DYNAMIC FEATURE SELECTION VIA REINFORCEMENT LEARNING

A Thesis

by

JOHN MATHAI REJI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Anxiao (Andrew) Jiang
Co-Chair of Committee,	Theodora Chaspari
Committee Member,	Xiaoning Qian
Head of Department,	Scott Schaefer

August 2020

Major Subject: Computer Science

ABSTRACT

Monte Carlo REINFORCE is used to design an algorithm to not only find the optimal deep learning architecture but also the optimal set of features that can maximize the performance of the said deep learning model. The algorithm is applied to the problem of predicting the onset of severe sepsis (before 4 hours) and the results are compared with existing severe sepsis literature. Sepsis is a life-threatening condition caused by the patient body's extreme response to an infection, causing tissue damage and multiple organ failures. MIMIC-III dataset, a publicly available medical dataset is used for all the experiments. Apart from the 6 common vital sign measurements, the dataset also contains 127 physiological and laboratory features to predict the onset of severe sepsis, mostly observed in intensive care units (ICUs). Reinforcement learning is used to reduce the number of features (from 133) without sacrificing peak model performance that uses all 133 features. Among the discovered deep learning models, the CNN-LSTM model using 110 features achieves the best performance: an AUC of 0.933 in predicting the onset of severe sepsis.

DEDICATION

To my mother, my father, and my grandparents.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my research advisor Professor Anxio Jiang from the Department of Computer Science and Engineering, Texas A&M for supporting and guiding me throughout this research. This work would not be possible without his ideas and expertise in the field of deep learning and information theory. I would also like to thank Professor Xiaoqian Jiang and his research group from UTHealth School of Biomedical Informatics, Houston for data collection and preprocessing.

I extend my gratitude to Professor Dileep Kalathil from the Department of Electrical & Computer Engineering, Texas A&M for helping me understand the working concepts behind neural architecture search and reinforcement learning that was necessary to finish my research.

Lastly, I wish to acknowledge the support of my mother Siny Reji, my father A.Y. Reji, my sister Meenu Sara Mathai and my friends for being a constant source of inspiration.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Anxiao (Andrew) Jiang (Chair) and Professor Xiaoning Qian of the Department of Electrical & Computer Engineering and Professor Theodora Chaspari of the Department of Computer Science & Engineering.

The data used for Chapter 2 and 3 was provided by Professor Xiaoqian Jiang and his research group from UTHealth School of Biomedical Informatics.

All other work conducted for the thesis (or) dissertation was completed by the student independently.

Funding Sources

There are no outside funding contributions to acknowledge related to the research and compilation of this document.

NOMENCLATURE

$\vec{A}$	Sampled Action Vector
$\vec{A^*}$	Optimal Action Vector
H	Sampled Hyperparameter Set
H^*	Optimal Hyperparameter Set
val_auc	Child Network Validation AUC(ROC)
R_k^A	Reward by feature controller for kth Child Network
R_k^H	Reward by hyperparameter controller for kth Child Network
G_t	Discounted Reward
b	Mean of Rewards
θ	Controller Weight Parameters
$J(\theta)$	Expected Cumulative Reward
$\nabla_\theta J(\theta)$	Gradient of Expected Cumulative Reward
$a_{k,t}$	Ouput from the tth Controller LSTM Cell from the kth cycle
$s_{k-1,T}$	State from the last Controller LSTM Cell from the kth cycle
λ	Feature Selection Coefficient
$\|\vec{A}\|$	l_1 norm of $\vec{A}$
NAS	Neural Architecture Search
ENAS	Efficient Neural Architecture Search
EWS 2.0	Early Warning Systems 2.0

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Introduction

In the past few years, we saw many advancements in the field of AutoML and Neural Architectural Search (NAS) that can dynamically design deep neural networks without human intervention. In this thesis, we propse to build a reinforcement model that can dynamically choose the optimal feature space for the deep learning network. There are numerous instances where biomedical and financial models involve a large feature space and its often the case that not all these features are required by the model itself. Removing these less important features can save valuable time and resources in building and maintaining these deep learning models. Take for example in hospitals, for detecting or diagnosing a disease numerous patient vitals may be collected over time, if we can identify the the right subset of features we can reduce the cost and time required to come to a conclusion on the patient's condition.

Our motivation for dynamic feature selection comes from our research on severe sepsis prediction, a well known medical problem. Severe Sepsis is estimated to affect about 750,000 people and causes about 200,000 deaths in the United States alone every year. The abundance of electronic health records (EHR) makes it possible to predict severe sepsis in patients based on real-world evidence when appropriate machine learning models are developed. But its not yet established the optimal set of features required to make a good severe sepsis prediction in patients. In this paper, we aim to develop a reinforcement framework to reduce the features required by severe sepsis models by identifying the best subset from the possible feature space. Collecting fewer patient vitals (medical features) increases the chances of early accurate diagnoses, as our sepsis prediction model will require fewer features to perform well.

1.2 Literature Review

Feature selection is a crucial preprocessing step in traditional machine learning. It involves the process of selecting a subset of most relevant or important features in order to reduce the dimension of data. Additionally, it also provides a means to understand the data better and improve model performance at the same time. The art of feature selection is mostly tied to domain knowledge, it is imperative that the person building the machine learning model has in-depth knowledge of the problem domain. However, researchers have come up with solutions for feature selection because having such constraints will limit the use of machine learning as a statistical tool. Historically, feature selection has been performed using filters, wrappers, and embedded approaches [2]. A common method to measure feature importance is through the random forest model in [3] using random forest permutation importance. [4] studied in detail how features affect model accuracies by proposing a permutation-based model-free version of feature importance called model reliance. Researchers in [5] leveraged a variety of random forest to select the optimal subset of features that minimizes cost, where each feature is assigned an acquisition cost. [6] used Bayesian multiple instance learning algorithms to automatically choose relevant features from a larger feature pool. In [7], the authors have used reinforcement learning for selecting the best features. They have modeled this as a Markov Decision Process (MDP) problem where the states are all possible subsets of the features, actions are adding new features to the states, and the reward is the increase or decrease in model accuracy after training using the selected features. The value of the states are learned using a temporal difference approach and the best actions are determined using a criterion called Average of Rewards (AOR). The performance is not only better than other feature selection algorithms, but it was recorded that this method has a faster execution time. In another paper [8], the authors use imitation learning for dynamic feature selection to achieve high accuracy using fewer features. They use the complete feature set to train a model and then use fewer features on the test set to achieve similar generalization accuracy given each feature acquisition comes with a cost.

In the past, feature engineering used to take at most precedence when it comes to building machine learning models but recent advancements in neural networks have completely changed that process. Neural Networks are considered universal approximators and that makes them excellent feature extractors. This desirable property in neural networks has made researchers focus more on designing these networks for a specific task. Now building complex neural networks have always been a challenging engineering problem that involves long hours of network design and hyper-parameter tuning. But in recent years, researchers have found success in automating the process of network architecture design. The success of neural architecture search meant that researchers were now able to automatically generate networks that convincingly outperform human-designed networks. Successful attempts from [1, 9, 10] paved the way for research in automating neural network design. They modeled Neural Architecture Search (NAS) as a search problem to discover state-of-the-art networks (for CIFAR-10 and ImageNet) using reinforcement learning and evolutionary algorithms. Bayesian Optimization is generally used for model selection for hyper-parameter tuning but in [11], a Gaussian process-based Bayesian Optimization framework was developed for NAS. [12] proposed a new method to efficiently learn the structure of convolutional neural networks using sequential model-based optimization (SMBO) strategy. However, the above-mentioned research attempts often required large computing resources and infrastructure that made NAS inaccessible to an average researcher. More recent efforts were put into making NAS more efficient in terms of reducing the number of model evaluations thereby decreasing the total memory and compute time. One such attempt is by sharing parameters across evaluated models from the search space as described in [13, 14]. Most of the above described are limited to macro or chain-structured architecture search (single chained neural network layers). Progress has been made in not only in the field of macro search but also micro search where a NAS is performed at the cell level (eg: Gated Recurrents Units, Long short-term memory cells) and then further optimization is done on a macro level as described in [15]. NAS is usually considered to be a search space problems on a discrete domain until more recently where optimization is done through gradient descent without reinforcement learning as proposed in [16].

We discuss NAS here because our work shares elements similar to the ones discussed above. In this thesis, we are going to treat feature selection as a reinforcement learning problem taking cues from NAS and applying this to the task of optimal feature selection for severe sepsis prediction.

1.3 Thesis Overview

This thesis is organized into 2 main chapters: In Chapter 2 we discuss severe sepsis causes, past work done on severs sepsis, the data set used for this work (MIMICS-III) and finally, present our insights involved in designing an efficient deep learning model for severe sepsis prediction. In Chapter 3 we address the problem of large feature space used for severe sepsis prediction, we shall discuss the working components of neural architecture search and reinforcement learning, develop an automated learning framework for feature selection and share its results. Chapter 4 concludes the thesis and proposes recommendations for future work.

2. SEVERE SEPSIS STUDY

Sepsis is a life threatening condition caused due to the over reaction of the immune system to an infection. This causes inflammation through out the body leading to increased heart rate, high body temperature and fever. Severe sepsis starts when the patient experiences organ failures coupled with fever and delirium (disturbed mental state). When severe sepsis is left untreated for an extended period of time, the patient goes into a septic shock that affects the cardiovascular system possibly leading to death. The goal is to develop a deep learning model that can predict the onset of severe sepsis before a 4 hour period. Taking proactive measures before a 4 hour period avoids life threatening complications and aids speedy recovery. This chapter is dedicated to building a deep learning model for severe sepsis prediction using the openly available MIMIC-III dataset.

2.1 Introduction to Severe Sepsis

Severe Sepsis (sepsis associated with at least one new organ dysfunction) is estimated to affect about 750,000 people and causes about 200,000 deaths in the United States alone every year. Over the past few decades, with better understanding of pathophysiology, many changes have been made to the definition of sepsis. Earlier, the conditions for sepsis is recognized by 2 or more of systemic inflammatory response syndrome (SIRS) criteria, namely tachycardia (heart rate $>$ 90 beats/min), tachypnea (respiratory rate $>$ 20 breaths/min), fever or hypothermia (temperature $>$ 38 or $<$ 36 C), and leukocytosis or leukopenia (white blood cells $>$ 1,200/mm3, $<$ 4,000/mm3 or bandemia $\geq$ 10 %) [17]. But there are certain limitations to SIRS guidelines. For example, a pregnant woman or a healthy human being after an intense cardio activity could satisfy more than 2 SIRS criteria [18]. Due to failures of sepsis-1 [19], The International Consensus Conference (2001) introduced sepsis-2 with additional diagnostic standards. Under sepsis-2, a patient needs to meet at least 2 SIRS criteria with suspected infection to be classified as sepsis patient. Later on, a new definition under sepsis-3 in 2016 was formed, which define sepsis as a life-threatening organ dysfunction caused by dysregulated host response to infection [20]. Sepsis-3 adopted the quick Sequential Organ Failure

Assessment (qSOFA) [20] and SOFA due its ease of calculation and high specificity as its scoring system over the SIRS criteria. There are controversial discussions regarding the applicability of sepsis-2 and sepsis-3, but recent studies show sepsis-3 has higher specificity [21]. Several other scoring systems were also introduced to address the issue of early sepsis identification, e.g., the Modified Early Warning Score (MEWS) [22] and National Early Warning Score (NEWS) [23].

The abundance of electronic health records (EHR) makes it possible to further push the sensitivity and specificity based on real-world evidence when appropriate machine learning models are developed. In this chapter, we explore various machine learning and deep learning models that can convincingly outperform conventional sepsis scoring systems. The main area of focus is to predict the onset of *severe sepsis*, since its associated mortality rates are very high (between 20%-30% [24]). As severe sepsis cases are mostly observed in ICU [25], early identification of severe sepsis in patients can dramatically reduce the chances of going into septic shock, hence increasing the odds of survival and recovery.

There have been a number of works studying the predictions of sepsis onsets. The standard techniques involve the practice of continuously sampling data from electronic health records of the patient to generate early warning scores for prediction of sepsis and sepsis shock. The most common scoring systems include SIRS, SOFA [26], qSOFA [20], MEWS [27], NEWS [28], *etc*. However, these scores are considered to give more insights into the deteriorating condition of the patient and cannot be used directly for sepsis diagnosis. Various machine learning and deep learning techniques have been used, including SVM[29], HMM[30], gradient boosted trees[31], DNN [32] and LSTMs [33], which achieve AUCs between 0.8 and 0.92 for sepsis and septic shock predictions. Note that the existing results focus mainly on general sepsis/septic shock, instead of severe sepsis.

This work focuses on predicting the onset of severe sepsis before a 4-hr period. We present a new deep learning model based on CNN-LSTM, and compare it to machine learning techniques including Adaboost, Xgboost, Random Forests and multiple deep learning models. Beside 6 common vital sign measurements, our model also uses 127 physiological and laboratory features together with demographic information to predict the onset of severe sepsis. The features are mostly observed in intensive care units (ICUs), which match the cohort of this study: the MIMIC-III dataset for patients admitted in ICUs. Among the compared techniques, the deep learning model we present achieved the best performance: an AUC of 0.936 with a 95% confidence interval between 0.931 and 0.941.

2.2 Description of Dataset

The dataset used for sepsis prediction is constructed from the publicly available MIMIC-III database that comprises of health records linked to approximately 40,000 patients admitted to the intensive care units of the Beth Israel Deaconess Medical Center (BIDMS, Boston, MA) between the years 2001 and 2012. The MIMIC-III database comprises of clinical patient data such as vital sign measurements, demographics, lab test results, doctors notes and bedside medical reports.

