 10.1093/bjaceaccp/mkt078.

[12] D. De Backer and J.-L. Vincent, “Should we measure the central venous pressure to guide fluid management? Ten answers to 10 questions,” *Crit. Care*, vol. 22, no. 1, p. 43, Feb. 2018, doi: 10.1186/s13054-018-1959-3.

[13] T. G. V. Cherpanath *et al.*, “Predicting Fluid Responsiveness by Passive Leg Raising: A Systematic Review and Meta-Analysis of 23 Clinical Trials,” *Crit. Care Med.*, vol. 44, no. 5, pp. 981–991, May 2016, doi: 10.1097/CCM.0000000000001556.

[14] J. I. Alvarado Sánchez, J. D. Caicedo Ruiz, J. J. Diaztagle Fernández, W. F. Amaya Zuñiga, G. A. Ospina-Tascón, and L. E. Cruz Martínez, “Predictors of fluid responsiveness in critically ill patients mechanically ventilated at low tidal volumes: systematic review and meta-analysis,” *Ann. Intensive Care*, vol. 11, p. 28, Feb. 2021, doi: 10.1186/s13613-021-00817-5.

[15] P. E. Marik, R. Cavallazzi, T. Vasu, and A. Hirani, “Dynamic changes in arterial waveform derived variables and fluid responsiveness in mechanically ventilated patients: a systematic review of the literature,” *Crit. Care Med.*, vol. 37, no. 9, pp. 2642–2647, Sep. 2009, doi: 10.1097/CCM.0b013e3181a590da.

[16] J.-L. Teboul, X. Monnet, D. Chemla, and F. Michard, “Arterial Pulse Pressure Variation with Mechanical Ventilation,” *Am. J. Respir. Crit. Care Med.*, vol. 199, no. 1, pp. 22–31, Jan. 2019, doi: 10.1164/rccm.201801-0088CI.

[17] F. Hatib *et al.*, “Machine-learning Algorithm to Predict Hypotension Based on High-fidelity Arterial Pressure Waveform Analysis,” *Anesthesiology*, vol. 129, no. 4, pp. 663–674, Oct. 2018, doi: 10.1097/ALN.0000000000002300.

[18] Z. Zhang, K. M. Ho, and Y. Hong, “Machine learning for the prediction of volume responsiveness in patients with oliguric acute kidney injury in critical care,” *Crit. Care*, vol. 23, no. 1, Apr. 2019, doi: 10.1186/s13054-019-2411-z.

[19] R. Kamaleswaran *et al.*, “Predicting Volume Responsiveness Among Sepsis Patients Using Clinical Data and Continuous Physiological Waveforms,” *AMIA Annu. Symp. Proc. AMIA Symp.*, vol. 2020, pp. 619–628, 2020.

[20] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[21] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. New York, NY: Springer New York, 2013. doi: 10.1007/978-1-4614-6849-3.

[22] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*, 2nd edition. New York, NY: Springer, 2016.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[24] F. Karim, S. Majumdar, H. Darabi, and S. Harford, “Multivariate LSTM-FCNs for Time Series Classification,” *Neural Netw.*, vol. 116, pp. 237–245, Aug. 2019, doi: 10.1016/j.neunet.2019.04.014.

[25] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” *ArXiv160806993 Cs*, Jan. 2018, Accessed: Sep. 28, 2021. [Online]. Available: http://arxiv.org/abs/1608.06993
[26] Z. Wang, W. Yan, and T. Oates, “Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline,” *ArXiv161106455 Cs Stat*, Dec. 2016, Accessed: Sep. 28, 2021. [Online]. Available: http://arxiv.org/abs/1611.06455
[27] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *ArXiv150203167 Cs*, Mar. 2015, Accessed: Sep. 29, 2021. [Online]. Available: http://arxiv.org/abs/1502.03167
[28] X. Glorot, A. Bordes, and Y. Bengio, “Deep Sparse Rectifier Neural Networks,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, Jun. 2011, pp. 315–323. Accessed: Sep. 29, 2021. [Online]. Available: https://proceedings.mlr.press/v15/glorot11a.html
[29] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *ArXiv151203385 Cs*, Dec. 2015, Accessed: Sep. 29, 2021. [Online]. Available: http://arxiv.org/abs/1512.03385
[30] K. He, X. Zhang, S. Ren, and J. Sun, “Identity Mappings in Deep Residual Networks,” *ArXiv160305027 Cs*, Jul. 2016, Accessed: Sep. 29, 2021. [Online]. Available: http://arxiv.org/abs/1603.05027
[31] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” *ArXiv151200567 Cs*, Dec. 2015, Accessed: Sep. 29, 2021. [Online]. Available: http://arxiv.org/abs/1512.00567
[32] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images,” *Univ. Tor.*, May 2012.
[33] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading Digits in Natural Images with Unsupervised Feature Learning,” 2011. Accessed: Sep. 29, 2021. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf
[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” p. 8.
[35] K. Mehmood, H. A. Imran, and U. Latif, “HARDenseNet: A 1D DenseNet Inspired Convolutional Neural Network for Human Activity Recognition with Inertial Sensors,” in *2020 IEEE 23rd International Multitopic Conference (INMIC)*, Nov. 2020, pp. 1–6. doi: 10.1109/INMIC50486.2020.9318067.
[36] J. Azar, A. Makhoul, and R. Couturier, “Using DenseNet for IoT multivariate time series classification,” in *2020 IEEE Symposium on Computers and Communications (ISCC)*, Rennes, France, Jul. 2020, pp. 1–6. doi: 10.1109/ISCC50000.2020.9219631.

