University of California
Santa Barbara

Data Mining in Neuroscience and Healthcare

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

Yun Zhao

Committee in charge:

Professor Linda Petzold, Chair
Professor Yu-Xiang Wang
Professor Tao Yang

June 2021

The Dissertation of Yun Zhao is approved.

Professor Yu-Xiang Wang

Professor Tao Yang

Professor Linda Petzold, Committee Chair

April 2021

Data Mining in Neuroscience and Healthcare

by

Yun Zhao

To my parents,
Yueying Li and Jianbing Zhao, who provided me with
unconditional love and support

Acknowledgements

First and foremost, my deepest gratitude goes to my Ph.D. advisor, Prof. Linda Petzold. I'm grateful to have benefited tremendously from her broad knowledge in the field, her trust and her attention to details. Prof. Linda led me to the research areas of data mining, neuroscience and healthcare and provided me with endless support. I have always felt enlightened when she generously shared her invaluable experience in her research career and life. I always remembered the details she shared with me every Monday happy hour, which makes my best activities during the pandemic. She inspired me in so many ways that helped shape who I am today. I am truly thankful for her guidance.

Next I would like to thank Prof. Yuxiang Wang and Prof. Tao Yang for serving on my committee and providing insightful comments on my dissertation and defense. Prof. Yuxiang sometimes challenge me to make me think harder and deeper, which I appreciate a lot. Prof. Tao gave me lots of valuable advice for my professional career planning.

I am also very lucky to have worked with many brilliant people. I owe my thanks to my collaborators: Prof. Paul Hansma and Dr. Ken Tovar. Not only did they help me significantly in every piece of my work, but also I learned a lot from them during our collaborations. I also want to thank my lab mates and friends who have inspired me and supported me. My Ph.D. journey was a very pleasant experience because of them.

Last but not least, I would like to express my special thanks to my family, my friends, and my loved ones. They always show care and support unconditionally, which greatly encourage me to overcome any barriers throughout my Ph.D. life.

Curriculum Vitæ

Yun Zhao

Education

2021	Ph.D. in Computer Science, University of California, Santa Barbara.
2020	M.S. in Computer Science, University of California, Santa Barbara.
2016	M.E. in Electronic Engineering, Tsinghua University.
2012	B.A. in Electronic Engineering, Zhejiang University.

Professional Experience

Summer 2020	Machine Learning Software Engineer Intern, Facebook, Menlo Park, CA. Topic: Text Asset Selection in Ads Creation

Publications

Yun Zhao, Qinghang Hong, Xinlu Zhang, Yu Deng, Yuqing Wang, PaulK. Hansma, and Linda Petzold, "BERTSurv: BERT based Survival Models for Predicting Outcomes for Trauma Patients", to appear in ICDM 2021.

Yuqing Wang*, **Yun Zhao***, Rachael Callcut, and Linda Petzold, "Influence of machine learning configurations on Multiple Organ Failure (MOF) Prediction for Trauma Patients", to appear in ICDM 2021.

Yun Zhao, Franklin Ly, Qinghang Hong, Tyler Santander, Henry T. Yang, Paul K. Hansma, and Linda Petzold, How Much Does It Hurt: A Deep Learning Framework for Chronic Pain Score Assessment, in DMBIH 2020, Sorrento, Italy, Nov. 2020.

Yun Zhao, Richard Jiang, Zhenni Xu, Elmer Guzman, Paul K. Hansma and Linda Petzold, Scalable Bayesian Functional Connectivity Inference for Multi-Electrode Array Recordings, in BIOKDD 2020 , San Diego, USA, Aug. 2020.

Yun Zhao, Elmer Guzman, Morgane Audouard, Zhuowei Cheng, Paul K. Hansma, Kenneth S. Kosik, and Linda Petzold, A Deep Learning Framework for Classification of in vitro Multi-Electrode Array Recordings, in ICDM 2019, New York, USA, Jul. 2019.

Yun Zhao, "An Auxiliary Classifier Generative Adversarial Framework for Relation Extraction." arXiv preprint arXiv:1909.05370 (2019).

Yun Zhao, Sheng Zhou, Tianchu Zhao, and Zhisheng Niu, Energy-Efficient Task Offloading for Multiuser Mobile Cloud Computing, in IEEE ICCC 2015, Shenzhen, China, Nov. 2015.

Xinyi Zhang, Shiliang Tang, **Yun Zhao**, Gang Wang, Haitao Zheng and Ben Y. Zhao, Cold Hard E-Cash: Friends and Vendors in the Venmo Digital Payments System, in ICWSM 2017.

Tianchu Zhao, Sheng Zhou, Xueying Guo, **Yun Zhao**, and Zhisheng Niu, A Cooperative Scheduling Scheme of Local Cloud and Internet Cloud for Delay-Aware Mobile Cloud Computing, in IEEE Globecom Workshop 2015, San Diego, CA, USA, Dec. 2015.

Yangtian Yan, Bangcheng Sun, **Yun Zhao**, Zhenhui Huang, Hui Yang, and Jian Song, A Bi-directional Visible Light Communication System Based on DTMB-A, in IEEE VTC 2016, Nanjing, China, May, 2016.

Tianchu Zhao, Sheng Zhou, Xueying Guo, **Yun Zhao**, and Zhisheng Niu, A Cooperative Scheduling Pricing Policy and Computational Resource Provisioning for Delay-aware Mobile Edge Computing, in IEEE ICCC 2016, Xian, China, Jul. 2016.

Yun Zhao, Energy-efficient Resource Allocation in Mobile Cloud Computing, Master Thesis.

In Submission

Xinlu Zhang*, **Yun Zhao***, Rachael Callcut, and Linda Petzold, Multiple Organ Failure Prediction with Classifier-Guided Generative Adversarial Imputation Networks.

Zhuowei Cheng, Franklin Ly, Tyler Santander, Elyes Turki, **Yun Zhao**, Henry Yang, Michael Miller, Paul Hansma, Linda Petzold, Quantification of Chronic Pain from Physiological Data.

Awards and Honors

2016 - 2021	Chancellors Fellowship (Only one in CS Department), UCSB.
2017	Summer Research Project Fellowship, UCSB.
2016	Holbrook Fellowship (annually awarded to 6 freshmen), UCSB.
2015, 2014	Department Set Scholarship (Only one in the Communication Class), Tsinghua University.

Professional Service

Served as reviewer for a number of computer science journals, including:

IEEE Transactions on Green Communications and Networking,

Journal of Communications and Information Networks,

Peer-to-Peer Networking and Applications.

Abstract

Data Mining in Neuroscience and Healthcare

by

Yun Zhao

Statistical methods, and in particular deep learning models, have achieved remarkable success in computer vision, speech recognition, and natural language processing due to the availability of powerful computational resources. Recently, neuroscience and healthcare have entered an exciting new age. Modern recording technologies, like Multi-Electrode Arrays (MEA) and electronic health records (EHR), offer unprecedented opportunities to explore neural systems and to improve health care. At the same time, they present extraordinary computational and statistical challenges. This Ph.D. dissertation presents knowledge we mined from neuroscience data and healthcare data.

In the neuroscience data mining part, we propose a deep learning framework for MEA classification of mouse and human derived induced Pluripotent Stem Cell recordings. We also introduce a scalable Bayesian framework for inference of functional neural networks from MEA data.

In the healthcare mining part, we first perform quantitative analysis on early multiple organ failure (MOF) prediction with comprehensive machine learning (ML) configurations, including data preprocessing (missing value treatment, label balancing, feature scaling), feature selection,classifier choice), and hyperparameter tuning. We introduce BERTSurv, a deep learning survival framework which applies Bidirectional Encoder Representations from Transformers (BERT) as a language representation model on unstructured clinical notes, for mortality prediction and survival analysis. We propose classifier-guided generative adversarial imputation networks (Classifier-GAIN) for MOF

prediction, by incorporating both observed data and label information. Finally, we propose an end-to-end deep learning framework for chronic pain score assessment.

In the end, we summarize the strengths, weaknesses, and implications of our work, and discuss future research directions.

Contents

Chapter 1

Introduction

AI methods, and in particular deep learning models, have achieved remarkable success in computer vision, speech recognition, and natural language processing due to the availability of powerful computational resources. Recently, neuroscience and healthcare have entered an exciting new age. Modern recording technologies in neuroscience, like Multi-Electrode Arrays (MEA), enable simultaneous measurements of thousands of neurons activities. Similarly, more and more electronic health record (EHR) data are available. Such recordings offer an unprecedented opportunity to learn the mechanistics in neuroscience and healthcare, but they also present an extraordinary computational and statistical challenge: How do we make sense of these large scale recordings?

Significant work about trauma have been done by observing and studying individual recordings or a small group of recordings. With the increasing ability to store and manage recording data, and with the development of data mining and machine learning research, more and more attention is going to the application of data mining and machine learning techniques on recordings at a much larger scale. In this thesis, we demonstrate our uses of the data-driven approaches to study MEA and electronic health recordings.

1.1 Data Mining in Multi-Electrode Array Recordings

MEAs have been widely used to record neuronal activities, which could be used in the diagnosis of gene defects and drug effects. In Chapter 2, we first address the problem of classifying *in vitro* MEA recordings of mouse and human neuronal cultures from different genotypes, where there is no easy way to directly utilize raw sequences as inputs to train an end-to-end classification model [1]. While carefully extracting some features by hand could partially solve the problem, this approach suffers from obvious drawbacks such as difficulty of generalizing. We propose a deep learning framework to address this challenge. Our approach correctly classifies neuronal culture data prepared from two different genotypes — a mouse Knockout of the delta-catenin gene and human induced Pluripotent Stem Cell-derived neurons from Williams syndrome. By splitting the long recordings into short slices for training, and applying Consensus Prediction during testing, our deep learning approach improves the prediction accuracy by 16.69% compared with feature based Logistic Regression for mouse MEA recordings. We further achieve an accuracy of 95.91% using Consensus Prediction in one subset of mouse MEA recording data, which were all recorded at the age of six days *in vitro*. As high-density MEA recordings become more widely available, this approach could be generalized for classification of neurons carrying different mutations and classification of drug responses.

Also, with the advancement of MEA technology, it has become increasingly crucial to develop statistical tools for analyzing multiple neuronal activity as a network. In Chapter 3, we propose a scalable Bayesian framework for inference of functional networks from MEA data [2]. Our framework makes use of the hierarchical structure of networks of neurons. We split the large scale recordings into smaller local networks for network inference, which not only eases the computational burden from Bayesian sampling but

also provides useful insights on regional connections in organoids and brains. We speed up the expensive Bayesian sampling process by using parallel computing. Experiments on both synthetic datasets and large-scale real-world MEA recordings show the effectiveness and efficiency of the scalable Bayesian framework. Inference of networks from controlled experiments exposing neural cultures to cadmium presents distinguishable results and further confirms the utility of our framework.

1.2 Data Mining in Healthcare

Multiple organ failure (MOF) is a life-threatening condition. Due to its urgency and high mortality rate, early detection is critical for clinicians to provide appropriate treatment. In Chapter 4, we perform quantitative analysis on early MOF prediction with comprehensive machine learning (ML) configurations, including data preprocessing (missing value treatment, label balancing, feature scaling), feature selection, classifier choice, and hyperparameter tuning [3]. Results show that classifier choice impacts both the performance and variation the most, among all of the configurations. In general, complex classifiers including ensemble methods can provide better performance than simple classifiers. However, blindly pursuing complex classifiers is unwise as it also brings the risk of greater performance variation.

An important challenge in the application of machine learning models to electronic health records (EHRs) is the pervasiveness of missing values. Most existing imputation methods are applied in the data preprocessing phase, failing to capture the relationship between data and outcome for downstream predictions. In Chapter 5, we propose classifier-guided generative adversarial imputation networks (Classifier-GAIN) for MOF prediction, by incorporating both observed data and label information. Specifically, the classifier takes imputed values from the generator to predict task outcomes and provides

additional supervision signals to the generator by joint training. The classifier-guided generator (imputer) can impute missing values with label-awareness during training, which can improve the classifier's performance during inference. We conduct experiments showing that our approach consistently outperforms classical and state-of-art neural baselines across a range of missing data scenarios and evaluation metrics.

Survival analysis is a technique to predict the times of specific outcomes, and is widely used in predicting the outcomes for intensive care unit (ICU) trauma patients. Recently, deep learning models have drawn increasing attention in healthcare. However, there is a lack of deep learning methods that can model the relationship between measurements, clinical notes and mortality outcomes. In Chapter 6, we introduce BERTSurv, a deep learning survival framework which applies Bidirectional Encoder Representations from Transformers (BERT) as a language representation model on unstructured clinical notes, for mortality prediction and survival analysis [4]. We also incorporate clinical measurements in BERTSurv. With binary cross-entropy (BCE) loss, BERTSurv can predict mortality as a binary outcome (mortality prediction). With partial log-likelihood (PLL) loss, BERTSurv predicts the probability of mortality as a time-to-event outcome (survival analysis). We apply BERTSurv on Medical Information Mart for Intensive Care III (MIMIC III) trauma patient data. For mortality prediction, BERTSurv obtained an area under the curve of receiver operating characteristic curve (AUC-ROC) of 0.86, which is an improvement of 3.6% over baseline of multilayer perceptron (MLP) without notes. For survival analysis, BERTSurv achieved a concordance index (C-index) of 0.7. In addition, visualizations of BERT's attention heads help to extract patterns in clinical notes and improve model interpretability by showing how the model assigns weights to different inputs.

Chronic pain is defined as pain that lasts or recurs for more than 3 to 6 months, often long after the injury or illness that initially caused the pain has healed. The

gold standard for chronic pain assessment remains self report and clinical assessment via a biopsychosocial interview, since there has been no device that can measure it. A device to measure pain would be useful not only for clinical assessment, but potentially also as a biofeedback device leading to pain reduction. In Chapter 7, we propose an end-to-end deep learning framework for chronic pain score assessment [5]. Our deep learning framework splits the long time-course data samples into shorter sequences, and uses Consensus Prediction to classify the results. We evaluate the performance of our framework on two chronic pain score datasets collected from two iterations of prototype Pain Meters that we have developed to help chronic pain subjects better understand their health condition.

Part I

Data Mining in Multi-Electrode Array Recordings

Chapter 2

Classification of Multi-Electrode Array Recordings

2.1 Introduction

Deep learning models have achieved remarkable success in computer vision [6], speech recognition [7], natural language processing [8] and the game of Go [9]. Recently there has been increasing interest in using deep learning in end-to-end neuroscience data analysis [10, 11, 12]. Inspired by biology, deep learning models share many common properties with neuron functions. Deep learning models enable the extraction of information from action potential recordings of neuron activity, playing a vital role in several important neuron-based research and application areas [13].

Convolutional neural networks (CNN) can learn local patterns in data by using convolution filters as their key components [14]. Originally developed for computer vision, CNN models have recently been shown to be effective for neuroscience data analysis. Deep learning has recently been used to identify abnormal EEG signals [11]. In [10], researchers designed an end-to-end EEG decoding for movement-related information using

deep CNNs. With the latest development in fabrication of MEAs, a CNN was used to classify different neuronal cell types using simulated in-vivo extracellular recordings [15]. However, most of the work in this area has focused on simulated data [15, 16] since the experimental *in vitro* recordings are too noisy and there are not sufficient training samples for deep learning models. Researchers have also manually extracted features for deep learning training [15, 16]. However, this does not fully exploit the deep learning model's ability of end-to-end learning, which learns from the raw data without any prior feature selection.

MEAs with advanced neural probes have been widely utilized to measure neuronal activity by recording local field potential [17]. Since the same units are measured on multiple recording sites, MEA recordings provide rich spatial information, which could be used to help diagnose diseases and genetic abnormalities. Our objective in this work has been to develop a deep learning framework which can distinguish MEA recordings of different genotypes. For example, delta-catenin is a crucial brain-specific protein of the adherens junction complex that localizes to the postsynaptic and dendritic compartments. It is enriched in dendrites and can be localized to the post-synaptic compartment. Recent studies indicate that delta-catenin is required for the maintenance of neural structure and function in the mature cortex [18, 19, 20]. Williams syndrome (WS) is a neurodevelopmental disorder caused by a genomic deletion of about 28 genes [21, 22]. As a result of this hemideletion, the subjects display a characteristic phenotype with mild to moderate intellectual disability as well as behavioral features such as an outgoing personality and conserved communication skills. Studying those genes is of particular interest in order to decipher the social behaviors in humans [23].

In this chapter, we propose an end-to-end CNN architecture to classify *in vitro* MEA recordings with different genotypes. We test our framework on mouse recordings to classify Wild Type and delta-catenin Knockout. We also attempt to classify human

derived induced Pluripotent Stem Cell (iPSC) neuron cultures from Williams syndrome versus Control cultures. We split the long recordings into smaller slices for training to provide more training samples, and then apply Consensus Prediction during testing.

The key contributions of this chapter include:

1) We propose a CNN based model to classify the genotype of *in vitro* MEA recordings, which outperforms Logistic Regression by 16.69%. To the best of our knowledge, this is the first work using deep leaning to classify *in vitro* MEA recordings.

2) We split the long recordings into smaller slices for training, which not only eases the burden on GPU memory but also provides many training samples for deep learning models.

3) We define Consensus Prediction as the majority voting result of the sampled short slices for testing, since not all of the short slices can be expected to contain enough useful information. We achieve an accuracy of 95.91% using Consensus Prediction in one subset of MEA recording data, which were all recorded at 6 days *in vitro* (DIV).

The rest of this chapter is organized as follows. Section 2.2 describes how our MEAs are recorded and introduces the classification problem. We delineate the deep learning model in Section 2.3 and describe the experimental setup in Section 2.4. Evaluation and discussion are provided in Sections 2.5 and 2.6, respectively.

2.2 Data Collection and Classification

2.2.1 Mouse Neuron Culture

Commercial MEAs (MultiChannel Systems) were sterilized with UV irradiation for $>$ 30 minutes, incubated with poly-L-lysine(0.1 mg/ml) solution for at least one hour at $37°C$, rinsed several times with sterile deionized water and allowed to dry before

cell plating. Wild-type mice were in a C57BL/6 background and littermate controls were obtained by breeding heterozygote male and female delta-catenin transgenic mice. For the delta-catenin transgenic mice, a targeted mutation in the delta-catenin gene is located within axon 9 of the delta-catenin locus and consists of a GFP reporter fused to a PGK-hyygro-pA cassette followed by a stop condon, which results in the prevention of transcription of the rest of the delta-catenin gene. Mouse pups were decapitated at P0 or P1, the brains were removed from the skulls and the hippocampi were dissected from the brain followed by manual dissociation and plating of 250,000 cells in the MEA chamber [24]. After one week, cultures were treated with 200 μM glutamate to kill any remaining neurons, followed by a new batch of cells added at the same density as before. Cultures were grown in a tissue culture incubator ($37°C$, 5% CO_2), in a medium made with Minimum Essential Media with 2 mM Glutamax (Life Technologies), 5% heat-inactivated fetal calf serum (Life Technologies), 1 ml/L of Mito+ Serum Extender (BD Bioscience) and supplemented with glucose to an added concentration of 21 mM. All animals were treated in accordance with University of California and NIH policies on animal care and use.

2.2.2 Culture of iPSCs Neurons

iPSCs were cultured in mTeSR1 media (Stem Cell Technologies) and routinely passaged with ReleSR (Stem Cell Technologies). The cells were subsequently infected with TetO-hNgn2-UBC-puro (plasmid from Addgene # 61474) and rtTA (plasmid from Addgene # 20342) lentiviruses. Briefly, the cells were passaged as single cells into 4 wells with accutase (Life Technologies) and Y-27632 dihydrochloride (Tocris) at a final concentration of 10 M. On day 2 the cells were infected with hNgn2 in fresh mTeSR1 media. On day 3, the infected iPSCs were selected by adding puromycin at 2 ug/ml for a 2 day

period. The cells were infected with rtTA virus on day 5 and incubated overnight. The neurons were differentiated by adding doxycycline at a final concentration of 2 ug/ml. Two days after addition of doxycycline, the neurons were replated on poly-l-lysine coated MEAs at a density of 180,000 cells concentrated in a 15 ul droplet. iPSCs-derived neurons were cocultured with mouse primary astrocytes in BrainPhys complete medium (Stem Cell Technologies). Doxycycline was kept in the media for 14 days total.

2.2.3 Electrophysiology

We used 120 electrode MEAs (120MEA100/30iR-ITO arrays; MultiChannel Systems) for recording as is shown in Fig. 2.1. All recordings were performed in cell culture medium so as to minimally disturb the neurons. The osmolality of the culture medium was adjusted to 320 mosmol. Recordings were performed using MultiChannel Systems MEA 2100 acquisition system. Data were sampled at 20 kHz. Recordings were performed at $30°C$. All recordings were performed on neurons at 2-30 DIV. Data recordings were typically 3 minutes long. The recording duration was controlled to minimize the effects of removing MEAs from the incubator.

2.2.4 Spike Detection

For each MEA recording, we performed spike detection [25]. Extracellular signals were band pass filtered using a digital 2nd order Butterworth filter with cutoff frequencies of 0.2 and 4 kHz. Spikes were then detected using a threshold of 5 times the standard deviation of the median noise level. Since there are 120 electrodes in our MultiChannel Systems, the spike detection result of a 3 min recording is a 120×180000 shape matrix made up of 1s and 0s, where 1 represents neuron firing and 0 represents not firing.

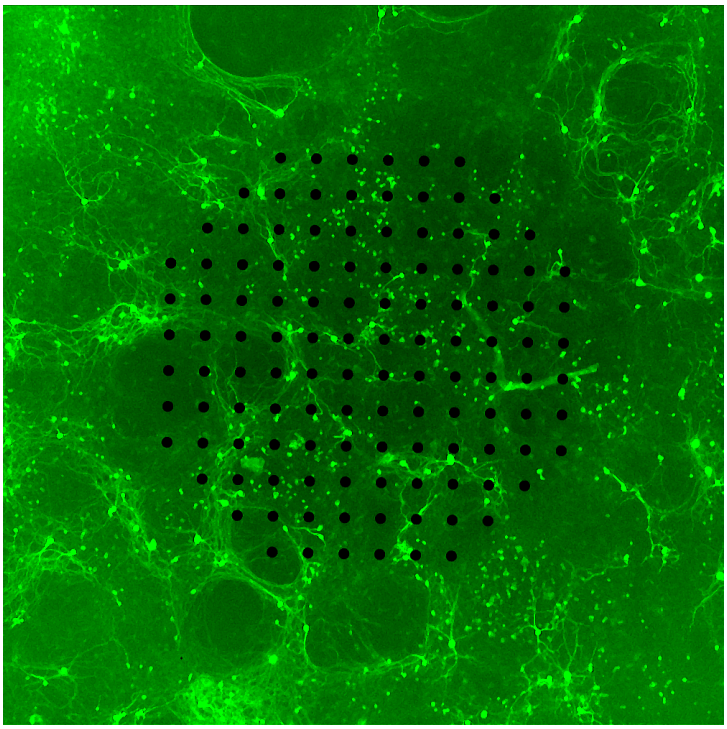
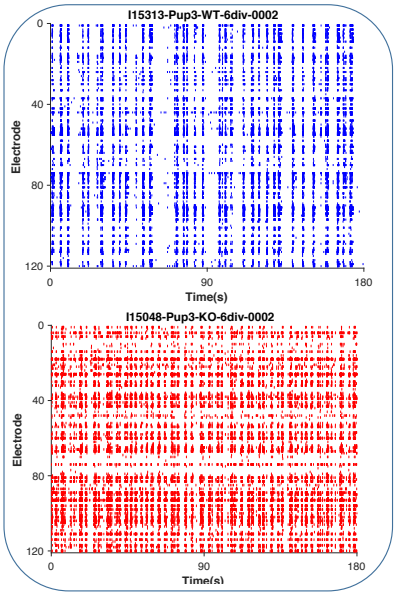


Figure 2.1: Neural networks were grown on arrays of 120 electrodes. The purpose of this research was to determine whether neural cultures derived from genetically different neurons could be distinguished by analysis of their electrical activity.

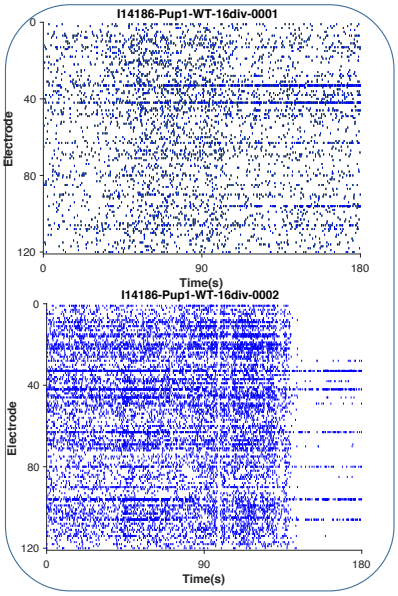
2.2.5 Classification

For the remainder of this chapter, Wild Type (WT) means that there is no gene mutation. Knockout (KO) means that the gene delta-catenin is knocked out or not expressed in the mouse neurons. WS is Williams syndrome neurons, compared with Control. Fig. 2.2 shows Raster Plots for some sample mouse MEA recordings from WT and KO. From the figure, the recording patterns vary drastically according to different mice, different DIV and even different recording numbers. However, recordings of different genotypes sometimes perform similarly. It is challenging for human eyes to distinguish KO from WT. There are several reasons: 1. The recordings are noisy due to the errors in measuring potentials and spike detection. 2. The firing pattern will change drastically according to many factors like different DIV, different mice and even different recordings. A deep learning classification framework is therefore introduced to automatically predict the genotype, given an MEA recording.

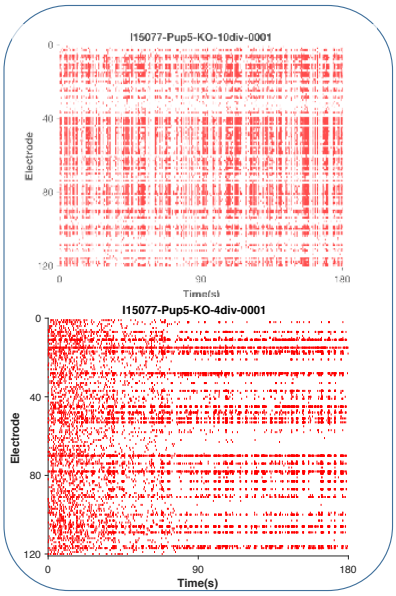
We use two separate sets of MEA recordings in our classification: one dataset consists of mouse neuron recordings to classify KO and WT, while the other dataset consists of human iPSC neuron recordings to distinguish WS and Control human cells. Our mouse recordings consist of 5 separate experiments (Exp1, Exp2, Exp3, Exp4, Exp5) and 331 180000 ms recordings in total, of which 198 recordings are WT and the remainder are delta-catenin KO. Our iPSC recording data are made up of 12 WS recordings and 8 Control recordings. Considering the size of the two datasets, we randomly shuffle and split the mouse MEA data into training, validation and testing by 70%, 10% and 20%, while we apply 5-fold cross-validation for human iPSC recordings.



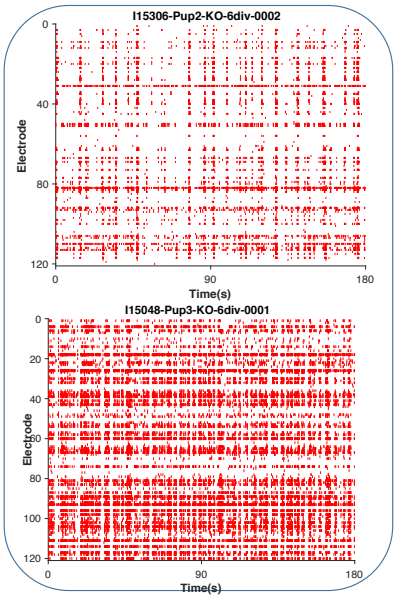
(a) Different genotype



(b) Different recording number



(c) Different DIVs



(d) Different mice

Figure 2.2: Raster Plots of WT and delta-catenin KO. Blue represents WT and red indicates delta-catenin KO. The title of each raster plot is formatted as "MEA device-Mouse-Gene type-DIV-Record Number". (a) KO and WT share some common firing patterns. (b) Different recordings with the same gene type, as well as the same DIV look different. (c) Recordings with the same mouse, the same gene type but different DIV look different. (d) Recordings with the same gene type, the same DIV, but different mouse look different.

2.3 Deep Learning Model

The model architecture, shown in Fig. 2.3, consists of convolution-pooling layers followed by fully connected layers. To learn temporal and spatial invariant features, the convolution is performed on both time and space dimensions. We split the long recordings into smaller slices with length of seq_length. Detected spikes with shape of (120, seq_length) serve as input x for the neural network. A convolution operation involves a filter $w \in \mathbb{R}^{st}$, which is applied to a window of s electrodes and t ms to produce a new feature. For example, a feature $f_{i,j}, (0 \leq i \leq 120-s+1, 0 \leq j \leq seq\_length-t+1)$ is generated from a window size (s,t) of the spike train:

$$f_{i,j} = ReLU(wx_{i:i+s-1,j:j+t-1} + b), \tag{2.1}$$

where $b \in \mathbb{R}$ is a bias term. This filter is applied to each possible window of the spike trains to produce a feature map:

$$f = \begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,seq\_length-t+1} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,seq\_length-t+1} \\ \cdots & \cdots & \cdots & \cdots \\ f_{120-s+1,1} & f_{120-s+1,2} & \cdots & f_{120-s+1,seq\_length-t+1} \end{bmatrix}, \tag{2.2}$$

with $f \in \mathbb{R}^{120-s+1,seq\_length-t+1}$. We then apply a max-pooling operation over the feature map and take the maximum value $m = \max f$ as the feature corresponding to this particular filter. The idea is to capture the most important feature, the one with the highest value, for each feature map. Our model uses multiple filters to obtain multiple features. These features form the penultimate layer and are passed to a fully connected softmax layer whose output is the probability distribution over two different genotypes. We adjust the number of convolutional ReLU layers from 2 to 5, based on the choice of

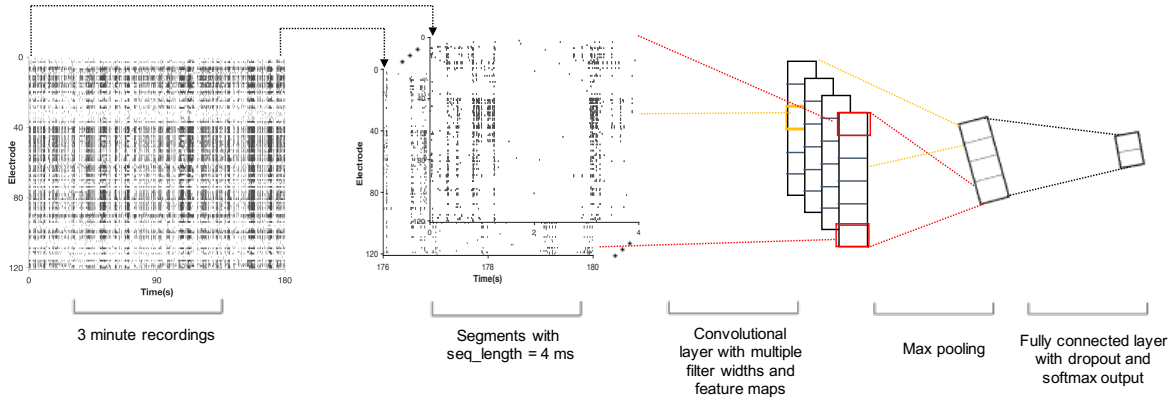


Figure 2.3: Model architecture: 3 minute recordings of the electrical potentials measured on the 120 electrodes are collected from the neuron cultures. Segments with seq_length = 4 ms of these recordings are individually classified. These individual classifications are conducted for Consensus Prediction in mouse MEA recordings.

seq_length.

We use Batch Normalization [26] to accelerate training. For regulaization, dropout [27] and early stopping methods [28] are implemented to avoid overfitting. Dropout prevents co-adaptation of hidden units by randomly dropping out a proportion of the hidden units during backpropagation. Model training is ended when no improvement is seen during the last 100 validations. Softmax cross entropy loss is minimized with the Adam optimizer [29] for training.

2.4 Experimental Setup

2.4.1 Training and Hyperparameters

We use 1 ms time bins for our spike train data, thus the dimensionality of time is extremely high. For example, a slice of 10 seconds has 10,000 data points along the time dimension. Thus, the CNN model has a very high demand for memory, while the memory for the graphics processing unit (GPU) is limited. In practice, we randomly

sample segments from each recording for training, which not only decreases the GPU memory usage by reducing the dimensionality of time but also increases the number of training samples. For example, if we use seq_length of 1000ms, then a 180000ms recording can provide 180 independent samples.

We implement the deep learning framework using Tensorflow [30] with the following configurations. The (120, seq_length) spike detected matrices (see Fig. 2.3) are input to convolutional ReLU layers which filter the input spike train with 2×5 kernels and stride of (1, 1). It is interesting to note several biologically inspired hyperparameters in Table 2.1. Seq_length is the slice length that we use to split the recordings. Kernel_size and stride in CNN correspond to propagation signals, synaptic coupling and correlation between channels. Short latency, monosynaptic, interactions are in a range of 2-4 ms. Propagation signals occurring between nearby electrodes have an average latency of 0.3 ms to 0.7 ms. We choose a kernel size of 2×5 and stride of (1, 1) to capture propagation signals and synaptic coupling. Hyperparameters are described in Table 2.2. Max pooling is then applied after each convolutional ReLU layer. The feature maps are input for fully connected layers with 2 output nodes for the binary classification.

2.4.2 Testing

For testing, we define Consensus Prediction to measure the performance of predictions for the whole recordings. Consensus Prediction synthesizes results from odd numbers of short slices by majority voting, which can significantly improve the prediction accuracy for a long recording. This is because not all of the short time-slices can be expected to contain useful information. The results of mouse MEA recordings in Section 2.5 are reported with Consensus Prediction.

Table 2.1: Bio-inspired parameters

Para	Biological Justification	value
Seq_length	The appropriate slice length which can represent a recording	4000 ms
Kernel_size	Propogation signals	(2,5)
Stride	Synaptic coupling, correlation between channels	(1,1)

Table 2.2: Hyperparameters

Hyperparameters	Value
Batch size	24
Epoch	5000
Dropout rate	0.5

2.4.3 Implementation

We implement a framework that can distribute the convolutional neural network into multiple (N) GPUs to ease the burden on GPU memory. Each GPU contains an entire copy of the deep learning model. We first split the training batch evenly into N sub-batches. Each GPU only processes one of the sub-batches. Then we collect gradients from each replicate of the deep learning model, aggregate them together and update all the replicates. With 3 NVIDIA GeForce GTX 1080s, each of which has a memory of 11178 MB, we can handle spike train segments of 14 seconds with batch size of 24.

2.5 Empirical Evaluation

We focus our evaluation mainly on the accuracy of predicting genotype. We use Consensus Prediction, which is the majority voting result of the sampled short slices, for the mouse recordings. We report the initial prediction accuracy of short slices for human iPSC recordings without Consensus Prediction, since the recording experiments

are better controlled.