Individual patient's consent was waived by the Institutional Review Boards of BIDMS and the Massachusetts Institute of Technology (MIT) because the data records in MIMIC-III is anonymized to comply with the Health Insurance Portability and Accountability Act (HIPAA) standards [34]. In this thesis, we chose to use patient's demographics, vital signs, laboratory test, and physiological measurements from MIMIC-III dataset.

2.2.1 Data Extraction

Patients identified with severe sepsis ICD-9 code 995.92 were selected for the study, and the onset time was defined by the time when the patient's SIRS ≥ 2 and at least one organ dysfunction symptoms occurred at the same time, signs for which are discussed below:

- Urine output <0.5 mL/kg, over 2 hours, prior to organ dysfunction after fluid resuscitation
- Creatinine > 2 mg/dL without renal insufficiency or chronic dialysis
- Bilirubin > 2 mg/dL without having liver disease or cirrhosis
- Platelet count $< 100\,000$ μL
- International normalised ratio > 1.5

Table 2.1: Demographic features in the studied cohort

Demographic Characteristics	Attributes	Count
Gender	FEMALE	1425 (58.0%)
	MALE	1028 (42.0%)
Age	16.91 - 30.67	139 (5.67%)
	30.67 - 44.33	241 (9.82%)
	44.33 - 58.00	463 (18.87%)
	58.00 - 71.67	654 (26.66%)
	71.67 - 85.33	657 (26.78%)
	85.33 - 99.00	299 (12.19%)
Ethnicity	ASIAN	57 (2.32%)
	BLACK	205 (8.35%)
	HISPANIC	75 (3.05%)
	WHITE	1803 (73.5%)
	OTHER	313 (21.7%)

- Arterial oxygen tension (PaO2)/fractional inspired oxygen (FiO2) $<$ 200 in addition to pneumonia, or <2 50 with acute kidney injury but without pneumonia.

Based on the above conditions, we selected 432 severe sepsis patients (the cases) and randomly selected another 2021 patients without any sepsis ICD-9 code (the controls)

2.2.2 Data Preprocessing

The features are based on the studies in [35]. After removing the features with insufficient data, the final 3 datasets contains 'n' (n $\in$ {6, 14, 133}) variables per patient shown in Table 3.1, each of which is preprocessed as: vital sign measurement channels 5 hours, 6 hours and 7 hours, respectively, prior to onset; the difference in values between the current hour and the prior hour; and the difference in values between the prior hour and the hour before it. Those five values for each vital sign are concatenated into a causal feature vector x (with 5 values from each of the 'n' measurement channels). Hence each of the patient record contains ('n' vital measurements) $\times$ (5 features) = ('n' x 5 non-sequential features) corresponding to a single patient. The label for the vector is the sepsis onset label $\in$ {0,1}.

Due to the sparse nature of the dataset, Min-Max scaling is used to normalize the features. All the models are trained with 80% and 20% train-test split respectively. For sequential models (e.g., LSTM, CNN-LSTM, *etc*.), each patient record is converted to a timesteps $\times$ feature matrix. This is done by removing the hourly differences and then reshaping the corresponding row into a 3 $\times$ 'n' (timesteps $\times$ feature) matrix.

2.3 Learning Method

After the initial data extraction from the MIMIC-III database, the number of data samples are reduced drastically. Deep learning models are generally data hungry. So to make sure the performance estimate is less biased, k-fold validation was used to reduce variance by averaging over k different folds. All the models are trained with 5-fold stratified cross-validation. After cross-validation the model is again trained using the complete training set and evaluated on the 25% test split. All models are trained with early stopping criteria on validation loss, even in the cross-validation process. Both average validation accuracy and average validation ROC curve (AUC) were calculated during training and validation; however importance is given to AUC as a metric for evaluating the models.

For this prediction task, we experimented with multiple deep learning models and conventional machine learning models including Random Forest, Adaboost and XgBoost. All the models are trained using the same methodology as described earlier. We chose ensemble based learning algorithms that combine individual weak classifiers to form a single strong classifier similar to Random Forests, Adaboost, *etc*. The ensemble models accept an input feature vector of size 1$\times$677 (including demographic features). All the ensemble models are trained using 100 estimators (weak classifiers). In addition, we experimented with deep learning models including DNNs (dense neural networks), CNNs and LSTMs. The hyperparameters for all the models in this experiment is tuned for maximum performance.

Because of the nature of the dataset, sequential models like RNN-LSTM [36] were also considered, which can memorize time dependencies across different features. The CNN and LSTM models take an input vector of (3$\times$133) (timesteps $\times$ feature) matrix whereas the DNNs use a flattened 399-dimensional input vector (1$\times$399).

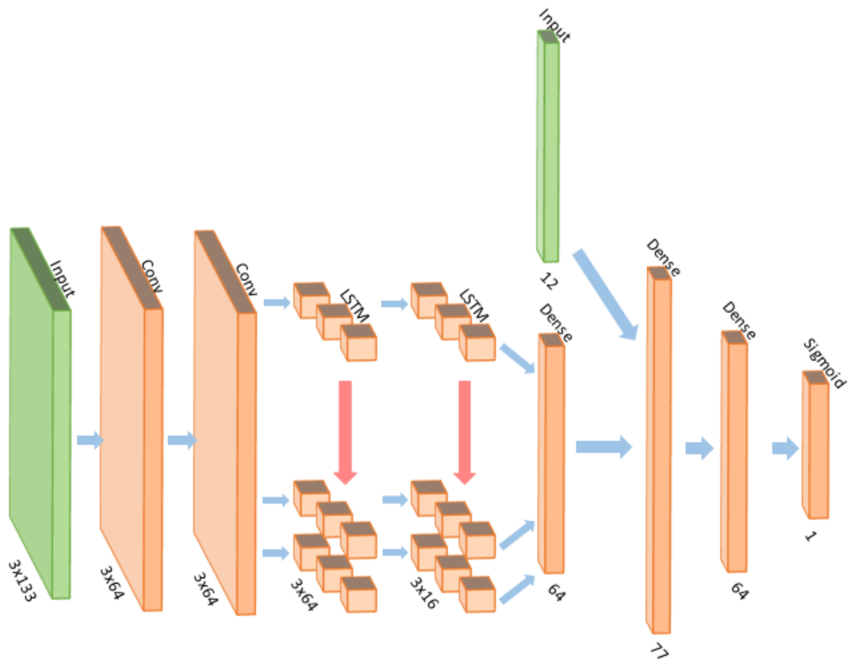


Figure 2.1: The CNN-LSTM model. The numbers next to each layer indicate the shape of its output tensor.

The best performing classifier uses a CNN-LSTM architecture, which is presented in Figure 2.1. From the 3×133 input features, the 2 convolutional layers (of kernel size 1×133 and 1×64 respectively) generate highly discriminative features [37]. Each convolutional layers is followed by a batch normalization layer. The 2 subsequent LSTM layers capture the temporal correlation of the features. Their outputs, together with the 13 demographic features as shown in Table 2.1, are transformed by the dense layers to obtain the final prediction.

2.3.1 Evaluation Metrics

Area under the receiver operating characteristics curve (AUROC) is generally the metric used for evaluating binary classification tasks such as the one described in this study, especially in a clinical setting. AUC (short for AUROC) is a better metric over accuracy as it provides more insights when there is a class imbalance in the dataset. However there are cases when the AUC curve gets too optimistic if the dataset gets heavily skewed towards true negative cases. For this study we are primarily going to stick with AUC metric because several sepsis literature studies in

Table 2.2: Performance comparison of different models

Model	VAL AUC	TRAIN AUC	TEST AUC
CNN-LSTM (ours)	0.99065	1.0	0.96542
LSTM	0.98055	0.99705	0.94162
CNN	0.98312	0.99853	0.89698
DNN	0.95464	0.98431	0.93158
AdaBoost	0.97227	1.0	0.86413
Xboost	0.97963	1.0	0.87999
RandomForest	0.98586	0.95472	0.73549

the past has chosen this metric over the others.

2.4 Experimental Evaluation

The experimental results are summarized in Table 2.2. It can be seen that the deep learning models outperforms the conventional machine learning models. The DNN model with AUC of 0.93 compares favourably to the previous best results on severe sepsis prediction [31]. The CNNs did not yield favourable results possibly due to the relatively small size of our dataset. LSTM slightly outperforms the CNN and DNN models with an AUC of 0.94, due to its utilization of temporal correlation. However, the LSTM layers have many cells and are hard to train optimally given the limited size of the dataset. The CNN-LSTM model addresses this challenge, where the convolutional layers generate useful features and reduce the sizes of the LSTM layers [33, 38]. The optimized CNN-LSTM model achieves the best performance with an AUC of 0.965.

To compare the results to existing results on severe sepsis onset prediction, we list some major works in Table 2.3. The work [29] used non-linear SVM, achieving an AUC of 0.78 with sensitivity of 0.94 and specificity of 0.63. The Insight model in [31] used a gradient boosted tree model with 1000 estimators, achieving an AUC of 0.85 with sensitivity of 0.80 and specificity of 0.84. The EWS 2.0 model in [39] used random forest classifiers with 587 features, including demographics, vital signs, and laboratory results. It achieved an AUC of 0.88 with sensitivity of 0.26 and specificity of 0.98. Although it would not be strictly fair to compare our results to the existing ones due to the difference in datasets, our model does demonstrate its clear strengths. To measure the

Table 2.3: Some prediction performance on severe sepsis reported in the literature

Model	Data Source	Method	AUC	SEN	SPE
EWS 2.0 [39]	UPenn. Health System	Random Forest	0.88	0.26	0.98
Insight [31]	UCSF + MIMIC-III	GBM	0.85	0.80	0.84
PCA + SVM [29]	The Prince of Wales Hospital	SVM	0.78	0.94	0.63
CNN-LSTM (ours)	MIMIC-III	Deep Neural Net	0.94	0.77	0.95

robustness of the model, it is trained 50 times and its average performance is measured. The model achieved an average AUC of 0.936 (95% CI 0.931 0.941), with sensitivity of 0.77 and specificity of 0.95.

To understand how the model performs on different demographics, the detailed results are presented for 491 patients in Table 2.4. Three demographics are considered: gender, ethnicity, and age. In the table, TN, TP, FP and FN represent 'true negative', 'true positive', 'false positive' and 'false negative', respectively. And SEN, SPE and ACC represent 'sensitivity', 'specificity' and 'accuracy', respectively. It can be seen that overall, for each demographic measure, the prediction performance is well balanced for its different subgroups. That is, the sepsis onset prediction model accommodates both good performance and good diversity.

2.5 Conclusion

In this chapter we proposed a deep learning network for severe sepsis prediction that outperforms machine learning models from past severe sepsis literature. However, we do acknowledge the limitation of this work for the inclusion of a single site data and the limited sample size (partially due to the focus on the severe sepsis patients for this study). For future research, we can explore transfer learning techniques to combat potentially biased modeling common for deep learning models when they are applied to unseen data from different hospitals, and develop more robust models that do not necessarily require continuous collection of patient data in real time. Collecting over 133 unique features to monitor a single patient over a period of time is tedious hence in the next chapter we will primarily focus on reducing the number of unique features used by our deep learning model.

Table 2.4: Results based on Demographics evaluated on our 20% reserved test data (491 samples)

Characteristics	Attributes	TN	TP	FP	FN	SEN	SPE	ACC
Gender	FEMALE	177	23	9	8	0.72	0.96	0.92
	MALE	207	43	11	13	0.8	0.94	0.91
	TOTAL	384	66	20	21			
Ethnicity	ASIAN	13	1	2	1	0.33	0.93	0.82
	BLACK	35	4	2	1	0.67	0.97	0.93
	HISPANIC	12	0	0	0	nan	1.0	1.0
	WHITE	276	55	14	12	0.8	0.96	0.93
	OTHER	48	6	2	7	0.75	0.87	0.86
	TOTAL	384	66	20	21			
Age	16.92 - 30.67	32	1	0	0	1.0	1.0	1.0
	30.67 - 44.33	41	4	3	0	0.57	1.0	0.94
	44.33 - 58.00	64	9	1	7	0.9	0.9	0.9
	58.00 - 71.67	94	22	6	7	0.79	0.93	0.9
	71.67 - 85.33	110	22	7	5	0.76	0.96	0.92
	85.33 - 99.00	43	8	3	2	0.73	0.96	0.91
	TOTAL	384	66	20	21			

3. DYNAMIC FEATURE SELECTION

In the previous chapter, we have discussed how we use all 133 features from electronic health records to build a novel deep learning model that improves the performance of severe sepsis onset predictions. The model takes advantage of time dependencies between the features (of 4 hrs before onset) to achieve state-of-the-art performance. When synthesized carefully, the vital signs from severe sepsis patience contains temporal-spatial patterns that offer better discriminative power than previous methodologies.

Most of the predictions models in sepsis literature (discussed in the previous chapter) uses features anywhere between 6 and 30. On the other hand, this study uses 133 features to achieve significantly higher performance using deep learning. In a real-world scenario, it might not be viable for hospitals to continuously monitor 133 features. In this chapter, we implement a reinforcement learning framework to select the most optimal set of features (from 133) required for severe sepsis diagnosis.

3.1 General Working

Now we shall discuss how Neural Architecture Search (NAS) works and how it relates to our research in this thesis. This section gives you a holistic view of the working of a simple Neural Architecture Search (NAS) framework and its components followed by our research on Dynamic Feature Selection & Hyperparameter Tuning.