[37] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[38] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” *ArXiv14090473 Cs Stat*, May 2016, Accessed: Sep. 29, 2021. [Online]. Available: http://arxiv.org/abs/1409.0473

[39] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, “Squeeze-and-Excitation Networks,” *ArXiv170901507 Cs*, May 2019, Accessed: Sep. 29, 2021. [Online]. Available: http://arxiv.org/abs/1709.01507

[40] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, vol. 385. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-24797-2.

[41] H. Minasyan, “Sepsis: mechanisms of bacterial injury to the patient,” *Scand. J. Trauma Resusc. Emerg. Med.*, vol. 27, no. 1, p. 19, Feb. 2019, doi: 10.1186/s13049-019-0596-4.

[42] T. K. Williams *et al.*, “Endovascular variable aortic control (EVAC) versus resuscitative endovascular balloon occlusion of the aorta (REBOA) in a swine model of hemorrhage and ischemia reperfusion injury,” *J. Trauma Acute Care Surg.*, vol. 85, no. 3, pp. 519–526, Sep. 2018, doi: 10.1097/TA.0000000000002008.

[43] D. Basu *et al.*, “Prediction of Fluid Responsiveness Using Machine Learning and Arterial Blood Pressure Waveform Data”.

[44] H. Odenstedt, A. Aneman, Y. Oi, M. Svensson, O. Stenqvist, and S. Lundin, “Descending aortic blood flow and cardiac output: a clinical and experimental study of continuous oesophageal echo-Doppler flowmetry,” *Acta Anaesthesiol. Scand.*, vol. 45, no. 2, pp. 180–187, Feb. 2001, doi: 10.1034/j.1399-6576.2001.450208.x.

[45] X. Monnet and J.-L. Teboul, “Assessment of fluid responsiveness: recent advances,” *Curr. Opin. Crit. Care*, vol. 24, no. 3, pp. 190–195, Jun. 2018, doi: 10.1097/MCC.0000000000000501.

[46] P. Liashchynskyi and P. Liashchynskyi, “Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS,” *ArXiv191206059 Cs Stat*, Dec. 2019, Accessed: Nov. 03, 2021. [Online]. Available: http://arxiv.org/abs/1912.06059

[47] “Improving Deep Neural Networks: Hyperparameter Tuning, Regularization and Optimization,” *Coursera*. https://www.coursera.org/learn/deep-neural-network (accessed Nov. 03, 2021).

[48] C. Huang, Y. Li, C. C. Loy, and X. Tang, “Learning Deep Representation for Imbalanced Classification,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 5375–5384. doi: 10.1109/CVPR.2016.580.

[49] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality,” in

Advances in Neural Information Processing Systems, 2013, vol. 26. Accessed: Nov. 03, 2021. [Online]. Available: https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html

[50] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, “Class-Balanced Loss Based on Effective Number of Samples,” p. 10.

[51] P. Baldi, “Gradient descent learning algorithm overview: a general dynamical systems perspective,” *IEEE Trans. Neural Netw.*, vol. 6, no. 1, pp. 182–195, Jan. 1995, doi: 10.1109/72.363438.

[52] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson, “Mini-batch gradient descent: Faster convergence under data sparsity,” in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Dec. 2017, pp. 2880–2887. doi: 10.1109/CDC.2017.8264077.

[53] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *Proceedings of the 30th International Conference on Machine Learning*, May 2013, pp. 1139–1147. Accessed: Nov. 03, 2021. [Online]. Available: https://proceedings.mlr.press/v28/sutskever13.html

[54] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *ArXiv14126980 Cs*, Jan. 2017, Accessed: Nov. 03, 2021. [Online]. Available: http://arxiv.org/abs/1412.6980

[55] K. You, M. Long, J. Wang, and M. I. Jordan, “How Does Learning Rate Decay Help Modern Neural Networks?,” *ArXiv190801878 Cs Stat*, Sep. 2019, Accessed: Nov. 07, 2021. [Online]. Available: http://arxiv.org/abs/1908.01878

[56] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, “Don’t Decay the Learning Rate, Increase the Batch Size,” *ArXiv171100489 Cs Stat*, Feb. 2018, Accessed: Nov. 07, 2021. [Online]. Available: http://arxiv.org/abs/1711.00489

[57] M. W. Sjoding *et al.*, “Deep learning to detect acute respiratory distress syndrome on chest radiographs: a retrospective study with external validation,” *Lancet Digit. Health*, vol. 3, no. 6, pp. e340–e348, Jun. 2021, doi: 10.1016/S2589-7500(21)00056-X.