2.5.1 Performance Analysis

Results of our framework compared against other machine learning models on mouse recordings and human iPSC recordings are shown in Table 2.3 and Table 2.4 respectively. We compare our CNN model with Multilayer Perceptron(MLP) and feature based Logistic Regression. We use a two layer MLP, which shares the same hyperparameters with our model's fully connected layers. For Logistic Regression, we first extract features of firing rate and Pearson correlation coefficient between different electrodes for each recording, and then classify neuron genotypes based on these two features. For the mouse recordings, our CNN based deep learning approach improves the Consensus Prediction accuracy by 16.69% compared with feature based Logistic Regression. Fig 2.4 shows the Consensus Prediction accuracy. The accuracy improves by 5.92% using Consensus Prediction. Although not all of the short slices can be expected to contain enough useful spike patterns, we can overcome that when we synthesize multiple individual classification results from these short slices. For the human iPSC recordings, we report the prediction accuracy of short recording slices. Our model achieves accuracy of 96.18% even without Consensus Prediction, which is a 15.59% improvement over feature based Logistic Regression. Our CNN based deep learning model also outperforms MLP on both of the two sets of recordings by 7.00% and 7.81% respectively, which shows CNN's advantage of local feature extraction using convolutional kernels over MLP.

Fig 2.5 shows the trend of accuracy versus the choice of seq_length for human iPSC. For the effect of seq_length on accuracy, there exists a trade off between number of training samples and representation of a whole recording. The short slices contain less information but can provide more independent training samples. For deep learning models, larger

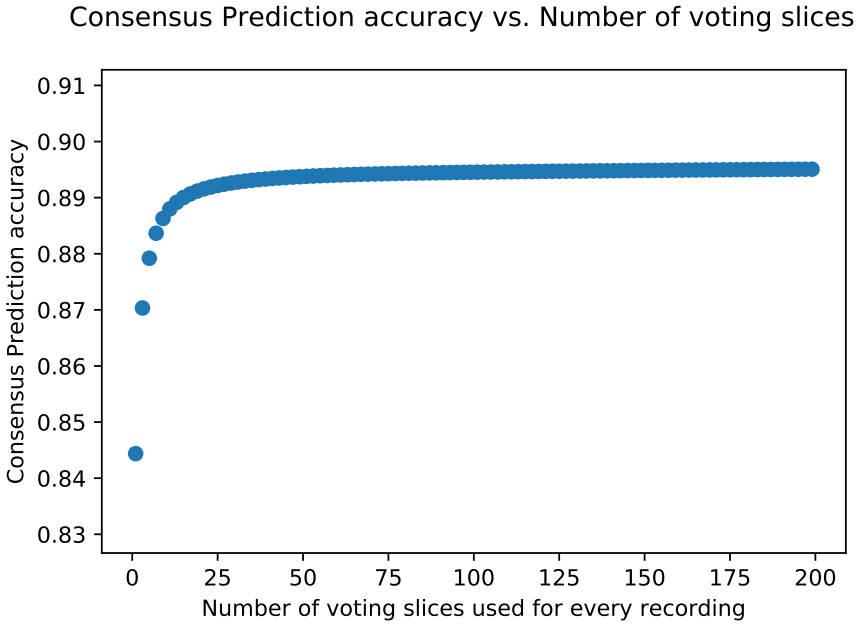


Figure 2.4: Consensus Prediction accuracy vs. number of short slices used for mouse recordings.

Table 2.3: Consensus Prediction performance comparison of our deep learning model with Multilayer Perceptrons and Logistic Regression on mouse recordings.

Model	Accuracy on Testing
Convolutional Neural Network	**0.8951**
Multilayer Perceptron	0.8366
Logistic Regression	0.7671

numbers of training slices help more than a larger sample. However, we still cannot choose too small of a seq_length, since a too short slice is not representative for a recording. Given the data we currently have, we use a seq_length of 4000 ms.

Dropout proved to be such a good regularizer that it was fine to use a larger than necessary network or train too many epochs and simply let dropout regularize it [31]. Dropout consistently added 2% - 4% relative performance. Our model converged best with Adam optimizer compared with Vanilla gradient descent, Adagrad [32], Adadelta [33] and RMSprop [34].

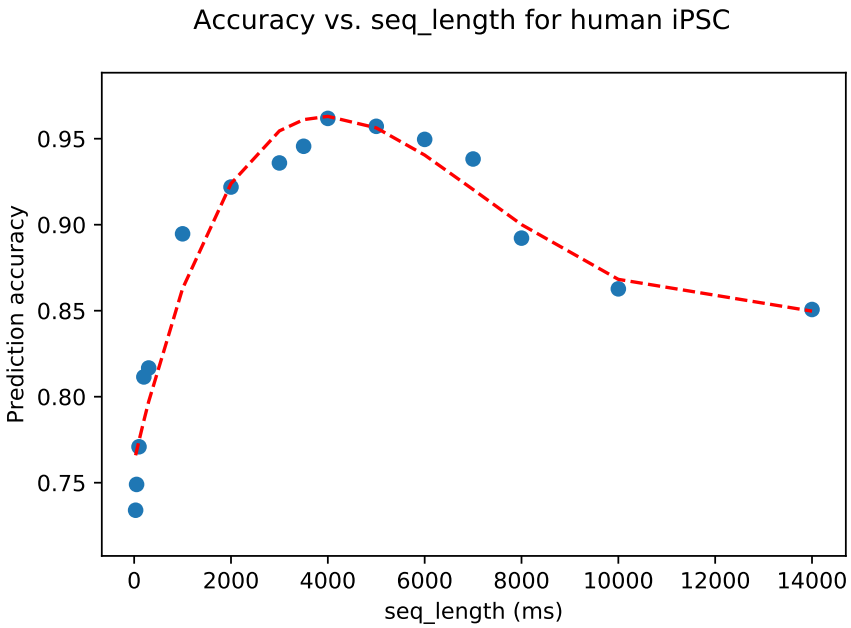


Figure 2.5: Accuracy vs. seq_length trend for human iPSC recordings.

Table 2.4: Performance comparison of our deep learning model with Multilayer Perceptrons and Logistic Regression on iPSC recordings.

Model	Accuracy on Testing
Convolutional Neural Network	**0.9618**
Multilayer Perceptron	0.8921
Logistic Regression	0.8321

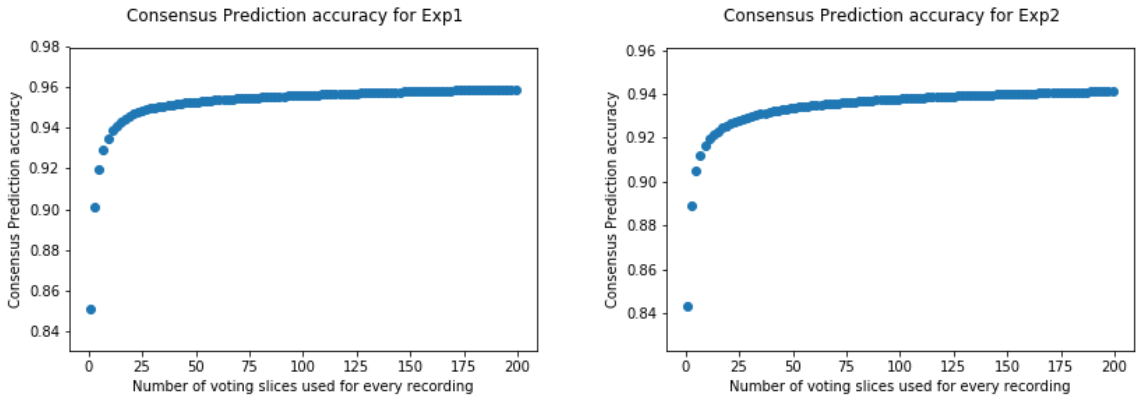


Figure 2.6: Consensus Prediction accuracy for Exp1 and Exp2.

2.5.2 Case Study

It is challenging to classify the genotype of mouse MEA recordings due to the differences in recordings taken from neurons of different DIV, different mice and different recordings. Considering that the neuron firing patterns change drastically with different DIV, we use two subsets of mouse recording data (Exp1 and Exp2), recorded at 6 DIV and 10 DIV respectively, to study the effect of Consensus Prediction. Fig. 2.6 shows the prediction accuracy versus number of voting slices in Consensus Prediction. By taking one subset of experiments all recorded at 6 DIV, we achieve a Consensus Prediction accuracy of 95.91% for Exp1. Similarly, we achieve a Consensus Prediction accuracy of 94.12% for recordings in Exp2, which are all at 10 DIV. Using Consensus Prediction, we improve the prediction accuracy by 12.70% and 11.68% for Exp1 and Exp2 respectively, which indicates that combining information from different parts of one recording significantly helps improve the performance.

2.6 Discussion

We have addressed the issue of classifying different genotype MEA recordings by proposing a deep learning framework. We split the long recordings into smaller slices, which not only eases the burden on GPU memory but also provides more training samples for the deep learning model. We use Consensus Prediction during testing, to predict the genotype for a recording. This work is a proof of principle for classification via deep learning of in-virtro MEA recordings. Clearly, however, more work is needed before it can be known if deep learning will be a generally useful technique for classification of neural cell genotypes or drug effects from *in vitro* MEA recordings. For example, one can use more recordings and MEAs with larger numbers of probes in future work.

Chapter 3

Scalable Bayesian Functional Connectivity Inference for Multi-Electrode Array Recordings

3.1 Introduction

Neuroscience deals with how networks of neurons are organized and how they function [35]. Understanding connectivity between neurons and within the brain is a fundamental problem in neurobiology [36]. Functional connectivity, defined as the statistical dependencies between different brain regions with similar patterns, is widely used in various neural tasks [37, 38]. For instance, functional connectivity magnetic resonance imaging (MRI) is crucial for diagnosing and comprehending autism spectrum disorders [39]. MEAs [40] can record extracellular action potentials from hundreds or thousands of neurons and provide insights on neuronal connectivity [41]. For hours or weeks, action potentials can be non-invasively monitored, when neurons are grown on planar MEAs [42]. Further, there is a trend towards increasing the density of the arrays [43] to better un-

derstand the neuron connectivities.

MEA recordings provide researchers opportunities to understand neuron activities in many regions such as the brain, retina, and heart [44]. However, the analysis of this data is challenging, in part because of its high dimensionality. Summary statistics could be used to measure the connection weights between electrodes. These include, for example, Pearson correlation [45], cross correlogram (CCG), the maximal information coefficient (MIC) [46] as well as biophysically-inspired metrics [47]. However, a data generative model is required to understand the underlying structure and to make full use of the domain expert knowledge [48]. Furthermore, these summary statistic methods provide different functional connectivity results for the same recording since they are all deterministic metrics, which present fixed connection weights between every two electrodes instead of a probabilistic estimation.

Bayesian inference can address the requirements for the inference, since it provides distributions for parameters using probabilistic models and observation data. In contrast to deterministic optimization procedures that give point estimates of the unknown functional connectivity, computing a Bayesian posterior yields probability distributions for the neuronal network functional connectivity. Bayesian inference has been combined with the generalized linear model (GLM), with graph-based priors to infer the neuron connectivity pattern for analysis [49]. However, there is a lack of scalable Bayesian techniques for inference of network structure, which is particularly acute for inference from high-density recordings.

A considerable challenge for Bayesian techniques is the rapid growth of computation time in accordance with the increasing scale of the network. In this chapter, we propose a scalable framework of Bayesian inference, inspired by the hierarchical structure of networks of neurons. Experiments on both synthetic datasets and large scale real-world MEA recordings show that our framework provides accurate and insightful results. Fur-

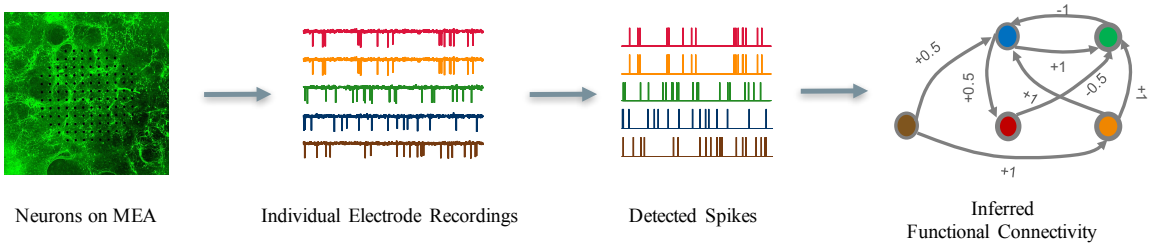


Figure 3.1: The workflow for Bayesian functional connectivity inference. We consider only negative deflections that exceeded 6 times the standard deviation of the median noise level as spikes.

thermore, we apply the proposed framework to a controlled cadmium dataset, and the results confirm its utility.

The key contributions of this chapter include:

1) We propose a scalable functional connectivity inference framework shown in Fig. 3.1 for MEA recording data. We speed up the expensive Bayesian sampling process through the use of parallel computing.

2) We infer the network by splitting the large scale recordings into smaller local networks. We also provide a strategy for inferring the regional connectivity between local networks. This not only eases the burden from sampling but also provides useful insights on regional connections in organoids and brains.

3) Experiments on both synthetic dataset and large-scale real-world MEA recordings show the effectiveness and efficiency of the Bayesian framework. Inference of network structure of Cadmium-exposed neuron cultures further demonstrates the usefulness of our framework.

The remainder of this chapter is organized as follows. Section 3.2 describes the MEA data collection. We delineate the probabilistic models in Section 3.3 and demonstrate the Bayesian inference details in Section 3.4. Section 3.5 describes the hierarchical setup. Results for both synthetic and real data are provided in Sections 3.6 and 3.7, respectively.

Related work is described in Section 3.8. Section 3.9 is the Discussion.

3.2 Data Collection

3.2.1 Cell Culture

We prepared hippocampal neurons from postnatal day 0 (P0) mice with C57BL/6 genetic background using a previously described protocol [50]. Cleaned and sterilized MEAs (120MEA100/30iR-ITO arrays; Multi Channel Systems) were incubated with poly-L-lysine (0.1 mg/ml) for at least one hour at 37 °C, rinsed 3 times with sterile deionized water and allowed to air dry before cell plating. Glial cultures were maintained in separate T-75 flasks. 100, 000 - 125, 000 dissociated glial cells were used for the first plating of MEAs to obtain a confluent glial culture over the surface of the electrodes. Once glia were confluent, the hippocampi dissected from the brain followed by manual dissociation were plated at 250, 000 cells in the MEA chamber. Cultures were grown in a tissue culture incubator (37 °C, 5% CO_2) in a medium made with minimum essential medium + Earles salts (Thermo Scientific, catalog # 11090081) with 2mM Glutamax (Thermo Scientific), 5% heat-inactivated fetal bovine serum (Thermo Scientific), and 1 ml/l Mito+ serum extender (Corning) and supplemented with glucose to an added concentration of 21mM. To minimize the effects of evaporation, maintain cell culture sterility, and decrease degassing of the medium during recordings, the MEA chamber was covered by a gas permeable membrane that permits exchange of CO_2 when the plate is in the CO_2 incubator.

3.2.2 MEA recordings

Extracellular voltage recordings of neuronal cultures were performed using an MEA 2100-System (Multichannel Systems, Reutlingen, Germany). Each MEA contained 120 electrodes with a 100 m inter-electrode distance. All data were acquired at a 20 kHz sampling rate. All recordings were performed in culture media. The head stage temperature was set to 30C with an external temperature controller, and the MEAs were equilibrated for 5 min on the head stage before data acquisition or after any pharmacological or temperature manipulation. Recording duration was 3 minutes. Only cultures at 14 days in vitro (DIV) or older were used for pharmacological experiments.

3.2.3 Data Processing

Raw data was converted to HDF5 file format and processed offline. Spike detection was done with Matlab tools Waveclus [51]. Note that we did not apply spike sorting, since it may introduce considerable noise due to unsupervised clustering methods when trying to obtain the neuron (or unit) information [52], and there are many different spike sorting algorithms [51, 53], which give different outputs. Extracellular voltage recordings were bandpass filtered using cutoff frequencies of 200Hz and 4000Hz. Only negative deflections in the voltage records were labelled as spikes when the amplitude exceeded 6 times the standard deviation of the median noise level. Spike times and amplitudes were recorded and used for downstream analysis.

3.3 Probabilistic Model

In this section, we briefly review the probabilistic model of neuronal spike trains introduced in [49] along with our choice of parameterization. Table 3.1 summarizes

Table 3.1: Notations

Notations	Description
$X_{t,n}$	The observed spike at time bin t for electrode n
A	Adjacency matrix
W	Weight matrix
b_n	The baseline activation of electrode n
N	Number of electrodes
T	Autoregressive window of influence
$\psi_{t,n}$	The activation of electrode n at time bin t
N_o	Number of overlapped electrodes when split
W_o	Weight matrix of the overlapped region
N_s	Sample number of Bayesian inference
ρ	The prior for connection probability
μ_{w_n}	Mean for the nth row of W
μ_b	Mean of the bias vector
S_{w_n}	Covariance for the nth row of W
S_b	Covariance of the bias vector

some common notations that we will use in this chapter. At a high level, the model describes how the underlying connectivity network affects the activation propensity of each electrode over time, producing the observed spike firing pattern measured over the entire MEA. Specifically, the model is composed of three parts: a network model specifying the underlying connectivity of the electrodes, an activation propensity model detailing how a network along with past spike history affects the probability of a spike at a time bin, and a spiking observation model mapping the activation propensity to the observed binary spike trains. Note that an electrode can fire no more than once in one time bin because of the refractory period in neurons. A probabilistic graphical model of this is shown in Fig 3.2.

3.3.1 Network Model

The network model aims to capture the key properties of the underlying functional network of the electrode population. Specifically, it seeks to represent that those con-

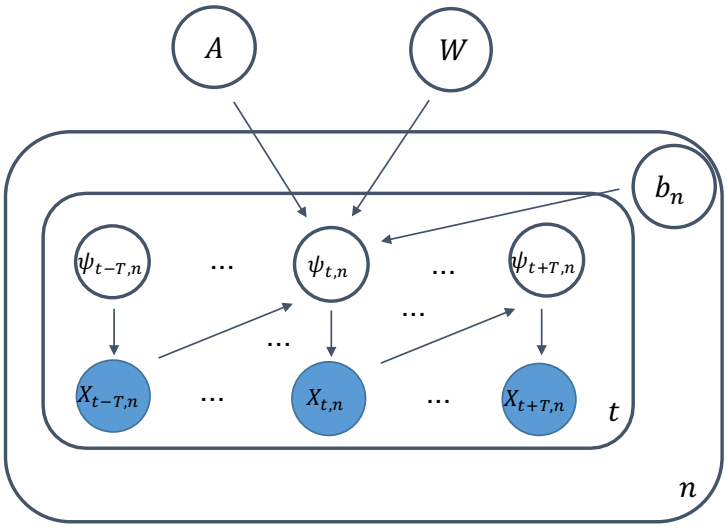


Figure 3.2: Probabilistic graphical model for MEA. The model describes how the underlying connectivity of a network of electrodes (A and W) can lead to the observed spike trains $X_{t,n}$.

nections are potentially directional and different in strength. To accommodate this, a weighted directed graph is used where the edge weights represent the strength of the connection between two electrodes. This is incorporated as two matrix-valued latent variables $A \in \{0,1\}^{N\times N}$ and $W \in \mathbb{R}^{N\times N}$ corresponding to a binary adjacency matrix and a real-valued weight matrix respectively.

3.3.2 Activation Propensity Model

A neuron can either fire spontaneously or as a response to communications (spikes) it receives from incoming, connected neurons. Given a particular realization of the electrode network, A and W, the instantaneous activation of electrode n at time bin t, $\psi_{t,n}$ is modeled as a linear, autoregressive function of the lagged spikes from neighboring electrodes:

$$\psi_{t,n} = b_n + \sum_{m=1}^{N}\sum_{\Delta t=1}^{T} A_{m\to n} W_{m\to n} e^{-\Delta t/\tau} X_{t-\Delta t,m}, \tag{3.1}$$

Here, b_n represents the baseline activation rate for electrode n in the absence of influence from any other electrode. $A_{m\to n} \in \{0, 1\}$ is a binary variable indicating whether or not there exist directed connections from electrode m to electrode n. The weight $W_{m\to n}$ is the connection strength from electrode m to electrode n. The activation rate ψ is linearly adjusted by the lagged spikes from neighboring electrodes. The strength of the lagged spike is weighted by the strength of the connection to the neighbor and an exponentially decreasing function of time, inspired by the synapse connectivity measurement in [47], with time constant of $\tau = 15ms$. This prioritizes recent spikes from strongly connected neighbors. We consider both positive and negative $W_{m\to n}$, which captures that neuronal connections may be excitatory or inhibitory in nature. It is possible for a spike to decrease the propensity of firing when a weight is negative.

3.3.3 Bernoulli Observation Model

Our spike train data consists of binary observations of whether electrode n fired at time bin t, $X_{t,n}$. This is modeled as a Bernoulli random variable with probability dependent on the activation propensity, $\sigma(\psi_{t,n}) = e^{\psi_{t,n}}(1+e^{\psi_{t,n}})^{-1}$, where σ is the logistic function that maps the propensity to a probability.

3.4 Bayesian Inference

Based on the previous section, the full model, including the priors, is as follows:

$$\begin{aligned} A_{i,j} &\sim \text{Bernoulli}(\rho) \\ \{\mu_{w_n}, \mu_b\}, S_{w_n}, S_b &\sim \text{Normal-Inverse-Wishart}(0, 1, \mathbf{I}, 3) \\ w_n | \mu_{w_n}, S_{w_n} &\sim \text{Normal}(\mu_{w_n}, S_{w_n}) \\ b | \mu_b, S_b &\sim \text{Normal}(\mu_b, S_b) \\ \psi_{t,n} &= b_n + \sum_{m=1}^{N} \sum_{\Delta t=1}^{T} A_{m \to n} W_{m \to n} e^{-\Delta t/\tau} s_{t-\Delta t, m} \\ X_{t,n} &\sim \text{Bernoulli}(\sigma(\psi_{t,n})), \end{aligned} \tag{3.2}$$

where w_n denotes the nth row of the matrix W. The hyper-parameter ρ affects the prior over the connectivity matrix A.

We apply an efficient Gibbs sampler, which exhibits scalable parallelism, derived from [49]. Sampling the posterior over the discrete adjacency matrix A is the most challenging step. Due to conjugacy, we can integrate over W and sample A from its collapsed conditional distribution. We update A and W by collapsing out W to directly sample each of As elements. We iterate over each $A_{m \to n}$ and sample it from its conditional distribution. After that, we sample W from its conditional Gaussian likelihood.

We employ a novel splitting strategy to make the inference method amenable to high density arrays. We present and verify the methodology in the following sections.

3.5 Split

We have 120 electrodes in the MEA device. However, there are 26,400 electrodes in the MaxWell complementary metal oxide semiconductor (CMOS) MEA device. As we

can see in Fig. 3.3, the average time for one sample increases drastically with the size of the array. That is because the dimensions of parameters (A and W) to be estimated exhibit quadratic growth according to the number of electrodes. The time reported in Fig. 3.3 is the sampling time, computed in parallel with 24 CPU cores. To deal with the time complexity challenge, we propose a hierarchical inference procedure. As shown in Fig. 3.4, we split the whole large array into fixed number (for example, 4) of regions. In the first level of the algorithm, we perform Bayesian inference individually on each of the smaller regions. In the second level, we treat each whole region as one group by taking the mean of all the spike trains in the region. We apply the same probabilistic model to infer the regional connectivity, which is the connection strength between each pair of regions. This heuristic splitting strategy is inspired by the biological phenomenon of regional connections in brains. The splitting strategy decreases the average sampling time quadratically in accordance with the number of sub-regions. To get a better understanding of the connections on the border of any two regions, we further propose an overlapping split mechanism, as shown in Fig. 3.5.

3.6 Results on Synthetic Data

Due to the lack of ground truth functional connectivity in an MEA, we first use synthetic data to check the effectiveness of the probabilistic model and our splitting strategies. We use the ground truth model to generate synthetic data and compare the functional connectivity inferred from synthetic data with the ground truth. For all the experiments in this section and the next section, we use spike trains with shape of 180, 000 * 120. We apply parallel sampling with 24 CPUs. The Gibbs sampler was run for 1000 iterations. The first 500 samples were discarded to account for burn-in. We verify that the chain has reached a steady state by observing that the parameter traces and the

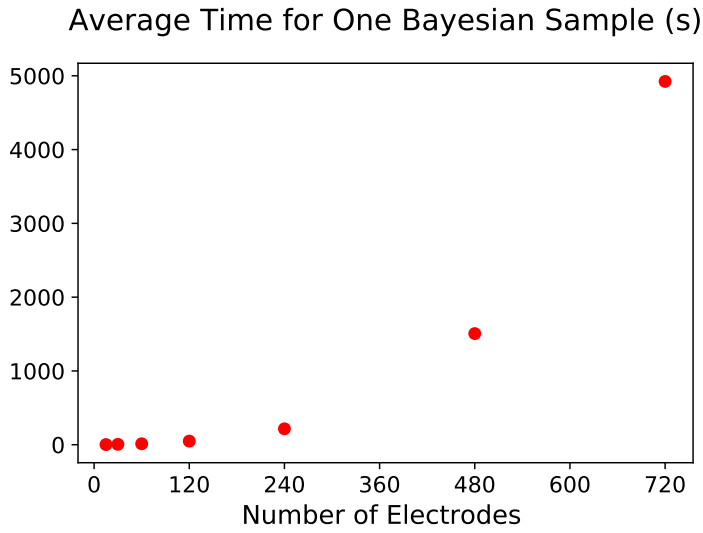


Figure 3.3: Average time for one Bayesian sampling exhibits quadratic growth according to the number of electrodes. The time reported is the sampling time computed in parallel with 24 central processing units (CPU) cores.

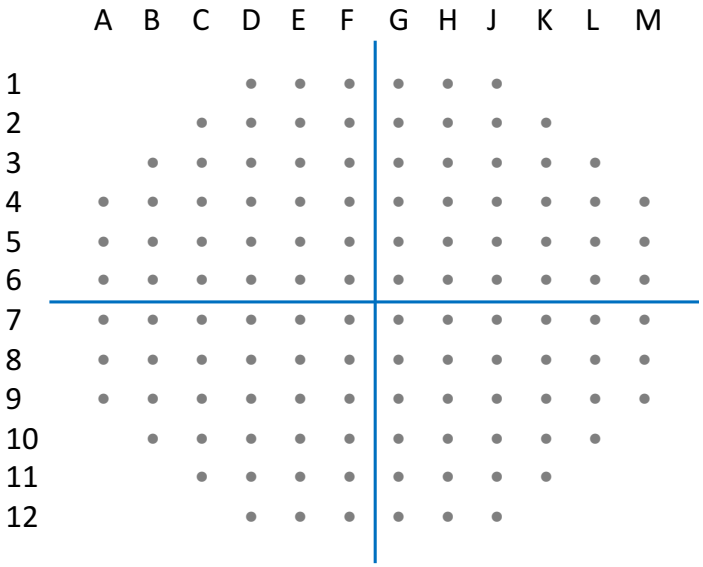


Figure 3.4: Non-overlapping split. We split the large array into four regions. For the first level, we infer individually on the smaller regions. For the second level, we treat each whole region as one hidden super node and infer the regional connection strength between each pair of sub-regions.

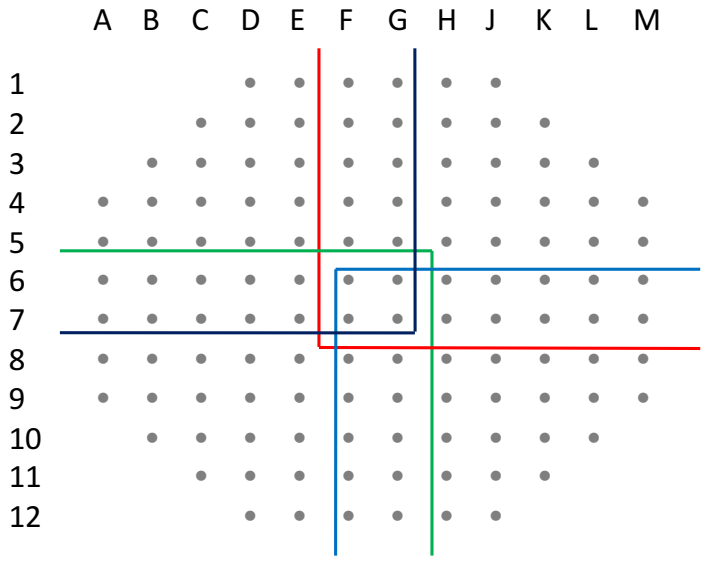


Figure 3.5: Overlapping split. To better capture the connections on the border of the regions, an overlapping split is used.

log-likelihood have converged.

In Fig. 3.6, the inferred posterior mean is almost the same as the ground truth connectivity matrices. Cosine similarity measures the similarity between two matrices of an inner product space, which is widely used in high dimensional spaces [54]. The inferred functional network A and W both achieved a high cosine similarity of 0.99 compared with ground truth in this case. Similarly in Fig. 3.7, we obtain cosine similarity of 0.95 and 0.99 for A and W when the number of electrodes increases to 10.

We also use synthetic datasets to verify the effectiveness of our splitting strategies. Here, we assume all the electrodes lie in a line. For the non-overlapping split, we use N electrodes and split them into two sub-regions, "front" and "back", with equal size and apply the Bayesian inference on each region. We compared the separate split results with the Bayesian inference results without split and reported the cosine similarity in Tab. 3.2. It is shown that the cosine similarities are consistently high with all different parameter settings, which validates the effectiveness of our splitting strategy. Similarly,

Table 3.2: Non-overlapping Results: Cosine Similarity

N	S_w	μ_b	ρ	W_{front}	A_{front}	W_{back}	A_{back}
10	1	0	0.5	0.93	0.99	0.94	0.98
10	1	0	1	0.99	0.99	0.98	0.99
10	1	5	0.5	0.99	0.99	0.98	0.99
10	2	0	0.5	0.95	0.98	0.88	0.97
20	1	0	0.5	0.91	0.99	0.95	0.99
30	1	0	0.5	0.94	0.99	0.93	0.99

for overlapping split, we test the results from split with N_o overlapped electrodes. In Tab. 3.3, N_o indicates the number of overlapped electrodes. W_o is the weight matrix for the overlapped region. High cosine similarities in Tab. 3.3 confirm the effectiveness of overlapping split strategy. From both Tab. 3.2 and Tab. 3.3, the network prior does affect the inference of A and W but in an indirect way, which is small in degree compared to the effect of the data.

Fig. 3.8 shows the effectiveness of regional connection inference after split. Each element in the inferred regional connectivity matrix W summarizes the elements of the corresponding regions in ground truth W altogether. Regional inferences after split precisely indicate the strength and the nature of the connections between regions.

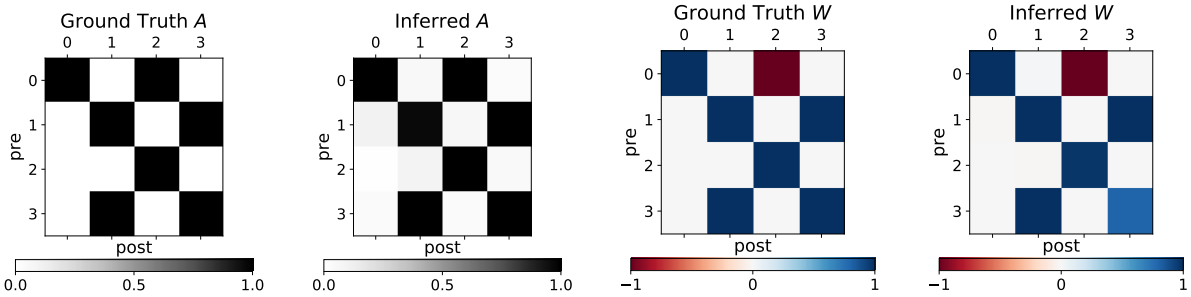


Figure 3.6: Comparison of network inferred from synthetic data with 4 electrodes and ground truth. Zero indicates no connection. Minus one indicates an inhibitory connection, and one indicates an excitatory connection. Cosine similarity between the two As is 0.99. Cosine similarity between the two Ws is 0.99.

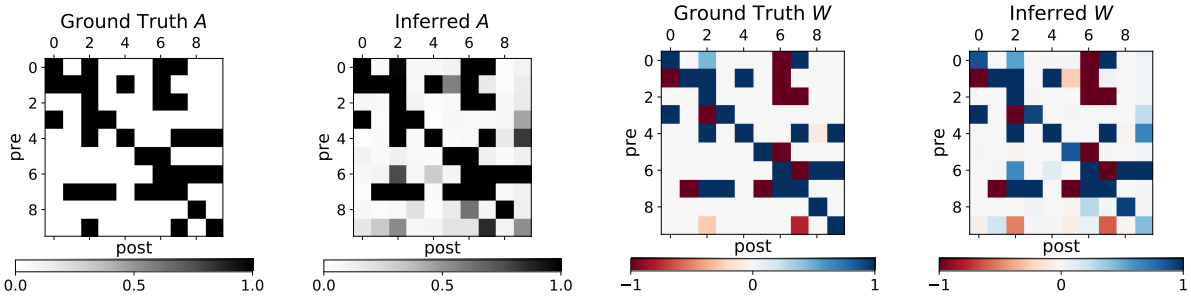


Figure 3.7: Comparison of network inferred from synthetic data with 10 electrodes and ground truth. Cosine similarity between the two As is 0.95. Cosine similarity between the two Ws is 0.99.

Table 3.3: Overlapping Results: Cosine Similarity

N	N_o	S_w	μ_b	ρ	W_{front}	A_{front}	W_{back}	A_{back}	W_o
10	2	1	0	0.5	0.97	0.9	0.95	0.99	0.98
10	2	1	0	1	0.99	0.99	0.99	0.99	0.99
10	2	1	5	0.5	0.99	0.99	0.99	0.99	0.99
10	4	2	0	0.5	0.92	0.98	0.95	0.99	0.93
20	4	1	0	0.5	0.94	0.99	0.97	0.99	0.96
30	4	1	0	0.5	0.90	0.99	0.93	0.99	0.97

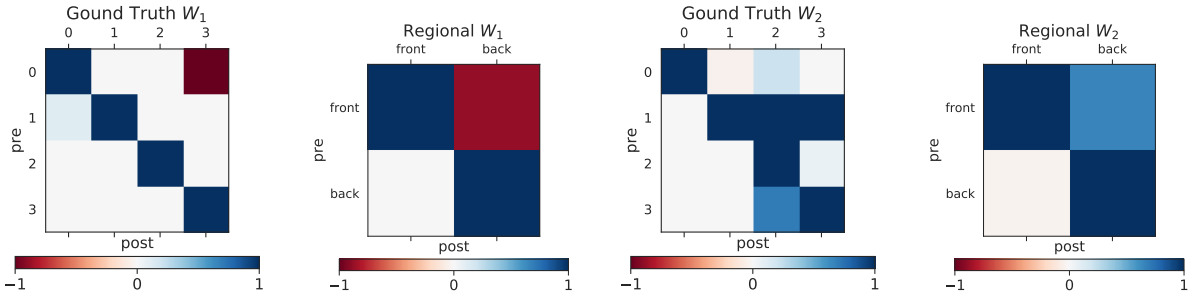


Figure 3.8: Regional connections in non-overlapping split. "Front" indicates the region composed of electrodes 0 and 1 in the ground truth figure, while "back" indicates the region of electrode 2 and 3. Each element in the inferred regional connectivity matrix W reflects the elements of the corresponding regions in ground truth W altogether. Regional inferences after split precisely reveal the overall connectivity between regions.