3.1.1 Neural Architecture Search

The first attempt to develop a NAS framework was described in [1], the authors used a controller network for sampling architectures and a child network for evaluating the suggested architecture against a validation dataset. The validation accuracy is fed back to the controller as a reward signal R, which is used to update the controller's parameters θ using a policy gradient approach shown in Figure 3.1.

Similarily in [10], the authors use Q-learning with experience replay and epsilon greedy strat-

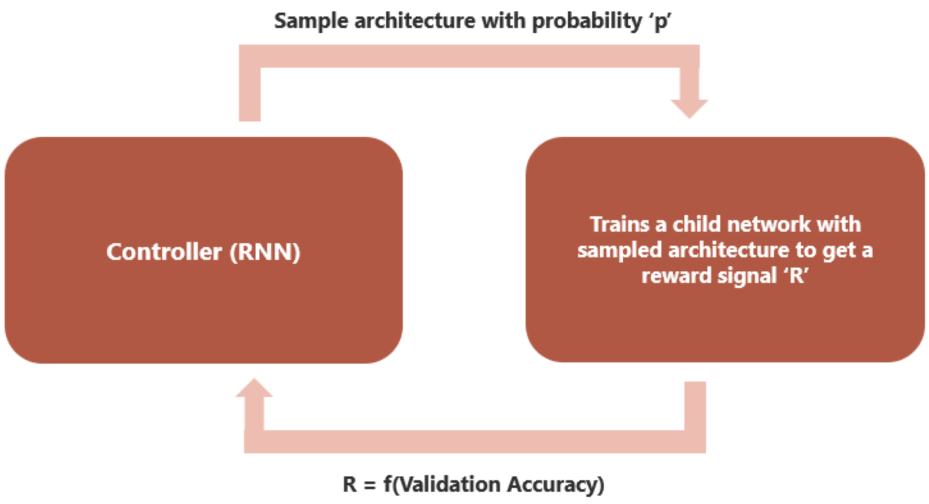

Sample architecture with probability 'p'
Controller (RNN)
Trains a child network with sampled architecture to get a reward signal 'R'
R = f(Validation Accuracy)

Figure 3.1: Neural Architecture Search using Reinforcement Learning

egy to generate CNN architecture using validation accuracy from the selected child network. Controller agents in [1, 10] generates only single chained neural network models. Since the severe sepsis prediction discussed in the previous chapter is a time-series problem, the main focus is on designing single chained networks (1D convolution and recurrent networks) for the child model. We have given more emphasis on choosing the right features using dynamic feature selection. We have taken cues from [1, 10] in designing the controllers in this research.

3.1.2 Dynamic Feature Selection

So far we have briefly discussed the NAS cycle and how the NAS controller builds a child network model. We cannot directly apply the NAS methodology because building a Neural network and selecting appropriate features are not the same. In this research we articulate 2 main cycles 1) Dynamic Feature Selection 2) Hyperparameter Tuning as shown Figure 3.3 and 3.4 respectively. In the feature selection cycle our objective is to find the optimal set of features $\vec{A}^*$ and then subsequently use it for Hyperparameter Tuning to find optimal parameters H^*.

In Figure 3.3 we have the feature selection controller (a LSTM based deep neural network) that samples a feature vector $\vec{A}$. This feature vector $\vec{A}$ decides which set of features are selected. The dataset is modified such that only the selected features are used to train the baseline child network

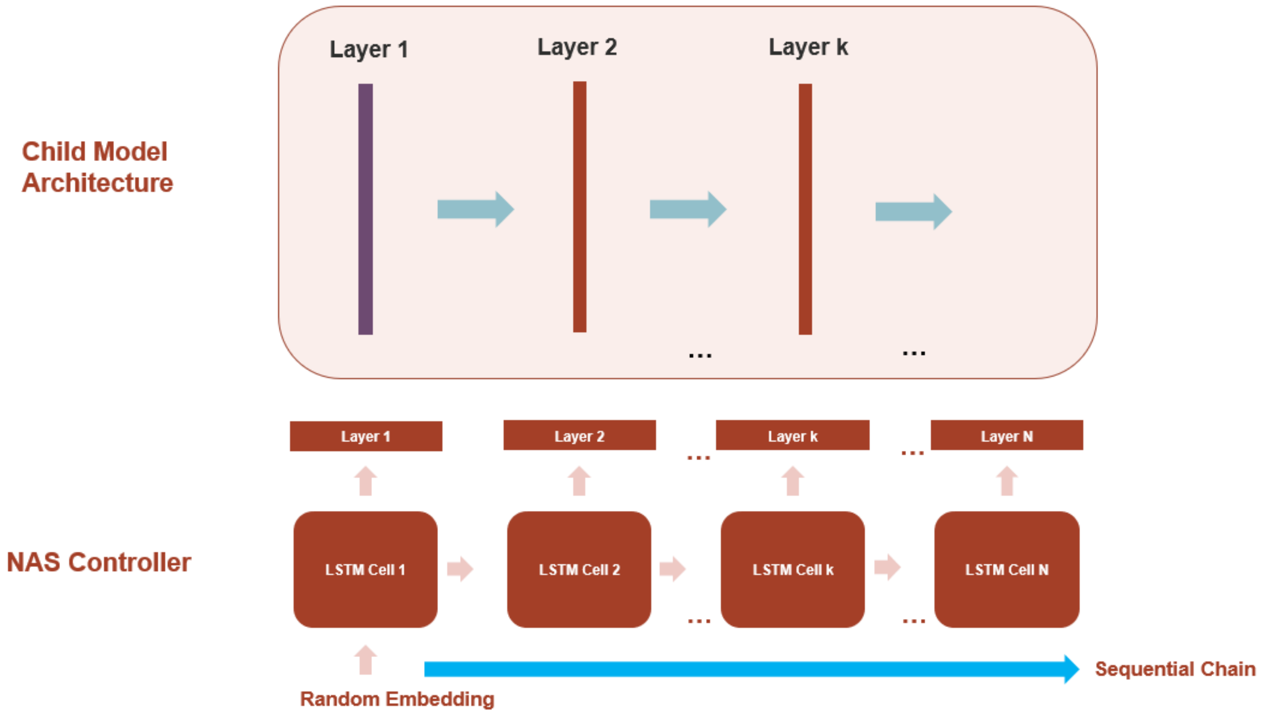
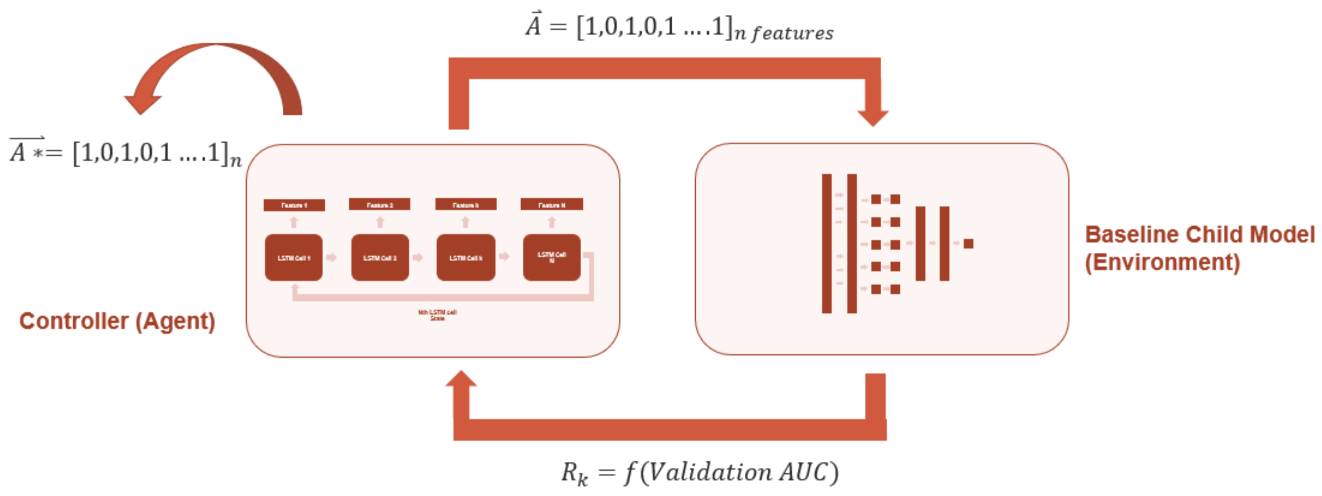


Figure 3.2: A simplified model of NAS [1] controller sampling a child network architecture. Each lstm cell outputs the parameters of a single layer building the neural network layer by layer sequentially.



$\vec{A} = [1,0,1,0,1 \ldots .1]_{n\ features}$
$\overrightarrow{A*} = [1,0,1,0,1 \ldots .1]_n$
Feature 1
Feature 2
Feature k
Feature N
LSTM Cell 1
LSTM Cell 2
LSTM Cell k
LSTM Cell N
Nth LSTM cell State
Controller (Agent)
Baseline Child Model (Environment)
$R_k = f(Validation\ AUC)$

Figure 3.3: Dynamic Feature Selection cycle.

model. In feature selection, our child network model (in our experiment we use a variation of the CNN-LSTM model we discussed for severe sepsis prediction) is fixed. The reward signal sent to the feature selection controller is a function of the validation accuracy from the baseline child network model. The cycle stops when the feature controller converges to an optimal set of features $\vec{A}*$.

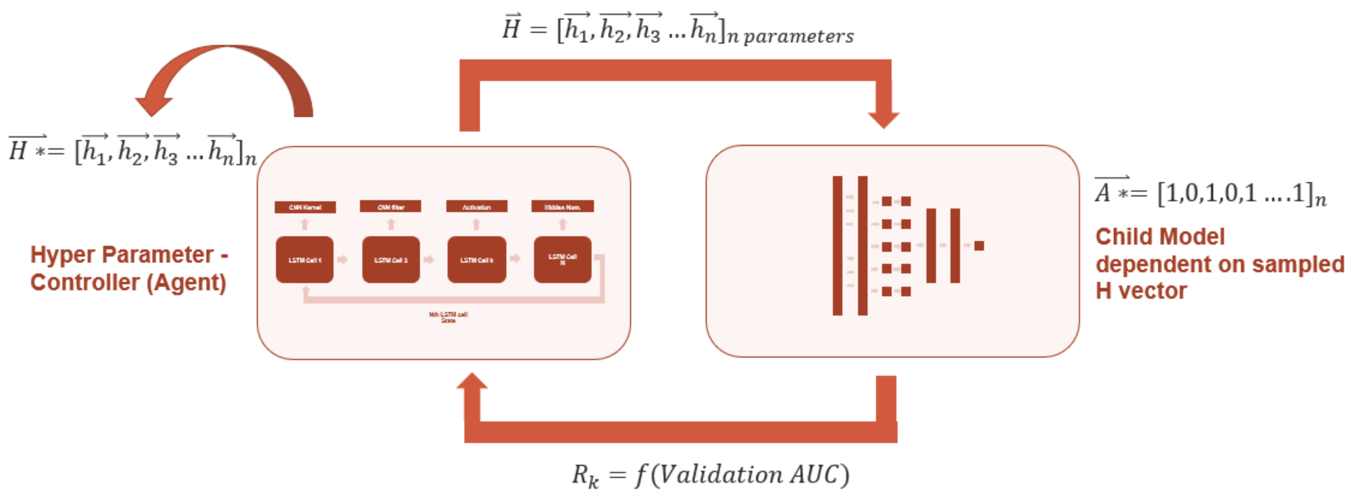
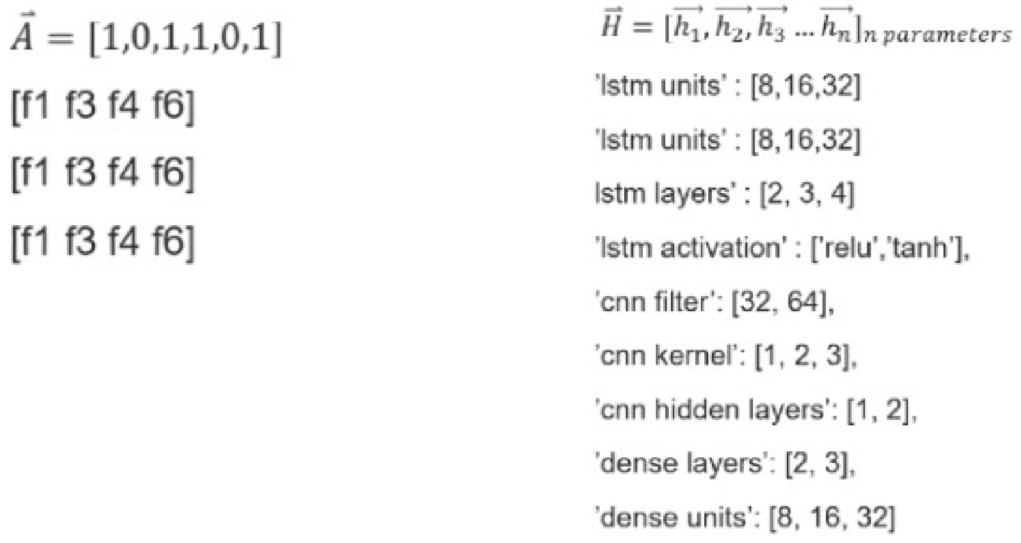

$\vec{H} = [\vec{h_1}, \vec{h_2}, \vec{h_3} \ldots \vec{h_n}]_{n\,parameters}$
$\vec{H}* = [\vec{h_1}, \vec{h_2}, \vec{h_3} \ldots \vec{h_n}]_n$
Hyper Parameter - Controller (Agent)
CNN Kernel
CNN filter
Activation
Hidden Num.
LSTM Cell 1
LSTM Cell 2
LSTM Cell k
LSTM Cell N
Nth LSTM cell State
$\vec{A}* = [1,0,1,0,1 \ldots .1]_n$
Child Model dependent on sampled H vector
$R_k = f(Validation\ AUC)$

Figure 3.4: Hyperparameter Tuning cycle.