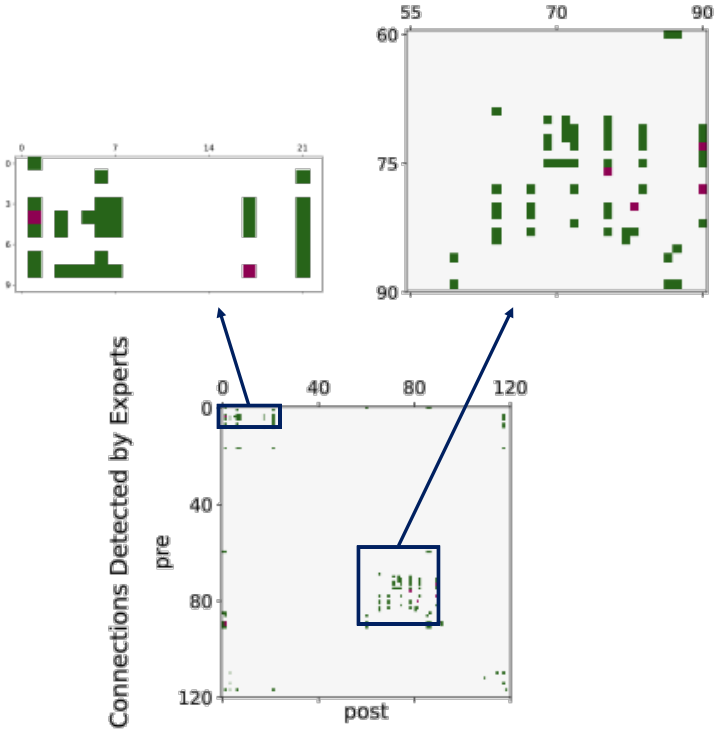


Figure 3.9: Comparison between our Bayesian inference and ground truth detected by neuroscience experts. Green represents the connections found by Bayesian inference, while red denotes the connections not found. 93.33% of the connections detected by the neuroscience experts are detected by Bayesian inference.

3.7 Results on Real Data

In this section, we apply our framework onto two sets of real MEA recordings. Throughout this section, we apply non-overlapping split with 4 regions of equal size as in Fig. 3.4. As we mentioned in Section 3.2.2, we use 3-minute recordings, which present as spike trains with shape of 180, 000 * 120. For the Bayesian inference, we adjusted the prior hyper-parameters and verified that the results stayed consistent.

3.7.1 Comparison with Neuron Experts' Labeled Ground Truth

For the real dataset, since we don't have the ground truth, we adopted neuroscience experts' labelling as our reference. The neuroscience experts labelled those connections

by watching MEA viewer [55]. We compare our Bayesian analysis with the expert labeled result. As is shown in Fig. 3.9, 93.33% of all the connections detected by neuron experts are detected by our inference. We can see that the real recording has some regional patterns, which satisfies the intuition of our splitting strategy. Because of the difficulty in detecting those connections in neuroscience experiments, our neuroscience experts can find some connections that they are confident in but cannot guarantee to find all the connections. However, in our Bayesian inference, we gave the overall possible connections based on our probabilistic model. Thus we found more connections using Bayesian inference compared with the neuroscience experts' result. We also calculated the latency between electrodes using CCG based on the neuroscience experts labeled ground truth connectivity. The latency shows a similar pattern as our inferred weights.

3.7.2 Cadmium Concentration vs. Controlled Experiment

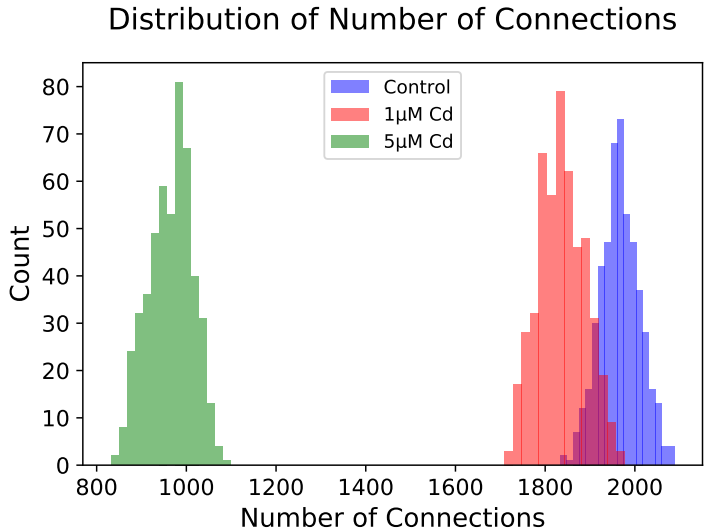


Figure 3.10: Posterior of number of connections with Cadmium concentration. Cd is short for Cadmium. The mean connection counts decrease with introduction of cadmium to the culture. Higher concentrations of cadmium lead to a further decrease in the number of connections in the functional networks.

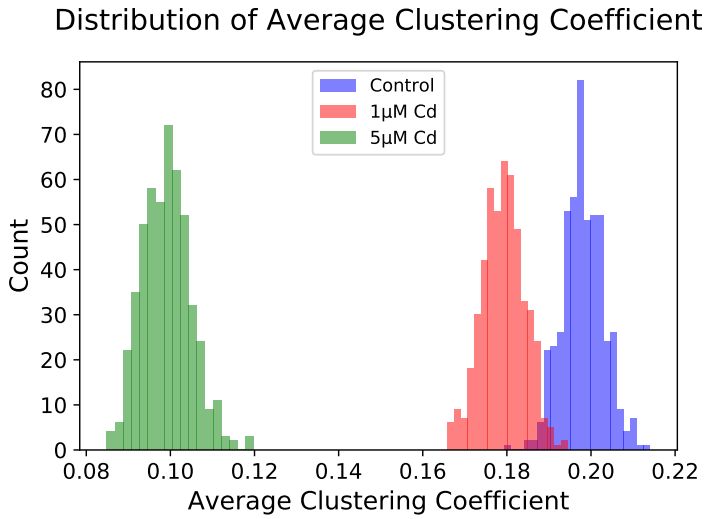


Figure 3.11: Posterior of average clustering coefficient with Cadmium concentration. Average clustering coefficient decreases with statistical significance with the introduction of Cadmium into the culture.

Cadmium is a toxic heavy metal that accumulates in living systems and is currently one of the most important occupational and environmental pollutants [56]. Cells have a calcium signalling toolkit whose components can be mixed and matched to create various spatial and temporal signals [57]. Cadmium can change the intracellular concentration of calcium, which is a universal and versatile intracellular messenger [58].

We reduced the release probability of presynaptic vesicles by titrating in increasing amounts of cadmium in order to decrease the influx of calcium current into the presynaptic terminal. Baseline recordings were performed 5 minutes prior to any cadmium addition. Following the baseline recording, the cadmium concentration was brought to $1\mu M$ and the solution was mixed gently. The MEA was placed on the recording headstage and allowed to equilibrate for 3 minutes, followed by a 3-minute recording. The same procedure was performed for $5\mu M$ cadmium in the same MEA.

To analyze how cadmium affects the structure of neuronal networks, we compared the obtained posterior matrices A and W for a set of MEA recordings, comprised of one control with no cadmium introduced and 2 others with $1\mu M$ and $5\mu M$ of cadmium.

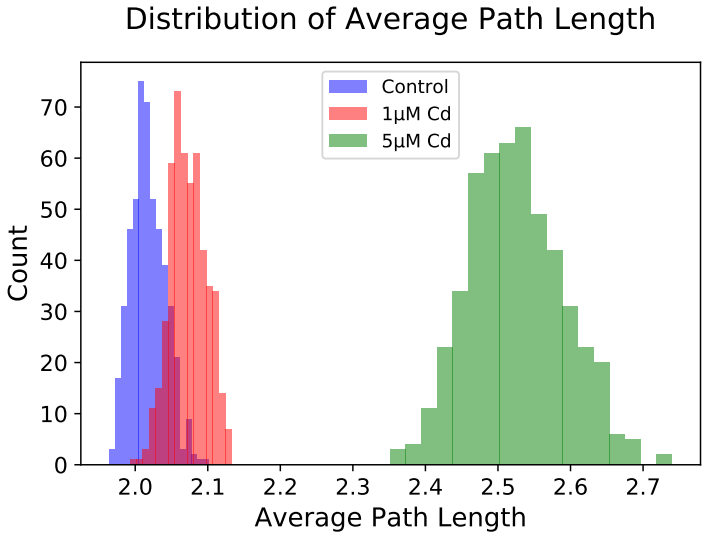


Figure 3.12: Posterior of average path length distribution with Cadmium concentration. The average path length increases significantly with the increase of cadmium concentration.

We used a threshold of 0.05 (the value is chosen based on the outlier of the weight distribution) to eliminate those tiny weights in the inferred weight matrix, which is noise in biology. To understand the topology, we look at the following graph metrics for each posterior sample: number of connections, average clustering coefficient, and average path length. Overall, we find that Cadmium does change the topology of the estimated networks with statistical significance in posterior regions. As is shown in Fig. 3.10, the mean connection counts decrease with introduction of cadmium to the culture. Higher concentrations of cadmium lead to a drastic decrease in the number of connections in the functional networks.

Clustering coefficient is the number of edges between a electrodes immediate neighbors divided by all possible connections that could exist among them. It measures the level of local connectivity between electrodes. In Fig. 3.11, average clustering coefficient decreases with statistical significance when cadmium is in the culture, indicating that cadmium cultures are less likely to have connections within tightly connected groups or

communities.

Average path length is the average shortest path length for all possible pairs of electrodes. Fig. 3.12 shows that the average path length increases significantly with the increase of cadmium concentration. That validates the idea that cadmium can impede neuron signal transmissions by changing the intracellular concentration of calcium.

In Fig. 3.13, we compare the indegree of each electrode for control and $5\mu M$ Cadmium concentration. We can see that the overall connection pattern maintains but the indegree decreases prominently with the introduction of Cadmium into the culture. Similar phenomenon can be detected for outdegree of each electrode for control and $5\mu M$ Cadmium concentration in Fig. 3.14.

3.8 Related Work

MEAs provide opportunities for researchers to understand neuronal connectivity, by recording spike trains but also presents severe data analysis challenges. Inferring functional connectivity networks is critical to many applications in neuroscience. To analyze high-dimensional spike trains, there exist several methods to measure connection weights. For example, CCG [45, 59] is a metric that was built from two electrodes spike trains. It presents the probability of an electrode firing to a spike in a τ milliseconds time window before or after another electrode fires. MIC has been widely used to identify known and novel relationships for data sets in gene expression and global health [46]. MIC is a two-variable dependent measure that captures functional relationship, by providing a score that roughly equals the coefficient of determination of the data relative to the regression function. Biophysically-inspired metrics extracts a directed functional connectivity matrix, based on the spike train [47]. It uses exponential decay property in axon potential signals to quantify the connection coefficients. However, despite the popularity of apply-

ing those metrics on MEA recordings, we can not rely on them to get reliable functional networks because they are deterministic and there is no model for the data.

GLM, a commonly used modeling framework [52, 60], can model the binary fire/not (1/0) and spike train recordings. We can use the *logit* link function to model the binomial distribution. GLMs are often fit by maximum a posteriori (MAP) estimation [61, 62]. However, when it comes to high-density recordings, MAP cannot fully convey the information in the posterior with a point estimate. Bayesian inference is a process using Bayes' theorem based on the prior beliefs about the probabilistic data generation process [63]. It updates the posterior for parameters in data generation model, as more data becomes available. GLM and graph-based models have been combined to infer the neuron connectivity patterns in a Bayesian way [49]. However, the Bayesian techniques have a considerable downside of the increased computation time, especially when the number of electrodes is large. In this chapter, we introduced a scalable Bayesian inference framework on large scale MEA recordings. The functional connectivities are inferred from a joint probabilistic model of GLM and networking.

3.9 Discussion

In this chapter we have focused on inferring functional neuronal network connections using MEA recordings. Specifically, our goal has been to infer the functional connectivity for MEAs with large numbers of probes. Along this line, we proposed a scalable Bayesian inference framework. Our framework makes use of the hierarchical structure of networks of neurons and splits the whole array into smaller local networks for network inference. The splitting strategy decreases the average sampling time quadratically with the number of sub-regions. We also provide a strategy for inferring the connectivity between local networks. By comparing with ground truth for both synthetic data and real world human

expert labeled MEA recordings, our experimental results provide compelling evidence that our framework can infer the underlying functional connectivity. Furthermore, we applied the proposed framework to a controlled cadmium dataset, and the results confirm its utility. As the density of the MEA continues to increase, our method will become more valuable to be able to infer the neuronal network structure efficiently.

Our experimental results demonstrate the usefulness of this framework. Here we suggest some avenues for future work. First, the model could be extended to account for more detailed biological knowledge. For example, there are different types of neurons such as motor neurons and interneurons, which exhibit different types of behaviors in response to incoming signals. The auto-regressive propensity model, which is currently linear, can also be improved to incorporate nonlinear effects or other mechanistic firing models. Secondly, developing a method to determine when and how to split can allow for the framework to be used more robustly. Moreover, using the framework to study different factors instead of Cadmium, such as the presence of certain genes, may produce a greater understanding of the biologically complex systems.

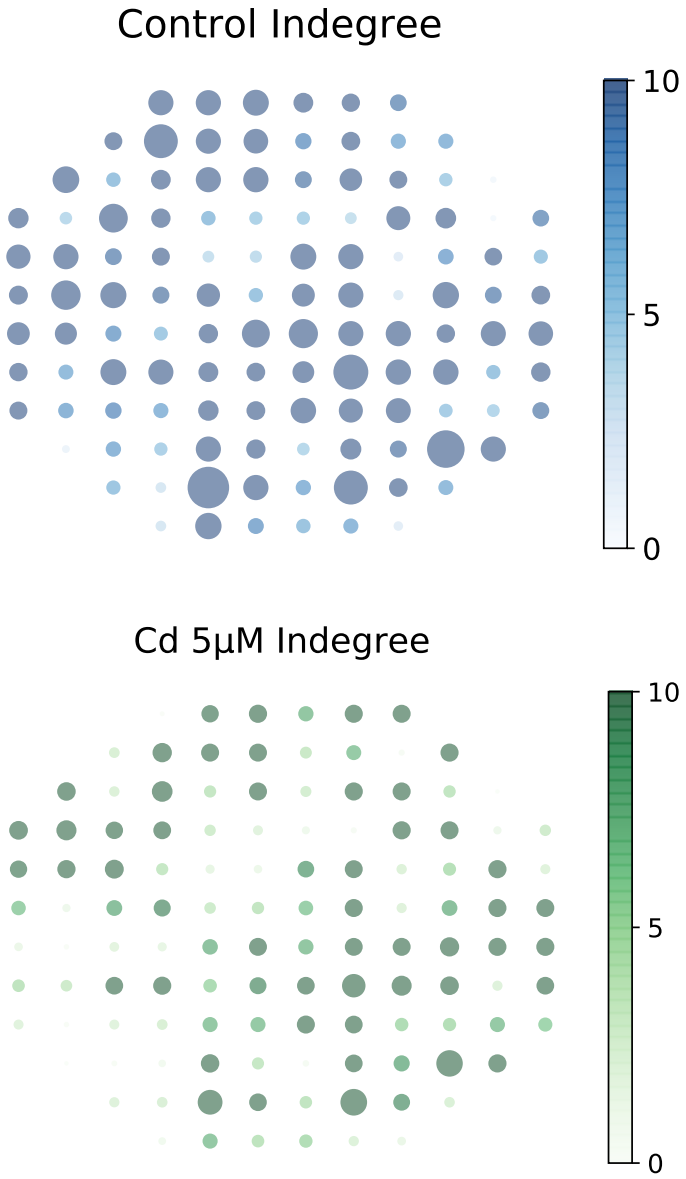


Figure 3.13: Incoming connections for control and Cadmium $5\mu M$. Nodes are sized and colored according to the number of incoming connections. The overall connection pattern remains but the indegree decreases prominently with the introduction of Cadmium into the culture.

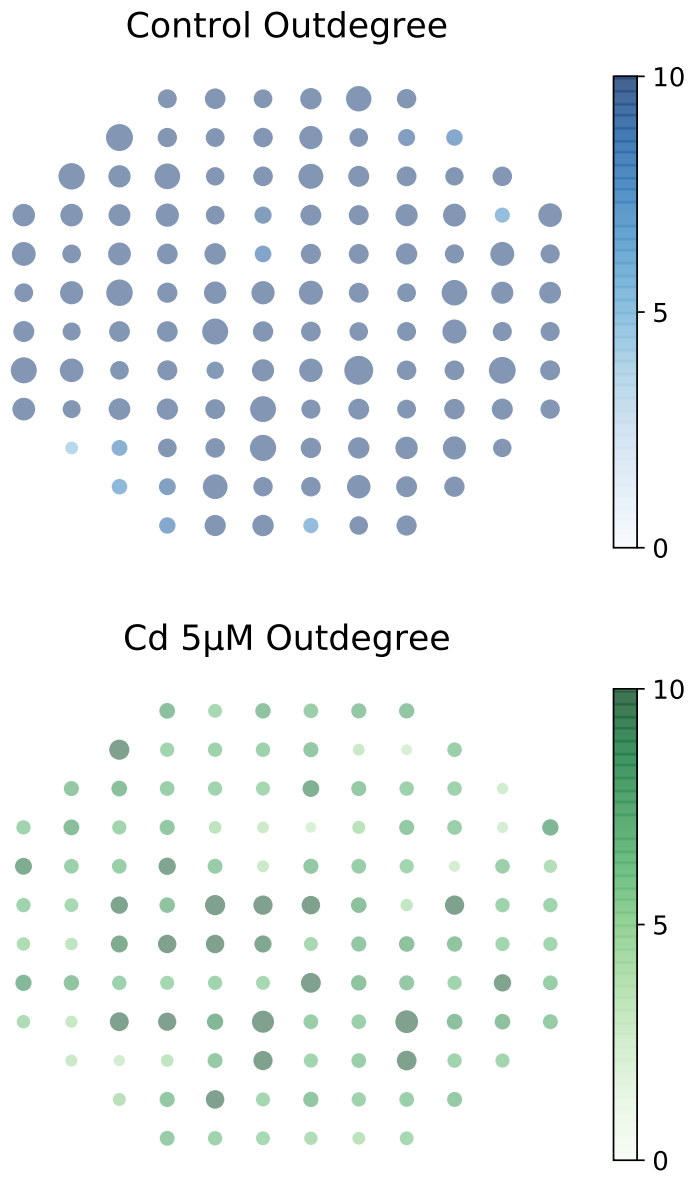


Figure 3.14: Outgoing connections for control and Cadmium $5\mu M$. Nodes are sized and colored according to the number of outgoing connections. The overall connection pattern remains but the outdegree decreases prominently with the introduction of Cadmium into the culture.

Part II

Data Mining in Healthcare

Chapter 4

Empirical Analysis of Machine Learning Configurations for Prediction of Multiple Organ Failure in Trauma Patients

4.1 Introduction

Multiple organ failure (MOF) is a clinical syndrome with variable causes including pathogens [64], complicated pathogenesis [65], and a major cause of mortality and morbidity for trauma patients who are admitted to Intensive Care Units (ICU) [66]. Based on recent studies on ICU trauma patients, up to 47% have developed MOF, and MOF increased the overall risk of death 6 times compared to patients without MOF [67]. To prevent the development of MOF for trauma patients from progression to an irreversible stage, it is essential to diagnose MOF early and effectively. Many scoring systems have been proposed to predict MOF [68, 69, 70, 71] and researchers have attempted to predict

MOF on trauma patients using predictive models in an early phase [72, 73].

The rapid growth of data availability in clinical medicine requires doctors to handle extensive amounts of data. As medical technologies become more complicated, technological advances like machine learning (ML) are increasingly needed to improve real-time analysis and interpretation of the results [74]. In recent years, practical uses of ML in healthcare have grown tremendously, including cancer diagnosis and prediction [75, 76, 77], tumor detection [78, 79], medical image analysis [80], and health monitoring [81, 82].

Compared to traditional medical care, ML-assisted clinical decision support enables a more standardized process for interpreting complex multi-modality data. In the long term, ML can provide an objective viewpoint for clinical practitioners to improve performance and efficiency [83]. ML is often referred to as a black box: explicit input data and output decisions, but opaque at intermediate learning process. Additionally, in medical domains, there is no universal rule for selecting the best configuration to achieve the optimal outcome. Moreover, medical data has its own challenges such as numerous missing values [84] and colinear variables [85]. Thus it is difficult to process the data and choose the proper model and corresponding parameters, even for a ML expert. Furthermore, detailed quantitative analysis of the potential impacts of different settings of ML systems on MOF has been missing.

In this chapter, we experiment with comprehensive ML settings for prediction of MOF, considering 6 different dimensions from data preprocessing (missing value treatment, label balancing, feature scaling), feature selection, classifier choice, to hyperparameter tuning. To predict MOF for trauma patients at an early stage, we use only initial time measurements (hour 0) as inputs. We mainly use area under the receiver operating characteristic curve (AUC) to evaluate MOF prediction outcomes. We focus on analyzing the relationships among configuration complexity, predicted performance, and

performance variation. Additionally, we quantify the relative impacts of each dimension.

The main contributions of this chapter include:

1. To the best of our knowledge, this is the first work to conduct a thorough empirical analysis quantifying the predictive performance with exhaustive ML configurations for MOF prediction.

2. We provide general guidance for ML practitioners in healthcare and medical fields through quantitative analysis of different dimensions commonly used in ML tasks.

3. Experimental results indicate that classifier choice contributes most to both performance improvement and variation. Complex classifiers including ensemble methods bring higher default/optimized performance, along with a higher risk of inferior performance compared to simple ones on average.

The remainder of this chapter is organized as follows. Section 4.2 describes the dataset and features we use. All of the ML configurations are available in Section 4.3. Experimental results are discussed in Section 5.4. Finally, our conclusions are presented in Section 4.5.

4.2 Dataset

Our dataset, collected from the San Francisco General Hospital and Trauma Center, contains 2190 highest level trauma activation patients evaluated at the level I trauma center. Due to the urgency of medical treatment, there are numerous missing values for time-dependent measurements. Thus we have chosen to consider only those features with a maximum missing value percentage of 30% over all patients. To obtain a timely prediction, early lab measurements (hour 0) as well as patients' demographic and illness

information were extracted as the set of features. Detailed feature statistics are available in Table 4.1.

Feature type	# of extracted features	Features
Demographic	5	*gender*, age, weight, *race, blood type*
Illness	2	*comorbidities, drug usage*
Injury factors	4	*blunt/penetrating trauma*, # of rib fractures, *orthopedic injury*, *traumatic brain injury*
Injury scores	8	injury severity score, 6 abbreviated injury scale (head, face, chest, abdomen, extremity, skin), Glasgow coma scale score
Vital sign measurements	4	heart rate, respiratory rate, systolic blood pressure, mean arterial pressure
Blood-related measurements	13	white blood cell count, hemoglobin, hematocrit, serum CO_2, prothrombin time, international normalized ratio, partial thromboplastin time, blood urine nitrogen, creatinine, blood pH, platelets, base deficit, *factor VII*

Table 4.1: MOF dataset statistics. Italicized features are categorical.

Our target variable consists of binary class labels (0 for no MOF and 1 for MOF). Then, the data with feature and target variables is randomly split into training and testing sets at the ratio of $7:3$.

4.3 Methods

Based on ML pipelines and special characteristics of our data such as large number of missing values and varying scales in feature values, we consider comprehensive ML con-

figurations from the following 6 dimensions: data preprocessing (missing value treatment (MV), label balancing (LB), feature scaling (SCALE)), feature selection (FS), classifier choice (CC), and hyperparameter tuning (HT). In the remainder of the chapter, we will interchangeably use the full name and corresponding abbreviations shown in parentheses. Further details on each dimension are described below.

4.3.1 Data Preprocessing

Methods to handle the dataset with missing values, imbalanced labels, and unscaled variables are essential for the data preprocessing process. We use several different methods to deal with each of these problems.

Missing Value Treatment

In our dataset, numerous time-dependent features cannot be recorded on a timely basis, and missing data is a serious issue. We consider three different ways to deal with missing values, where the first method serves as the baseline setting for MV, and the latter two methods are common techniques of missing value imputation in ML.

1. Remove all patients with any missing values for the features listed in Section 4.2.
2. Replace missing values with mean for numerical features and mode for categorical features over all patients.
3. Impute missing values by finding the k-nearest neighbors with the Euclidean distance metric for each feature respectively.

Label Balancing

Our dataset is imbalanced as the sample class ratio between class 0 and class 1 is $11:1$. Keeping imbalanced class labels serves as the baseline setting for LB. Three

different ways are considered to resample the training set.

1. Oversampling the minority class (label 1)

   1.1 Method: SMOTE (synthetic minority over-sampling technique) [86].

   1.2 Explanation: choose k-nearest neighbors for every minority sample and then create new samples halfway between the original sample and its neighbors.

2. Undersampling the majority class (label 0)

   2.1 Method: NearMiss [87].

   2.2 Explanation: when samples of both classes are close to each other, remove the samples of the majority class to provide more space for both classes.

3. Combination of oversampling and undersampling

   3.1 Method: SMOTE & Tomek link [88].

   3.2 Tomek link: two samples are k-nearest neighbors to each other but come from different classes.

   3.3 Explanation: first create new samples for the minority class and then remove the majority class sample in any Tomek link.

Feature Scaling

Since the range of feature values in our dataset varies widely, we perform feature scaling. No scaling on any feature serves as the baseline setting for SCALE. Two common scaling techniques are used for numerical features.

1. Normalization: rescale values to range between 0 and 1.

2. Standardization: rescale values with mean 0 and standard deviation 1.

4.3.2 Feature Selection

In medical datasets, there usually exist many highly correlated features, and some features that are weakly correlated to the target [85, 89]. Thus it is essential to identify the most relevant features that may help to improve the outcome of the analysis. Using all of the features described in Section 2 serves as the baseline setting for FS. We consider two main feature selection techniques: filter and wrapper methods.

1. Filter-based methods (independent of classifiers):

    1.1 Use correlation between features and the target to select features which are highly dependent on the target.

    1.2 Filter out numerical features using ANOVA F-test and categorical features using χ^2 test.

2. Wrapper-based methods (dependent on classifiers):

    2.1 Method: RFE (recursive feature elimination) in random forest.

    2.2 Explanation: perform RFE repeatedly such that features are ranked by importance, and the least important features are disregarded until a specific number of features remains.

4.3.3 Classifier Choice

We experimented with 15 classifiers on the dataset. In general, these classifiers can be divided into two main categories: single and ensemble. Lists of all classifiers are available in Table 4.2. For ensemble classifiers (combination of individual classifiers), we tried bagging (BAG, RF, ET), boosting (GB, ABC, XGB, LGBM), voting (VOTE) and stacking (STACK). In bagging, DT is a homogeneous weak learner. Multiple DTs learn

the dataset independently from each other in parallel and the final outcome is obtained by averaging the results of each DT. In boosting, DT also serves as a homogeneous weak learner. However, DTs learn the dataset sequentially in an adaptive way (new learner depends on previous learners' success), and the final outcome is determined by weighted sum of previous learners. In voting, heterogeneous base estimators (LR, RF, SVM, MLP, ET) are considered, where each estimator learns the original dataset and the final prediction is determined by majority voting. In stacking, several heterogeneous base learners (RF, KNN, SVM) learn the dataset in parallel, and there exists a meta learner (LR) that combines the predictions of the weak learners. Abbreviations of classifiers shown in parentheses for voting and stacking are the ones we use.

Single classifiers	Ensemble classifiers
Logistic Regression (LR)	Bagged Trees (BAG)
Support Vector Machine (SVM)	Random Forest (RF)
Naive Bayes (NB)	Extra Trees (ET)
K-nearest Neighbors (KNN)	Gradient Boosting (GB)
Decision Tree (DT)	Adaptive Boosting (ABC)
Multi-layer Perceptron (MLP)	Extreme Gradient Boosting (XGB)
	Light Gradient Boosting Machine (LGBM)
	Voting (VOTE)
	Stacking (STACK)

Table 4.2: List of 6 single classifiers and 9 ensemble classifiers. Corresponding abbreviations of each classifier are shown in parentheses.

4.3.4 Hyperparameter Tuning

Hyperparameters are crucial for controlling the overall behavior of classifiers. Default hyperparameters of classifiers serve as the baseline setting for HT. We apply grid search to perform hyperparameter tuning for all classifiers. Detailed information about tuned hyperparameters is available in Table 4.3.

Classifiers	♯ of tuned hyperparameters	Hyperparameter lists
LR	3	C, class_weight, penalty
SVM	4	C, gamma, kernel, class_weight
KNN	3	n_neighbors, weights, algorithm
NB	1	var_smoothing
DT	5	min_samples_split, max_depth, min_samples, leaf_max_features, class_weight
MLP	3	activation, solver, alpha
BAG	2	base_estimator, n_estimators
RF	2	n_estimators, max_features
ET	2	n_estimators, max_features
GB	2	n_estimators, max_depth
ABC	3	base_estimator, n_estimators, learning_rate
XGB	2	min_child_weight, max_depth
LGBM	4	num_leaves, colsample_bytree, subsample, max_depth
VOTE	2	C (SVM), n_estimators (ET)
STACK	2	C (SVM), n_neighbors (KNN)

Table 4.3: Detailed configurations of tuned hyperparameters for all classifiers. All of the hyperparameter names come from *scikit-learn* [90].

4.4 Experiments and Results

We formulated MOF prediction as a binary classification task. All of the experiments in this chapter were implemented using *scikit-learn* [90]. As mentioned in Section 4.2, our training and testing dataset is randomly split with a ratio of 7 : 3. One-hot encoding is applied to all categorical features. For each classifier, we use the same training and testing dataset. We use AUC as our main performance metric, as it is commonly used for MOF prediction in the literature [69, 91, 92]. It provides a summary" of classifier performance compared to single metrics such as precision and recall. AUC represents the probability that a classifier ranks a randomly chosen positive sample (class 1) higher than a randomly chosen negative sample (class 0), and thus useful for imbalanced datasets. In this section, we quantify the impacts (improvement and variation) of each dimension on the predicted performance over our testing dataset.

4.4.1 Influence of Individual Dimensions

First, we evaluate how much each dimension contributes to the AUC score improvement and variation respectively, and find the correlation between performance improvement and variation over all dimensions.

Performance Improvement across Dimensions

For HT, MV, LB, SCALE, and FS, we define the ***baseline*** as default hyperparameter choices, using no missing value imputation, no label balancing, no feature scaling, and no feature selection, respectively. For CC, we choose SVM, which achieves the median score among all classifiers, as the ***baseline***. Then we quantify the performance improvement of each dimension. Fig. 4.1 shows the percentage that each dimension contributes to the improvement in the AUC score over baseline by tuning only one dimension at a

time while leaving others at baseline settings. We observe that CC contributes most to the performance improvement (15.00%) for MOF prediction. After CC, LB (10.81%), FS (10.09%), MV (7.90%), HT (6.94%), and FS (2.45%) bring decreasing degrees of performance improvement in the AUC score.

Table 4.4 shows the improvement of every single dimension on each classifier over the baseline. In general, MV and LB tend to provide the greatest performance improvement for most classifiers. For RF, ET, and LGBM, FS contributes the most to improvement in performance since these classifiers require feature importance ranking intrinsically, and external FS improves their prediction outcomes to a large extent. Note that the classifier for which SCALE has the largest impact is KNN, as it is a distance-based classifier which is sensitive to the range of feature values. Also, due to instability and tendency to overfit, HT is the most critical for DT improvement.

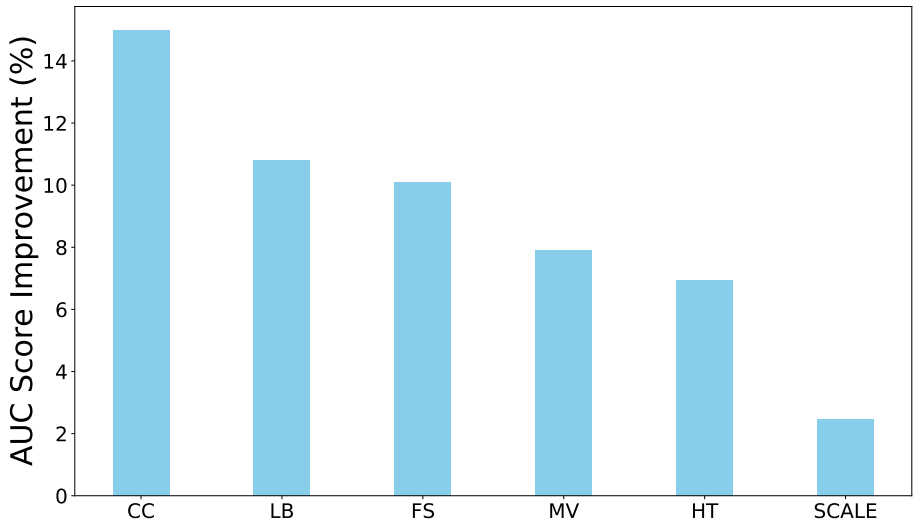


Figure 4.1: Performance improvement in the AUC score of each dimension over the baseline when tuning only one dimension at a time while leaving others at baseline settings. CC brings the greatest performance improvement, followed by LB, FS, MV, HT, and SCALE in decreasing order of improvement.

Classifier	MV (%)	LB (%)	SCALE (%)	FS (%)	HT (%)
LR	2.78	**11.48**	0.30	5.50	3.03
SVM	3.37	**26.83**	2.38	20.95	3.21
KNN	13.60	11.85	**17.68**	13.12	15.60
NB	0.60	**38.90**	0.17	4.12	2.84
DT	12.87	16.22	0.42	15.34	**38.85**
BAG	2.94	**8.91**	0.28	7.05	5.43
RF	4.13	5.34	0.28	**5.85**	1.04
ET	3.82	7.87	0.00	**18.96**	1.33
ABC	**19.33**	7.02	0.00	16.99	12.99
GB	**12.44**	3.81	0.02	6.63	4.08
LGBM	7.03	1.85	2.75	**10.39**	3.13
XGB	**11.46**	3.97	0.02	7.47	4.27
MLP	**10.78**	5.08	6.05	7.53	5.69
STACK	6.94	**8.94**	4.32	5.48	1.82
VOTE	**6.38**	4.00	2.11	6.04	0.85

Table 4.4: Column 1 shows a total of 15 classifiers. Columns 2 to 6 represent the percentage (two decimal places accuracy) of AUC score improvement when tuning each individual dimension while leaving other dimensions at baseline settings for each classifier. Bold entries represent the dimension that contributes to the largest improvement for the specific classifier. MV and LB tend to dominate in performance improvement for most classifiers.

In addition to AUC, 6 other performance metrics are used to measure the performance improvement degree of each dimension. The results in Table 4.5 reveal that CC brings the greatest improvement regardless of the metrics we use. Contributions from HT and SCALE are relatively small compared to other dimensions.

	AUC	F-score	G-mean	Precision	Sensitivity/ Recall	Specificity	Accuracy
CC (%)	15.00	15.58	10.50	16.41	10.50	11.86	10.50
LB (%)	10.81	11.34	9.33	13.27	9.33	10.72	9.34
FS (%)	10.09	7.33	6.30	10.61	6.30	6.94	6.30
MV (%)	7.90	5.30	4.60	5.83	4.59	4.95	4.59
HT (%)	7.46	2.11	3.21	3.41	3.20	4.64	3.20
SCALE (%)	2.45	1.04	0.65	3.03	0.65	0.48	0.65

Table 4.5: Performance improvement in different metrics of each dimension. The performance improvement of each dimension on other metrics displays an order consistent with that of the AUC score.

Performance Variation across Dimensions

For all of the ML configurations, we further investigate how much each dimension contributes to the performance variation in the AUC score. By tuning only one dimension at a time while leaving other dimensions at baseline settings, we obtain a range of AUC scores. Performance variation is the difference between the maximum and the minimum score of each dimension. Fig. 4.2 shows the proportion of each dimension that brings the performance variation in the AUC score. Based on Fig. 4.2, we notice that CC, which brings the largest performance improvement, also brings the largest performance variation (10.98 %). After CC, LB (7.00 %), FS (6.93 %), MV (5.64 %), HT (4.97 %), and SCALE (1.66 %) bring decreasing degrees of performance variation in the AUC score.

Table 4.6 shows the variation of every single dimension on each classifier over the baseline. We observe that for each classifier, if one dimension brings a larger performance improvement, it also results in a larger performance variation. For our assessment of performance variation, the same metrics as above are used for evaluation on each dimension.

Using the same metrics as above, Table 4.7 shows that the proportion of performance variation in different metrics from each dimension follows an order that is consistent with the performance improvement in Table 4.5. Thus, for different metrics, greater improvement brings greater variation of each dimension. For every step that researchers take when predicting MOF using ML, they should always be aware of the trade-off between benefits (improvement in performance) and risks (variation in performance) when adjusting each dimension.

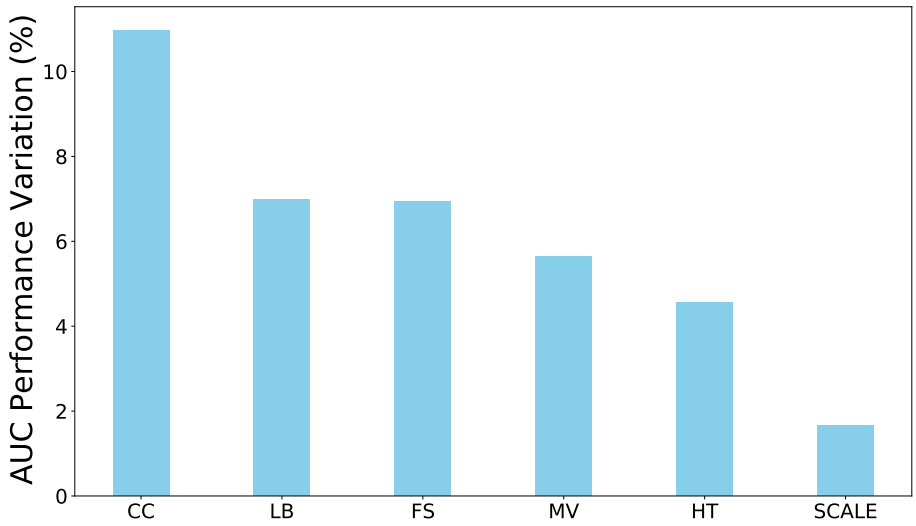


Figure 4.2: Performance variation in the AUC score when tuning only one dimension at a time while leaving others at baseline settings. CC brings the greatest performance variation, followed by LB, FS, MV, HT, and SCALE in decreasing order of variation. Larger improvement also brings the risk of larger variation for each dimension.

Classifier	MV (%)	LB (%)	SCALE (%)	FS (%)	HT (%)
LR	2.28	**8.44**	0.25	4.27	2.49
SVM	2.45	**16.87**	1.79	13.04	2.42
KNN	7.97	6.95	**10.36**	7.69	9.14
NB	0.48	**22.22**	0.13	3.25	2.25
DT	7.13	8.99	0.23	8.50	**21.53**
BAG	2.22	**6.17**	0.21	5.26	4.10
RF	3.35	4.14	0.23	**4.49**	0.84
ET	3.22	6.14	0.00	**13.41**	1.12
ABC	**13.09**	4.61	0.00	11.51	9.40
GB	**9.59**	2.83	0.02	5.11	3.14
LGBM	5.54	1.43	2.16	**7.76**	2.47
XGB	**8.70**	3.01	0.02	5.59	3.24
MLP	**7.89**	3.55	4.43	5.30	4.16
STACK	5.47	**6.47**	3.41	4.20	1.43
VOTE	**5.19**	3.13	1.71	4.63	0.69

Table 4.6: Columns 2 to 6 represent the proportion (two decimal places accuracy) of each dimension that contributes to the performance variation in the AUC score. Bold entries represent the dimension that contributes to the largest variation for the specific classifier. MV and LB tend to result in larger performance variation for most classifiers.

	AUC	F-score	G-mean	Precision	Sensitivity/ Recall	Specificity	Accuracy
CC (%)	10.98	11.87	8.57	12.86	8.57	10.60	8.57
LB (%)	7.00	7.29	6.83	9.02	6.83	10.14	6.82
FS (%)	6.93	5.27	4.38	7.62	4.37	4.70	4.38
MV (%)	5.64	4.36	3.69	4.77	3.69	2.98	3.68
HT (%)	4.87	2.55	3.52	1.54	3.52	2.72	3.53
SCALE (%)	1.66	0.88	0.57	1.47	0.57	0.46	0.57

Table 4.7: Performance variations in different metrics of each dimension. The performance variation of each dimension on other metrics displays an order that is consistent with that of the AUC score.

4.4.2 Performance Comparison across Classifiers

We have shown that classifier choice is the largest contributor to both performance improvement and variation in the AUC score. Hence, we further investigate the performance differences among classifiers. Specifically, we investigate the relationships among classifier complexity, performance, and performance variation.

Default versus Optimized Performance

Default classifiers are defined as classifiers with default parameters, while ***optimized*** classifiers are those for which hyperparameter tuning with 10-fold cross validation is applied using grid search. We compare the performance of default and optimized classifiers in consideration of all other dimensions, i.e., MV, LB, SCALE, and FS. The average AUC scores of all classifiers with default and optimized settings are shown in Fig. 4.3. In general, ensemble classifiers perform better than single classifiers regardless of default or optimized performance.

In addition to AUC, 6 other performance metrics are used to evaluate the performance of all classifiers. We use the median score to rank classifiers with both default and optimized settings. Then, NDCG (normalized discounted cumulative gain), one of the most prevalent measures of ranking quality [93], is used to compare classifier rankings between each of these metrics and the AUC score. Detailed relevance scores are shown in Table 4.8. The result indicates that the median performance of each classifier is similar no matter which metric is used. This also suggests that the AUC score can represent classifiers' overall performance well.

Based on the above experiments, ensemble classifiers should be prioritized in MOF prediction since they usually bring better predictive performance than single classifiers.

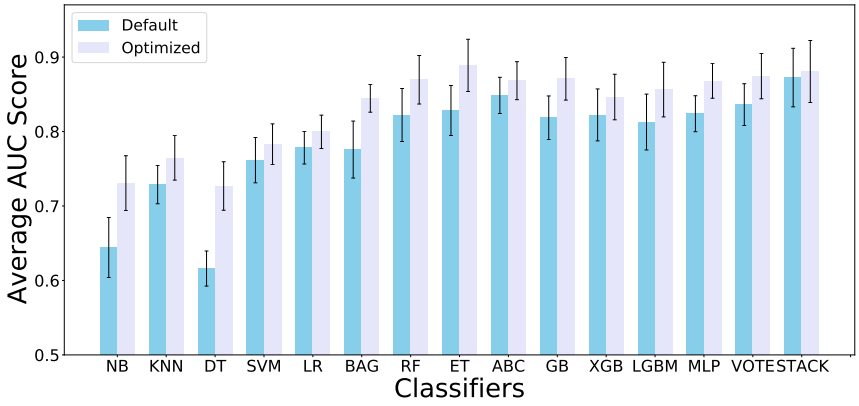


Figure 4.3: Comparison of default and optimized performance over all classifiers. Classifiers listed on the left-hand side of BAG are single while the ones on the right–hand side are ensemble and MLP. Overall, ensemble methods have better default and optimized performance compared with single classifiers.

	Default (%)	Optimized (%)
F-score	96.92	97.92
G-mean	96.46	97.63
Precision	95.49	90.01
Sensitivity/Recall	98.42	97.59
Specificity	95.35	97.46
Accuracy	96.46	97.59

Table 4.8: Column 1 represents 6 other performance metrics. Columns 2 and 3 show the NDCG score between each of these metrics and the AUC score when ranking 15 classifiers by their median performance in default and optimized settings, respectively. Median performance of classifiers is similar regardless of which metric to use.

Performance Variation across Classifiers

We measure the performance variation for each classifier in consideration of all other dimensions, i.e., MV, LB, SCALE, FS, and HT. For each classifier, we get a range of AUC scores. The size of the range determines the extent of performance variation. Fig. 4.4 shows the performance variation in the AUC score of all classifiers. The order of listed classifiers on the x-axis is based on increasing model complexity, which is measured by classifier training time with default settings. The complexity of classifiers and performance variation demonstrates an evident U-shaped' relationship. When the classifier is too simple', its performance variation is relatively large. When the complexity of the classifier is appropriate', the performance variation is relatively small. If the classifier becomes too complex', it is also at the risk of larger performance variation. Therefore, classifiers with appropriate' complexity are more stable, with smaller changes in performance, while too simple' or too complex' classifiers are relatively unstable with larger changes in performance in general.

In addition to AUC, the same metrics as above were used to validate the performance variation of all of the classifiers. We use the range (difference between maximum and minimum scores) to rank classifiers in consideration of MV, LB, SCALE, FS, and HT. Then, NDCG is used to compare classifier rankings between each of these metrics and the AUC score. Table 4.9 displays detailed relevance scores. The result suggests that other metrics show a similar U-shaped' relationship between classifier complexity and performance variation as the AUC score. When predicting MOF, it is inappropriate for clinical practitioners to choose too simple' and too complex' classifiers since they may run the risk of underfitting and overfitting, respectively.

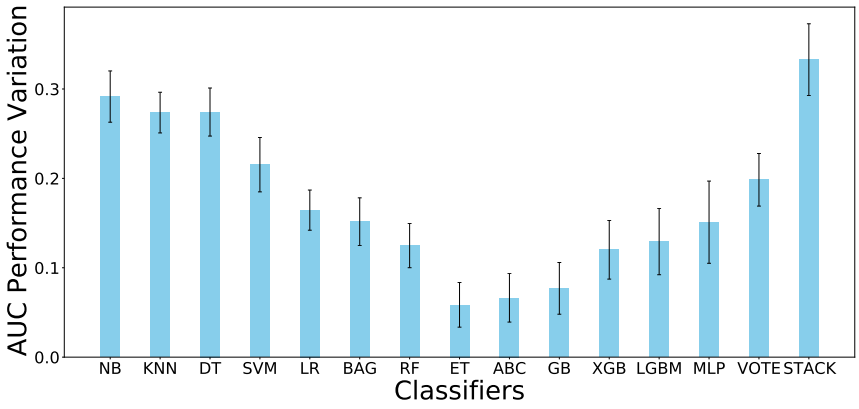


Figure 4.4: Performance variation comparison over all classifiers. The order of classifiers listed on the x-axis is based on increasing model complexity. Too simple' and too complex' classifiers result in larger performance variation. The performance variation of classifiers with appropriate' complexity is relatively small.

	F-score	G-mean	Precision	Sensitivity/ Recall	Specificity	Accuracy
Relevance (%)	93.15	94.98	94.37	93.24	93.13	93.77

Table 4.9: NDCG score between each of 6 other performance metrics and the AUC score in terms of classifier complexity and performance variation. Different metrics show a similar U-shaped' relationship.

4.5 Discussion

We have provided a timely MOF prediction using early lab measurements (hour 0), patients demographic and illness information. Our study quantitatively analyzes the performance via the AUC score in consideration of a wide range of ML configurations for MOF prediction, with a focus on the correlations among configuration complexity, predicted performance, and performance variation. Our results indicate that choosing the correct classifier is the most crucial step that has the largest impact (performance and variation) on the outcome. More complex classifiers including ensemble methods can provide better default/optimized performance, but may also lead to larger performance degradation, without careful selection. Clearly, more MOF data is needed to provide a more general conclusion. Our work can potentially serve as a practical guide for ML practitioners whenever they conduct data analysis in healthcare and medical fields.

Chapter 5

Multiple Organ Failure Prediction with Classifier-Guided Generative Adversarial Imputation Networks

5.1 Introduction

Multiple organ failure (MOF) is a life-threatening syndrome with variable causes, including sepsis [94], pathogens [95], and complicated pathogenesis [96]. It is a major cause of death in the surgical intensive care unit (ICU) [97]. Care in the first few hours after admission is critical to patient outcomes. This period is also more prone to medical decision errors in ICUs than later times [98]. Thus, an effective and real-time prediction is essential for clinicians to provide appropriate treatment and increase the survival rates for MOF patients.

With the rapid growth of electronic health record (EHR) data availability, machine learning models have drawn increasing attention for MOF prediction. Missing values are a pervasive and serious medical data issue, which could be caused by various reasons such as lost records or inability to collect the data during some time periods [99]. There exist many preprocessing imputation algorithms, such as mean value imputation [100], multivariate imputation by chained equations (MICE) [101] and generative adversarial imputation nets (GAIN) [102] which impute missing components by adapting generative adversarial networks (GANs). However, these methods focus only on constructing the distribution between the unobserved components and the observed ones, without considering the underlying connections with specific downstream tasks, as is shown in the left of Figure 5.1.

Recently, GANs [103] have made significant progress on data generation. Labels can be incorporated in the GAN framework, e.g. CGAN [104] and AC-GAN [105], to generate label-aware outputs. Semi-supervised GAN [106] introduces a classifying discriminator to output either the validity of data or its class. Triple-GAN [107] proposes a three-player adversarial network, which contains an additional classifier to separate the role from the discriminator. Although the classifier helps the generator control the semantics of the generated samples and improves class prediction, the generator has to take ground truth label to generate label-aware data, which is not applicable in classification problems during inference.

In this chapter, we propose a classifier-guided missing value imputation framework for MOF prediction with early-stage measurements after admission. The generator uses observed data and random noise to impute missing components and obtains imputed data; the classifier takes the generators outcomes, models the relationship between imputed data and labels by joint training with the generator, and outputs estimated labels.

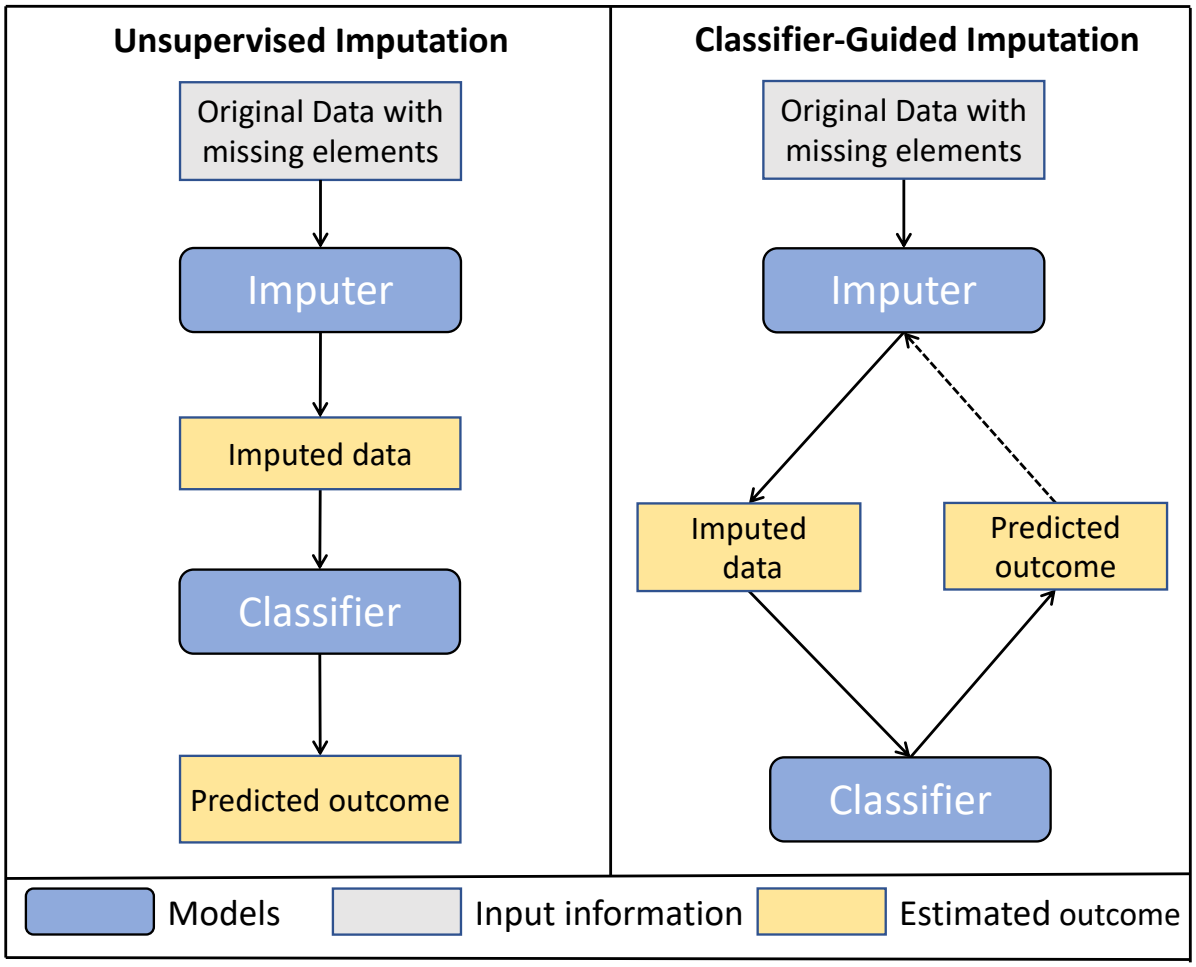


Figure 5.1: Workflow comparison of unsupervised imputers (left) and Classifier–GAIN (right). Classifier-guided imputers learn from the classifier during training and improve classification during inference, while unsupervised imputers learn only from the partially observed data in the data-prepossessing phase. The solid lines represent processes that occur during both the training and inference phase, while the dashed line represents the step that only occurs during training. Note that our final goal is to improve the classifier's performance utilizing imputed data.

The discriminator attempts to identify which component is observed by taking imputed data from the generator and predicted label information from the classifier.

The key contributions of this chapter include:

1) We propose a classifier-guided missing value imputation deep learning framework for MOF prediction, which incorporates both observed data and label for modeling label-aware imputation during training to help classification during inference. To the best of our knowledge, this is the first GAN-based end-to-end deep learning architecture for MOF prediction with missing values.

2) Experimental results on both synthetic and real-world MOF datasets show that our Classifier-GAIN outperforms GAIN and MICE consistently in different missing ratio scenarios and evaluation metrics.

3) Visualization of the values imputed by our approach further validates the effectiveness of Classifier-GAIN compared to various baselines.

The remainder of our chapter is organized as follows. Preliminaries are introduced in Section 5.2, followed by the details of our proposed approach in Section 5.3. Experimental results are reported in Section 5.4. In Section 5.5, we review the existing related work, and conclusions are given in Section 5.6.

5.2 Preliminaries

We formulate the MOF prediction as a binary classification problem with missing components in multiple features. In this section, we describe the problem definition in Section 5.2.1 and review the GAIN imputation algorithm in Section 5.2.2. The related notations are summarized in Table 5.1. Specifically, throughout this chapter, we utilize lower-case letters, e.g. $\mathbf{x}$, to denote the data vector. $p(\mathbf{x})$ is the probability distribution

function of $\mathbf{x}$. $\mathbf{1}$ denotes a d-dimensional vector of 1s, and letters with hats such as $\hat{\mathbf{x}}$ denote estimated vectors.

Table 5.1: Notation definitions

Notations	Description
i	index of observations
j	index of observed features
$d \in \mathbb{N}$	number of observed features
$N \in \mathbb{N}$	total number of observations
$n \in \mathbb{N}$	size of minibatch
$\mathbf{x}$	data vector
y	outcome indicator
$\mathbf{m}$	mask vector
$\mathbf{z}$	noise vector
$\mathbf{h}$	hint vector
$\mathbf{b}$	binary vector for calculating hint
$\tilde{\mathbf{x}}$	combination of partially observed data and *NA*s
$\ddot{\mathbf{x}}$	combination of partially observed data and noise
G	generator
C	classifier
D	discriminator
$\mathbf{g}$	reconstructed vector, the output of G
$\hat{\mathbf{x}}$	imputed data vector
$\hat{\mathbf{m}}$	estimated mask, the output of D
$\hat{y}$	estimated label, the output of C

5.2.1 Problem Definition

Let $\mathcal{X}^d$ be a d-dimensional space and $\mathbf{x}$ a data vector, taking values in $\mathcal{X}^d$ following distribution $p(\mathbf{x})$. We denote x_j as the j-th feature in $\mathbf{x}$. Binary mask vector $\mathbf{m} \in \{0,1\}^d$ indicates if the corresponding element in $\mathbf{x}$ is missing or not, where x_j is observed if $m_j = 1$, otherwise x_j is missing. To clarify the observed and missing components, we define a new vector $\tilde{\mathbf{x}} = (\tilde{x}_1, \cdots, \tilde{x}_d)$ as follows:

$$\tilde{x}_{j\in\{1,2,\ldots,d\}} = \begin{cases} x_j, & \text{if } m_j = 1, \\ NA, & \text{if } m_j = 0. \end{cases}$$

Supposing that $y \in \{0, 1\}$ is the binary outcome indicator for each sample, we can represent the dataset as a collection of N i.i.d. samples $\{(\tilde{\mathbf{x}}_i, \mathbf{m}_i), y_i\}_{i=1}^{N}$.

We aim to impute the missing components in every $\tilde{\mathbf{x}}$, and predict the outcome y for all samples by leveraging the imputed data vector. Formally, we seek to model $p(y|\tilde{\mathbf{x}})$: the conditional distribution of the task outcome given a partially observed data vector.

5.2.2 Generative Adversarial Imputation Networks (GAIN)

GAIN [102] was proposed to impute missing components with a GAN framework. In GAIN, the generator obtains the observed components in $\tilde{\mathbf{x}}$, mask vector $\mathbf{m}$ and a noise vector $\mathbf{z}$ as inputs, and outputs a completed data vector. The discriminator tries to distinguish the observed components and the missing ones. Furthermore, a hint vector $\mathbf{h}$ is introduced to provide additional missing information for alleviating the diversity of the imputation result.

The generator, G, takes

$$\ddot{\mathbf{x}} = \mathbf{m} \odot \tilde{\mathbf{x}} + (\mathbf{1} - \mathbf{m}) \odot \mathbf{z},$$

a combination of $\tilde{\mathbf{x}}$ and $\mathbf{z}$ by element-wise multiplication with $\mathbf{m}$, as input, and outputs $\mathbf{g}$, the reconstructed vector,

$$\mathbf{g} = G(\ddot{\mathbf{x}}).$$

Note that $\mathbf{g}$ is an output vector for every component, even if values are not missing in the data vector.

Thus, another element-wise multiplication is performed to calculate the imputed data vector via

$$\hat{\mathbf{x}} = \mathbf{m} \odot \tilde{\mathbf{x}} + (\mathbf{1} - \mathbf{m}) \odot \mathbf{g},$$

where $\hat{\mathbf{x}}$ is obtained by taking the observed part in $\tilde{\mathbf{x}}$ and replacing each NA by the corresponding value in $\mathbf{g}$.

The discriminator serves as an adversarial character to train G by taking in the imputed data vector $\hat{\mathbf{x}}$ and the hint vector $\mathbf{h}$, following the distribution $p(\mathbf{h}|\mathbf{m})$. The output of the discriminator is a distribution to identify which components in $\hat{\mathbf{x}}$ are observed. To help the discriminator distinguish imputations and observations, $\mathbf{h}$ provides certain information about $\mathbf{m}$ and its amount can be controlled by adjusting $\mathbf{h}$ in different settings. Specifically, a binary random variable $\mathbf{b} \in \{0, 1\}^d$ is randomly drawn with $P(b_j = 1) = p$. Then, $\mathbf{h}|\mathbf{m}$ is calculated by

$$\mathbf{h} = \mathbf{m} \odot \mathbf{b} + 0.5(\mathbf{1} - \mathbf{b}),$$

such that the discriminator will get mask information by $h_j = m_j$ if $b_j = 1$, otherwise, no information provided.

5.3 Methodology

Notably, conditional information such as labels can improve the performance of the generator [104, 105], and the completed data vector can enhance its task prediction result. However, state-of-art imputation methods, e.g. GAIN, do not make use of the relationship between observations and outcome labels, which could provide additional information to help downstream classification tasks. Therefore, we propose Classifier-GAIN to bridge this gap. Figure 5.2 depicts the overall architecture. We explain each of the

components and the training process of Classifier-GAIN in detail in Section 5.3.1∼ 5.3.2.

5.3.1 Classifier-guided Generative Adversarial Imputation Nets

In an imputation setting, we propose to fill in the missing components *NA*s in $\tilde{\mathbf{x}}$, using the distribution of data obtained by the generator, G. To guide the data imputation and predict the final task outcome, the classifier, C, takes imputed data and is trained together with G. The discriminator D plays an adversarial role to train G, with additional label prediction information from C.

Generator and Classifier

Similar to the structure of GAIN, G takes $\tilde{\mathbf{x}}$ as input and outputs $\hat{\mathbf{x}}$, trying to model $p(\mathbf{x}|\tilde{\mathbf{x}})$,the conditional distribution of a data vector given the partial observations. The classifier C is a supervised learning model to predict the task outcome, $\hat{y}$, by taking $\hat{\mathbf{x}}$ from G, which obtains the conditional distribution $P(y|\mathbf{x}) = P(y|\hat{\mathbf{x}})$. We define the estimated outcome $\hat{y}$ by

$$\hat{y} = C(\hat{\mathbf{x}}).$$

Then G and C are jointly trained to obtain the distribution $p(y, \mathbf{x}|\tilde{\mathbf{x}})$, making G label-aware during imputation, which is ignored by GAIN.

Discriminator

In our architecture, the discriminator D serves as an adversarial character to train G by receiving the predicted label information from C. We input $\hat{\mathbf{x}}$, $\hat{y}$ and $\mathbf{h}$ into D to obtain the probability that each component in $\hat{\mathbf{x}}$ is observed. Here, $\hat{\mathbf{x}}$ and $\hat{y}$ jointly provide information to enhance D by learning the relationship between data and task outcomes, which can further strengthen G and C. We define the estimated mask variable,

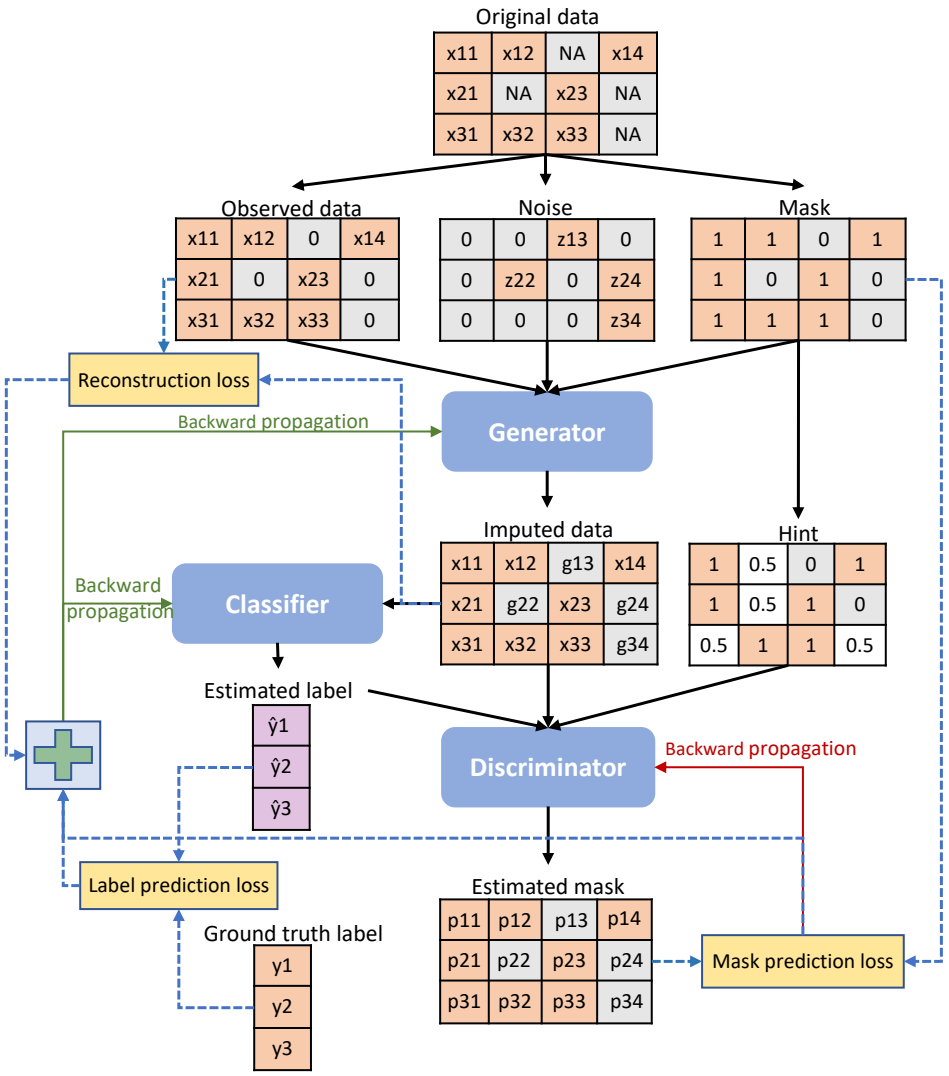


Figure 5.2: The overall architecture of Classifier-GAIN with three samples. Each row in the matrices and vectors corresponds to a sample. The input of our network is the original data vector with missing components. The generator takes partial observation data, a noise matrix filling missing components and mask, and outputs imputed data, which is then fed into the classifier to obtain the estimated labels. The imputed data and the estimated labels along with a hint matrix are fed into the discriminator to estimate the mask. Dashed lines represent information flows for loss calculation. Green solid lines represent backward propagation for the generator and the classifier training, and red solid line represents the backward propagation for the discriminator training.

$\hat{\mathbf{m}} \in [0,1]^d$, by

$$\hat{\mathbf{m}} = D(\hat{\mathbf{x}}, \hat{y}, \mathbf{h}),$$

with the j-th item in $\hat{\mathbf{m}}$ corresponding to the probability that the j-th item in $\hat{\mathbf{x}}$ is not NA in $\tilde{\mathbf{x}}$.

5.3.2 Classifier-GAIN Training

G, C and D are trained as a min-max game by

$$\min_{G,C} \max_{D} V(G,C,D) = \mathbb{E}_{\ddot{\mathbf{x}},\mathbf{m},\mathbf{h}} \Bigg[\mathbf{m} \log \Big(D\big(G(\ddot{\mathbf{x}}), C(G(\ddot{\mathbf{x}})), \mathbf{h}\big) \Big) + (\mathbf{1} - \mathbf{m}) \log \Big(\mathbf{1} - D\big(G(\ddot{\mathbf{x}}), C(G(\ddot{\mathbf{x}})), \mathbf{h}\big) \Big) \Bigg], \quad (5.1)$$

where log is an element-wise logarithm. Specifically, We train G and C together to minimize the probability of D identifying $\mathbf{m}$, maximize the probability of correctly predicting y and minimize the reconstruction loss of observed components. We train D to maximize the probability of correctly predicting $\mathbf{m}$.

On each iteration, G and C are updated k times with objective function, $\mathcal{L}_{C\&G}$, which is a weighted sum of three losses

$$\mathcal{L}_{C\&G} = \mathcal{L}_G(\mathbf{m}, \hat{\mathbf{m}}) + \alpha \mathcal{L}_R(\mathbf{x}, \hat{\mathbf{x}}) + \beta \mathcal{L}_C(y, \hat{y}), \quad (5.2)$$

where α and β are hyper-parameters.

The first loss, $\mathcal{L}_G$, is an adversarial loss, which applies to missing components ($m_j = 0$)

by

$$\mathcal{L}_G(\mathbf{m}, \hat{\mathbf{m}}) = -\sum_{j=1}^{d} \Big[(1 - m_j) \log(\hat{m}_j)\Big]. \tag{5.3}$$

The second loss, $\mathcal{L}_R$, is a reconstruction loss, which applies to observed components ($m_j = 1$) by

$$\mathcal{L}_R(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{j=1}^{d} m_j L_R(x_j, \hat{x}_j), \tag{5.4}$$

where

$$L_R(x_j, \hat{x}_j) = \begin{cases} (x_j - \hat{x}_j)^2, & \text{for numerical variables,} \\ -x_j \log(\hat{x}_j), & \text{for binary variables.} \end{cases}$$

The third loss, $\mathcal{L}_C$, is a binary cross entropy loss for task prediction given by

$$\mathcal{L}_C(y, \hat{y}) = -\big[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\big]. \tag{5.5}$$

Note that as G and C are updated together via Eq. 5.2, C 's performance will influence G's parameters to guide the missing component imputation.

D updates once at each iteration with objective function

$$\mathcal{L}_D(\mathbf{m}, \hat{\mathbf{m}}) = -\sum_{j=1}^{d} \Big[m_j \log(\hat{m}_j) + (1 - m_j)(1 - \hat{m}_j)\Big]. \tag{5.6}$$

The detailed training process is shown in Algorithm 1.

Algorithm 1 Minibatch Classifier-GAIN training

repeat
  Generator and Classifier
  Sample a batch of n binary vector $\{\mathbf{b}_i\}_{i=1}^{n} \sim \text{Bern}(p)^d$
  for k steps **do**
    Sample a batch of n noises $\{\mathbf{z}_i\}_{i=1}^{n} \sim U(0,1]^d$
    for $i \leftarrow 1$ to n **do**
      $\mathbf{h}_i \leftarrow \mathbf{m}_i \odot \mathbf{b}_i + 0.5(\mathbf{1} - \mathbf{b}_i)$
      $\ddot{\mathbf{x}}_i \leftarrow \mathbf{m}_i \odot \tilde{\mathbf{x}}_i + (\mathbf{1} - \mathbf{m}_i) \odot \mathbf{z}_i$
      Imputed data $\mathbf{g}_i \leftarrow G(\ddot{\mathbf{x}}_i)$
      $\hat{\mathbf{x}}_i \leftarrow \mathbf{m}_i \odot \tilde{\mathbf{x}}_i + (\mathbf{1} - \mathbf{m}_i) \odot \mathbf{g}_i$
      Obtain $\hat{y}_i \leftarrow C(\hat{\mathbf{x}}_i)$
      Obtain $\hat{\mathbf{m}}_i \leftarrow D(\hat{\mathbf{x}}_i, \hat{y}_i, \mathbf{h}_i)$
    end for
    Update generator G and classifier C together via stochastic gradient descent (SGD):
    $\bigtriangledown_G \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_G(\mathbf{m}_i, \hat{\mathbf{m}}_i) + \alpha \mathcal{L}_R(\mathbf{x}_i, \hat{\mathbf{x}}_i) + \beta \mathcal{L}_C(y_i, \hat{y}_i)$
  end for

  Discriminator
  Sample a batch of n binary vector $\{\mathbf{b}_i\}_{i=1}^{n} \sim \text{Bern}(p)^d$
  for $i \leftarrow 1$ to n **do**
    $\mathbf{h}_i \leftarrow \mathbf{m}_i \odot \mathbf{b}_i + 0.5(\mathbf{1} - \mathbf{b}_i)$
    Obtain $\hat{\mathbf{m}}_i \leftarrow D(\hat{\mathbf{x}}_i, \hat{y}_i, \mathbf{h}_i)$
  end for
  Update discriminator with fixed G and C via SGD
  $\bigtriangledown_D \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_D(\mathbf{m}_i, \hat{\mathbf{m}}_i)$
until Classifier-GAIN converges

5.4 Experiments

In this section, we conduct experiments on two datasets: the PhysioNet sepsis synthetic dataset and the UCSF real-world EHR dataset, introduced in Section 5.4.1, to evaluate Classifier-GAIN's performance. Particularly, we investigate

1. Does the classifier-guided imputation help the downstream MOF prediction?
2. How does the proposed algorithm perform across different missing ratio scenarios?
3. Are imputed values of Classifier-GAIN reasonable compared to values from other imputation methods in the real-world missing?