After obtaining the optimal feature vector $\vec{A}*$, its essential to further tune the child network model by using features selected in $\vec{A}*$. The second cycle for Hyperparameter Tuning follows the same strategy as in the feature selection cycle except that instead of choosing features we proceed to find the optimal hyperparameter H^* for the child network as shown in Figure 3.4. See the appendix for the different hyperparameter attributes used for tuning the child network. Figure 3.5 gives an example of $\vec{A}$ and H used in this research. Since we got an overall picture into the working of our research methodology, the upcoming sections describes in detail the controller design and learning algorithm used for the experiments.



$\vec{A} = [1,0,1,1,0,1]$

[f1 f3 f4 f6]

[f1 f3 f4 f6]

[f1 f3 f4 f6]

$\vec{H} = [\vec{h_1}, \vec{h_2}, \vec{h_3} \ldots \vec{h_n}]_{n\ parameters}$

'lstm units' : [8,16,32]

'lstm units' : [8,16,32]

lstm layers' : [2, 3, 4]

'lstm activation' : ['relu','tanh'],

'cnn filter': [32, 64],

'cnn kernel': [1, 2, 3],

'cnn hidden layers': [1, 2],

'dense layers': [2, 3],

'dense units': [8, 16, 32]

Figure 3.5: An example of feature vector $\vec{A}$ sampled by the feature controller (on the left), $\vec{A}$ is a binary vector 1 indicates the particular feature is selected and 0 indicates the particular feature is removed. An example of H hyperparameter sampled by the hyperparameter controller (on the right).

3.2 Controller Design Insights

The most important aspect of this research is the controller design since it models the relationships between features in case of dynamic feature selection or dependencies between neural network layers in the case of NAS. For this research we have experimented with 2 different controller agents 1) Controller A & 2) Controller B. These controllers are similar to the ones used in NAS but it is redesigned to make it suitable for the task of dynamic feature selection and hyperparameter tuning. Controller B is just an enhanced version of controller A. To understand the decisions we made to design the controllers it is essential to know how the NAS controller [1] is designed.

3.2.1 NAS Controller

In 3.1.1 we briefly discussed the NAS cycle and its working. In this section, we describe how the NAS controller spawns the child network. NAS uses an LSTM network to generate its child network models as shown in Figure (3). The controller takes an empty embedding (random tensor) as its input at each controller epoch (or cycle) and samples a child network model dictated by the decisions made by the lstm cells. In a single chained neural network model each Nth layer is dependent on the previous N-1 layers. In any neural network model the first layer is always the input layer, hence the first lstm cell in a NAS controller is always fed a random tensor (because the input layer is not conditioned on any previous layers). The following sections describes how the controllers are designed to specifically model the task of feature selection and hyperparameter tuning.

3.2.2 Controller A

This controller design is inspired from the ENAS [15] and NAS [1] LSTM controllers. The NAS controller is designed such that the nth LSTM cell, outputs parameters of the nth layer. And each nth layer is dependent on the previous 1 to $n-1$ layers of the child model. Hence, the Monte Carlo policy gradient is formulated in a similar way as given below:

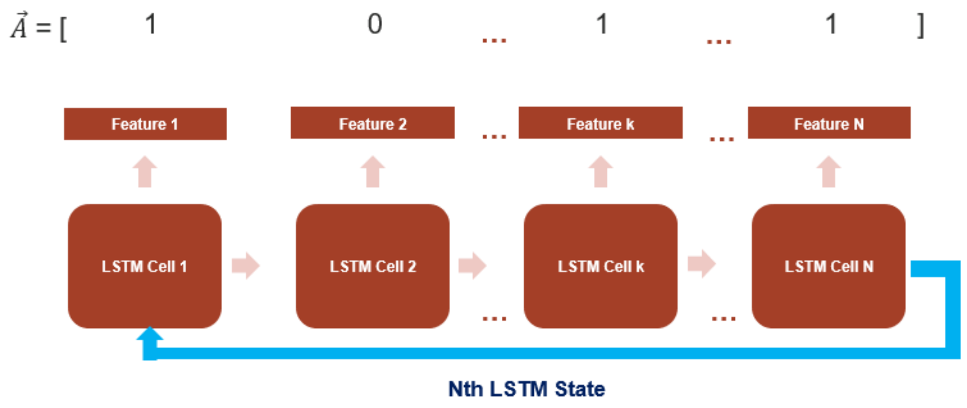


Figure 3.6: Controller Design A

In (3.1), $a_{k,t}$ is dependent on $a_{k,t-1:k,1}$ as explained above. But that's not the case for features as it is not necessary that nth feature is dependent on previous 1 to $n-1$ features. It is possible that the first feature might be dependent on the nth feature or vice versa. To model this behavior in this controller (Equation 3.9) the state & output of the last LSTM cell is fed as the input state to the controller in the next epoch, in contrast to NAS & ENAS controllers where an empty embedding is fed in every epoch. By doing so we are actually introducing a dependency for the first few features based on the states of all the features from the previous epoch. In this way, the controller will be able to decide whether it should give importance to the first few sets of features in the next epoch. Additionally, introducing the state variable also stabilized the controller training leading to an elegant convergence. The controller architecture and design are shown in the appendix below.

$$\nabla_\theta J(\theta) \approx \frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T}\nabla_\theta logP(a_{k,t}|a_{k,t-1:k,1};\theta)(R_k - b) \tag{3.1}$$

3.2.3 Controller B

Controller design discussed in section 3.2.2 is suitable for data-sets that contain 30-50 unique features. Selecting an optimal subset of features when there are more than 100 unique features becomes more tedious. And a single state vector s_{k-1} is not sufficient to encode the dependencies between 100 features. In such a situation we have proposed to use the hidden state from all the LSTM cells (each lstm cell corresponds to a single feature) to provide better context to the controller if a particular feature needs to be selected (see Figure(4)). The results from both the controller A & B are discussed in section 3.5

3.3 Monte Carlo REINFORCE

In this section, we introduce reinforcement learning and the Monte Carlo REINFORCE algorithm that is used to train the controllers. Reinforcement learning is one of the major paradigms in machine learning (along with supervised learning and unsupervised learning). It involves an agent that tries to maximize its cumulative reward through a set of actions in an unknown complex environment. The aim of the agent is to learn an optimal strategy from experiences interacting with the environment. Formally, reinforcement learning is modeled as a Markov Decision Process that consists of an agent that can belong to a state s from many states S in an environment and can choose to take an action from a set of defined actions A. The probability of transitioning from state s to another state s' under action a is given by the transition probability $P(s'|s,a)$ and the reward

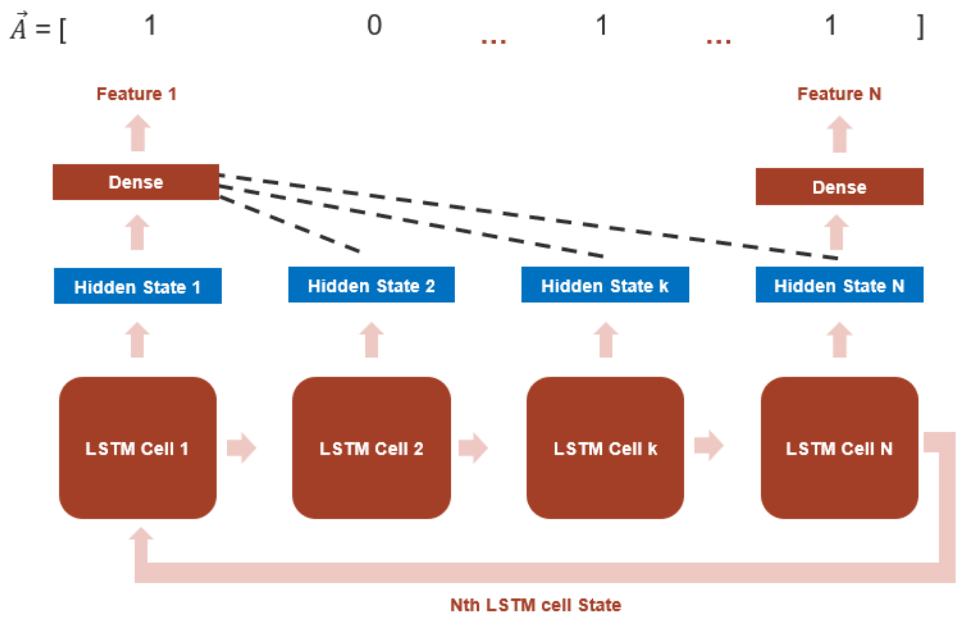


Figure 3.7: Controller Design B.

function is defined by $R(s', s)$ after transition to state s' from state s under action a.

Reinforcement algorithms can be classified mainly into two types: *model-based* learning and *model-free* learning. Model-based algorithms learn the transition probability implicitly from action-state pairs. However, for systems with large action-state spaces model-based learning becomes infeasible. On the other hand, model-free algorithms learn by sampling action-state pairs through trial and error. Model-free learning doesn't directly depend on the dynamics of the environment (transition probability and reward function) making them suitable for reinforcement learning problems with large action-state spaces.

In this thesis, we pose the problem of dynamic feature selection as a model-free reinforcement learning task. We adopt Monte Carlo REINFORCE as the reinforcement learning strategy. Monte Carlo REINFORCE is a policy gradient method that seeks to find the optimal policy directly. The policy π is defined as deterministic ($\pi(s) = a$) or stochastic mapping ($\pi(a|s) = P_\pi[A = a|S = s]$) from state s to an action a. In a policy gradient algorithm, the policy is often modeled as a function of parameter θ as $\pi_\theta(a|s)$. The reward function is calculated based on this policy $\pi_\theta(a|s)$. The goal of the policy gradient algorithm is to maximize the expected reward $J(\theta)$ by optimizing θ for a given total reward function $r(\tau)$ associated to the trajectory τ (or episode) by following the policy π as defined below:

$$J(\theta) = E_{\pi}[r(\tau)] \tag{3.2}$$

We optimize θ to maximize $J(\theta)$ using gradient ascent update rule:

$$\theta_{t+1} = \theta_t + \alpha \nabla_{\theta} J(\theta_t)) \tag{3.3}$$

Now the gradient of expectation $\nabla E_{\pi}[r(\tau)]$ can be written as $\nabla \int \pi(\tau) r(\tau) d\tau$. By expanding this expectation we get $\int \pi(\tau) \nabla \log \pi(\tau) r(\tau) d\tau$. Hence we can rewrite the expected reward as:

$$\nabla E_{\pi}[r(\tau)] = E_{\pi}[\nabla \log \pi(\tau) r(\tau)] \tag{3.4}$$

where $\pi(\tau)$ is a product of transistion probabilities $p(s_{t+1}, r_{t+1}|s_t, a_t)$ and policy $\pi(a_t|s_t)$ over time period T. Additionally we also subsitute $r(\tau)$ for G_t. In reinforcement learning the current reward is not dependent on past states, instead we try to take the future discounted rewards G_t that we might receive when we take an action from a given state s_t. G_t is defined as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3.5}$$

We discussed previously that REINFORCE is a model-free learning algorithm, by using the Williams (1992) rule we can rewrite Equation (3.4) as:

$$\nabla E_{\pi}[r(\tau)] = E_{\pi}[(\sum_{t=1}^{T} G_t \nabla \log \pi(a_t|s_t))] \tag{3.6}$$

By directly applying the REINFORCE Williams rule, we were able to get rid of the environment dynamics (deriving and proving Williams rule is not trivial and is beyond the scope of this work). Now computing the raw gradient using the scaling factor G_t directly is unreliable because of high variance. To reduce the variance we introduce a baseline b (b can be a function of current state s_t, etc. depends on the problem statement) in Equation (3.6):

$$\nabla J(\theta) = \nabla E_\pi[r(\tau)] = E_\pi[(\sum_{t=1}^{T}(G_t - b)\nabla \log \pi(a_t|s_t))] \tag{3.7}$$

Adding this bias provides stability by reducing variance and at the same time doesn't change the unbiased nature of the gradient estimator. The complete Monte Carlo REINFORCE procedure is given below:

Algorithm 1: REINFORCE Algorithm

```
Function REINFORCE ($\tau$, $\theta$)
    for t = 1, 2, ..., T do
        Estimate the discounted return $G_t$ from trajectory $\tau$
        $\theta \leftarrow \theta + \alpha\gamma^t(G_t - b)\nabla \log \pi(a_t|s_t)$
```

In [1], the authors used the Monte Carlo REINFORCE algorithm to train the controller given by Equation (3.7). The expectation is replaced by taking large samples and substitute R_k directly as G_t (generally REINFORCE algorithm is updated over an episode after generating a trajectory τ from policy π_θ but in NAS an episode generates only a single trajectory with reward R) hence Equation (3.7) is modified to:

$$\nabla_\theta J(\theta) \approx \frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T} \nabla_\theta \log P(a_t \mid a_{(t-1):1}; \theta)(R_k - b) \tag{3.8}$$

Where $a_{1:T}$ is the list of actions for designing a child network, m is the number of different child networks sampled by the controller in one batch and T is the number of hyperparameters of a neural network (eg. activation function, filter size, kernel size) to design a child network. As we discussed previously directly using a scaled factor R_k leads to high variance, a baseline b (exponential moving average of R_k) is used to reduce the variance of this unbiased gradient estimator. In the next section, we formulate the Monte Carlo REINFORCE algorithm to fit our methodology.