We explain the experimental settings in Section 5.4.2. The performance comparisons of Classifier-GAIN against other imputation algorithms for MOF prediction are shown in Section 5.4.3, followed by the visualizations of the imputed missing values of the UCSF MOF dataset in Section 5.4.4.

5.4.1 Dataset

The brief statistics of our datasets are shown in Table 5.2.

Table 5.2: Statistics of Datasets

Dataset	#Feature	#Patient
PhysioNet sepsis synthetic dataset	40	10,587
UCSF MOF real-world dataset	29	2,160

For the PhysioNet sepsis dataset, we randomly select 80% of the instances as the training set, 10% as the development set, and 10% as the testing set. For the UCSF MOF dataset, we perform a 5-fold cross validation, considering the dataset's size and models' training time. The detailed description of dataset as follows:

PhysioNet Sepsis Synthetic Dataset

Sepsis is a severe critical illness syndrome that can result in MOF [94]. Since MOF is the fatal end of sepsis progression[108], early detection of sepsis and antibiotic prescription are critical for improving MOF patient outcomes. We built a synthetic dataset based on the *physiological data* [109] provided by PhysioNet, sourced from ICU patients. Each patient contains 40 hourly measurements in three categories (vital signs, laboratory values and demographics) and the sepsis outcome in each hour. To obtain the sepsis outcome in the early stages, we focus on the first 6 hours' records of each feature. We take the first-appearance measurement of each feature in the first six hours after admission. If a value of a feature was not recorded in the first six hours, we assume that value was missing. We exclude the patients whose features were entirely missing in the first six hours. We label a patient with sepsis as 1, otherwise as 0. To obtain a completed dataset for further experiments, we apply SMOTE [110] to balance the data, and KNN to impute the original missing components. After data preprocessing, we obtained 10,587 patients, among which 5,808 patients are with sepsis and 4,779 without sepsis.

UCSF MOF Real-world Dataset

Our UCSF MOF dataset, collected from the UCSF/San Francisco General Hospital and Trauma Center, contains 2,190 patients admitted to a Level I trauma center. Both demographic information, such as gender, age, BMI (body mass index), and injury measurements, e.g. injury severity score (ISS), traumatic brain injury, and Glasgow Coma Scale (GCS), were measured at the admission time of each patient. Laboratory results (D-Dimer, creatinine, white blood cell etc.) and physical vital signs (for example heart rate, respiratory rate, systolic blood pressure etc.) were recorded at different hours. Unique ICU treatments such as blood transfusion units, fresh frozen plasma transfusion

Table 5.3: Rates of missing data in the UCSF dataset

Missing rate	Feature
41.9%	D-Dimer
41.6%	Factor VII (blood test)
17.4%	BMI
$10\% \geq \& > 5\%$	Factor VII treatment, PTT, Respiratory, SBP
$5\% \geq \& > 0\%$	HR, numribfxs, GCS, Vasopressor, Bun, Serumco2, PLTs, Crystalloids, Crystalloids, WBC, HGB, HCT, AIS scores, FFP_units, Blood_units, age, iss, Thromboembolic complication, Heparin_gtt
0%	Gender

and crystalloids for fluid resuscitation were slotted into time intervals such as 0 to 24 hours. Medical treatments (vasopressor, Heparin and Factor VII et al.) were reported in daily after admission.

To analyze the MOF states associated with patients' early-stage status, we select either the first day or the initial hour records manually. We extract features with importance score higher than 2% using forests of trees in Scikit-learn [111], and remove the patients whose data were utterly missing in the early stage or whose MOF outcome was not recorded. After data preprocessing and removal of abnormal values, we are left with 2,160 patients and 29 measurements. Selected features are categorized by types. Two blood test features, D-Dimer and Factor VII, had a missing rate higher than 40%. The body mass index (BMI) missed 17.4%. Factor VII treatment, partial thromboplastin time (PTT), respiratory rate and systolic blood pressure were missing at rates between 5% and 10%. The remainder of the features were missing at rates less than 5%. The rate of missing data for each feature is listed in Table 5.3, ordered from high to low. Missing values account for 6.42% among all observations and the labeling ratio between No MOF (class 0) and MOF (class 1) is $11:1$ in the dataset.

5.4.2 Experimental Settings

Evaluation Metrics

We measure the performance of Classifier-GAIN and baselines by both macro F1-score and area under the ROC curve (AUC-ROC).

Macro F1-score is defined as the mean of class-wise F1 scores:

$$Precision_l = \frac{TP_l}{TP_l + FP_l},$$

$$Recall_l = \frac{TP_l}{TP_l + FN_l}$$

$$F1_l = \frac{2 \times Precision_l \times Recall_l}{Precision_l + Recall_l}$$

$$Macro\ F1\text{-}Score = \frac{1}{L} \sum_{l=1}^{L} \mathrm{F1}_l,$$

where l is the class index and L is the number of classes. TN_l, TP_l, FP_l and FN_l denote the true positive, true negative, false positive and false negative rate, respectively for class l, and $\mathrm{F1}_l$ is the F1-score for the class. The macro F1-score metric is commonly applied to evaluate binary, multi-class and multi-label classification tasks [112]. It assigns equal importance to every class. It is low for models that perform well on the common classes, while performing poorly on the rare classes. AUC-ROC is also commonly used for MOF prediction [113, 114]. AUC-ROC assesses the overall preference of a classifier by summarizing over all possible classification thresholds. In binary classification problems, the higher the AUC-ROC, the better the model's performance in identifying the two classes.

Table 5.4: Hidden layer setting of different modules in UCSF and Sepsis datasets.

Dataset	Network	Hidden layer 1	Hidden layer 2	Dropout rate
UCSF	Classifier	32	16	0.1
	Generator	64	32	0.1
	Discriminator	64	32	0.1
Sepsis	Classifier	128	64	0.1
	Generator	64	32	0.1
	Discriminator	64	32	0.1

Missing Rate	macro F1-score			
	Classifier-GAIN	Simple imputation	MICE	GAIN
0%	Upper bound : 0.848 ± 0.001			
20%	**0.832 ± 0.003**	0.678 ± 0.021	0.686 ± 0.015	0.695 ± 0.018
25%	**0.829 ± 0.002**	0.683 ± 0.033	0.669 ± 0.015	0.659 ± 0.016
30%	**0.808 ± 0.008**	0.674 ± 0.021	0.673 ± 0.011	0.671 ± 0.020
35%	**0.797 ± 0.009**	0.669 ± 0.016	0.689 ± 0.015	0.662 ± 0.033
40%	**0.788 ± 0.007**	0.646 ± 0.026	0.669 ± 0.023	0.648 ± 0.035
45%	**0.772 ± 0.005**	0.656 ± 0.040	0.664 ± 0.019	0.654 ± 0.032
50%	**0.777 ± 0.008**	0.651 ± 0.016	0.637 ± 0.010	0.655 ± 0.021

Table 5.5: Model performance (F1-score) on PhysioNet sepsis dataset in different missing ratio settings.

Model Configurations

We compare Classifier-GAIN with both classical and state-of-art neural baselines for MOF prediction as follows:

1. **Simple imputation**: It imputes missing components by mean imputation and most frequent imputation for continuous and categorical variables, respectively.

2. **MICE** [101]: It is a multiple imputation method, accounting for the statistical uncertainty in the imputations.

3. **GAIN** [102]: It is a deep learning adversarial imputation framework, which we explained in Section 5.2.2 in detail.

Missing Rate	AUC-ROC			
	Classifier-GAIN	Simple imputation	MICE	GAIN
0%	Upper bound : 0.909 ± 0.003			
20%	**0.883 ± 0.006**	0.831 ± 0.009	0.843 ± 0.015	0.834 ± 0.014
25%	**0.871 ± 0.004**	0.818 ± 0.012	0.819 ± 0.006	0.813 ± 0.006
30%	**0.864 ± 0.006**	0.819 ± 0.010	0.810 ± 0.009	0.818 ± 0.010
35%	**0.848 ± 0.004**	0.812 ± 0.011	0.803 ± 0.010	0.817 ± 0.008
40%	**0.839 ± 0.006**	0.803 ± 0.004	0.800 ± 0.011	0.812 ± 0.009
45%	**0.811 ± 0.004**	0.795 ± 0.009	0.792 ± 0.009	0.790 ± 0.007
50%	**0.821 ± 0.007**	0.805 ± 0.004	0.747 ± 0.006	0.807 ± 0.006

Table 5.6: Model performance (AUC-ROC) on PhysioNet sepsis dataset in different missing ratio settings.

Each of methods (1), (2) and (3) is separated into two steps. First we impute missing components by the corresponding method. Then we utilize a binary classifier to predict the subjects' outcomes by taking imputed data. For our proposed Classifier-GAIN, we take the partially observed data as input, and output both an imputed data and classification outcomes.

In order to make the performance comparison as fair as possible, we assign the same structure and hidden size for all classifiers. GAIN has exactly the same structure in the generator and the same number of hidden layers in the discriminator as Classifier-GAIN. All of the networks are designed as multi-layer perceptrons with two hidden layers. We use batch normalization to normalize the input layer by re-centering and re-scaling. Relu activation function and dropout are applied after each hidden linear layer. All of the neural networks utilize Sigmoid activation at the last step for outputs. The hidden layer settings in all of our experiments are listed in Table 5.4.

We implement our model and its variants using PyTorch [115], and use a GeForce GTX TITAN X 12 GB GPU for training, validation as well as testing. All of the neural networks are trained by using the Adam optimizer [116], whose learning rates are selected

by grid search from 0.0005 to 0.002[1].

Table 5.7: Hyperparameters for models on PhysioNet sepsis dataset and UCSF MOF dataset

Model	parameter	PhysioNet dataset	UCSF dataset
Simple imputation &Classifier	epochs	30	
	batch size	128	16
	leaning rate	0.0005-0.002	
	classifier's weight decay	5e-4	
MICE & Classifier	epochs	30	
	batch size	128	16
	leaning rate	0.0005-0.002	
	generator's weight decay	5e-4	
GAIN & Classifier	epochs for GAIN	20	50
	batch size for GAIN	128	16
	generator's leaning rate discriminator 's leaning rate	0.0005-0.002	
	generator's weight decay discriminator s weight decay	5e-4	
	p_hint	0.9	
	alpha	1	
	weight decay	5e-4	
	epochs for classifier	30	
	batch size for classifier	128	16
	learning rate for classifier	0.0005-0.002	
	classifier s weight decay	5e-4	
Classifier-GAIN	epochs	50	
	batch size	128	16
	generator's leaning rate discriminator 's leaning rate classifier 's leaning rate	0.0005-0.002	
	p_hint	0.5	0.9
	alpha	5-20	
	generator's weight decay discriminator's weight decay classifier's weight decay	5e-4	

[1]Other hyper-parameters are detailed in Table 5.7

5.4.3 Performance Comparison

We conduct each experiment by running 5 times with different random initializations and show the results as the format "mean standard deviation" to answer Q1 and Q2. For readers' convenience, we make the best performance bold in each of the performance tables in this section.

Synthetic Data

To evaluate Classifier-GAINs capability to capture the relationship between clinical records (vital signs and laboratory values) and label outcome for downstream prediction, we randomly remove 20%, 25%, 30%, 35%, 40%, 45% and 50% of all components from clinical records, to simulate missingness resulting from the urgency of the clinical situation. We demonstrate the effectiveness of Classifier-GAIN against other baselines in Table 5.6 and Table 5.5. To understand the performance gap between different missing scenarios and the completed data, we train a binary classifier on the completed dataset, which we refer to as the upper bound. As shown in Table 5.6 and Table 5.5, Classifier-GAIN consistently outperforms the simple imputation, MICE and GAIN across the entire range of missing rates, for both evaluation metrics. Especially, when the missing rate is 25%, Classifier-GAIN improves 0.052 and 0.146 in AUC-ROC and macro F1-score, respectively, comparing with the best baselines.

To quantitatively evaluate the performance of Classifier-GAIN, we derive two additional metrics: (1) the relative improvement rate (RIR),

$$\frac{\text{Classifier-GAIN - best\_baseline}}{\text{best\_baseline}}, \tag{5.7}$$

to demonstrate how much Classifier-GAIN improves compared to the best baseline, and

(2) the relative gap reduction rate (RGRR),

$$\frac{\text{Classifier-GAIN - best\_baseline}}{\text{Upper bound - best\_baseline}}, \tag{5.8}$$

to measure the capability of Classifier-GAIN reducing the performance gap to the upper bound.

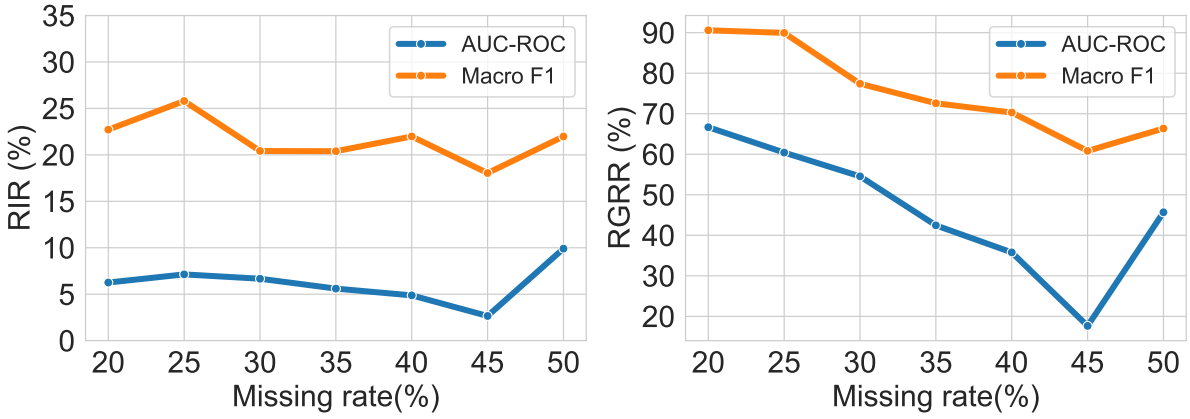
Missing Rate	macro F1-score			
	Classifier-GAIN	Simple imputation	MICE	GAIN
20%	**0.683 ± 0.004**	0.649 ± 0.016	0.652 ± 0.015	0.660 ± 0.009
25%	**0.679 ± 0.010**	0.643 ± 0.005	0.615 ± 0.017	0.646 ± 0.017
30%	**0.702 ± 0.007**	0.682 ± 0.013	0.616 ± 0.022	0.659 ± 0.010
35%	**0.677 ± 0.008**	0.645 ± 0.012	0.604 ± 0.021	0.650 ± 0.004
40%	**0.657 ± 0.014**	0.619 ± 0.008	0.581 ± 0.030	0.632 ± 0.012
45%	**0.657 ± 0.010**	0.625 ± 0.014	0.573 ± 0.027	0.617 ± 0.017
50%	**0.650 ± 0.002**	0.594 ± 0.016	0.573 ± 0.033	0.600 ± 0.017

Table 5.8: Model performance (macro F1-score) on UCSF MOF dataset with different additionally simulated missing ratios.

Missing Rate	AUC-ROC			
	Classifier-GAIN	Simple imputation	MICE	GAIN
20%	0.889 ± 0.005	0.882 ± 0.006	**0.891 ± 0.002**	0.887 ± 0.005
25%	**0.885 ± 0.004**	0.874 ± 0.005	0.884 ± 0.002	0.878 ± 0.003
30%	**0.895 ± 0.005**	0.890 ± 0.002	0.877 ± 0.002	0.887 ± 0.004
35%	**0.885 ± 0.004**	0.877 ± 0.004	0.870 ± 0.006	0.874 ± 0.005
40%	**0.878 ± 0.005**	0.868 ± 0.004	0.853 ± 0.002	0.865 ± 0.005
45%	**0.868 ± 0.006**	0.857 ± 0.007	0.846 ± 0.002	0.850 ± 0.007
50%	0.839 ± 0.008	0.837 ± 0.005	**0.842 ± 0.001**	0.835 ± 0.014

Table 5.9: Model performance (AUC-ROC) on UCSF MOF dataset with different additionally simulated missing ratios.

The relative improvement rates calculated by Eq. 5.7 across different settings are shown in Figure 5.3 (a). For both macro F1-score and AUC-ROC, Classifier-GAIN consistently achieves a high relative improvement rate, with 21.62% and 6.16% on average



(a) The relative improvement rate (%)

(b) The relative gap reduction rate (%)

Figure 5.3: The relative improvement rate (left) and the relative gap reduction rate (right) of Classifier-GAIN on AUC-ROC and macro F1-score for PhysioNet sepsis dataset across different missing ratio scenarios.

across different scenarios, respectively. Especially, the relative improvement rate of macro F1-score is 25.80% when the missing rate is 25%, and the relative improvement rate of AUC-ROC is 9.91% when the missing rate is 50%.

Figure 5.3 (b) shows the relative gap reduction rate of Classifier-GAIN with different missing ratio settings. Classifier-GAIN significantly reduces the performance gap to the upper bound, with a 75.43% relative reduction rate for macro F1-score on average compared to best baselines, making the prediction less susceptible to missingness across different scenarios. Especially when missing rates are 20% and 25% (relatively low), the relative gap reduction rates are as high as 90.58% and 89.94%, which significantly narrows the performance gap caused by missing components. Even when missing rates are 40% and 50% (very high), the relative gap reduction rates remain 60.82% and 66.35%, which further validates Classifier-GAIN's applicability in different missing scenarios.

Real-world Data

We further evaluate our model on the UCSF MOF real-world dataset, including early-stage clinical records for MOF prediction. In addition to the high missing ratio in bio-marker measurements, there is a serious label imbalance issue in this dataset, which is common in real-world clinical data. We evaluate the performance of Classifier-GAIN on the UCSF MOF dataset in the following settings: (1) imputing the original missing components and predicting MOF outcome; (2) adding additional random masks with 20%, 25%, 30%, 35%, 40%, 45% and 50% missing rates to simulate more serious missing situations in real-world data.

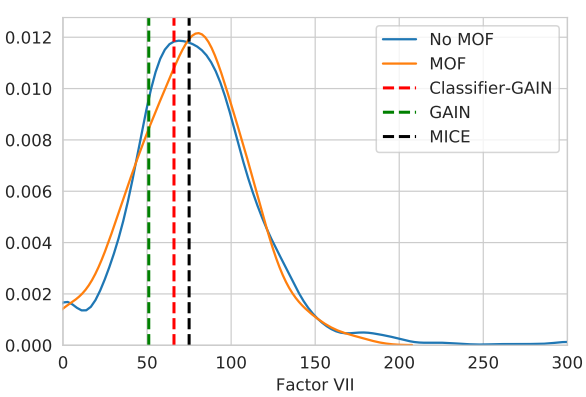
Table 6 reports the AUC-ROC and macro F1-score to evaluate Classifier-GAIN's prediction performance against other methods, on the UCSF MOF dataset with original missing components (The missing ratio of features is 6.42% among all patients on average.). Classifier-GAIN yields the best prediction performance as measured by both macro F1-score and AUC-ROC.

Algorithm	macro F1-score	AUC-ROC
Classifier-GAIN	**0.710 ± 0.010**	**0.906 ± 0.005**
Simple imputation	0.689 ± 0.010	0.903 ± 0.003
MICE	0.682 ± 0.008	0.900 ± 0.004
GAIN	0.682 ± 0.009	0.902 ± 0.003

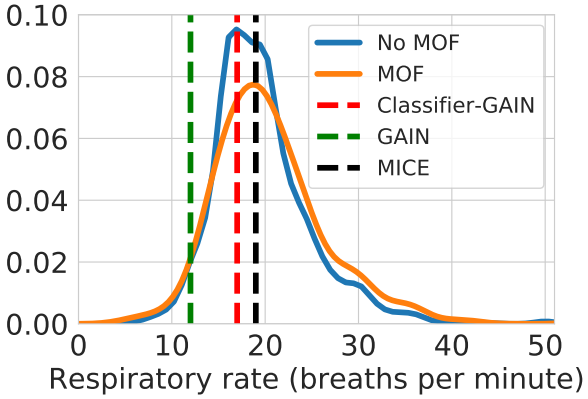
Table 6: Model performance on UCSF MOF dataset with original missing components.

For more missing ratios in our simulated setting, the corresponding macro F1-score and AUC-ROC are shown in Table 5.8 and Table 5.9. For macro F1-score, Classifier-GAIN consistently outperforms the simple imputation, MICE and GAIN across the entire range of missing rates. For AUC-ROC, Classifier-GAIN outperforms other post-

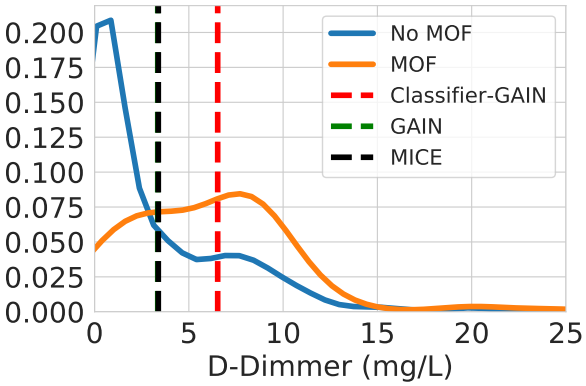
imputation predictions in 25%, 30%, 35%, 40%, 45% missing scenarios, and achieves comparable performance with MICE in 20% and 50% missing conditions.



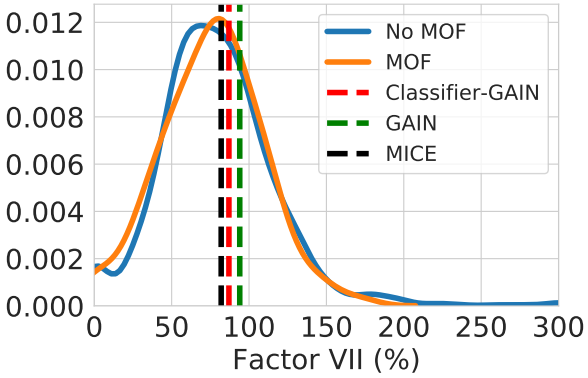
(a) Factor VII imputation for no MOF



(b) Respiratory rate imputation for no MOF



(c) D-Dimmer imputation for MOF



(d) Factor VII imputation for no MOF

Figure 5.4: Density plots of features and imputed values. The blue and orange curves are density curves of observed data points. The blue curve represents MOF = 0, and the orange represents MOF = 1. The dashed vertical lines are the imputation results of three different imputation methods.

5.4.4 Imputation Results

To answer Q3, we compare the imputation results for the original missingness in the UCSF MOF dataset of Classifier-GAIN and other baselines in Figure 5.4, which plots the univariate distributions of selected features for MOF (MOF=1) and no MOF (MOF=0)

patients, respectively. The blue and orange curves are density curves of observed components of features that need imputation. The blue curves represent the density curves of MOF = 0, and the orange ones represent MOF = 1. Dashed vertical lines are the imputed results of three different imputation methods: red is Classifier-GAIN, green is GAIN, and black is MICE imputation.

We select three features: D-Dimer, Factor VII (blood test)[2] and respiratory rate, for the imputation study. Both D-Dimer and Factor VII have more than 40% missing in the original dataset, and the missing rate of respiratory rate is 7.31%. D-Dimer is an indicator of patients who may develop organ failure in the further course of acute pancreatitis[117]. Factor VII and respiratory rate are highly related to pulmonary failure [118, 119]. In the UCSF MOF dataset, the feature values available with maximum density of D-Dimer, Factor VII and respiratory rate for patients who did not develop MOF are 0.86 mg/L, 68.82% and 16.92 breaths per minute, respectively. For MOF patients, the feature values available with maximum density are 7.71 mg/L, 80.69% and 18.75 breaths per minute, respectively.

For panels (a) and (b), Classifier-GAIN predicts correctly for a patient without MOF, while the other classifiers, whose input data are imputed by MICE or GAIN, predict incorrectly. The imputed values of Classifier-GAIN for Factor VII and respiratory rate are relatively closer to the feature values with maximum density for no-MOF's in both cases. Panel (c) shows a MOF patient who's data was missing the D-Dimer record. MICE and GAIN have very similar imputed value which are 3.38 mg/L and 3.35 mg/L, Classifier-GAIN imputes the D-Dimer as 6.53,mg/L which follows the trend of MOF patients in this dataset. Panel (d) shows a situation that all of the classifiers predict a no-MOF case incorrectly. In this case, all three methods impute the Factor VII closer to the feature value with maximum density for MOFs.

[2]In the remainder of this subsection, we use Factor VII to represent Factor VII (blood test).

5.5 Related Work

Generative Adversarial Networks (GAN)

GAN, introduced in [103], is a game-theoretic framework for estimating the implicit distribution of data via an adversarial process. CGAN conditions the GAN framework on class labels to direct the data generation process [104]. AC-GAN further improves generation performance by modifying the discriminator to contain an auxiliary decoder network [105]. Semi-supervised GAN [106] performs GAN in a semi-supervised context to make the discriminator output either data validation or class labels. Triple-GAN facilitates the convergence of both the generator and the discriminator by introducing the "third player" – classifier [107].

Researchers have also applied GANs on missing value imputation. In GAIN [102], the generator imputes the missing components while the discriminator takes a completed vector and attempts to determine which components were actually observed and which were imputed with some additional information in the form of a hint vector. MISGAN learns a complete data generator along with a mask generator that models the missing data distribution and an adversarially trained imputer [120]. However, those existing methods ignore the connection between observations and classification information, which can make use for learning label-aware imputation during training and help to improve downstream task prediction during inference.

Multiple Organ Failure (MOF)

MOF is a major threat to the survival of patients with sepsis and is becoming the most common cause of death for surgical ICU patients [121]. According to a recent study of ICU trauma patients, almost half of them developed MOF, and MOF increased the overall risk of death 6.0 times compared to the patients without MOF [122]. Sepsis

is viewed as an immune storm that leads to MOF and death, which still is a leading cause of death in critically ill patients, though modern antibiotics and new resuscitation therapies have been used [123]. The Acute Physiology and Chronic Health Evaluation (APACHE) score and the Ranson score are widely used for seriously ill patients, but their empirical utilization for predicting the risk of MOF at an early stage is limited by cumbersomeness and needs to record some indexes dynamically [124]. Therefore, a prognostic tool that can reliably predict MOF in the early phase is essential for improving patient outcomes.Different machine learning methods have been applied to MOF and sepsis predictions. [124] performed support vector machine, logistic regression analysis and artificial neural networks models to predict MOF in moderately severe and severely acute pancreatitis. [125] applied CNN to predict severe sepsis in critically ill children, and found that the performance of the CNN approach continuously improved when more training data was available. However, this study could utilize only 493 out of 5053 patients for training and testing, as other observations have missing features. In this work, we have chosen to base our MOF prediction on highly-related vital signs at the initial stage, to predict outcomes with classifier-guided imputation, in order to handle the data sparsity problem.

Missing Data Mechanisms

Depending on the underlying reasons, missingness is divided into three categories: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). MCAR refers to a situation in which the occurrence for a data point to be missing is entirely random. MAR assumes that the missingness does not have any relationship with the missing data but may depend on the observed data. MNAR indicates that the missing elements are related to the reasons for which the data is missing. In general, we assume that the EHR data is MAR data because, in most EHR

instances, those collected features would be expected to explain some, but not all, of the variation among patients who's data have missing values[126].

Various methodologies are available to address missing data. The most common and straightforward imputation methodologies are deletion based methods, such as listwise deletion [127] and pairwise deletion [128], which exclude all measurements with missing elements. These methods are easy and practical if the missingness rate is low (e.g., less than about 5%) [129], and the missing data is MCAR. However, because EHR data is high dimensional and the missing elements usually are not MCAR, deletion-based methods may not be useful in predictive modelling and regression analyses in EHR data. Single imputation algorithms only impute missing components in one iteration, which can utilize some unique numbers (e.g., 0) or statistical characteristics, such as mean value imputation [100], median imputation [130] and most common value imputation [131]. MICE [101] is one of the most commonly used multiple imputation algorithms, applying multiple regression models iteratively to impute missing values for different types of variables [132]. In this work, we have explored the algorithm with missing components in EHRs datasets to resolve the real-world MOF prediction task.

5.6 Conclusion

In this chapter we present Classifier-GAIN, an end-to-end deep learning framework to improve performance of MOF prediction on datasets with a wide range of missingness ratios. In contrast to most of the label-aware GANs, which focus on improving the generator outputs, we design a three-player adversarial imputation network, aiming to optimize the downstream prediction while imputing missing values. Classifier-GAIN uses a classifier to provide label supervision signals to the generator in training, and the trained generator to improve the classifiers downstream prediction performance in

inference. Extensive experimental results on both a synthetic sepsis dataset and a real world MOF dataset demonstrate the usefulness of this framework. Although we only demonstrate the effeteness of Classifier-GAIN in MOF prediction tasks, its applications in other domains are worth exploring, which we leave as further work.

Chapter 6

BERTSurv: BERT-Based Survival Models for Predicting Outcomes of Trauma Patients

6.1 Introduction

Trauma is the leading cause of death from age 1 to 44. More than 180,000 deaths from trauma occur each year in the United States [133]. Most trauma patients die or are discharged quickly after being admitted to the ICU. Care in the first few hours after admission is critical to patient outcome, yet this time period is more prone to medical decision errors in ICUs [98] than later periods. Therefore, early and accurate prediction for trauma patient outcomes is essential for ICU decision making.

Medical practitioners use survival models to predict the outcomes for trauma patients [134]. Survival analysis is a technique to model the distribution of the outcome time. The Cox model [135] is one of the most widely used survival models with linear proportional hazards. Faraggi-Simon's network [136] is an extension of the Cox

model to nonlinear proportional hazards using a neural network. DeepSurv [137] models interactions between a patients covariates and treatment effectiveness with a Cox proportional hazards deep neural network. However, these existing models deal only with well-structured measurements and do not incorporate information from unstructured clinical notes, which can offer significant insight into patients' conditions.

The transformer architecture has taken over sequence transduction tasks in natural language processing (NLP). Transformer is a sequence model that adopts a fully attention-based approach instead of traditional recurrent architectures. Based on Transformer, BERT [138] was proposed for language representation and achieved state-of-the-art performance on many NLP tasks. There has also been increasing interest in applying deep learning to end-to-end e-health data analysis [139]. Biobert [140] extends BERT to model biomedical language representation. Med-BERT [141] modifies BERT by leveraging domain specific hierarchical code embedding and layer representation to generate sequential relationships in the clinical domain. G-BERT [142] combines Graph Neural Networks (GNNs) and BERT for medical code representation and medication recommendation. Clinical BERT [143, 144] explores and pre-trains BERT using clinical notes. Clearly, there is an unmet need to include unstructured text information in deep learning survival models for patient outcome predictions.

In this chapter we propose BERTSurv, a deep learning survival framework for trauma patients which incorporates clinical notes and measurements for outcome prediction. BERTSurv allows for both mortality prediction and survival analysis by using BCE and PLL loss, respectively. Our experimental results indicate that BERTSurv can achieve an AUC-ROC of 0.86, which is an improvement of 3.6% over the baseline of MLP without notes on mortality prediction.

The key contributions of this chapter are:

1. We propose BERTSurv: a BERT-based deep learning framework to predict the risk

of death for trauma patients. To the best of our knowledge, this is the first work applying BERT on unstructured text data combined with measurements for survival analysis.

2. We evaluate BERTSurv on the trauma patients in MIMIC III. For mortality prediction, BERTSurv achieves an AUC-ROC of 0.86, which outperforms baseline of MLP without notes by 3.6%. For survival analysis, BERTSurv achieved a C-index of 0.7 on trauma patients, which outperforms a Cox model with a C-index of 0.68.

3. We extract patterns in clinical notes by performing attention mechanism visualization, which improves model interpretability by showing how the model assigns weights to different clinical input texts with respect to survival outcomes.

This chapter is organized as follows: Section 6.2 describes how we processed the MIMIC trauma dataset. We present BERTSurv in Section 6.3.1 and describe the background of BERT and survival analysis in Section 6.3.2 and Section 6.3.3. Evaluation and discussion are given in Sections 6.4 and 6.5, respectively.

6.2 Dataset

BERTSurv is applied to the data from trauma patients selected using the ICD-9 code from the publicly available MIMIC III dataset [145], which provides extensive electronic medical records for ICU admissions at the Beth Israel Deaconess Medical Center between 2001 and 2012. The measurements, clinical notes, expire flag (0 for discharge and 1 for death), and death/discharge time for each patient were used to train and test BERTSurv. The patient data were aggregated over the first 4 hours to obtain the initial state of each individual admission. We took the average for each of the measurements taken during this time period, and concatenated all of the clinical notes together. Considering the missing value issue and redundancy in MIMIC III, we selected 21 common features as our representative set: blood pressure, temperature, respiratory rate, arterial

PaO2, hematocrit, WBC, creatinine, chloride, lactic acid, BUN, sodium (Na), glucose, PaCO2, pH, GCS, heart rate, FiO2, potassium, calcium, PTT and INR. Our feature set overlaps 65% of the measurements required by APACHE III [146]. We also extracted 4 demographic predictors: weight, gender, ethnicity and age.

As is common in medical data, MIMIC III contains many missing values in the measurements, and the notes are not well-formatted. Thus, data preprocessing is very important to predict outcomes. To deal with the missing data issue, we first removed patients who have a missing value proportion greater than 0.4 and then applied MICE [101] data imputation for the remainder of the missing values. For the clinical notes, we removed formatting, punctuation, non-punctuation symbols and stop words. In addition to the most commonly used English stop words, our stop word dictionary includes a few specific clinical and trauma related stop words: *doctor, nurse and measurement*, etc. Following this preprocessing, our trauma dataset includes 1860 ICU patients, with 21 endogenous measurements, 4 exogenous measurements and notes. The sample class ratio between class 0 (discharge) and class 1 (death) is 1206 : 654.

6.3 Methods

In this section we first describe the framework of BERTSurv. Then we introduce some basics of BERT and survival analysis, which are the key components of BERTSurv.

6.3.1 BERTSurv

Our model architecture, shown in Fig 6.1, consists of BERT embedding of clinical notes concatenated with measurements followed by feed forward layers. The output for BERTSurv is a single node $h_\theta(x_i)$ parameterized by the weights of the neural network θ, which estimates either the probability of mortality or the hazard risk. For mortality

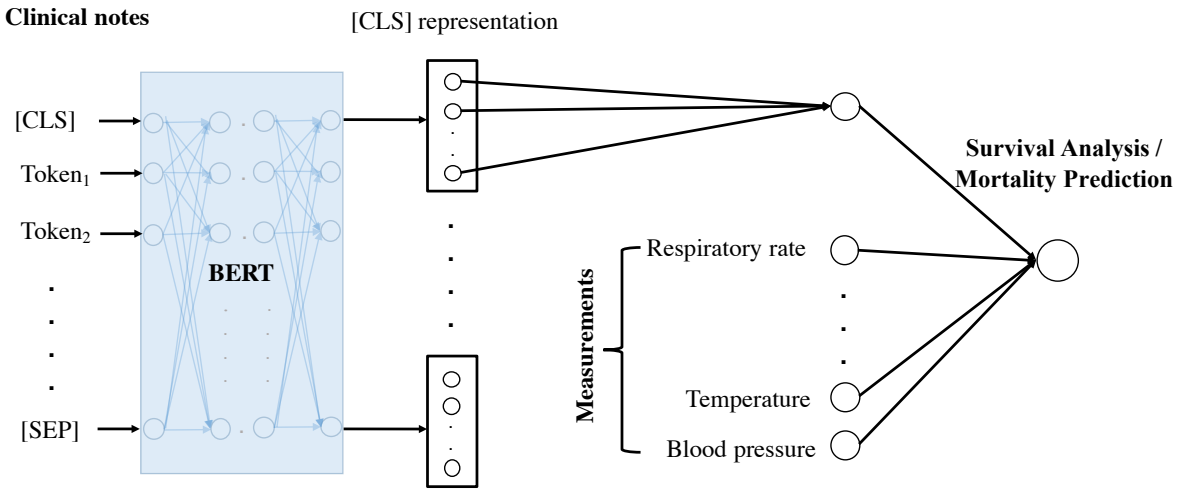


Figure 6.1: The framework of BERTSurv. [CLS] is a special symbol added in front of every clinical note sample, and [SEP] stands for a special separator token. BERTSurv consists of three main parts: BERT, measurements and output layer for mortality prediction or survival analysis. First, we input a set of diagnostics and nurse notes to BERT pretrained on masked language modeling and next sentence prediction. The [CLS] representation, is treated as the representation of the input notes. Then we concatenate the [CLS] representation and measurements as input and fine-tune BERTSurv for downstream survival analysis.

prediction, we apply BCE loss to predict outcomes of death or discharge:

$$\text{BCELoss} := \sum_{i=1}^{n} p(y_i) \log(h_\theta(x_i)), \tag{6.1}$$

where x_i and y_i represent inputs and outcomes for the ith patient, respectively.

To estimate θ in survival analysis, similar to the Faraggi-Simon network [136, 137], we minimize the PLL loss function, which is the average negative log partial likelihood:

$$\text{PLLLoss} := -\frac{1}{N_{D=1}} \sum_{i:D_i=1} (h_\theta(x_i) - \log \sum_{j \in R(T_i)} \exp(h_\theta(x_j))), \tag{6.2}$$

where $N_{D=1}$ is the number of patients with an observable death. The risk set $R_i = \{j : T_j \geq T_i\}$ is the set of those patients under risk at T_i.

We use batch normalization through normalization of the input layer by re-centering and re-scaling [26]. We apply rectified linear unit(ReLU) or scaled exponential linear units (SELU) as the activation function. For regularization, dropout [27] is implemented to avoid overfitting. Dropout prevents co-adaptation of hidden units by randomly dropping out a proportion of the hidden units during backpropagation. BCE/PLL loss is minimized with the Adam optimizer [29] for training.

BERTSurv is implemented in Pytorch [147]. We use a Dell 32GB NVIDIA Tesla M10 Graphics Card GPU (and significant CPU power and memory for pre-processing tasks) for training, validation and testing. The hyperparameters of the network include: BERT choice ($\text{BERT}_{\text{BASE}}$ or clinical BERT [143]), sequence length, batch size, learning rate, dropout rate, training epochs and activation function (ReLU or SELU).

6.3.2 BERT

A key component of BERTSurv is the BERT language representation model. BERT is a Transformer-based language representation model, which is designed to pre-train deep bidirectional representations from unlabeled text by jointly considering context from both directions (left and right). Using BERT, the input representation for each token in the clinical notes is the sum of the corresponding token embeddings, segmentation embeddings and position embeddings. WordPiece token embeddings [148] with a 30,000 token vocabulary are applied as input to BERT. The segment embeddings identify which sentence the token is associated with. The position embeddings of a token are a learned set of parameters corresponding to the tokens position in the input sequence. An attention function maps a query and a set of key-value pairs to an output. The attention function takes a set of queries Q, keys K, and values V as inputs and is computed on an input sequence using the embeddings associated with the input tokens. To construct Q, K and

V, every input embedding is multiplied by the learned sets of weights. The attention function is

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}}V), \tag{6.3}$$

where d_k is the dimensionality of Q and K. The dimension of V is d_v. A multi-head attention mechanism allows BERT to jointly deal with information from different representation subspaces at different positions with several (h) attention layers running in parallel:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O, \tag{6.4}$$

where $\text{head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$. Parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ are the learned linear projections from Q, K, V to d_k, d_k and d_v dimensions.

In BERTSurv, we use pretrained BERT of $\text{BERT}_{\text{BASE}}$ and clinical BERT [143] for clinical note embedding, and focus on fine-tuning for survival analysis.

6.3.3 Survival Analysis

Another key component of BERTSurv is survival analysis. Survival analysis [149, 150] is a statistical methodology for analyzing the expected duration until one or more events occur. The survival function $S(t)$, defined as $S(t) = P(T \geq t)$, gives the probability that the time to the event occurs later than a given time t. The cumulative distribution function (CDF) of the time to event gives the cumulative probability for a given t-value:

$$F(t) = P(T < t) = 1 - S(t). \tag{6.5}$$

The hazard function $h(t)$ models the probability that an event will occur in the time interval $[t, t + \Delta t)$ given that the event has not occurred before:

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t \mid T \geq t)}{\Delta t} = \frac{f(t)}{S(t)}, \tag{6.6}$$

where $f(t)$ is the probability density function (PDF) of the time to event. A greater hazard implies a greater probability of event occurrence. Note from Equ. 6.5 that $-f(t)$ is the derivative of $S(t)$. Thus Equ. 6.6 can be rewritten as

$$h(t) = -\frac{dS(t)}{dt} * \frac{1}{S(t)} = -\frac{d}{dt} \log(S(t)). \tag{6.7}$$

By solving Equ. 6.7 and introducing the boundary condition $S(0) = 1$ (the event can not occur before duration 0), the relationship between $S(t)$ and $h(t)$ is given by

$$S(t) = \exp\left(-\int_0^t h(s)\mathrm{d}s\right). \tag{6.8}$$

The Cox model [135] is a well-recognized survival model. It defines the hazard function given input data $h(t \mid \mathbf{y}, \boldsymbol{\eta})$ to be the product of a baseline function, which is a function of time, and a parametric function of the input data $\mathbf{y}$ and $\boldsymbol{\eta}$. $\mathbf{y}$ and $\boldsymbol{\eta}$ denote endogenous measurements and exogenous measurements, respectively. Using the assumption of a linear relationship between the log-risk function and the covariates, the Cox model has the form

$$h(t \mid \mathbf{y}, \boldsymbol{\eta}) = h_0(t) \exp(\boldsymbol{\tau}^T \mathbf{y} + \boldsymbol{\gamma}^T \boldsymbol{\eta}), \tag{6.9}$$

where $h_0(t)$ is the baseline hazard function, and $\boldsymbol{\tau}$ and $\boldsymbol{\gamma}$ are the vectors of weights for $\mathbf{y}$ and $\boldsymbol{\eta}$.

In BERTSurv, the log-risk function $h_\theta(\mathbf{x})$ is the output node from the neural network:

$$h(t \mid \mathbf{y}, \boldsymbol{\eta}) = h_0(t) \exp(h_\theta(\mathbf{x})), \tag{6.10}$$

where the input $\mathbf{x}$ includes $\mathbf{y}$, $\boldsymbol{\eta}$ and clinical notes. The likelihood function for the survival model is as follows:

$$p(T, \delta) = h(T)^\delta S(T). \tag{6.11}$$

When $\delta = 1$, it means that the event is observed at time T. When $\delta = 0$, the event has not occurred before T and it will be unknown after T. The time T when $\delta = 0$ is called the censoring time, which means the event is no longer observable.

The Cox partial likelihood is parameterized by $\boldsymbol{\tau}$ and $\boldsymbol{\gamma}$ and defined as

$$\mathrm{PL}(\boldsymbol{\tau}, \boldsymbol{\gamma}) = \prod_{i=1}^{n} \{ \frac{\exp(\boldsymbol{\tau}^T \mathbf{y} + \boldsymbol{\gamma}^T \boldsymbol{\eta})}{\sum_{j \in R_i} \exp(\boldsymbol{\tau}^T \mathbf{y} + \boldsymbol{\gamma}^T \boldsymbol{\eta})} \}^{\Delta_i}, \tag{6.12}$$

where $\Delta_i = I(T_i \leq C_i)$. C_i is the censoring time for the ith patient, and $I(*)$ is the indicator function.

We use the Breslow estimator [151] to estimate the cumulative baseline hazard $\widehat{H_0}(t) = \int_0^t \widehat{h_0}(u)du$:

$$\widehat{H_0}(t) = \sum_{i=1}^{n} \frac{I(T_i \leq t)\Delta_i}{\sum_{j \in R_i} \exp(\boldsymbol{\tau}^T \mathbf{y} + \boldsymbol{\gamma}^T \boldsymbol{\eta})}. \tag{6.13}$$

6.4 Experiments and Analysis

Throughout this section, we randomly pick 70% of the trauma data as training and the rest as testing. Considering the size of our dataset and the training time, we apply 5-fold cross-validation on the trauma training dataset and grid search for hyperparameter choice. Our hyperparameters are described in Table 6.1. Note that the sequence length and batch size choices are limited by GPU memory.

Table 6.1: Hyperparameters

Hyperparameters	Survival analysis	Mortality prediction
Batch size	24	16
Sequence length	512	512
Epoch	4	4
Dropout rate	0.1	0.1
Learning rate	1e-2	4e-2
BERT choice	clinical BERT	clinical BERT
Activation	SELU	ReLU

Using the clinical notes and measurements, we formulate the mortality prediction problem as a binary classification problem. Fig. 6.2 shows the averaged cross validation confusion matrix for mortality prediction in the trauma training dataset. The testing confusion matrix for mortality prediction is presented in Fig. 6.3. Dominant numbers on the diagonals of both confusion matrices indicate that BERTSurv achieves high accuracy for both of the outcomes (death/discharge). With BCE loss, we apply two baselines: MLP without notes and the TF-IDF mortality model. In MLP without notes, we consider only the measurements and build a MLP with 3 feed-forward layers for mortality outcomes. In the TF-IDF mortality model, we represent notes with TF-IDF vectors and build a support vector machine (SVM) on TF-IDF vectors combined with measurements for mortality prediction. We use AUC-ROC as our performance metric for mortality prediction, as it is commonly used in survival prediction [152, 153]. AUC-ROC represents the probability that a classifier ranks the risk of a randomly chosen death patient (class 1) higher than a randomly chosen discharged patient (class 0). As is shown in Fig. 6.5, BERTSurv achieved an AUC-ROC of 0.86, which outperforms MLP without notes by 3.6%. BERTSurv also outperforms MLP without notes, with 5-fold cross validation as shown for our trauma training dataset in Fig. 6.4.

To evaluate the model's predictive performance with PLL loss on survival analysis,

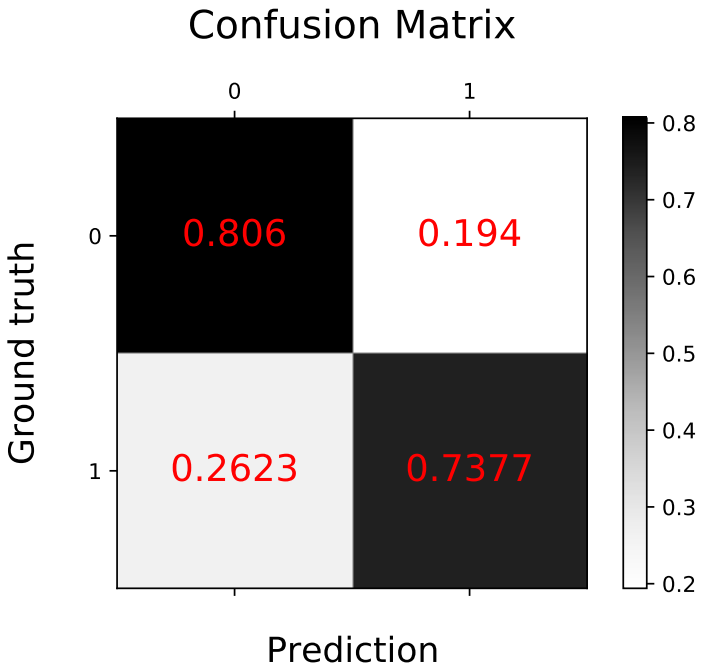


Figure 6.2: Averaged confusion matrix for mortality prediction over 5-fold cross validation on our trauma training dataset.

we measure the concordance-index (C-index) as outlined by [154]. BERTSurv achieved a C-index of 0.7 on trauma patients, which outperforms a Cox model with a C-index of 0.68. To reduce the risk of ICU trauma patients progressing to an irreversible stage, accurate and early prediction of patient condition is crucial for timely medical decisions. Mortality and cumulative hazard curves for two patients with different outcomes from BERTSurv are shown in Fig. 6.6 and Fig. 6.7. Fig. 6.6(c) indicates that an earlier discharged patients have a lower risk than later discharged patients, while Fig. 6.6(b) shows that patients who die early are at a relatively higher risk compared with patients who die later. Comparing Fig. 6.6(a) and Fig. 6.6(d), the gap between early discharge vs. early death is larger than that of late discharge vs. late death. Similar conclusions can be drawn from the hazard curves in Fig. 6.7. Such survival and hazard curves can provide physicians with comprehensive insight into patients condition change with time.

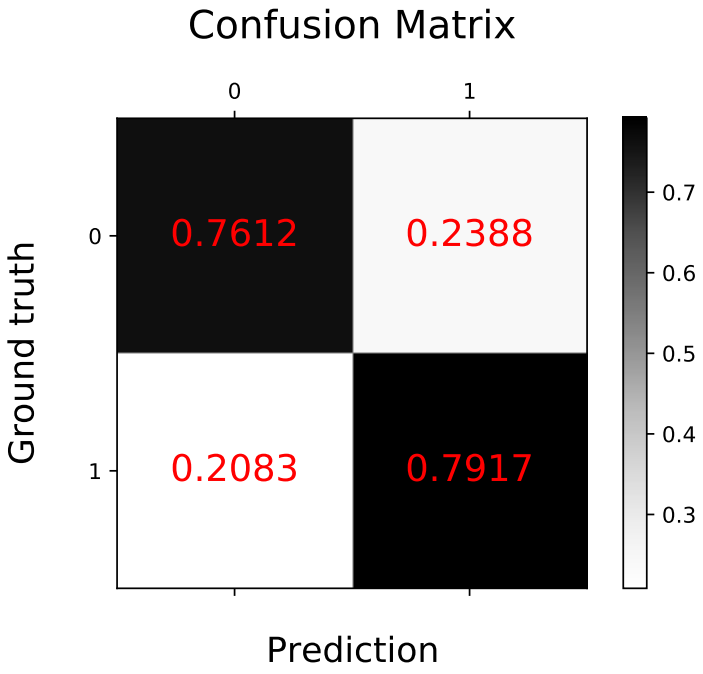


Figure 6.3: Confusion matrix for mortality prediction on trauma testing dataset.

Fig. 6.8 depicts four self-attention mechanisms in BERTSurv which help to understand patterns in the clinical notes. In all of the panels, the x-axis represents the query tokens and the y-axis represents the key tokens. In panels (a) and (b), we analyze a clinical note "left apical cap and left lateral pneumothorax suggests severe chest trauma " from a patient that died at hour 76. Panels (a) and (b) are two different head attention mechanisms. Panel (a) indicates "severe chest" and panel (b) extracts "trauma" as prominent patterns, respectively. Similarly, panels (c) and (d) are two head attention mechanisms for a patient discharged at hour 85. The input note to BERTSurv is "the endotracheal tube terminates in good position approximately 4 cm above the carina". BERTSurv finds "good" and "good position" in panels (c) and (d), respectively. Both "severe chest and "good position" help in understanding the patients' conditions and strongly correlate with the final outcomes. The indications from extracted patterns to patient outcomes show the effectiveness of BERT representation for clinical notes.

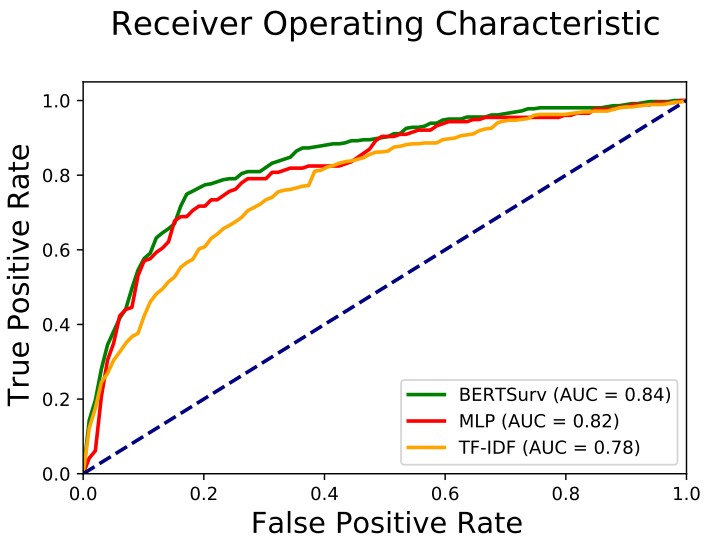


Figure 6.4: Receiver operating characteristic (ROC) curve for mortality prediction over 5-fold cross validation on our trauma training dataset. BERTSurv outperforms both baselines.

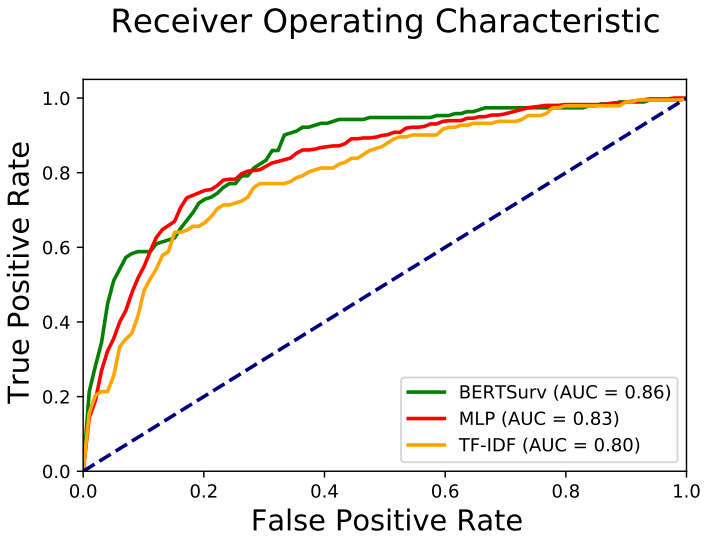
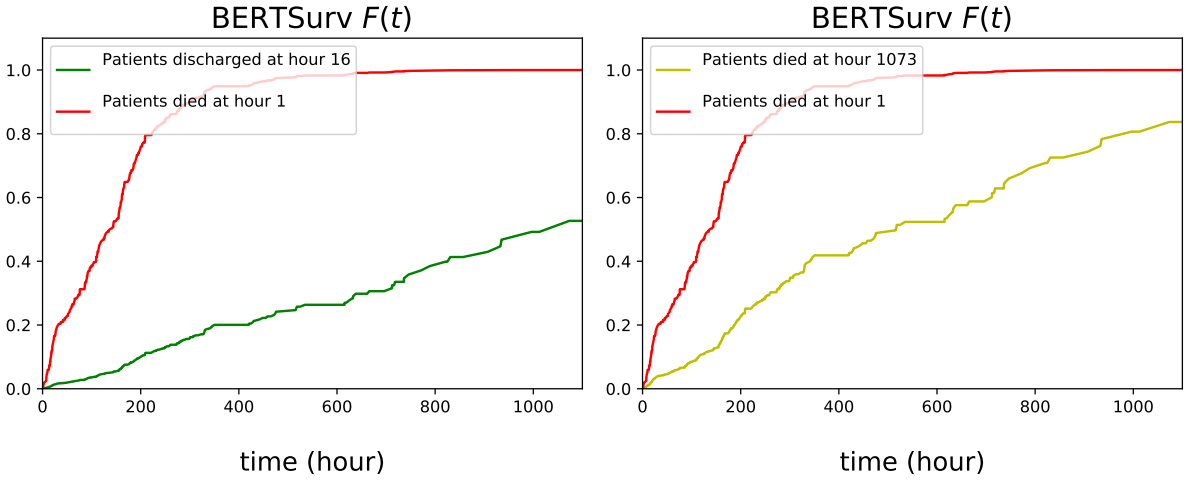
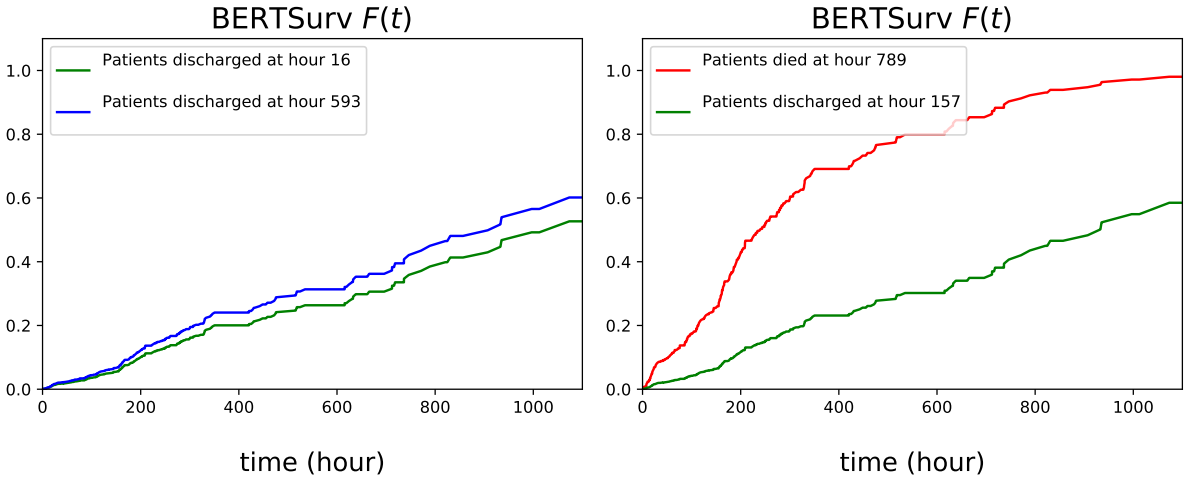


Figure 6.5: Receiver operating characteristic (ROC) curve for mortality prediction in trauma testing dataset. BERTSurv outperforms both baselines.



(a) early discharge vs. early death

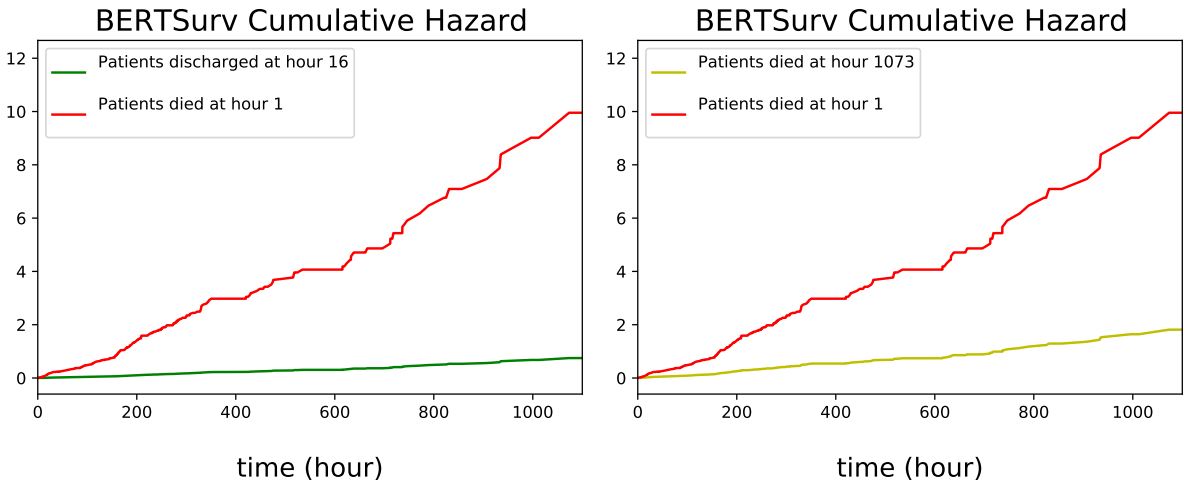
(b) early death vs. late death



(c) early discharge vs. late discharge

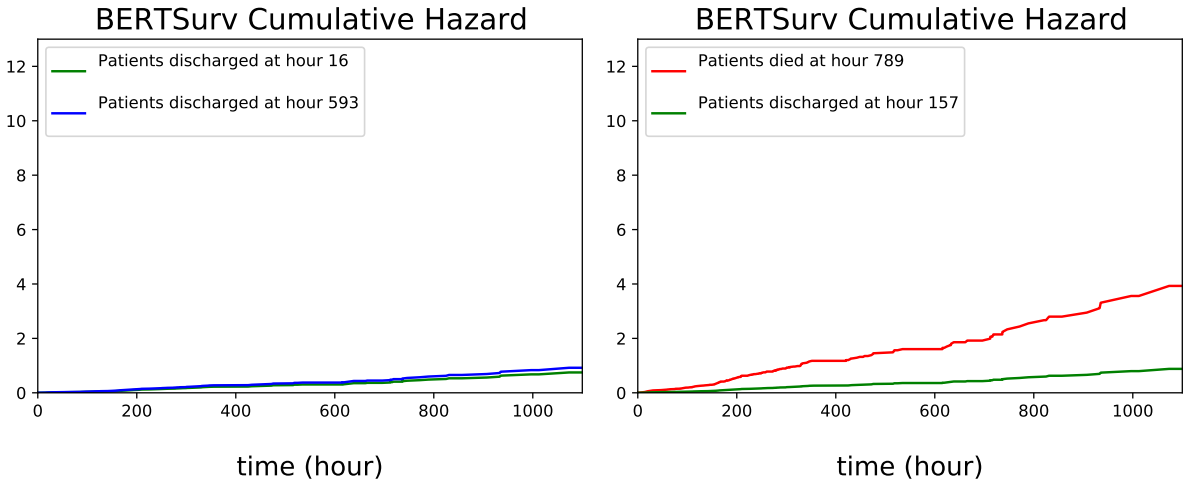
(d) late discharge vs. late death

Figure 6.6: Prediction of mortality as a function of time after admission to ICU using BERTSurv.



(a) early discharge vs. early death

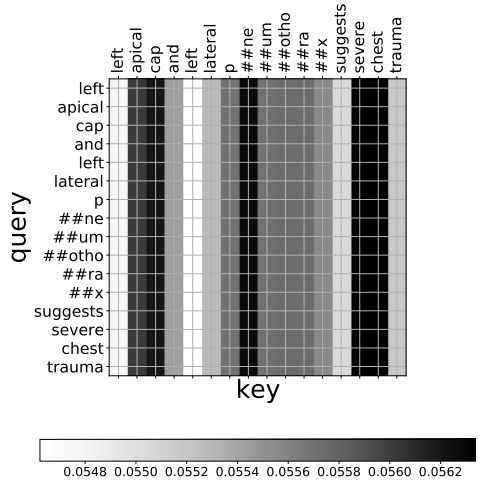
(b) early death vs. late death



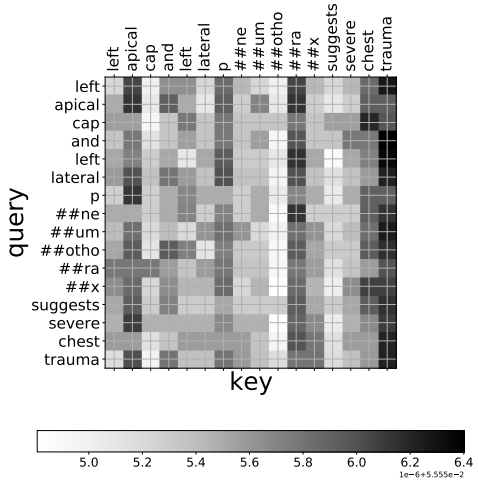
(c) early discharge vs. late discharge

(d) late discharge vs. late death

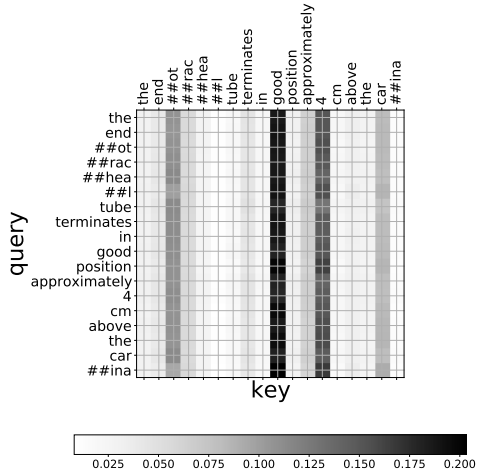
Figure 6.7: Prediction of cumulative hazard function as a function of time after admission to ICU using BERTSurv.



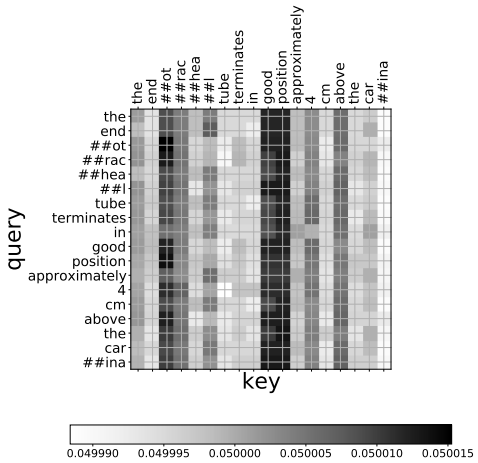
(a) patient died at hour 76



(b) patient died at hour 76



(c) patient discharged at hour 85



(d) patient discharged at hour 85

Figure 6.8: BERT visualization. The x-axis are the query tokens and the y-axis are the key tokens. Panels (a) and (b) are two head attention mechanisms for a patient that died at hour 76. The input notes to BERTSurv read "left apical cap and left lateral pneumothorax suggests severe chest trauma". Panels (a) and (b) extract "severe chest" and "trauma" as prominent patterns from the two heads, respectively. "severe chest" and "trauma" provide insight on the patient's critically ill condition. Similarly, panels (b) and (c) are two head attention mechanisms for a patient discharged at hour 85. The input notes include "the endotracheal tube terminates in good position approximately 4 cm above the carina". "good stands out in panel (c) and "good position" emerges in panel (d). Both "good" and "good position" are strong indications that the patient is in a relatively benign condition.

6.5 Discussion

We have proposed a deep learning framework based on BERT for survival analysis to include unstructured clinical notes and measurements. Our results, based on MIMIC III trauma patient data, indicate that BERTSurv outperforms the Cox model and two other baselines. We also extracted patterns in the clinical texts with attention mechanism visualization and correlated the assigned weights with survival outcomes. This work is a proof of principle for the incorporation of clinical notes into survival analysis with deep learning models. Given the current human and financial resources allocated in preliminary clinical note analysis, our method has foreseeable potential to save labor costs, and further improve trauma care. Additional data and work are needed, however, before the extent to which survival analysis can benefit from deep learning and NLP methods can be determined.

Chapter 7

How Much Does It Hurt: A Deep Learning Framework for Chronic Pain Score Assessment

7.1 Introduction

Deep learning models have achieved remarkable success in computer vision [6], natural language processing [8], speech recognition [7] and the game of Go [9]. Recently there has been increasing interest in applying deep learning for end-to-end health data analysis [155]. However, e-health data analysis is even more challenging since well-being data can be affected by many factors, for example individual differences, measurement errors in data collection and missing data.

Chronic pain is described as persistent or recurrent pain that lasts for at least 3 to 6 months [156]. According to the 2016 National Health Interview Survey (NHIS), roughly 20.4% (50.0 million) of U.S. adults suffer from chronic pain. Chronic pain affects individuals, their families, and society, and results in complications harming both physical

and mental health. The economic costs of chronic pain and pain-related disability in the United States cannot be overstated. One influential report [157] conservatively estimated an annual toll of $560-$650 billion dollarsfar exceeding the costs of cardiovascular disease, cancer, and diabetes. Therefore, identifying the score of chronic pain is of significant value to reduce further complications.

Neurophysiological signals have been used to quantify pain [158, 159, 160]. In the last decade, there has been some progress towards discovering the neurobiological substrates of pain. However, none of those methods is low-cost or easy-to-use. Clearly, there is an unmet need for a low-cost, easy-to-use Pain Meter. According to recent research results [161, 162, 163], classical physiological measurements can effectively quantify pain. Inexpensive technology is now available to measure relevant physiological features. Taken together, it is clear that objectively quantifying pain is possible. We thus investigated a number of inexpensive commercial sensors capable of measuring physiological variables for pain score assessment. We have thus far built prototype Pain Meters that offer the immense potential to revolutionize pain treatment and the development of therapeutics.

To this end, we collected two new chronic pain datasets, using prototype Pain Meter 1 and 2, respectively. For Dataset 1, our subject has been suffering chronic pain for more than 10 years. Neck and shoulder pain were causing her difficulty to perform daily activities. For Dataset 2, we recruited chronic pain subjects from our local community. All subjects signed an informed consent form according to a protocol approved by a Human Subjects Committee. Our chronic pain datasets are characterized by the following unique properties: First, we use Photoplethysmography (PPG) [164], which is low cost and comfortable for patients, to collect pulse signals. Secondly, we also include several temperature signals and Galvanic Skin Response(GSR) signals to detect the chronic pain symptoms like nervousness. For Dataset 2, we also use accelerometers and gyros to detect movements.

Given the pain score datasets, we introduce the task of chronic pain score prediction, which is to train an end-to-end oridinal classifier to accurately predict pain score. The illustration of our workflow is shown in Fig. 7.1. An accurate chronic pain score assessment will

1. Facilitate the development of new therapies both in the laboratory and in Phase II and Phase III clinical trials.

2. Make it possible for physicians to quantify the effects of existing therapies on individual patients.

3. Minimize the harm caused by diagnostic delays and the under/overtreatment of pain due to the influence of gender, race, or age.

4. Allow a patient to decide, with objective personal data, whether current treatments are effective in his/her quest to reduce chronic pain.

5. Serve as a biofeedback device for chronic pain subjects. The unconscious mind can learn to control things it can monitor [165]. If people are enabled to accurately monitor their chronic pain score with a Pain Meter, their unconscious mind can figure out how to decrease the chronic pain score. It is trained and rewarded by the tiny decreases in pain score that are accurately and continuously monitored.

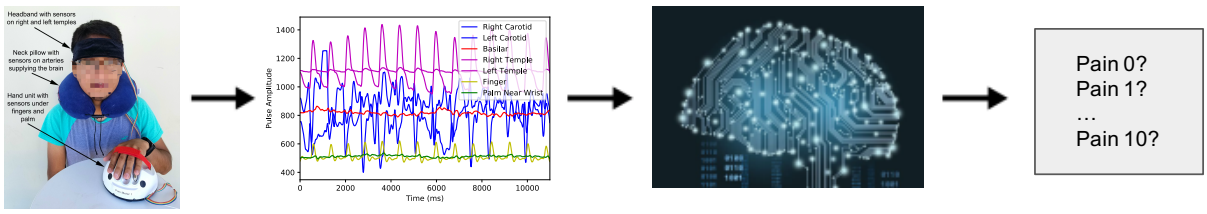


Figure 7.1: The workflow for chronic pain score assessment. The brain image is from Google.

Our main contributions are threefold:

1. We propose a deep learning ordinal regression framework for chronic pain score assessment. To the best of our knowledge, this is the first work to use deep learning for chronic pain score assessment.