3.4 Problem formulation

The algorithm design consists of 2 reinforcement agents - Feature selection controller & Hyperparameter search controller. Initially the optimal set of features are achieved by training the feature selection controller via the Monte Carlo REINFORCE policy gradient. The optimal feature $\vec{A}^*$ is used to search for the optimal deep learning hyperparameters H^* for predicting severe sepsis using the Hyperparameter search controller.

Monte Carlo REINFORCE specified in NAS[1] given by the equation (3.8). But in this case it is slightly altered to take into account the nth LSTM state as given below:

$$\nabla_\theta J(\theta) \approx \frac{1}{m} \sum_{k=1}^{m} \sum_{t=1}^{T} \nabla_\theta log P(a_{k,t} | a_{k,t-1:k,1}, s_{k-1,T}; \theta)(R_k^A - b_a) \tag{3.9}$$

where b_a is:

$$b_a = \frac{1}{n} \sum_{k=1}^{m} R_k^A \tag{3.10}$$

In this reinforcement problem, the state $s_{k-1,T}$ is the output of the last lstm cell of the controller. Action space $\vec{A}$, is defined as the output generated by the policy controller given state $s_{k-1,T}$. For example, the dataset used here consists of 6 features, so the action space is a binary vector of size 6. At every step, the controller generates a $\vec{A}$ which is used to select the features from the dataset specified by the action vector. The selected features are then used to train a fixed baseline child network (baseline CNN-LSTM network), the validation AUC (area under the curve) from the child network is sent to the controller as the reward signal R_k^A. Since we do not have a simulated environment, the success of the agent largely depends on tuning the reward function to create a feature space (less than the maximum feature space) to give an optimal validation AUC. The reward function, in this case, is given by taking the summation of the validation AUC times 4 of the last 5 epochs of the child model including the feature regularizer as defined below:

$$R_k^A = \sum_{i=n-5}^{n} (val\_auc)_i^4 - \frac{|\vec{A}|}{dim|\vec{A}|} \tag{3.11}$$

The stochastic policy here is the probability of choosing a particular feature a_i in $\vec{A}$, sampled from a Bernoulli distribution.

Once we converge to an optimal $\vec{A^*}$, we use this information to run the hyperparameter search using the Monte Carlo REINFORCE policy gradient for the child model given by equation below:

$$\nabla_\theta J(\theta) \approx \frac{1}{m} \sum_{k=1}^{m} \sum_{t=1}^{T} \nabla_\theta log P(h_{k,t} | h_{k,t-1:k,1}, s_{k-1,T}; \theta)(R_k^H - b_h) \tag{3.12}$$

where b_h is:

$$b_h = \frac{1}{n} \sum_{k=1}^{m} R_k^H \tag{3.13}$$

In the feature selection loop, we fixed a baseline child network but in the hyperparameter search loop, our goal is to find the optimal set of parameters H for the child model by fixing $\vec{A} = \vec{A^*}$. The controller is modeled the same way as we did above for the feature selection controller. Here the reward R_k^H is just a function of the validation AUC as we want to obtain a policy that gives H^* for the maximum performance (best validation AUC) defined as:

$$R_k^H = \sum_{i=n-5}^{n} (val\_auc)_i^4 \tag{3.14}$$

The stochastic policy here is the probability if you will choose a particular hyperparameter h_i in H, sampled by a Multinomial distribution.

3.5 Research Approach

After preprocessing the MIMIC-III dataset, we have 3 sets of data for testing purposes as shown in Table 3.1. We follow the same algorithm as discussed previously for all the 3 datasets. Features in all 3 datasets are reshaped from vectors into time-series features by removing the hourly differences and folding them into timesteps of 3 hrs. For example, Dataset-I (from 30 features) is

Algorithm 2: Algorithm Design

```
Function FindChildModel (features, search params)
    while all $a_i$ is $\leq$ 0.9 or $\geq$ 0.1 in $\vec{A}$ do
        Sample a feature vector $\vec{A}$ from feature selection controller;
        Train baseline child model with $\vec{A}$;
        Train feature selection controller with $R_k^A, \vec{A}, s_{k-1}$;
    while all max$|\vec{h_i}|$ is $\leq$ 0.8 in $H$ do
        Sample a set of hyperparameters $H$ from hyperparameter controller;
        Train sampled child model architecture $H$ with $\vec{A}^*$;
        Train hyperparameter controller with $R_k^H, H, s_{k-1}$;
```

Table 3.1: Sepsis data input feature dimensions

Dataset-I	**Dataset-II**	**Dataset-III**
30 (6x5) features	70 (14x5) features	665 (133x5) features

transformed into 6 time-series features by removing hourly differences (12 features removed) and folding the remaining 18 features into 3x6 (timesteps x features).

The goal is to find the optimal set of features needed for getting equivalent test results (that uses all features) or better using only the selected optimal features. The results for each of three datasets are summarized in Table 3.2, Table 3.3 and Table 3.4. For comparing results for Dataset-I we used Insight [40] as the benchmark. Insight uses the exact same features as we used for this experiment with Dataset-I. Using Dynamic feature selection and Hyperparameter search on CNN-LSTM model, the number of features were reduced from 6 to 4 without no performance loss as shown in Table 3.2. For Dataset-II (Table 3.3) and Dataset-III (Table 3.4), we are not aware of any known severe sepsis literature that uses a feature set of 14 and 133 respectively. Hence as a benchmark, just a Hyperparameter search is performed using the complete feature set in Dataset-II

Table 3.2: Controller A & B Results for Dataset-I

Controller A Attribute	Benchmark	Controller A	Controller B
Model	Insight [40]	CNN-LSTM	CNN-LSTM
Features	6	4	4
Dataset	UCSF + MIMIC-III	MIMIC-III	MIMIC-III
Result	0.85	0.85	0.85

Table 3.3: Controller A & B Results for Dataset-II

Attribute	Benchmark	Controller A	Controller B
Model	CNN-LSTM	CNN-LSTM	CNN-LSTM
Features	14	9	9
Dataset	MIMIC-III	MIMIC-III	MIMIC-III
Result	0.85	0.88	0.86

and Dataset-III and compared it to results obtained by performing both Dynamic feature selection (to select optimal features) and then Hyperparameter search using the optimal selected features. In all the above experiments, the CNN-LSTM model was used as the child model.

It is evident from Table 3.2, Table 3.3 and Table 3.4 that the performance of deep learning models trained with the optimal selected features are comparable to the performance of those models that use the entire feature set. The average run time for finding the best CNN-LSTM model including feature selection for Dataset-I (6 features) and Dataset-II (14 features) was around 400 controller epochs, and for Dataset-III (133 features) it took around 800 controller epochs. The reward history plots for all training experiments are shown in the appendix below.

Table 2.3 from the previous chapter has been updated to Table 3.5, to show the number of

Table 3.4: Controller A & B Results for Dataset-III

Attribute	Benchmark	Controller A	Controller B
Model	CNN-LSTM	CNN-LSTM	CNN-LSTM
Features	133	-	110
Dataset	MIMIC-III	MIMIC-III	MIMIC-III
Result	0.937	-	0.933

Table 3.5: Features vs performance on Severe Sepsis reported in the literature

Model	Num. of features	Method	AUC	SEN	SPE
EWS 2.0 [39]	587	Random Forest	0.88	0.26	0.98
Insight [31]	6	GBM	0.85	0.80	0.84
PCA + SVM [29]	6	SVM	0.78	0.94	0.63
CNN-LSTM (Original)	133 + 13(demographics)	Deep Learning	0.94	0.77	0.95
CNN-LSTM (with Dyn.)	110	Deep Learning	0.93	0.76	0.94

features used by past results on severe sepsis prediction. Most of the models that exist in sepsis literature use fewer features with few exceptions (EWS 2.0 [39]) for sepsis diagnosis and treatment. [29] and [40] uses only 6 features for severe sepsis prediction and achieved AUC scores of 0.78 and 0.85. Again we need to remind ourselves that the set of features used in [29, 40, 39] are different. To measure the robustness of the CNN-LSTM model that uses 110 features, it is trained 50 times and then its average AUC performance is measured. The model achieves an average AUC of 0.933, with a sensitivity of 0.76 and specificity of 0.94.

Reducing the number of features from 133 to 110 is a step in the right direction. However, collecting 110 features is still a daunting task. We can force the controller to choose even fewer features by adjusting the value of λ. A value of λ set between $(1, 1.5]$ should force the controller to pick features more aggressively. To accomodate λ we modify Equation 3.11 to:

$$R_k^A = \sum_{i=n-5}^{n} (val\_auc)_i^4 - \lambda(\frac{|\vec{A}|}{dim|\vec{A}|}) \tag{3.15}$$

Optimal feature selection is a complicated task, in this thesis we would like to point to some inconsistencies we faced when running the experiments. We designed 2 controllers for feature selection - 1) Controller A and 2) Controller B. Even though Controller A and Controller B picked an equal number of features for both Dataset-I and Dataset-II, the feature set (the action vector A^*) chosen by each one of them are different as given in Table 3.6.

Dynamic feature selection with Controller B repeatedly converged to the same set of features for Dataset-I and Dataset-II (because Controller B gets better context about the feature depen-

Table 3.6: Features selected ($\vec{A}^*$) by Controller A and B. See the Appendix for more details on features selected by each controller for the given dataset.

Data	Controller A	Controller B
DataSet-I	[1 1 0 1 1 0]	[1 1 0 0 1 1]
DataSet-II	[0 0 1 1 1 1 0 1 0 1 1 1 1 0]	[0 0 1 1 1 0 1 1 0 1 0 1 1 1]

dencies), as shown in Table 3.6. However, that wasn't case for Controller A as there were small deviations from the features selected when the experiment is run every single time. Unfortunately, Controller B is also not consistent when performing dynamic selection using Dataset-III. This inconsistency can be attributed to 2 main factors, one being that the dimensions of the feature set is too high (133 feature) and second is the fact that there can exist 2 or more features that might be highly correlated and the controller can choose only either one of them because of the constraints set by the reward function R_k during feature selection. In an event where there are 2 correlated features in a set, the controller picks the feature that gets sampled more frequently over the other.

4. SUMMARY AND CONCLUSIONS

Reinforcement learning (MONTE Carlo REINFORCE) was used to optimize the feature space as well as the hyperparameter space to discover deep learning models that perform equally well or better when compared to models that use the complete set of features from the dataset. The initial proposal was to perform a hyperparameter search H for every $\vec{A}$ sampled. But this process seemed extremely slow and tedious. For Dataset-I (6 features) it took approximately 48 hours to converge. Therefore a more viable solution was proposed to separate the reinforcements loops into first dynamic feature selection using a baseline child model and then perform a hyperparameter search using the optimal feature vector $\vec{A^*}$ to find optimal hyperparameters H^*. This technique was a good approximation to the previously thought out algorithm.

We performed dynamic feature selection and hyperparameter tuning for 3 different datasets with 2 different controllers (A & B). The experiment brought down the total number of features used in the severe sepsis prediction model (described in Chapter 2) from 133 to 110. Reducing the features had little to no impact on the prediction performance.

4.1 Challenges

The ability to achieve consistent results if a researcher tries to replicate the experiment is known as reproducibility. One of the main challenges we face in the field of deep reinforcement learning is reproducibility. Most of these models are black boxes hence studying and replicating these models become difficult because of the level of uncertainty involved. In this thesis, we have tried to address the issues of optimal feature convergence and why certain features get picked over the other. The evaluation metric used for the controller reward function is AUC (area under the curve). A small change in misclassification can cause a significant change in AUC, as a result, the variance of the reward function is high for the same set of sampled features by the controller. This also affected the reproducibility of our research. Feature space can become large pretty quickly hence it is important to scale the framework. Our initial codebase was in Keras, this significantly affected the performance of the algorithm. Moving it to PyTorch resulted in algorithms speed up by a factor of 3.

4.2 Further Study

In this thesis the architectures of the child model (CNN-LSTM) was fixed to a certain degree, in future work we would like to find custom micro architectures [15] specific to the dataset in question, this leads to the discovery of new deep learning models that use fewer features. Interpreting deep learning models and relationships between features is an active field of study. We would ideally like to find new dependencies between features previously unknown to human experts. In a real-world, each unique feature comes with an associated cost, we cannot assume all features cost the same. Dynamic cost-based feature selection is definitely a research problem we can explore especially in the medical field where collecting more features exponentially increases the cost of diagnosis or treatment.

In the future, we would like to work on reducing the time taken to find optimal controller policy (exploring other policy gradient algorithms like Actor-Critic) and to experiment with additional datasets to further validate our results.

REFERENCES

[1] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” *CoRR*, vol. abs/1611.01578, 2016.

[2] Y. Saeys, I. Inza, and P. Larrañaga, “A review of feature selection techniques in bioinformatics,” *bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[3] B. Gregorutti, B. Michel, and P. Saint-Pierre, “Correlation and variable importance in random forests,” *Statistics and Computing*, vol. 27, no. 3, pp. 659–678, 2017.