2. We collect two new chronic pain datasets using our prototype Pain Meters to predict the score of chronic pain.

3. We split the long recordings into smaller slices for training, which not only eases the burden on GPU memory but also provides many training samples for deep learning models. We define Consensus Prediction as the majority voting result of the sampled short slices for testing, since not all of the short slices can be expected to contain enough useful information.

The remainder of this chapter is organized as follows. Section 7.2 discusses related work. Section 7.3 describes the Pain Meters we used for data collection and introduces the classification problem. Section 7.4 delineates the methodology. Section 7.5 introduces the experimental setup for comparison of our results to those obtained via multilayer perceptron and logistic regression. Results and case studies are described in Section 7.6. Section 7.7 shows that our deep learning framework can also be used to provide feedback to improve the design of the Pain Meter. Section 7.8 is the Discussion.

7.2 Related Work

7.2.1 Deep Learning for E-Heath

Since the emergence of deep learning in e-health, more and more researchers have been implementing deep learning models for medicine, aiming to improve health care [155], classify diseases [166], and prevent misdiagnosis [167]. More specifically, the prediction of

medical events has been popular, including the prediction of death rates [168], prescriptions [169], and successful extubation [170]. Although many researchers have applied deep learning to e-health, to the best of our knowledge no researcher has applied deep learning from pulse signals to the prediction of chronic pain scores [171]. There are image based models for automatic estimation of pain [172, 173]. Deep Pain [172] uses long short term-memory (LSTM) and analyzes images of facial expression. However, images alone are not reliable since images alone may not contain enough relevant information for pain score prediction. Many factors affect pain score, including stress and mood, which might not be fully captured by image based methods. Physio-based models have been developed for assessments of physiological pain [174] but these have not been applied to chronic pain. Motivated by this need, we propose a Convolutional Neural Network (CNN)-based framework that uses physiological signals to assess chronic pain scores.

7.2.2 Chronic Pain and Traditional Machine Learning

In the context of pain assessment research, physiologically-based pain has been the main focus for many pain researchers [175]. Chronic pain, on the other hand, is prolonged, lasting anywhere from months to years [176]. In the past, traditional machine learning methods have been applied to e-health, but this has required extensive feature extraction [177]. Human feature extraction has many disadvantages. For example, it is costly and can result in the loss of data interpretability [178]. Random Forest has been used to monitor the nociception (perception of pain) level, which requires feature extraction [179]. Physiological parameters like heart rate, heart rate variability, plethysmograph wave amplitude, skin conductance level, number of skin conductance fluctuations, and their time derivatives are extracted for prediction. In contrast, our end-to-end deep learning framework requires no feature extraction and little data preprocessing.

7.2.3 Physiological Sensors

Photoplethysmography (PPG) is a widely used non-invasive method for measuring pain intensity, in which changes in blood volume and light absorption are detected [164]. This type of technology is commonly used because it is simple and can easily collect data, while being cost-efficient and accessible [180]. A PPG device has also been developed to monitor respiratory and heart rates of infants, and this has been proven to perform well [181]. PPG is favored not only for clinical use, but also for home use as a biofeedback device. In addition, it is more comfortable for patients because it does not involve gel and electrodes contacting their skin [182]. Thus, in our prototype Pain Meter, we use PPG [183] to collect pulse recordings. We also included axis accelerometer gyroscope modules (MPU-6050), force sensitive resistors to measure the forces that cause low frequency motion (DF9-40), and a GSR sensor (ZIYUN Grove GSR sensor).

7.3 Data Collection and Classification

7.3.1 Prototype Pain Meter

Fig. 7.2 shows the device we used for collecting Dataset 1. Pain Meter 1 contains: 1) two PPG pulse sensors held to the temples via a headband, three at the three arteries supplying blood to the brain mounted in a neck pillow, and two at the fingertip and palm, 2) temperature sensors at each location of the PPG pulse sensors, and 3) GSR sensors embedded in the block on which the hand rests. These provide a total of 15 signals, recorded in Dataset 1.

As is shown in Fig. 7.3, it turned out that the PPG sensors were sensing more than pulse. They were also sensing subtle motion. Motivated by these phenomena, we added actual motion sensors in our Pain Meter 2. As is shown in Fig. 7.4, Pain Meter 2 contains:

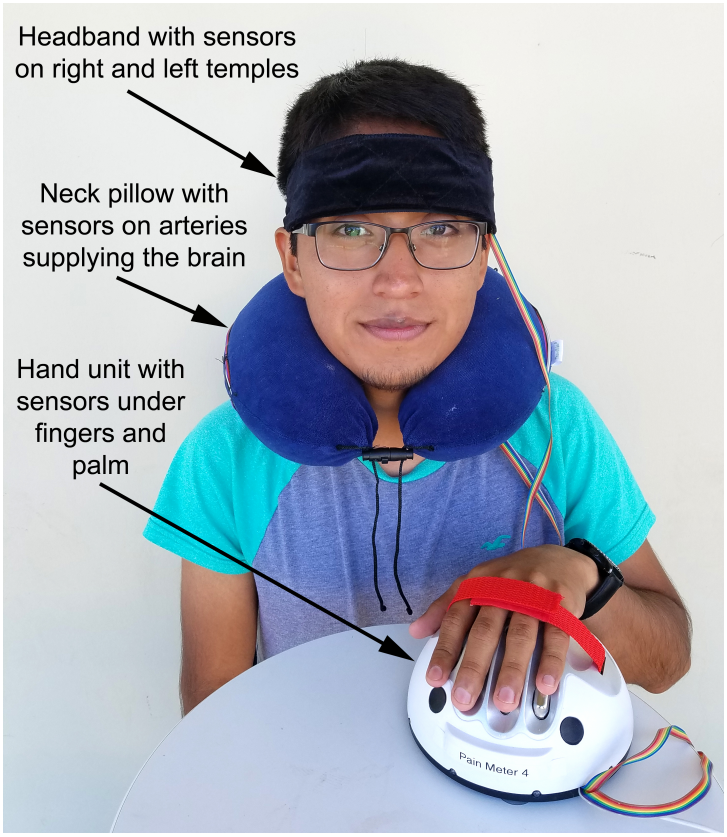


Figure 7.2: Pain Meter 1 sensed temperature, pulse and GSR, but it did not directly sense motion.

1) PPG pulse sensors in a headband, in a neck band, and on the fingertip, 2) temperature sensors on the neck and fingertip, 3) 3-axis accelerometers and 3-axis gyros in the head band and wrist band, 4) force sensors on the forehead, back of neck, side of neck, and wrist band, and 5) GSR sensors between the middle and ring fingers. These provide a total of 25 signals, recorded in Dataset 2.

For both Pain Meters, a Teensy 3.6 microcontroller with a 32-bit 180 MHz ARM Cortex-M4 processor is used to sample all signals every 15ms. A data acquisition program was designed in MegunoLink Pro for Pain Meter 1 data and a customized data acquisition program was written in Python for Pain Meter 2 data.

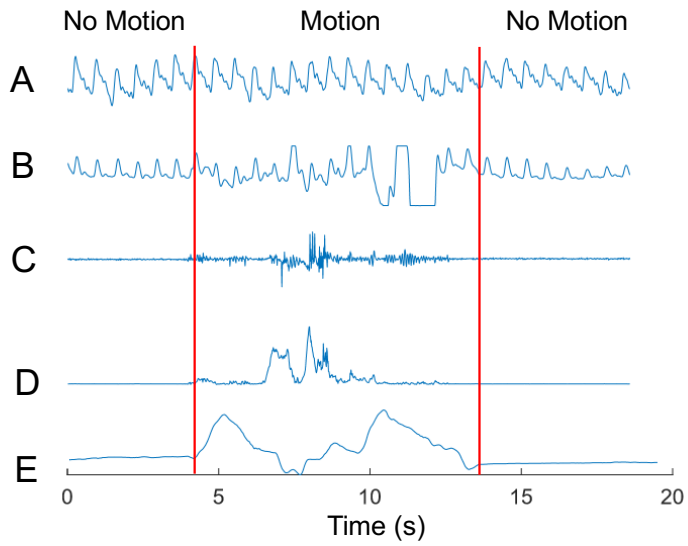


Figure 7.3: Motion affects the pulse sensor signals, which are based on measuring the intensity of light reflected from the skin. Part A shows a pulse sensor at the carotid artery, far from motion. Part B shows a pulse sensor at the finger. Motion is generated by flexing the wrist and is measured in part C by a wrist acceleration sensor. Part D shows data from wrist gyro sensor and part E shows data from a wrist force sensor. Note that the pulse sensor on the finger (sensor B), which is closer to the motion, is strongly affected by the motion, while the pulse sensor on the carotid artery (sensor A) is not.

Figure 7.4: Pain Meter 2 consists of: 1) PPG pulse sensors in a headband, in a neck band, and on the fingertip, 2) temperature sensors on the neck and fingertip, 3) 3-axis accelerometers and 3-axis gyros in the head band and wrist band, 4) force sensors on the forehead, back of neck, side of neck, and wrist band, and 5) GSR sensors between the middle and ring fingers.

7.3.2 Data Collection

Using Pain Meter 1 from Fig. 7.2, our chronic pain Dataset 1 was collected from one subject who self-reported the pain score. The subject was asked to fix the headband and neck pillow into a comfortable position. Once secured, real-time plots of the pulse and temperature data were viewed to verify that the head and neck sensors were collecting accurate and reliable signals for the subject. Minor adjustments to the sensor positions can be made if needed. After adjustments were made, the subject placed the left hand on the module to read the hand signals. A final verification step viewing all plots of data was performed. Then the subject was asked to close their eyes and relax before the recording begins. After 10 minutes, the recording was ended. Each recording was taken on a different day at the same time in the afternoon, the same temperature and the same environment brightness. During recordings with Pain Meter 1 we noticed that some of the pulse signals became erratic compared to other pulse signals and that this erratic behavior seemed to correlate with pain. We hypothesized that this was due to subtle movement [184] and constructed Pain Meter 2 (Fig. 7.4) to have motion sensors. With the addition of motion sensors we could confirm this hypothesis (Fig. 7.3) and measure the motion directly.

Pain score distributions for the 2-class Dataset 1 and 7-class Dataset 2 are shown in Table 7.1 and Fig 7.5, respectively. Each recording has a length of 10 minutes, with signals sampled every 15 milliseconds. We have 4 recordings from one subject in Dataset 1 and 62 recordings from 20 subjects in Dataset 2. We divide each 10-minute recording into ten mutually exclusive 1-minute samples.

Table 7.1: Data statistics for Dataset 1

Data	Classes	Pain score distribution
Chronic Pain	2	1 : 1

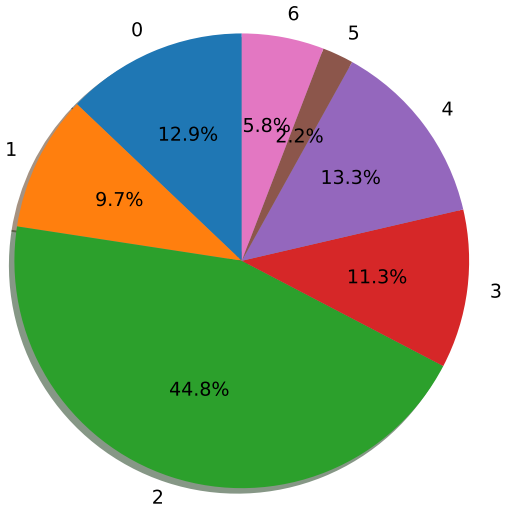


Figure 7.5: Distribution of pain scores in Dataset 2. The numbers outside the pie chart are the corresponding pain scores and the percentages of time for which each pain score is reported are shown in the pie chart.

7.3.3 Classification Problem

The classification problem for chronic pain assessment is to classify a Pain Meter dataset of time sequences into pain scores on a scale of 0 for no pain to 10 for the worst pain possible. The model is first trained on Dataset 1 from a chronic pain subject who self reports her pain score for each of the datasets on a 0 to 10 scale. Note that seldom has our subject reported a pain score higher than 2 in Dataset 1. Thus in Dataset 1, we use pain scores 1 and 2. In Dataset 2, we have pain scores from 0 to 6, which formulates to be a 7 class classification problem. Our goal is to use the trained deep learning model to accurately classify chronic pain datasets into the self reported pain scores.

7.4 Methodology

The model architecture, shown in Fig. 7.6, consists of convolution-pooling layers followed by fully connected layers. To learn temporal and correlation features, the convolution is performed on both the time and sensor dimensions. We split the 1-minute samples into smaller slices with length of seq_length. Sensor recording signals with a dimension of (number of sensors (N) $\times$ seq_length) serve as input x for the neural network. A convolution operation involves a filter $w \in \mathbb{R}^{st}$, which is applied to a window of s sensors and t samples to produce a new feature. For example, a feature $f_{i,j}, (0 \leq i \leq N-s+1, 0 \leq j \leq seq\_length-t+1)$ is generated from a window size (s,t) of the sensor signals:

$$f_{i,j} = ReLU(wx_{i:i+s-1,j:j+t-1} + b), \tag{7.1}$$

where $b \in \mathbb{R}$ is a bias term. This filter is applied to each possible window of the voltage signals to produce a feature map:

$$f = \begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,seq\_length-t+1} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,seq\_length-t+1} \\ \cdots & \cdots & \cdots & \cdots \\ f_{N-s+1,1} & f_{N-s+1,2} & \cdots & f_{N-s+1,seq\_length-t+1} \end{bmatrix}, \tag{7.2}$$

with $f \in \mathbb{R}^{N-s+1,seq\_length-t+1}$. We then apply a max-pooling operation over the feature map and take the maximum value $m = \max(f)$ as the feature corresponding to this particular filter. The idea is to capture the most important feature, the one with the highest value, for each feature map. Our model uses multiple filters to obtain multiple features. These features form the penultimate layer and are passed to a fully connected softmax layer whose output is the probability distribution over different pain scores. We adjust the number of convolutional ReLU layers from 2 to 5, based on the choice of

seq_length. The prediction of our model, parameterized by $\boldsymbol{W}$, is given by $\boldsymbol{p}(\hat{\boldsymbol{y}}|\boldsymbol{x}, \boldsymbol{W})$.

Due to the existence of the inherent ordering information in our chronic pain score, we apply ordinal regression, a setting that bridges metric regression and classification, to predict chronic pain scores of ordinal scale. Compared to regular regression problems, these pain scores are discrete. These pain scores are also different from the labels of multiple classes in classification problems due to the existence of the ordering information. The cross entropy loss of our ordinal regression for input vector $\boldsymbol{x}_n$ is as follows,

$$\begin{aligned} L_n(\boldsymbol{W}) =& (1 + |\frac{argmax(\boldsymbol{p}(\hat{\boldsymbol{y}}|\boldsymbol{x}_n, \boldsymbol{W})) - \boldsymbol{y}}{C-1}|) \\ & \times \frac{1}{C}(\sum_{i=1}^{C} -\boldsymbol{y}_i \cdot log(\boldsymbol{p}(\hat{\boldsymbol{y}}|\boldsymbol{x}_n, \boldsymbol{W}))), \end{aligned} \tag{7.3}$$

where $\boldsymbol{y}$ represents the true pain scores, $\boldsymbol{p}(\hat{\boldsymbol{y}})$ denotes the predicted probability vector with one value for each possible pain score, and C is number of pain scores. Divided by $C-1$, the absolute error $|argmax(\boldsymbol{p}(\hat{\boldsymbol{y}}|\boldsymbol{x}_n, \boldsymbol{W})) - \boldsymbol{y}|$ is normalized between 0 and 1, with C classes in total. By multiplying $|\frac{argmax(\boldsymbol{p}(\hat{\boldsymbol{y}}|\boldsymbol{x}_n, \boldsymbol{W})) - \boldsymbol{y}}{C-1}|$, the normalized absolute error between prediction and ground truth, with cross entropy loss, we include the ordinal information in our loss function. We penalize more in our loss function if the absolute error between ground truth and prediction is larger.

For testing, we define Consensus Prediction to measure the performance of predictions for the whole sensor signal sample. Consensus Prediction synthesizes results from multiple short slices by majority voting, which can significantly improve the prediction accuracy for a long sample. This is because not all of the short time slices can be expected to contain useful information for classification.

We use Batch Normalization [26] to accelerate training. For regularization, dropout [27] and early stopping methods [28] are implemented to avoid overfitting. Dropout prevents co-adaptation of hidden units by randomly dropping out a proportion of the hidden units

during backpropagation. Model training is ended when no improvement is seen during the last 100 validations. Softmax cross entropy loss is minimized with the Adam optimizer [29] for training. Since both frequency information and correlation between sensors are captured through convolution filters, our CNN-based framework automatically deals with the features needed for classificaiton. We use grid search for hyperparameter tuning. The hyperparameters are described in Table 7.2.

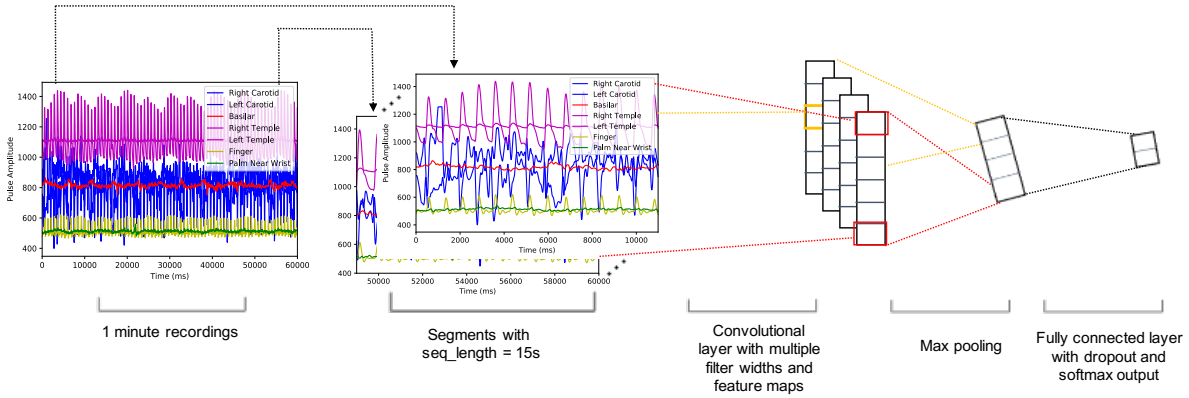


Figure 7.6: Model architecture: 1-minute samples of the voltages measured on the physiological sensors are collected from our prototype Pain Meter. Segments with seq_length = 15 s of these samples are individually classified. These individual classifications are conducted for Consensus Prediction in Pain score recordings.

7.5 Experimental Setup

We introduce two baselines: Multilayer Perceptron (MLP) and Logistic Regression, to compare with our proposed CNN framework for the two classification problems.

7.5.1 CNN based Model and Consensus Prediction

We implement a parallel processing framework that distributes the convolutional neural network into multiple (N) GPUs to ease the burden on GPU memory. Each GPU

contains an entire copy of the deep learning model. We first split the training batch evenly into N sub-batches. Each GPU processes only one of the sub-batches. Then we collect gradients from each replicate of the deep learning model, aggregate them together and update all the replicates. We train our CNN based framework with two NVIDIA GeForce GTX 1080s, each of which has a memory of 11178 MB.

7.5.2 Multilayer Perceptron

We use a four layer fully connected network, whose output is the probability distribution over two different classes, as a baseline. We use a parallel processing framework similar to our CNN model implementation. We train our MLP on two NVIDIA FeForce GTX 1080s.

7.5.3 Logistic Regression

Since the signals are periodic, we extract individual voltage signal FFT (x_1) and pairwise sensor signal Pearson Correlation (x_2) as features (X) for a Logistic Regression classifier:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(w_0+w_1x_1+w_2x_2)}}, \tag{7.4}$$

where Y is the label for classification and P is the probability of predicting Y as label 1 (pain score 1). w_0, w_1 and w_2 are model parameters to be learned during training.

7.5.4 Split Training and Testing

Since each of our recordings is long enough to split into multiple informative samples and there exists different settings among different recordings, we implement two methods of splitting the training and testing data. We first divide each 10-minute recording into ten mutually exclusive 1-minute samples.

1. Considering the size of our datasets in prediction, we use 5-fold cross validation. For each of the 10-minute recordings, we use between $2*(i-1)$ and $2*i$ minutes as testing, and the rest for training, for the i-th fold. This training/testing split fits well with the scenario of practical use of pain score assessment. It is known that different subjects have different perceptions of pain. For real use, our pain meters need some self-calibration before getting accurate pain score readouts.

2. Leave-one-recording-out cross validation on all the recordings. Each recording is used once as a test set while the remaining recordings form the training set. We note that, due to differences in subjects' sensitivity to pain, different settings while recordings, and the size of our data, it is much more difficult to predict across subjects than within one recording. We report the result using this splitting method in Section 7.8.

7.6 Results and Analysis

Considering the size of our dataset in prediction, we use 5-fold cross validation and report the average results in this section. Note that Dataset 2 is unbalanced. We also report the confusion matrix for evaluation.

Table 7.2: Hyperparameters

Hyperparameters	Value
Batch size	24
Epoch	2000
Dropout rate	0.5
Seq_length	15 seconds
Learning rate	0.5

Confusion matrices are shown in Fig. 7.7 and Fig. 7.8 for chronic pain score prediction. Dominant numbers on the confusion matrix diagonal indicates that our model achieves high accuracy for each class. Fig. 7.9 shows the individual class prediction distribution for Dataset 2. The infrequent prediction mistakes scatter around the ground truth. For example, although 11% of the predictions from pain score 3 are incorrect, they are still close to 3 (either 2 or 4). Fig 7.10 further demonstrates that the probability of making an error decreases as the absolute prediction error increases. This is the benefit from ordinal regression, which penalizes more in the loss if the absolute error between ground truth and prediction is larger. Fig. 7.11 is a scatter plot of expected predicted pain score vs. self-reported pain score. The relationship between predicted pain score and ground truth is highly linear, with an R-squared (R^2) of 0.9463.

Since we use seq_length to split long recordings into shorter slices, some of the short slices may not contain enough information for pain score prediction. However, these effects can be eliminated using Consensus Prediction. Although we have imblanced data in Dataset 2, we still use accuracy to compare different models and check the benefits obtained from Consensus Prediction, since chronic pain subjects are most interested in prediction accuracy.

Results of our framework compared against other machine learning models on chronic pain recordings are shown in Tables 7.3 and 7.4 respectively. We compare our CNN based model with two baselines: Multilayer Perceptron (MLP) and feature-based Logistic Re-

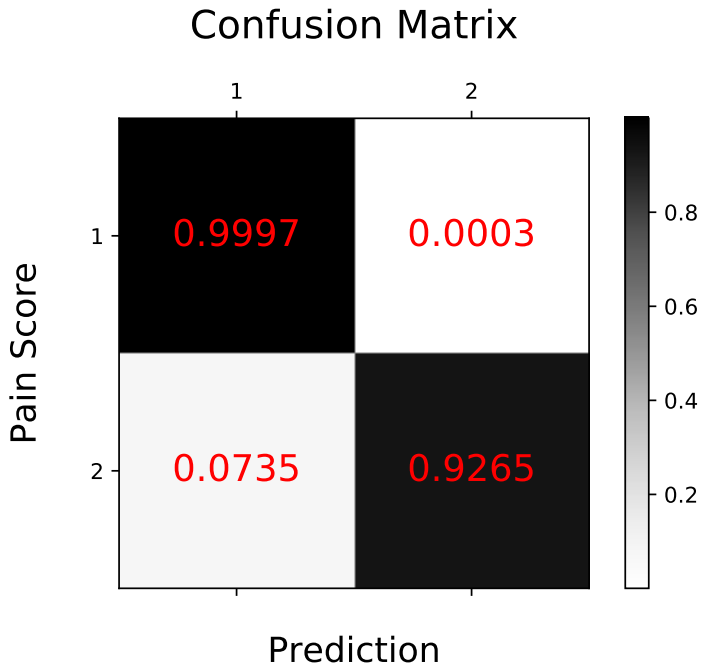


Figure 7.7: Confusion matrix for chronic pain score prediction in Dataset 1.

gression. For Dataset 1, our CNN-based deep learning approach improves the prediction accuracy by 5.25% compared to feature-based Logistic Regression. Fig. 7.12 shows the Consensus Prediction accuracy. The accuracy improves by 3.84% using Consensus Prediction. Although not all of the short slices can be expected to contain enough useful recording patterns, we can overcome that when we synthesize multiple individual classification results from these short slices. For Dataset 2, our model achieves accuracy of 95.23% for short recording slices, which is a 2.8% improvement over feature based Logistic Regression. The accuracy further improves to 98.30% with Consensus Prediction as shown in Fig. 7.13. Our CNN based deep learning model also outperforms MLP on both of the two sets of recordings by 3.32% and 2.91% respectively, which shows CNN's advantage of local feature extraction using convolutional kernels over MLP. Also from Fig. 7.12 and Fig. 7.13, 100 time slices for each recording are sufficient in Consensus Prediction.

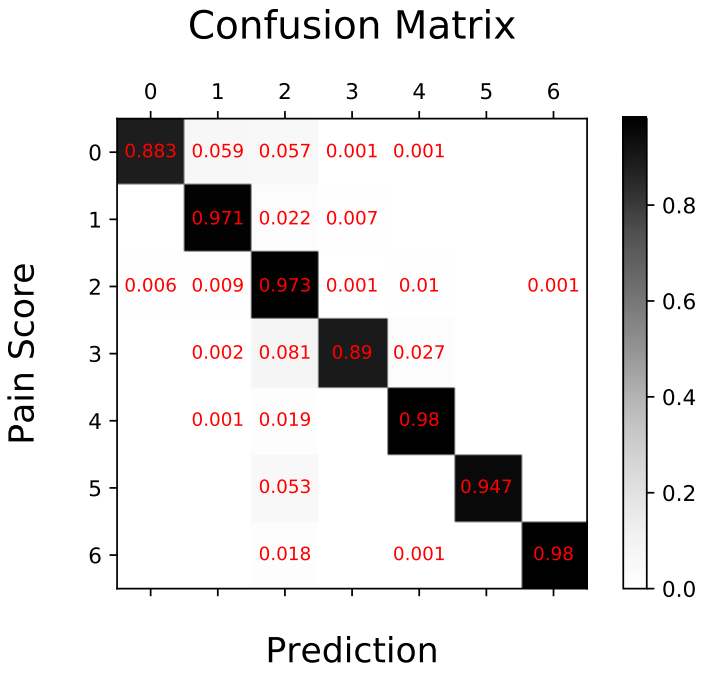


Figure 7.8: Confusion matrix for chronic pain score prediction in Dataset 2. Our model achieves high accuracy for each class.

Table 7.3: Cross-validation performance comparison of our deep learning model with Multilayer Perceptrons and Logistic Regression on Dataset 1.

Model	Accuracy on Testing
Convolutional Neural Network	**0.9630**
Multilayer Perceptron	0.9321
Logistic Regression	0.9150

Table 7.4: Cross-validation performance comparison of our deep learning model with Multilayer Perceptrons and Logistic Regression on Dataset 2. Although we have imblanced data for 7 classes, subjects are still most interested in prediction accuracy. We showed confusion matrix for our CNN model in the previous figure.

Model	Accuracy on Testing
Convolutional Neural Network	**0.9523**
Multilayer Perceptron	0.9238
Logistic Regression	0.9063

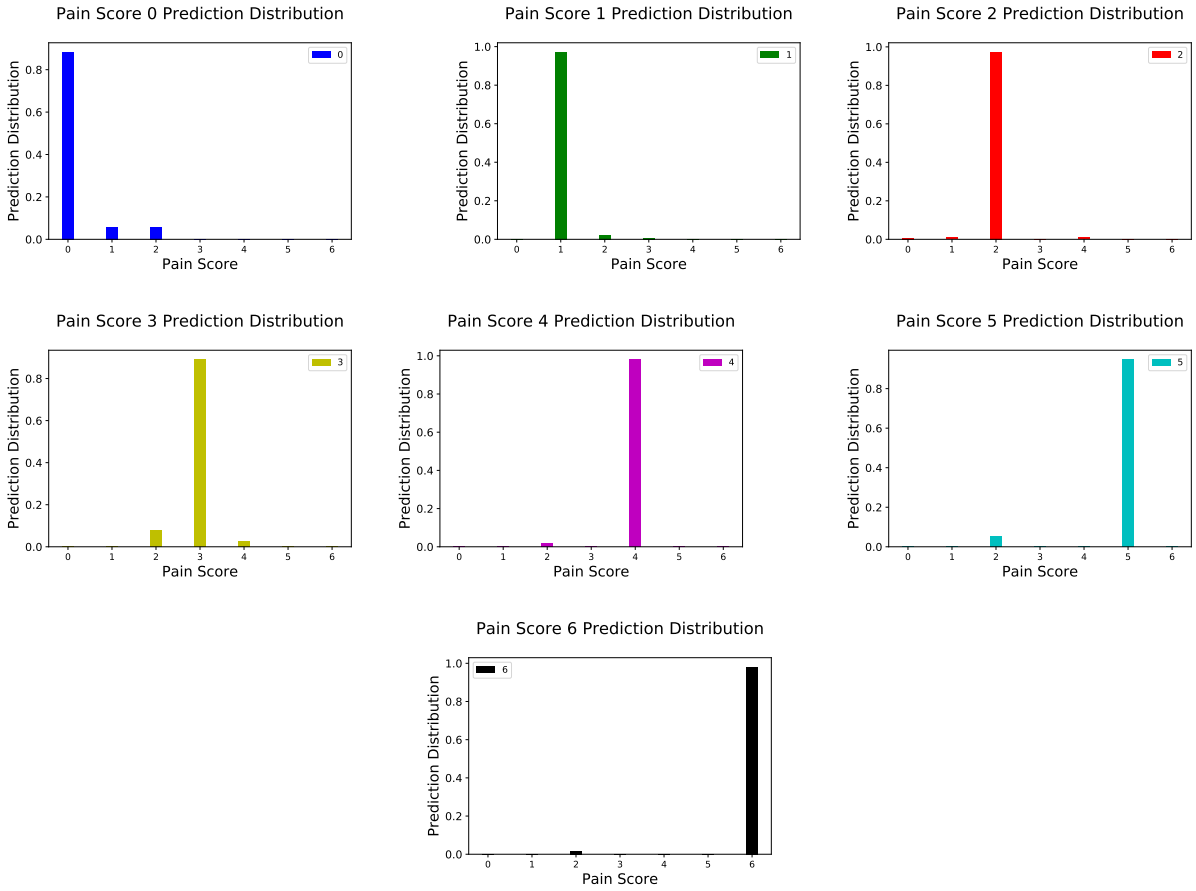


Figure 7.9: Individual class prediction distribution for Dataset 2. The occasional incorrect predictions scatter close to the ground truth.

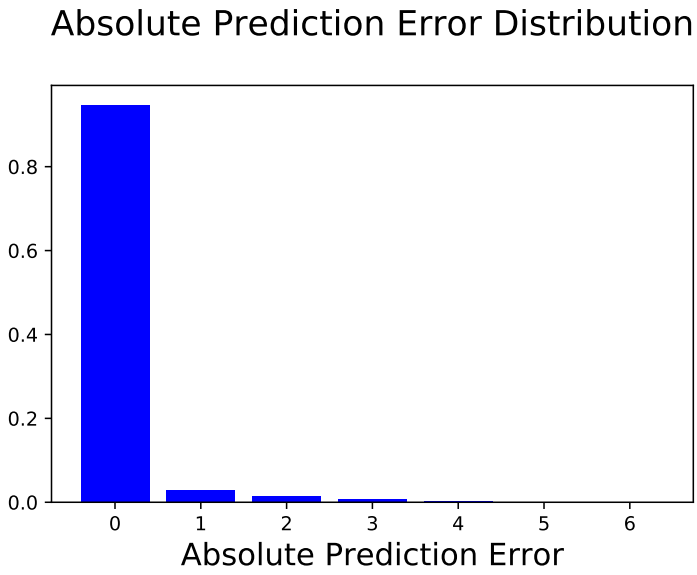


Figure 7.10: The distribution of absolute prediction error using ordinal regression for Dataset 2. The chance of making an error decreases with the increase of absolute prediction error.

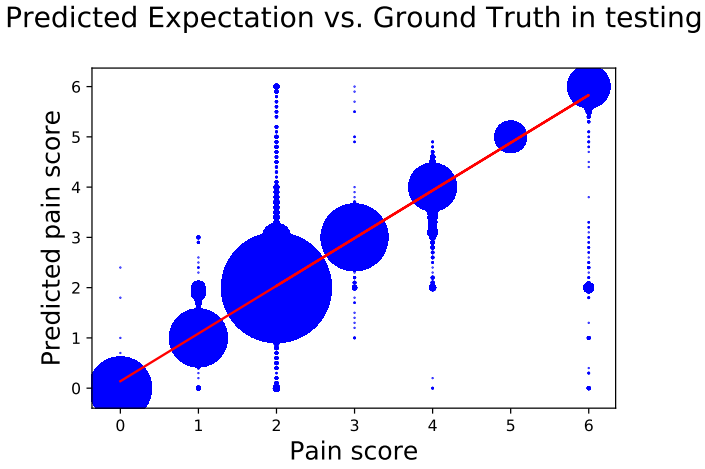


Figure 7.11: Scatter plot of expected predicted pain score vs. self-reported pain score for Dataset 2. The larger size of the point corresponds to the higher appearance frequency of the data point. We keep a decimal in calculating the expected predicted pain score because 0.1 is a reasonable precision to estimate pain in real life. The predicted pain score exerts high linear relationship with ground truth with a high R^2 of 0.9463.

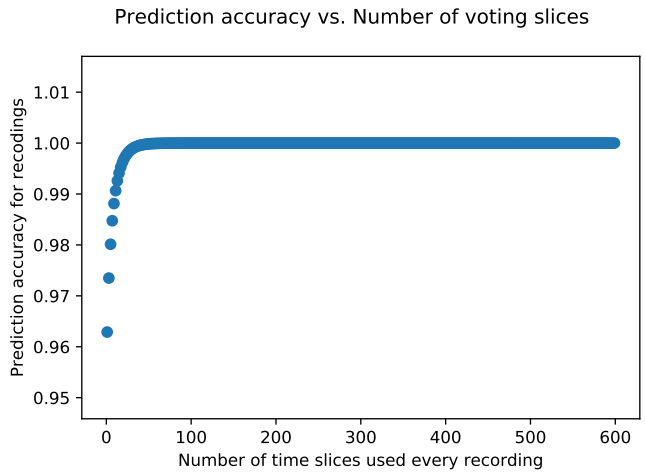


Figure 7.12: Consensus Prediction for chronic pain score prediction in Dataset 1.

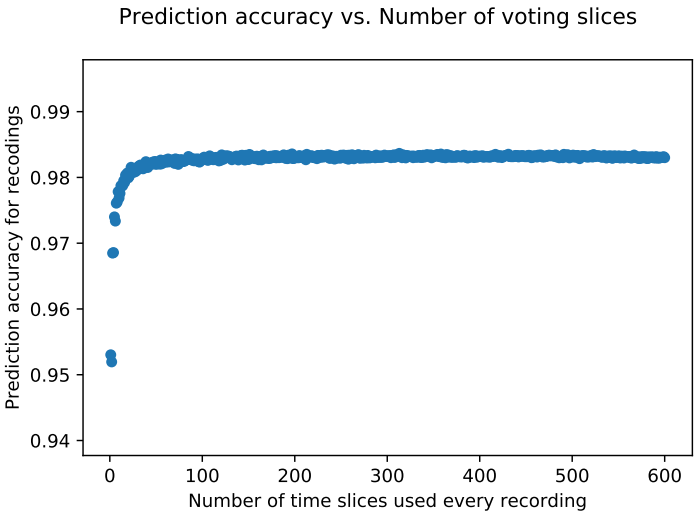


Figure 7.13: Consensus Prediction for chronic pain score prediction in Dataset 2.