[4] A. J. Fisher, C. Rudin, and F. Dominici, “All models are wrong, but many are useful: Learning a variable’s importance by studying an entire class of prediction models simultaneously.,” 2019.

[5] Q. Zhou, H. Zhou, and T. Li, “Cost-sensitive feature selection using random forest: Selecting low-cost subsets of informative features,” *Knowledge-based systems*, vol. 95, pp. 1–11, 2016.

[6] V. C. Raykar, B. Krishnapuram, J. Bi, M. Dundar, and R. B. Rao, “Bayesian multiple instance learning: automatic feature selection and inductive transfer,” in *Proceedings of the 25th international conference on Machine learning*, pp. 808–815, 2008.

[7] S. M. Hazrati Fard, A. Hamzeh, and S. Hashemi, “Using reinforcement learning to find an optimal set of features,” *Comput. Math. Appl.*, vol. 66, pp. 1892–1904, Dec. 2013.

[8] H. He and H. D. Iii, “Cost-sensitive dynamic feature selection.”

[9] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, “Large-scale evolution of image classifiers,” in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, (International Convention Centre, Sydney, Australia), pp. 2902–2911, PMLR, 06–11 Aug 2017.

[10] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *arXiv preprint arXiv:1611.02167*, 2016.

[11] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing, "Neural architecture search with bayesian optimisation and optimal transport," in *Advances in Neural Information Processing Systems*, pp. 2016–2025, 2018.

[12] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[13] H. Cai, J. Yang, W. Zhang, S. Han, and Y. Yu, "Path-level network transformation for efficient architecture search," *arXiv preprint arXiv:1806.02639*, 2018.

[14] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[15] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *CoRR*, vol. abs/1802.03268, 2018.

[16] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[17] M. LEVY and M. FINK, "Sccm/esicm/accp/ats/sis international sepsis definitions conference," 2003.

[18] M. M. Churpek, F. J. Zadravecz, C. Winslow, M. D. Howell, and D. P. Edelson, "Incidence and prognostic value of the systemic inflammatory response syndrome and organ dysfunctions in ward patients," *American journal of respiratory and critical care medicine*, vol. 192, no. 8, pp. 958–964, 2015.

[19] R. C. Bone, R. A. Balk, F. B. Cerra, R. P. Dellinger, A. M. Fein, W. A. Knaus, R. M. Schein, and W. J. Sibbald, "Definitions for sepsis and organ failure and guidelines for the use of innovative therapies in sepsis," *Chest*, vol. 101, no. 6, pp. 1644–1655, 1992.

[20] M. Singer, C. S. Deutschman, C. W. Seymour, M. Shankar-Hari, D. Annane, M. Bauer, R. Bellomo, G. R. Bernard, J.-D. Chiche, C. M. Coopersmith, *et al.*, "The third international consensus definitions for sepsis and septic shock (sepsis-3)," *Jama*, vol. 315, no. 8, pp. 801–810, 2016.

[21] D. D. Poutsiaka, M. Porto, W. Perry, J. Hudcova, D. Tybor, S. Hadley, S. Doron, J. A. Reich, D. Snydman, and S. Nasraway, "Comparison of the sepsis-2 and sepsis-3 definitions of sepsis and their ability to predict mortality in a prospective intensive care unit cohort," in *Open Forum Infectious Diseases*, vol. 4, p. S602, Oxford University Press, 2017.

[22] J. Ludikhuize, S. M. Smorenburg, S. E. de Rooij, and E. de Jonge, "Identification of deteriorating patients on general wards; measurement of vital parameters and potential effectiveness of the modified early warning score," *Journal of critical care*, vol. 27, no. 4, pp. 424–e7, 2012.

[23] A. McGinley and R. M. Pearse, "A national early warning score for acutely ill patients," 2012.

[24] M. M. Levy, A. Artigas, G. S. Phillips, A. Rhodes, R. Beale, T. Osborn, J.-L. Vincent, S. Townsend, S. Lemeshow, and R. P. Dellinger, "Outcomes of the surviving sepsis campaign in intensive care units in the usa and europe: a prospective cohort study," *The Lancet infectious diseases*, vol. 12, no. 12, pp. 919–924, 2012.

[25] D. C. Angus and T. Van der Poll, "Severe sepsis and septic shock," *New England Journal of Medicine*, vol. 369, no. 9, pp. 840–851, 2013.

[26] J.-L. Vincent, R. Moreno, J. Takala, S. Willatts, A. De Mendonça, H. Bruining, C. Reinhart, P. Suter, and L. Thijs, "The sofa (sepsis-related organ failure assessment) score to describe organ dysfunction/failure," *Intensive care medicine*, vol. 22, no. 7, pp. 707–710, 1996.

[27] J. Gardner-Thorpe, N. Love, J. Wrightson, S. Walsh, and N. Keeling, "The value of modified early warning score (mews) in surgical in-patients: a prospective observational study," *The Annals of The Royal College of Surgeons of England*, vol. 88, no. 6, pp. 571–575, 2006.

[28] G. B. Smith, D. R. Prytherch, P. Meredith, P. E. Schmidt, and P. I. Featherstone, “The ability of the national early warning score (news) to discriminate patients at risk of early cardiac arrest, unanticipated intensive care unit admission, and death,” *Resuscitation*, vol. 84, no. 4, pp. 465–470, 2013.

[29] C. H. Tang, P. M. Middleton, A. V. Savkin, G. S. Chan, S. Bishop, and N. H. Lovell, “Non-invasive classification of severe sepsis and systemic inflammatory response syndrome using a nonlinear support vector machine: a preliminary study,” *Physiological measurement*, vol. 31, no. 6, p. 775, 2010.

[30] S. Ghosh, J. Li, L. Cao, and K. Ramamohanarao, “Septic shock prediction for icu patients via coupled hmm walking on sequential contrast patterns,” *Journal of biomedical informatics*, vol. 66, pp. 19–31, 2017.

[31] Q. Mao, M. Jay, J. L. Hoffman, J. Calvert, C. Barton, D. Shimabukuro, L. Shieh, U. Chettipally, G. Fletcher, Y. Kerem, *et al.*, “Multicentre validation of a sepsis prediction algorithm using only vital sign data in the emergency department, general ward and icu,” *BMJ open*, vol. 8, no. 1, p. e017833, 2018.

[32] R. A. Lukaszewski, A. M. Yates, M. C. Jackson, K. Swingler, J. M. Scherer, A. Simpson, P. Sadler, P. McQuillan, R. W. Titball, T. J. Brooks, *et al.*, “Presymptomatic prediction of sepsis in intensive care unit patients,” *Clin. Vaccine Immunol.*, vol. 15, no. 7, pp. 1089–1094, 2008.

[33] H. J. Kam and H. Y. Kim, “Learning representations for the early detection of sepsis with deep neural networks,” *Computers in biology and medicine*, vol. 89, pp. 248–255, 2017.

[34] J. J. Trinckes Jr, *The definitive guide to complying with the HIPAA/HITECH privacy and security rules*. CRC Press, 2012.

[35] Johnson, A. EW, D. J. Stone, L. A. Celi, and T. J. Pollard, “The mimic code repository: enabling reproducibility in critical care research,” *Journal of the American Medical Informatics Association*, 2017.

[36] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, “A survey of the recent architectures of deep convolutional neural networks,” *arXiv preprint arXiv:1901.06032*, 2019.

[38] T. Chen, R. Xu, Y. He, and X. Wang, “Improving sentiment analysis via sentence type classification using bilstm-crf and cnn,” *Expert Systems with Applications*, vol. 72, pp. 221–230, 2017.

[39] H. M. Giannini, J. C. Ginestra, C. Chivers, M. Draugelis, A. Hanish, W. D. Schweickert, B. D. Fuchs, L. Meadows, M. Lynch, P. J. Donnelly, *et al.*, “A machine learning algorithm to predict severe sepsis and septic shock: Development, implementation, and impact on clinical practice,” *Critical care medicine*, vol. 47, no. 11, pp. 1485–1492, 2019.

[40] Q. Mao, M. Jay, J. L. Hoffman, J. Calvert, C. Barton, D. Shimabukuro, L. Shieh, U. Chettipally, G. Fletcher, Y. Kerem, Y. Zhou, and R. Das, “Multicentre validation of a sepsis prediction algorithm using only vital sign data in the emergency department, general ward and icu,” *BMJ Open*, vol. 8, no. 1, 2018.

APPENDIX A

APPENDIX FOR CHAPTER 3

A.1 Controller A

A.1.1 Controller Design

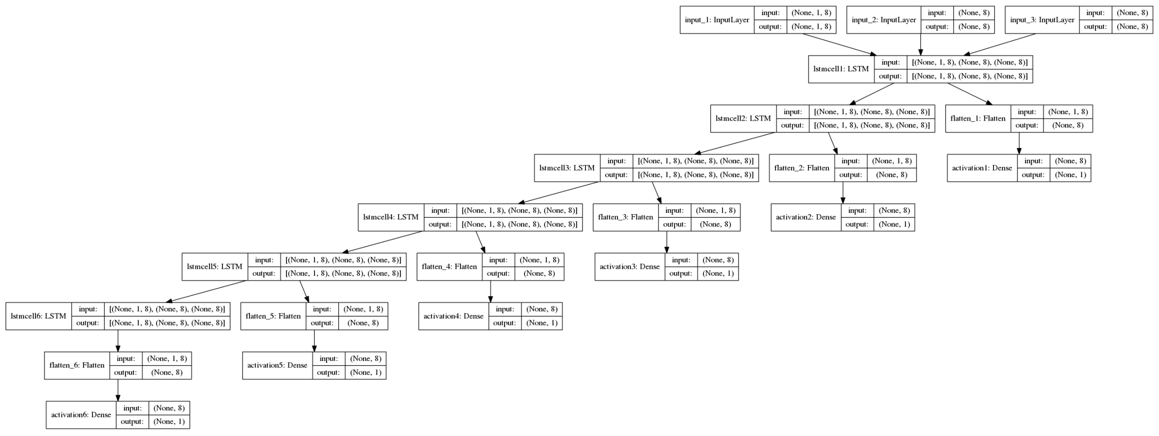


Figure A.1: Controller 'A' design for 6 features. The design is similar for 14 features.

A.1.2 Controller Hyperparameters

Adam optimizer with learning rate = 0.001 and decay = 0.001. Each controller cycle (episode) generates one sample and trained for 8 epochs.

A.1.3 DataSet-I

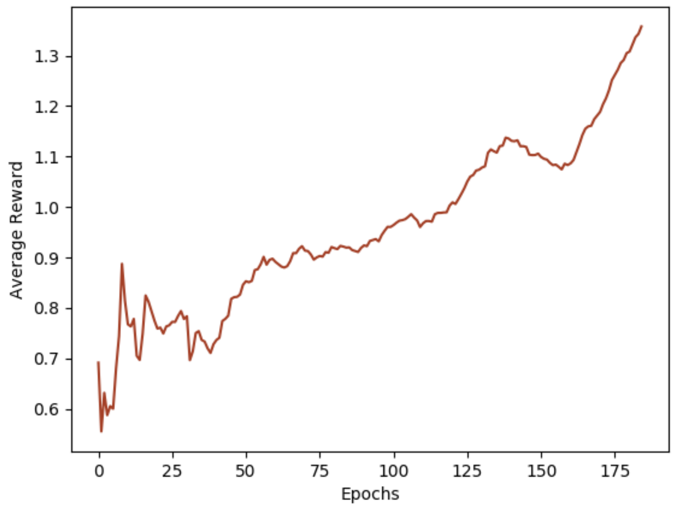


Figure A.2: Reward plot for Dynamic Feature Selection using Dataset-I (4/6 features) for Controller A.

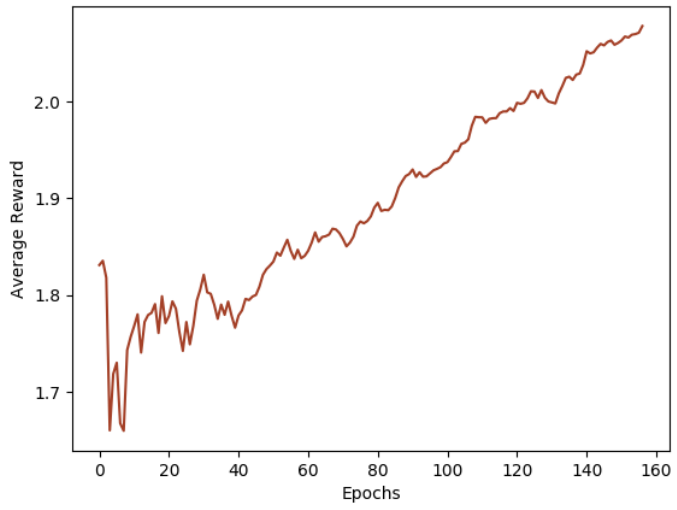


Figure A.3: Reward plot for Hyperparameter Selection using Dataset-I (4/6 features).

A.1.4 DataSet-II

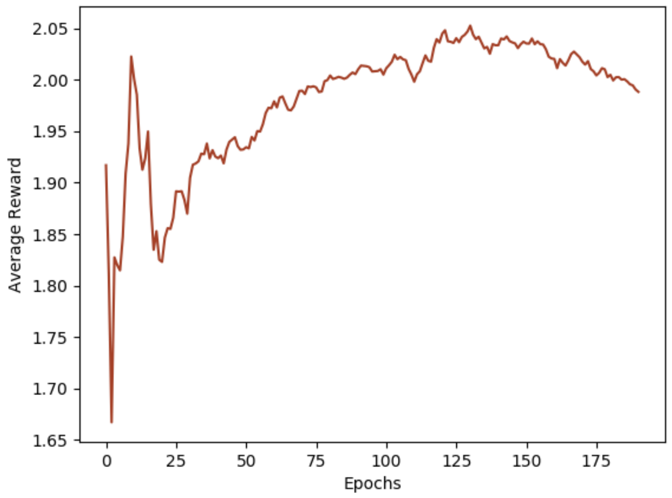


Figure A.4: Reward plot for Dynamic Feature Selection using Dataset-II (9/14 features) for Controller A.

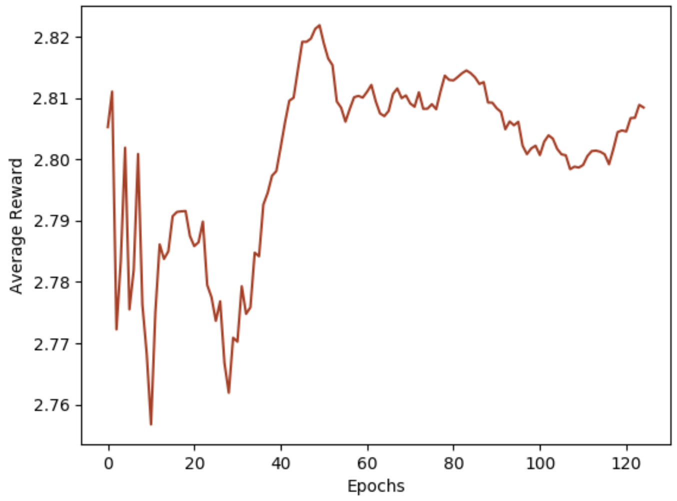
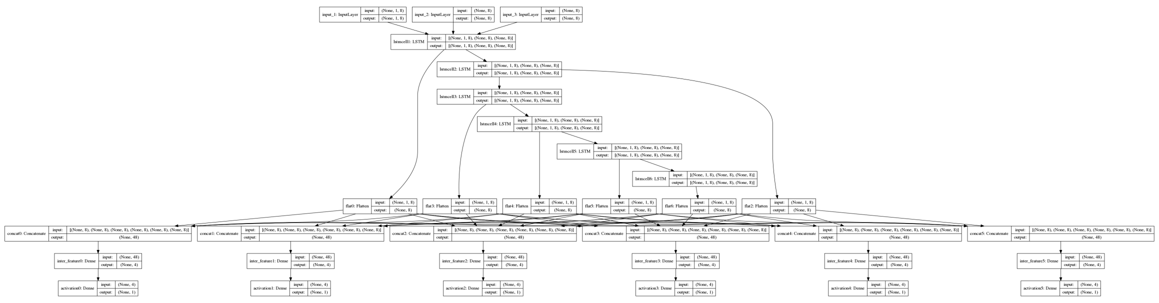


Figure A.5: Reward plot for Hyperparameter Selection using Dataset-II (9/14 features).

A.2 Controller B

A.2.1 Controller Design



input_1: InputLayer | input: (None, 1, 8) | output: (None, 1, 8)
input_2: InputLayer | input: (None, 8) | output: (None, 8)
input_3: InputLayer | input: (None, 8) | output: (None, 8)
lstmcell1: LSTM | input: [(None, 1, 8), (None, 8), (None, 8)] | output: [(None, 1, 8), (None, 8), (None, 8)]
lstmcell2: LSTM | input: [(None, 1, 8), (None, 8), (None, 8)] | output: [(None, 1, 8), (None, 8), (None, 8)]
lstmcell3: LSTM | input: [(None, 1, 8), (None, 8), (None, 8)] | output: [(None, 1, 8), (None, 8), (None, 8)]
lstmcell4: LSTM | input: [(None, 1, 8), (None, 8), (None, 8)] | output: [(None, 1, 8), (None, 8), (None, 8)]
lstmcell5: LSTM | input: [(None, 1, 8), (None, 8), (None, 8)] | output: [(None, 1, 8), (None, 8), (None, 8)]
lstmcell6: LSTM | input: [(None, 1, 8), (None, 8), (None, 8)] | output: [(None, 1, 8), (None, 8), (None, 8)]
flat0: Flatten | input: (None, 1, 8) | output: (None, 8)
flat3: Flatten | input: (None, 1, 8) | output: (None, 8)
flat4: Flatten | input: (None, 1, 8) | output: (None, 8)
flat5: Flatten | input: (None, 1, 8) | output: (None, 8)
flat6: Flatten | input: (None, 1, 8) | output: (None, 8)
flat2: Flatten | input: (None, 1, 8) | output: (None, 8)
concat0: Concatenate | input: [(None, 8), (None, 8), (None, 8), (None, 8), (None, 8), (None, 8)] | output: (None, 48)
concat1: Concatenate | input: [(None, 8), (None, 8), (None, 8), (None, 8), (None, 8), (None, 8)] | output: (None, 48)
concat2: Concatenate | input: [(None, 8), (None, 8), (None, 8), (None, 8), (None, 8), (None, 8)] | output: (None, 48)
concat3: Concatenate | input: [(None, 8), (None, 8), (None, 8), (None, 8), (None, 8), (None, 8)] | output: (None, 48)
concat4: Concatenate | input: [(None, 8), (None, 8), (None, 8), (None, 8), (None, 8), (None, 8)] | output: (None, 48)
concat5: Concatenate | input: [(None, 8), (None, 8), (None, 8), (None, 8), (None, 8), (None, 8)] | output: (None, 48)
inter_feature0: Dense | input: (None, 48) | output: (None, 4)
inter_feature1: Dense | input: (None, 48) | output: (None, 4)
inter_feature2: Dense | input: (None, 48) | output: (None, 4)
inter_feature3: Dense | input: (None, 48) | output: (None, 4)
inter_feature4: Dense | input: (None, 48) | output: (None, 4)
inter_feature5: Dense | input: (None, 48) | output: (None, 4)
activation0: Dense | input: (None, 4) | output: (None, 1)
activation1: Dense | input: (None, 4) | output: (None, 1)
activation2: Dense | input: (None, 4) | output: (None, 1)
activation3: Dense | input: (None, 4) | output: (None, 1)
activation4: Dense | input: (None, 4) | output: (None, 1)
activation5: Dense | input: (None, 4) | output: (None, 1)

Figure A.6: Controller 'B' design for 6 features. The design is similar for 14 and 133 features.

A.2.2 Controller Hyperparameters

Adam optimizer with learning rate = 0.001 and decay = 0. Each controller cycle (episode) generates one sample and trained for 5 epochs.

A.2.3 DataSet-I

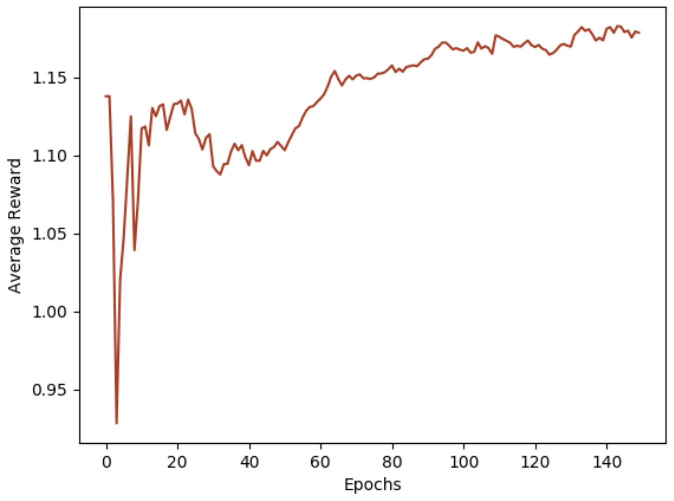


Figure A.7: Reward plot for Dynamic Feature Selection using Dataset-I (4/6 features) for Controller B.

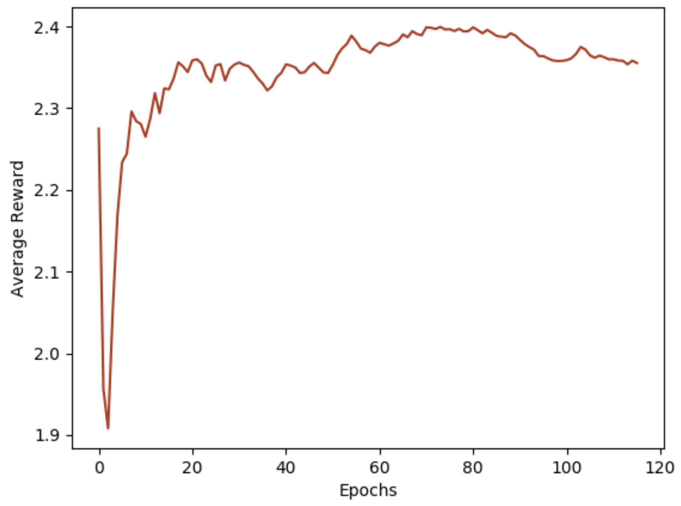


Figure A.8: Reward plot for Hyperparameter Selection using Dataset-I (4/6 features).

A.2.4 DataSet-II

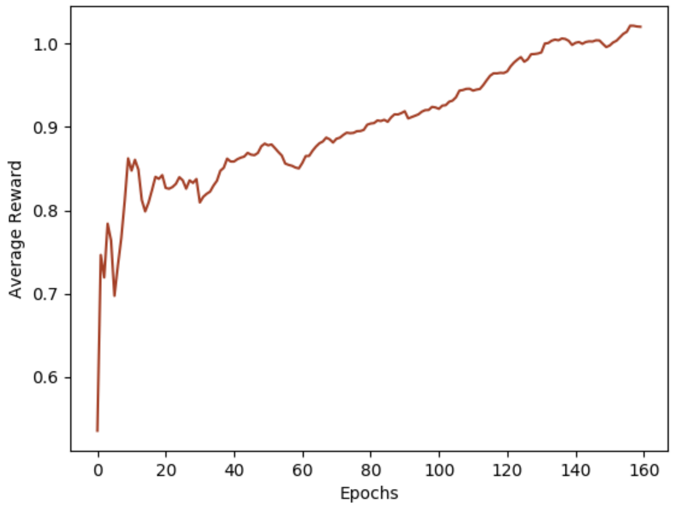


Figure A.9: Reward plot for Dynamic Feature Selection using Dataset-I (9/14 features) for Controller B.

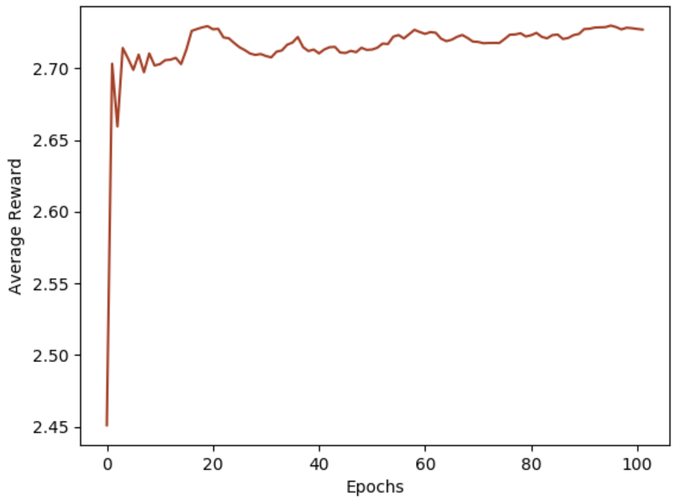


Figure A.10: Reward plot for Hyperparameter Selection using Dataset-I (9/14 features).

A.2.5 DataSet-III

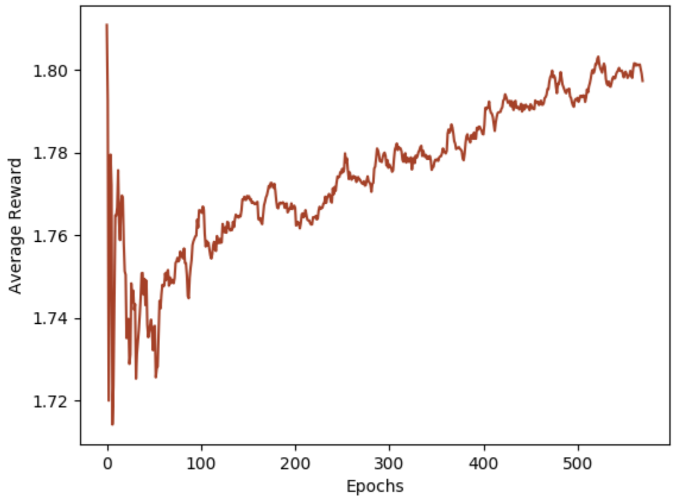


Figure A.11: Reward plot for Dynamic Feature Selection using Dataset-I (110/133 features) for Controller B.

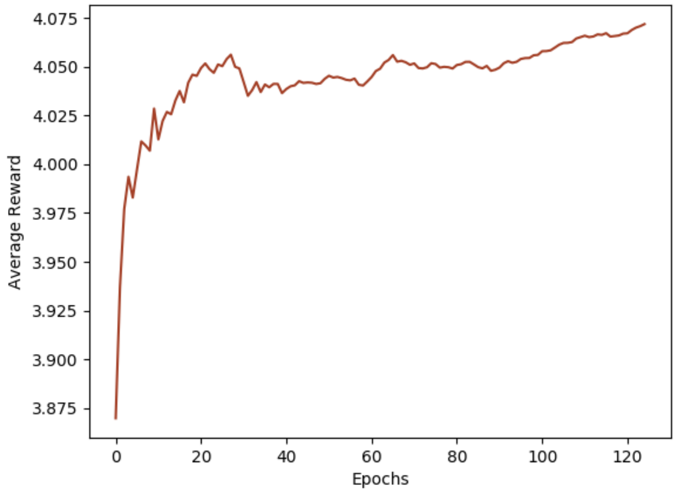
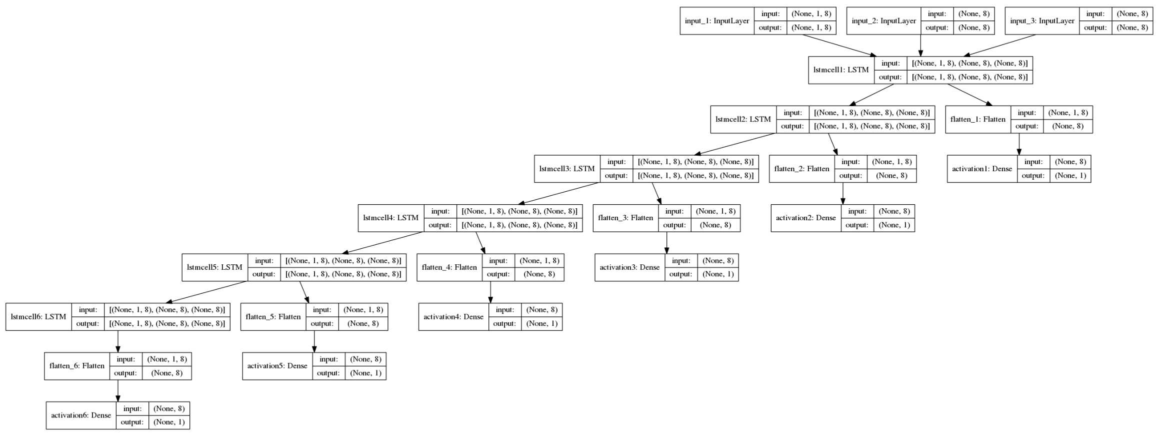


Figure A.12: Reward plot for Hyperparameter Selection using Dataset-I (110/133 features).

A.3 Hyperparameter Controller



input_1: InputLayer | input: (None, 1, 8) | output: (None, 1, 8)
input_2: InputLayer | input: (None, 8) | output: (None, 8)
input_3: InputLayer | input: (None, 8) | output: (None, 8)
lstmcell1: LSTM | input: [(None, 1, 8), (None, 8), (None, 8)] | output: [(None, 1, 8), (None, 8), (None, 8)]
lstmcell2: LSTM | input: [(None, 1, 8), (None, 8), (None, 8)] | output: [(None, 1, 8), (None, 8), (None, 8)]
flatten_1: Flatten | input: (None, 1, 8) | output: (None, 8)
lstmcell3: LSTM | input: [(None, 1, 8), (None, 8), (None, 8)] | output: [(None, 1, 8), (None, 8), (None, 8)]
flatten_2: Flatten | input: (None, 1, 8) | output: (None, 8)
activation1: Dense | input: (None, 8) | output: (None, 1)
lstmcell4: LSTM | input: [(None, 1, 8), (None, 8), (None, 8)] | output: [(None, 1, 8), (None, 8), (None, 8)]
flatten_3: Flatten | input: (None, 1, 8) | output: (None, 8)
activation2: Dense | input: (None, 8) | output: (None, 1)
lstmcell5: LSTM | input: [(None, 1, 8), (None, 8), (None, 8)] | output: [(None, 1, 8), (None, 8), (None, 8)]
flatten_4: Flatten | input: (None, 1, 8) | output: (None, 8)
activation3: Dense | input: (None, 8) | output: (None, 1)
lstmcell6: LSTM | input: [(None, 1, 8), (None, 8), (None, 8)] | output: [(None, 1, 8), (None, 8), (None, 8)]
flatten_5: Flatten | input: (None, 1, 8) | output: (None, 8)
activation4: Dense | input: (None, 8) | output: (None, 1)
flatten_6: Flatten | input: (None, 1, 8) | output: (None, 8)
activation5: Dense | input: (None, 8) | output: (None, 1)
activation6: Dense | input: (None, 8) | output: (None, 1)

Figure A.13: Hyperparameter Controller Design.

A.3.1 Hyperparameter Search Space

- ’lstm units’ : [8,16,32]
- ’lstm units’ : [8,16,32],
- ’lstm layers’ : [2, 3, 4],
- ’lstm activation’ : [’relu’,’tanh’],
- ’cnn filter’: [32, 64],
- ’cnn kernel’: [1, 2, 3],
- ’cnn hidden layers’: [1, 2],
- ’dense layers’: [2, 3],
- ’dense units’: [8, 16, 32]

A.3.2 Controller Hyperparameters

Adam optimizer with learning rate = 0.001 and decay = 0.001. Each controller cycle (episode) generates one sample and trained for 8 epochs.

A.4 CNN-LSTM Child Model Parameters

A.4.1 Feature Selection (Controller A) + Hyperparameter Tuning + Dataset-I

1 CNN layer (filter size of 64, kernel size of 3) + 3 LSTM layers (32 hidden units + tanh activation) + 3 Dense layers (16 hidden units)

A.4.2 Feature Selection (Controller A) + Hyperparameter Tuning + Dataset-II

2 CNN layer (filter size of 32, kernel size of 2) + 3 LSTM layers (16 hidden units + tanh activation) + 3 Dense layers (16 hidden units)

A.4.3 Feature Selection (Controller B) + Hyperparameter Tuning + Dataset-I

1 CNN layer (filter size of 64, kernel size of 3) + 3 LSTM layers (32 hidden units + tanh activation) + 3 Dense layers (16 hidden units)

A.4.4 Feature Selection (Controller B) + Hyperparameter Tuning + Dataset-II

1 CNN layer (filter size of 64, kernel size of 1) + 4 LSTM layers (32 hidden units + tanh activation) + 2 Dense layers (32 hidden units)

A.4.5 Feature Selection (Controller B) + Hyperparameter Tuning + Dataset-III

1 CNN layer (filter size of 64, kernel size of 2) + 3 LSTM layers (32 hidden units + tanh activation) + 3 Dense layers (16 hidden units)

APPENDIX B

COMPLETE FEATURE LIST

B.1 Original Feature Set

Original Feature Set contains the actual set of features available in DataSet-I, DataSet-II and DataSet-III used in this research work. The features are listed in their original order.

B.1.1 DataSet-I (6 features)

Systolic Blood Pressure Abp Mean, Diastolic Blood Pressure Mean, Heart Rate, Body Temperature, Respiratory Rate, SpO2 Peripheral.

B.1.2 DataSet-II (14 features)

Systolic Blood Pressure Abp Mean, Diastolic Blood Pressure Mean, Heart Rate, Body Temperature, Respiratory Rate, SpO2 Peripheral, Albumin, Lymphocytes, Calcium Total, Magnesium, Phosphate, PT, PTT, Serum Transferrin.

B.1.3 DataSet-III (133 features)

Gastric Gastric Tube, Stool Out Stool, Urine Out Incontinent, Chest Tube #1, Chest Tube #2, OR EBL, Pre-Admission, Fresh Frozen Plasma, Lorazepam (Ativan), Calcium Gluconate, Midazolam (Versed), Phenylephrine, Furosemide (Lasix), Hydralazine, Norepinephrine, Magnesium Sulfate, Nitroglycerin, Insulin-Regular, Morphine Sulfate, Potassium Chloride, Packed Red Blood Cells, Gastric Meds, D5 1/2NS, LR, Solution, Sterile Water, Piggyback, OR Crystalloid Intake, OR Cell Saver Intake, PO Intake, GT Flush, KCL (Bolus), Magnesium Sulfate (Bolus), Hematocrit, Platelet Count, Hemoglobin, MCHC, MCH, MCV, Red Blood Cells, RDW, Chloride, Anion Gap, Creatinine, Glucose, Magnesium, Calcium Total, Phosphate, INR (PT), PT, PTT, Lymphocytes, Monocytes, Neutrophils, Basophils, Eosinophils, PH, Base Excess, Calculated Total CO2, PCO2, Specific Gravity, Lactate, Alanine Aminotransferase (ALT), Asparate Aminotransferase (AST), Alkaline Phosphatase, Albumin, Aspirin, Bisacodyl, Docusate Sodium, Humulin-R Insulin, Metoprolol Tartrate, Pantoprazole, Arterial Blood Pressure Diastolic, Arterial Blood Pressure Mean, Respiratory Rate, Alarms On, MinuteVolumeAlarm-Low, Peakinsp. Pressure, PEEPset, Minute Volume, Tidal Volume (observed), Minute Volume Alarm-High, Mean Airway Pressure, Central Venous Pressure, Respiratory Rate (Set), Pulmonary Artery Pressure Mean, O2Flow, Glucosefingerstick, Heart Rate Alarm-Low, Pulmonary Artery Pressure Systolic, Tidal Volume (set), Pulmonary Artery Pressure Diastolic, SpO2 DesatLimit, RespAlarm-High, Skin Care, white Blood Cells Count Mean, Sodium Level Mean, Gcseyes, Serum Bicarbonate Level Mean, Systolic Blood Pressure Abp Mean, Potassium Level Mean, Heart Rate, Serum Urea Nitrogen Level, Urinary Output Sum, FiO2, Body Temperature, Bilirubin Level, Gcsmotor, Gcsverbal, PaO2, IE Ratio Mean, Ph, Arterial Pressure Mean, Midazolam, Weight, SpO2 Peripheral, Epinephrine, Glucose, Hgb, Propofol, Creatinine, Chloride, Fentanyl, Diastolic Blood Pressure Mean, Norepinephrine, Dopamine, Height, Serum Transferrin, Platelet, Phenylephrine, Peep, Respiratory Rate, Total Cholesterol.

B.2 Controller A

This section lists all the features selected (in order) by Controller A for the dynamic feature selection task using DataSet-I and DataSet-II.

B.2.1 DataSet-I (4 / 6 features)

Systolic Blood Pressure Abp Mean, Diastolic Blood Pressure Mean, Body Temperature, Respiratory Rate.

B.2.2 DataSet-II (9 / 14 features)

Heart Rate, Body Temperature, Respiratory Rate, SpO2 Peripheral, Lymphocytes, Magnesium, Phosphate, PT, PTT.

B.3 Controller B

This section lists all the features selected (in order) by Controller B for the dynamic feature selection task using DataSet-I, DataSet-II and DataSet-III.

B.3.1 DataSet-I (4 / 6 features)

Systolic Blood Pressure Abp Mean, Diastolic Blood Pressure Mean, Respiratory Rate, SpO2 Peripheral.

B.3.2 DataSet-II (9 / 14 features)

Heart Rate, Body Temperature, Respiratory Rate, Albumin, Lymphocytes, Magnesium, PT, PTT, Serum Transferrin.

B.3.3 DataSet-III (110 / 133 features)

Gastric Gastric Tube, Stool Out Stool, Urine Out Incontinent, Chest Tube #1, Chest Tube #2, OR EBL, Pre-Admission, Fresh Frozen Plasma, Lorazepam (Ativan), Calcium Gluconate, Furosemide (Lasix), Hydralazine, Magnesium Sulfate, Nitroglycerin, Insulin-Regular, Morphine Sulfate, Potassium Chloride, Packed Red Blood Cells, D5 1/2NS, LR, Solution, Sterile Water, Piggyback, OR Crystalloid Intake, OR Cell Saver Intake, PO Intake, GT Flush, KCL (Bolus), Hematocrit, Platelet Count, Hemoglobin, MCHC, MCH, MCV, Chloride, Anion Gap, Creatinine, Glucose, Magnesium, Calcium Total, Phosphate, INR (PT), PT, PTT, Lymphocytes, Monocytes, Neutrophils, Basophils, Eosinophils, PH, Base Excess, PCO2, Specific Gravity, Lactate, Asparate Aminotransferase (AST), Alkaline Phosphatase, Aspirin, Docusate Sodium, Humulin-R Insulin, Metoprolol Tartrate, Pantoprazole, Arterial Blood Pressure Diastolic, Arterial Blood Pressure Mean, Respiratory Rate, Alarms On, MinuteVolumeAlarm-Low, Peakinsp. Pressure, PEEPset, Minute Volume, Tidal Volume (observed), Minute Volume Alarm-High, Mean Airway Pressure, Central Venous Pressure, Respiratory Rate (Set), Pulmonary Artery Pressure Mean, O2Flow, Glucosefingerstick, Heart Rate Alarm-Low, Pulmonary Artery Pressure Systolic, Tidal Volume (set), SpO2 DesatLimit, RespAlarm-High, white Blood Cells Count Mean, Gcseyes, Serum Bicarbonate Level Mean, Systolic Blood Pressure Abp Mean, Heart Rate, Serum Urea Nitrogen Level, Urinary Output Sum, Body Temperature, Bilirubin Level, Gcsmotor, Gcsverbal, IE Ratio Mean','Arterial Pressure Mean, Midazolam, SpO2 Peripheral, Epinephrine, Hgb, Propofol, Fentanyl, Diastolic Blood Pressure Mean, Norepinephrine, Dopamine, Height, Serum Transferrin, Platelet, Peep, Respiratory Rate, Total Cholesterol.