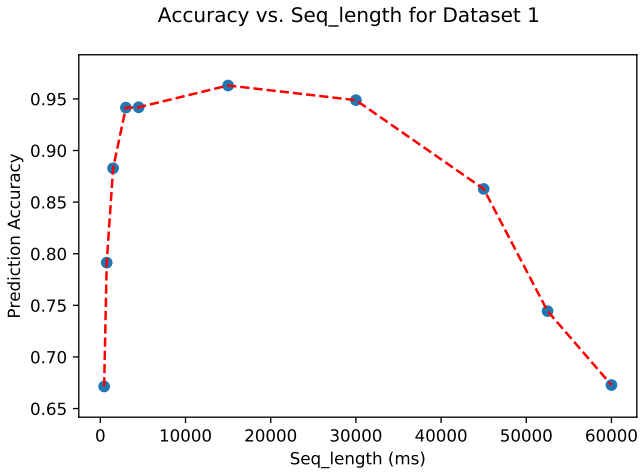


Figure 7.14: Accuracy vs. Seq_length trend for Dataset 1.

Fig. 7.14 shows the trend of accuracy versus the choice of seq_length for Dataset 1. For the effect of seq_length on accuracy, there exists a trade-off between number of training samples and representation of a whole recording. The short slices contain less information but can provide more independent training samples. For deep learning models, larger numbers of training slices help more than a larger sample. However, we still cannot choose too small of a seq_length, since a too short slice is not representative for a recording. Given the data we currently have, we use a seq_length of 15 seconds.

We show some failure cases for our CNN classifiers in Fig. 7.15. The upper figure shows the histogram of short sequence prediction for one recording, where blue indicates predicted successfully and red indicates predicted incorrectly. The x axis is the starting point of the sampled short sequences. The lower figure shows the corresponding voltage signals from our prototype Pain Meter recording. There is a period from 0.55 minute to 0.65 minute, during which the classifier consistently makes incorrect predictions. From the pulse signals, they also perform abnormally compared to other periods. This is also true for the short peak in the data coming from the palm near wrist at 0.28 minute. This illustrates that for those periods, by looking at shorter sequences, it is easy to make a

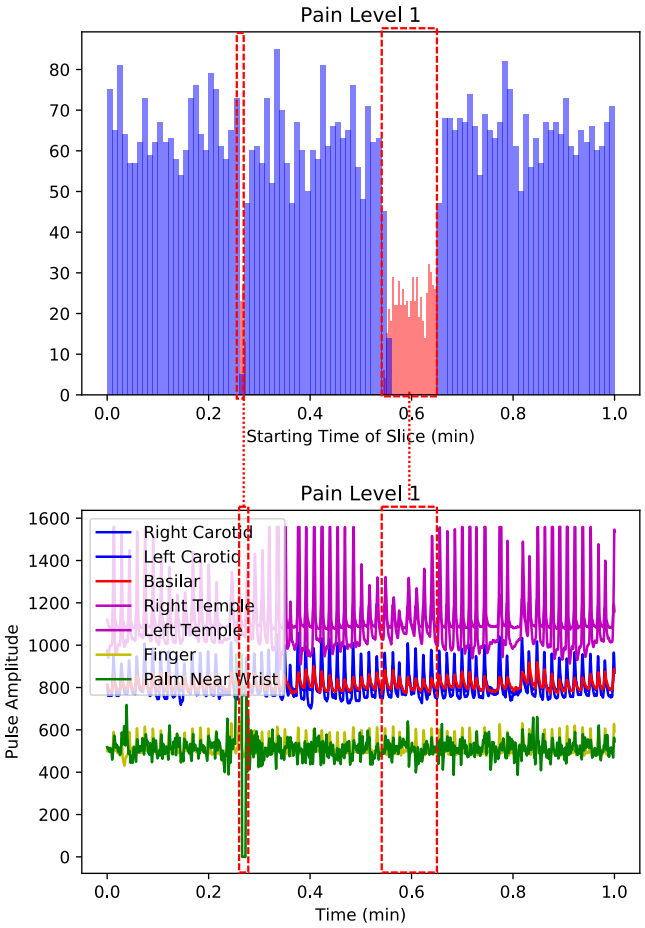


Figure 7.15: Case study for short slices that the classifier predicted incorrectly. In the upper figure, red color bars represent incorrect predictions while blue bars represent correct predictions.

wrong prediction for both human experts and a classifier. This is due to the fact that our sensors are sensitive to movements. Noise can be introduced with a contact position change between sensors and skin. However, by using Consensus Prediction, these errors have no effect on the final prediction.

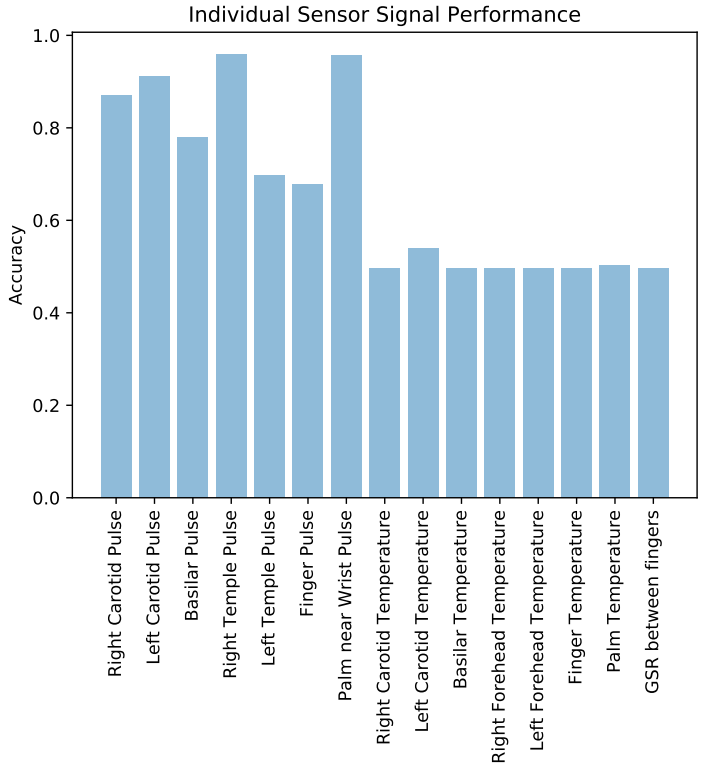


Figure 7.16: Individual sensor signal classification performance for Dataset 1.

7.7 Optimizing the Pain Meter Design

One of the benefits of our deep learning framework is that it can distinguish which sensors are most useful for pain score assessment. We compare the performance of each of the separate signals in Pain Meter 1 as shown in Fig. 7.16. We report the average accuracy using 5-fold cross validation. This can help us to further improve our chronic Pain Meter. By testing individual signal performance, we find that the accuracy for temperature signals are all around 0.5, which is similar to a random guess in binary classification. Thus the temperature signals are not very informative. We also find that the temple pulse and the palm near wrist pulse contribute substantially to our pain score prediction.

7.8 Discussion

We have addressed the issue of predicting a chronic pain score by proposing a deep learning ordinal regression framework. We split the long recordings into smaller slices, which not only eases the burden on GPU memory but also provides more training samples for the deep learning model. We define and use Consensus Prediction during testing. We present the Confusion Matrix of leave-one-recording-out in Fig. 7.17. Leave-one-recording-out does not perform well due to subjects' differing perceptions of pain intensities. Also since we have much more pain score 2 samples compared with other pain score samples in Dataset 2, as is shown in Fig. 7.5, the model is prone to make predictions of pain score 2.

This work is a proof of principle for chronic pain score assessment via deep learning. It can provide an objective pain assessment for each patient. More data for additional chronic pain subjects is needed before it can be definitively known if deep learning will

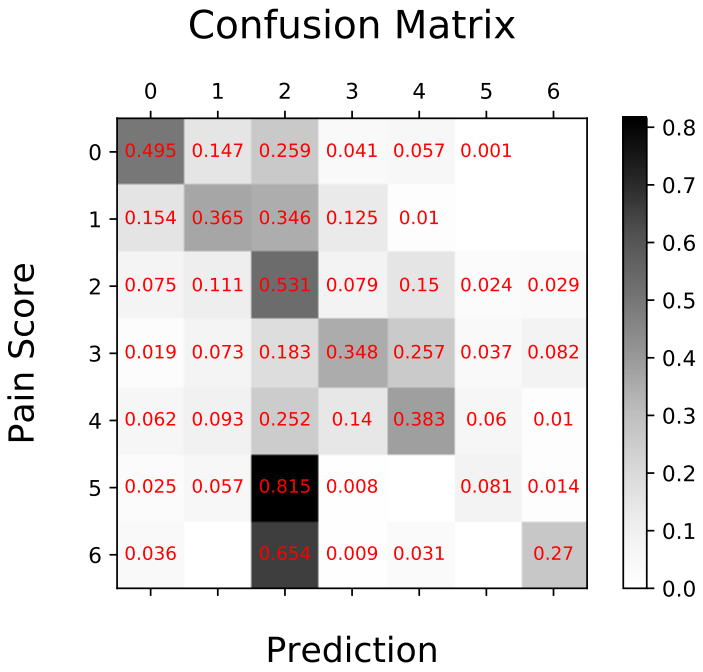


Figure 7.17: Confusion matrix for chronic pain score prediction in Dataset 2 using leave one recording out. The performance is poor due to subjects' differing sensitivities to pain, the use of different settings while recording and the limited unbalanced dataset.

be a generally useful technique for chronic pain assessment.

Chapter 8

Conclusion and Future Work

In this dissertation we have demonstrated our use of data-driven approaches for knowledge mining from MEA recordings and healthcare data.

In MEA recording mining, we proposed a deep learning framework for MEA classification on mouse and human derived induced Pluripotent Stem Cell recordings. We also introduced a scalable Bayesian framework for inference of functional networksfrom MEA data.

In healthcare data mining, we first performed quantitative analysis on early MOF prediction with comprehensive machine learning (ML) configurations. Then we introduced BERTSurv to include clinical notes in addition to measurements for mortality prediction and survival analysis. We also proposed Classifier-GAIN for MOF prediction to deal with missing data issue, by incorporating both observed data and label information. Finally we proposed an end-to-end deep learning framework for chronic pain score assessment.

Future work could include collecting more high-quality data, building more tools that can best provide real-time assessment of subjects, improving the accuracy and efficiency of the current tools, and in some cases providing a mechanism for bio-feedback.

Bibliography

[1] Y. Zhao, E. Guzman, M. Audouard, Z. Cheng, P. Hansma, K. S. Kosik, and L. Petzold, *A deep learning framework for classification of in vitro multi-electrode array recordings*, *arXiv preprint arXiv:1906.02241* (2019).

[2] Y. Zhao, R. Jiang, Z. Xu, E. Guzman, P. K. Hansma, and L. Petzold, *Scalable bayesian functional connectivity inference for multi-electrode array recordings*, *arXiv preprint arXiv:2007.02198* (2020).

[3] Y. Wang*, Y. Zhao*, R. Callcut, and L. Petzold, *Empirical analysis of machine learning configurations for prediction of multiple organ failure in trauma patients*, *arXiv preprint arXiv:2103.10929* (2021).

[4] Y. Zhao, Q. Hong, X. Zhang, Y. Deng, Y. Wang, and L. Petzold, *Bertsurv: Bert-based survival models for predicting outcomes of trauma patients*, *arXiv preprint arXiv:2103.10928* (2021).

[5] Y. Zhao, F. Ly, Q. Hong, Z. Cheng, T. Santander, H. T. Yang, P. K. Hansma, and L. Petzold, *How much does it hurt: A deep learning framework for chronic pain score assessment*, *arXiv preprint arXiv:2009.12202* (2020).

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, *Advances in neural information processing systems* **25** (2012) 1097–1105.

[7] A. Graves, A.-r. Mohamed, and G. Hinton, *Speech recognition with deep recurrent neural networks*, in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649, Ieee, 2013.

[8] R. Collobert and J. Weston, *A unified architecture for natural language processing: Deep neural networks with multitask learning*, in *Proceedings of the 25th international conference on Machine learning*, pp. 160–167, 2008.

[9] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et. al.*, *Mastering the game of go without human knowledge*, *nature* **550** (2017), no. 7676 354–359.

[10] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball, *Deep learning with convolutional neural networks for eeg decoding and visualization*, *Human brain mapping* **38** (2017), no. 11 5391–5420.

[11] K. Van Leeuwen, H. Sun, M. Tabaeizadeh, A. Struck, M. Van Putten, and M. Westover, *Detecting abnormal electroencephalograms using deep convolutional networks*, *Clinical neurophysiology* **130** (2019), no. 1 77–84.

[12] A. J. Kell, D. L. Yamins, E. N. Shook, S. V. Norman-Haignere, and J. H. McDermott, *A task-optimized neural network replicates human auditory behavior, predicts brain responses, and reveals a cortical processing hierarchy*, *Neuron* **98** (2018), no. 3 630–644.

[13] A. P. Buccino, T. V. Ness, G. T. Einevoll, G. Cauwenberghs, and P. D. Häfliger, *Localizing neuronal somata from multi-electrode array in-vivo recordings using deep learning*, in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 974–977, IEEE, 2017.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, *Proceedings of the IEEE* **86** (1998), no. 11 2278–2324.

[15] A. P. Buccino, M. Kordovan, T. V. Ness, B. Merkt, P. D. Häfliger, M. Fyhn, G. Cauwenberghs, S. Rotter, and G. T. Einevoll, *Combining biophysical modeling and deep learning for multielectrode array neuron localization and classification*, *Journal of neurophysiology* **120** (2018), no. 3 1212–1232.

[16] A. P. Buccino, T. V. Ness, G. T. Einevoll, G. Cauwenberghs, and P. D. Häfliger, *A deep learning approach for the classification of neuronal cell types*, in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 999–1002, IEEE, 2018.

[17] M. E. J. Obien, K. Deligkaris, T. Bullmann, D. J. Bakkum, and U. Frey, *Revealing neuronal function through microelectrode array recordings*, *Frontiers in neuroscience* **8** (2015) 423.

[18] T. N. Turner, K. Sharma, E. C. Oh, Y. P. Liu, R. L. Collins, M. X. Sosa, D. R. Auer, H. Brand, S. J. Sanders, D. Moreno-De-Luca, *et. al.*, *Loss of δ-catenin function in severe autism*, *Nature* **520** (2015), no. 7545 51–56.

[19] C. Matter, M. Pribadi, X. Liu, and J. T. Trachtenberg, *δ-catenin is required for the maintenance of neural structure and function in mature cortex in vivo*, *Neuron* **64** (2009), no. 3 320–327.

[20] K. S. Kosik, C. P. Donahue, I. Israely, X. Liu, and T. Ochiishi, *δ-catenin at the synaptic–adherens junction*, *Trends in cell biology* **15** (2005), no. 3 172–178.

[21] M. A. Lalli, J. Jang, J.-H. C. Park, Y. Wang, E. Guzman, H. Zhou, M. Audouard, D. Bridges, K. R. Tovar, S. M. Papuc, *et. al.*, *Haploinsufficiency of baz1b contributes to williams syndrome through transcriptional dysregulation of neurodevelopmental pathways*, *Human molecular genetics* **25** (2016), no. 7 1294–1306.

[22] M. Bayés, L. F. Magano, N. Rivera, R. Flores, and L. A. P. Jurado, *Mutational mechanisms of williams-beuren syndrome deletions*, *The American Journal of Human Genetics* **73** (2003), no. 1 131–151.

[23] C. A. Morris, H. M. Lenhoff, and P. P. Wang, *Williams-Beuren syndrome: Research, evaluation, and treatment*. JHU Press, 2006.

[24] K. R. Tovar and G. L. Westbrook, *Amino-terminal ligands prolong nmda receptor-mediated epscs*, *Journal of Neuroscience* **32** (2012), no. 23 8065–8073.

[25] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, *Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering*, *Neural computation* **16** (2004), no. 8 1661–1687.

[26] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, *The journal of machine learning research* **15** (2014), no. 1 1929–1958.

[28] L. Prechelt, *Early stopping-but when?*, in *Neural Networks: Tricks of the trade*, pp. 55–69. Springer, 1998.

[29] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, *arXiv preprint arXiv:1412.6980* (2014).

[30] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et. al.*, *Tensorflow: Large-scale machine learning on heterogeneous distributed systems*, *arXiv preprint arXiv:1603.04467* (2016).

[31] H. Wu and X. Gu, *Towards dropout training for convolutional neural networks*, *Neural Networks* **71** (2015) 1–10.

[32] J. Duchi, E. Hazan, and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization.*, *Journal of machine learning research* **12** (2011), no. 7.

[33] M. D. Zeiler, *Adadelta: an adaptive learning rate method*, *arXiv preprint arXiv:1212.5701* (2012).

[34] T. Tieleman and G. Hinton, *Lecture 6.5-rmsprop, coursera: Neural networks for machine learning*, *University of Toronto, Technical Report* (2012).

[35] M. F. Bear, B. W. Connors, and M. A. Paradiso, *Neuroscience*, vol. 2. Lippincott Williams & Wilkins, 2007.

[36] K. J. Friston, *Functional and effective connectivity in neuroimaging: a synthesis*, *Human brain mapping* **2** (1994), no. 1-2 56–78.

[37] A. Baldassarre, C. M. Lewis, G. Committeri, A. Z. Snyder, G. L. Romani, and M. Corbetta, *Individual variability in functional connectivity predicts performance of a perceptual task*, *Proceedings of the National Academy of Sciences* **109** (2012), no. 9 3516–3521.

[38] A. Gazzaley, J. Rissman, and M. Desposito, *Functional connectivity during working memory maintenance*, *Cognitive, Affective, & Behavioral Neuroscience* **4** (2004), no. 4 580–599.

[39] R.-A. Müller, P. Shih, B. Keehn, J. R. Deyoe, K. M. Leyden, and D. K. Shukla, *Underconnected, but how? a survey of functional connectivity mri studies in autism spectrum disorders*, *Cerebral cortex* **21** (2011), no. 10 2233–2243.

[40] R. Prevedel, Y.-G. Yoon, M. Hoffmann, N. Pak, G. Wetzstein, S. Kato, T. Schrödel, R. Raskar, M. Zimmer, E. S. Boyden, *et. al.*, *Simultaneous whole-animal 3d imaging of neuronal activity using light-field microscopy*, *Nature methods* **11** (2014), no. 7 727.

[41] M.-G. Liu, X.-F. Chen, T. He, Z. Li, and J. Chen, *Use of multi-electrode array recordings in studies of network synaptic plasticity in both time and space*, *Neuroscience bulletin* **28** (2012), no. 4 409–422.

[42] C. M. Lewis, C. A. Bosman, and P. Fries, *Recording of brain activity across spatial scales*, *Current opinion in neurobiology* **32** (2015) 68–77.

[43] I. D. Ruz and S. R. Schultz, *Localising and classifying neurons from high density mea recordings*, *Journal of neuroscience methods* **233** (2014) 115–128.

[44] M.-G. Liu, X.-F. Chen, T. He, Z. Li, and J. Chen, *Use of multi-electrode array recordings in studies of network synaptic plasticity in both time and space*, *Neuroscience bulletin* **28** (2012), no. 4 409–422.

[45] M. Garofalo, T. Nieus, P. Massobrio, and S. Martinoia, *Evaluation of the performance of information theory-based methods and cross-correlation to estimate the functional connectivity in cortical networks*, *PloS one* **4** (2009), no. 8 e6482.

[46] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti, *Detecting novel associations in large data sets*, *science* **334** (2011), no. 6062 1518–1524.

[47] Y. N. Billeh, M. T. Schaub, C. A. Anastassiou, M. Barahona, and C. Koch, *Revealing cell assemblies at multiple levels of granularity*, *Journal of neuroscience methods* **236** (2014) 92–106.

[48] H. Liu and B. Wu, *Active learning of functional networks from spike trains*, in *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 81–89, SIAM, 2017.

[49] S. Linderman, R. P. Adams, and J. W. Pillow, *Bayesian latent structure discovery from multi-neuron recordings*, in *Advances in neural information processing systems*, pp. 2002–2010, 2016.

[50] K. R. Tovar, D. C. Bridges, B. Wu, C. Randall, M. Audouard, J. Jang, P. K. Hansma, and K. S. Kosik, *Action potential propagation recorded from single axonal arbors using multielectrode arrays*, *Journal of neurophysiology* **120** (2018), no. 7 306–320.

[51] F. J. Chaure, H. G. Rey, and R. Quian Quiroga, *A novel and fully automatic spike-sorting implementation with variable number of features*, *Journal of neurophysiology* **120** (2018), no. 4 1859–1871.

[52] Z. Chen, *An overview of bayesian methods for neural spike train analysis*, *Computational intelligence and neuroscience* **2013** (2013).

[53] J. E. Chung, J. F. Magland, A. H. Barnett, V. M. Tolosa, A. C. Tooker, K. Y. Lee, K. G. Shah, S. H. Felix, L. M. Frank, and L. F. Greengard, *A fully automated approach to spike sorting*, *Neuron* **95** (2017), no. 6 1381–1394.

[54] C. Luo, J. Zhan, X. Xue, L. Wang, R. Ren, and Q. Yang, *Cosine normalization: Using cosine similarity instead of dot product in neural networks*, in *International Conference on Artificial Neural Networks*, pp. 382–391, Springer, 2018.

[55] D. C. Bridges, K. R. Tovar, B. Wu, P. K. Hansma, and K. S. Kosik, *Mea viewer: A high-performance interactive application for visualizing electrophysiological data*, *PloS one* **13** (2018), no. 2 e0192477.

[56] J. J. V. Branca, G. Morucci, and A. Pacini, *Cadmium-induced neurotoxicity: still much ado*, *Neural regeneration research* **13** (2018), no. 11 1879.

[57] M. J. Berridge, P. Lipp, and M. D. Bootman, *The versatility and universality of calcium signalling*, *Nature reviews Molecular cell biology* **1** (2000), no. 1 11–21.

[58] G. Choong, Y. Liu, and D. M. Templeton, *Interplay of calcium and cadmium in mediating cadmium toxicity*, *Chemico-biological interactions* **211** (2014) 54–65.

[59] E. Salinas and T. J. Sejnowski, *Correlated neuronal activity and the flow of neural information*, *Nature reviews neuroscience* **2** (2001), no. 8 539–550.

[60] J. W. Pillow, J. Shlens, L. Paninski, A. Sher, A. M. Litke, E. Chichilnisky, and E. P. Simoncelli, *Spatio-temporal correlations and visual signalling in a complete neuronal population*, *Nature* **454** (2008), no. 7207 995–999.

[61] L. Paninski, *Maximum likelihood estimation of cascade point-process neural encoding models*, *Network: Computation in Neural Systems* **15** (2004), no. 4 243–262.

[62] W. Truccolo, U. T. Eden, M. R. Fellows, J. P. Donoghue, and E. N. Brown, *A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects*, *Journal of neurophysiology* **93** (2005), no. 2 1074–1089.

[63] T. Parr, G. Rees, and K. J. Friston, *Computational neuropsychology and bayesian inference*, *Frontiers in human neuroscience* **12** (2018) 61.

[64] V.-P. Harjola, W. Mullens, M. Banaszewski, J. Bauersachs, H.-P. Brunner-La Rocca, O. Chioncel, S. P. Collins, W. Doehner, G. S. Filippatos, A. J. Flammer, *et. al.*, *Organ dysfunction, injury and failure in acute heart failure: from pathophysiology to diagnosis and management. a review on behalf of the acute heart failure committee of the heart failure association (hfa) of the european society of cardiology (esc)*, *European journal of heart failure* **19** (2017), no. 7 821–836.

[65] Z.-K. Wang, R.-J. Chen, S.-L. Wang, G.-W. Li, Z.-Z. Zhu, Q. Huang, Z.-L. Chen, F.-C. Chen, L. Deng, X.-P. Lan, *et. al.*, *Clinical application of a novel diagnostic scheme including pancreatic β-cell dysfunction for traumatic multiple organ dysfunction syndrome*, *Molecular medicine reports* **17** (2018), no. 1 683–693.

[66] R. M. Durham, J. Moran, J. E. Mazuski, M. J. Shapiro, A. E. Baue, and L. M. Flint, *Multiple organ failure in trauma patients*, *Journal of Trauma and Acute Care Surgery* **55** (2003), no. 4 608–616.

[67] A. Ulvik, R. Kvåle, T. Wentzel-Larsen, and H. Flaatten, *Multiple organ failure after trauma affects even long-term survival and functional status*, *Critical Care* **11** (2007), no. 5 1–8.

[68] P. S. Barie, L. J. Hydo, and E. Fischer, *A prospective comparison of two multiple organ dysfunction/failure scoring systems for prediction of mortality in critical surgical illness.*, *The Journal of trauma* **37** (1994), no. 4 660–666.

[69] D. P. Bota, C. Melot, F. L. Ferreira, V. N. Ba, and J.-L. Vincent, *The multiple organ dysfunction score (mods) versus the sequential organ failure assessment (sofa) score in outcome prediction*, *Intensive care medicine* **28** (2002), no. 11 1619–1624.

[70] D. C. Dewar, A. White, J. Attia, S. M. Tarrant, K. L. King, and Z. J. Balogh, *Comparison of postinjury multiple-organ failure scoring systems: Denver versus sequential organ failure assessment*, *Journal of trauma and acute care surgery* **77** (2014), no. 4 624–629.

[71] L. Hutchings, P. Watkinson, J. D. Young, and K. Willett, *Defining multiple organ failure after major trauma: a comparison of the denver, sequential organ failure assessment and marshall scoring systems*, *The journal of trauma and acute care surgery* **82** (2017), no. 3 534.

[72] A. Sauaia, F. A. Moore, E. E. Moore, J. M. Norris, D. C. Lezotte, and R. F. Hamman, *Multiple organ failure can be predicted as early as 12 hours after injury*, *Journal of Trauma and Acute Care Surgery* **45** (1998), no. 2 291–303.

[73] J. A. Vogel, M. M. Liao, E. Hopkins, N. Seleno, R. L. Byyny, E. E. Moore, C. Gravitz, and J. S. Haukoos, *Prediction of postinjury multiple-organ failure in the emergency department: development of the denver emergency department trauma organ failure score*, *The journal of trauma and acute care surgery* **76** (2014), no. 1 140.

[74] Z. Obermeyer and E. J. Emanuel, *Predicting the futurebig data, machine learning, and clinical medicine*, *The New England journal of medicine* **375** (2016), no. 13 1216.

[75] J. A. Cruz and D. S. Wishart, *Applications of machine learning in cancer prediction and prognosis*, *Cancer informatics* **2** (2006) 117693510600200030.

[76] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, *Machine learning applications in cancer prognosis and prediction*, *Computational and structural biotechnology journal* **13** (2015) 8–17.

[77] H. Asri, H. Mousannif, H. Al Moatassime, and T. Noel, *Using machine learning algorithms for breast cancer risk prediction and diagnosis*, *Procedia Computer Science* **83** (2016) 1064–1069.

[78] K. Sharma, A. Kaur, and S. Gujral, *Brain tumor detection based on machine learning algorithms*, *International Journal of Computer Applications* **103** (2014), no. 1 7–11.

[79] Z. Wang, G. Yu, Y. Kang, Y. Zhao, and Q. Qu, *Breast tumor detection in digital mammography based on extreme learning machine*, *Neurocomputing* **128** (2014) 175–184.

[80] M. De Bruijne, *Machine learning approaches in medical image analysis: From detection to diagnosis*, 2016.

[81] C. R. Farrar and K. Worden, *Structural health monitoring: a machine learning perspective*. John Wiley & Sons, 2012.

[82] K. Worden and G. Manson, *The application of machine learning to structural health monitoring*, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **365** (2007), no. 1851 515–537.

[83] Z. Ahmed, K. Mohamed, S. Zeeshan, and X. Dong, *Artificial intelligence with multi-functional machine learning platform development for better healthcare and precision medicine*, *Database* **2020** (2020).

[84] K. J. Janssen, A. R. T. Donders, F. E. Harrell Jr, Y. Vergouwe, Q. Chen, D. E. Grobbee, and K. G. Moons, *Missing covariate data in medical research: to impute is better than to ignore*, *Journal of clinical epidemiology* **63** (2010), no. 7 721–727.

[85] E. Tuba, I. Strumberger, T. Bezdan, N. Bacanin, and M. Tuba, *Classification and feature selection method for medical datasets by brain storm optimization algorithm and support vector machine*, *Procedia Computer Science* **162** (2019) 307–315.

[86] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, *Smote: synthetic minority over-sampling technique*, *Journal of artificial intelligence research* **16** (2002) 321–357.

[87] I. Mani and I. Zhang, *knn approach to unbalanced data distributions: a case study involving information extraction*, in *Proceedings of workshop on learning from imbalanced datasets*, vol. 126, ICML United States, 2003.

[88] G. E. Batista, A. L. Bazzan, M. C. Monard, *et. al.*, *Balancing training data for automated annotation of keywords: a case study.*, in *WOB*, pp. 10–18, 2003.

[89] H. Dağ, K. Sayin, I. Yenidoğan, S. Albayrak, and C. Acar, *Comparison of feature selection algorithms for medical data*, in *2012 International Symposium on Innovations in Intelligent Systems and Applications*, pp. 1–5, IEEE, 2012.

[90] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et. al.*, *Scikit-learn: Machine learning in python*, *the Journal of machine Learning research* **12** (2011) 2825–2830.

[91] J. Bakker, P. Gris, M. Coffernils, R. J. Kahn, and J.-L. Vincent, *Serial blood lactate levels can predict the development of multiple organ failure following septic shock*, *The American journal of surgery* **171** (1996), no. 2 221–226.

[92] G. I. Papachristou, V. Muddana, D. Yadav, M. O'connell, M. K. Sanders, A. Slivka, and D. C. Whitcomb, *Comparison of bisap, ranson's, apache-ii, and ctsi scores in predicting organ failure, complications, and mortality in acute pancreatitis*, *American Journal of Gastroenterology* **105** (2010), no. 2 435–441.

[93] W. Chen, T.-Y. Liu, Y. Lan, Z.-M. Ma, and H. Li, *Ranking measures and loss functions in learning to rank*, *Advances in Neural Information Processing Systems* **22** (2009) 315–323.

[94] J. Rossaint and A. Zarbock, *Pathogenesis of multiple organ failure in sepsis*, *Critical Reviews in Immunology* **35** (2015), no. 4.

[95] V.-P. Harjola, W. Mullens, M. Banaszewski, J. Bauersachs, H.-P. Brunner-La Rocca, O. Chioncel, S. P. Collins, W. Doehner, G. S. Filippatos, A. J. Flammer, *et. al.*, *Organ dysfunction, injury and failure in acute heart failure: from pathophysiology to diagnosis and management. a review on behalf of the acute heart failure committee of the heart failure association (hfa) of the european society of cardiology (esc)*, *European journal of heart failure* **19** (2017), no. 7 821–836.

[96] Z.-K. Wang, R.-J. Chen, S.-L. Wang, G.-W. Li, Z.-Z. Zhu, Q. Huang, Z.-L. Chen, F.-C. Chen, L. Deng, X.-P. Lan, *et. al.*, *Clinical application of a novel diagnostic scheme including pancreatic β-cell dysfunction for traumatic multiple organ dysfunction syndrome*, *Molecular Medicine Reports* **17** (2018), no. 1 683–693.

[97] R. M. Durham, J. Moran, J. E. Mazuski, M. J. Shapiro, A. E. Baue, and L. M. Flint, *Multiple organ failure in trauma patients*, *Journal of Trauma and Acute Care Surgery* **55** (2003), no. 4 608–616.

[98] M. J. Otero-López, P. Alonso-Hernández, J. A. Maderuelo-Fernández, B. Garrido-Corro, A. Domínguez-Gil, and A. Sánchez-Rodríguez, *Preventable adverse drug events in hospitalized patients*, *Medicina clinica* **126** (2006), no. 3 81–87.

[99] Y. Zhang, T. B. Wu, B. J. Daigle, M. Cohen, and L. Petzold, *Identification of disease states associated with coagulopathy in trauma*, *BMC medical informatics and decision making* **16** (2016), no. 1 1–9.

[100] E. Acuna and C. Rodriguez, *The treatment of missing values and its effect on classifier accuracy*, in *Classification, clustering, and data mining applications*, pp. 639–647. Springer, 2004.

[101] S. v. Buuren and K. Groothuis-Oudshoorn, *mice: Multivariate imputation by chained equations in r*, *Journal of statistical software* (2010) 1–68.

[102] J. Yoon, J. Jordon, and M. Van Der Schaar, *Gain: Missing data imputation using generative adversarial nets*, *arXiv preprint arXiv:1806.02920* (2018).

[103] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, *Advances in neural information processing systems* **27** (2014) 2672–2680.

[104] M. Mirza and S. Osindero, *Conditional generative adversarial nets*, *arXiv preprint arXiv:1411.1784* (2014).

[105] A. Odena, C. Olah, and J. Shlens, *Conditional image synthesis with auxiliary classifier gans*, in *International conference on machine learning*, pp. 2642–2651, PMLR, 2017.

[106] A. Odena, *Semi-supervised learning with generative adversarial networks*, *arXiv preprint arXiv:1606.01583* (2016).

[107] C. Li, T. Xu, J. Zhu, and B. Zhang, *Triple generative adversarial nets*, *Advances in neural information processing systems* **30** (2017) 4088–4098.

[108] L. Bravo-Merodio, A. Acharjee, J. Hazeldine, C. Bentley, M. Foster, G. V. Gkoutos, and J. M. Lord, *Machine learning for the detection of early immunological markers as predictors of multi-organ dysfunction*, *Scientific data* **6** (2019), no. 1 1–10.

[109] M. A. Reyna, C. Josef, S. Seyedi, R. Jeter, S. P. Shashikumar, M. B. Westover, A. Sharma, S. Nemati, and G. D. Clifford, *Early prediction of sepsis from clinical data: the physionet/computing in cardiology challenge 2019*, in *2019 Computing in Cardiology (CinC)*, pp. Page–1, IEEE, 2019.

[110] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, *Smote: synthetic minority over-sampling technique*, *Journal of artificial intelligence research* **16** (2002) 321–357.

[111] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et. al.*, *Scikit-learn: Machine learning in python*, *the Journal of machine Learning research* **12** (2011) 2825–2830.

[112] J. Opitz and S. Burst, *Macro f1 and macro f1*, *arXiv preprint arXiv:1911.03347* (2019).

[113] J. Bakker, P. Gris, M. Coffernils, R. J. Kahn, and J.-L. Vincent, *Serial blood lactate levels can predict the development of multiple organ failure following septic shock*, *The American journal of surgery* **171** (1996), no. 2 221–226.

[114] G. I. Papachristou, V. Muddana, D. Yadav, M. O'connell, M. K. Sanders, A. Slivka, and D. C. Whitcomb, *Comparison of bisap, ranson's, apache-ii, and ctsi scores in predicting organ failure, complications, and mortality in acute pancreatitis*, *American Journal of Gastroenterology* **105** (2010), no. 2 435–441.

[115] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et. al.*, *Pytorch: An imperative style, high-performance deep learning library*, in *Advances in neural information processing systems*, pp. 8026–8037, 2019.

[116] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, *arXiv preprint arXiv:1412.6980* (2014).

[117] D. Radenkovic, D. Bajec, N. Ivancevic, N. Milic, V. Bumbasirevic, V. Jeremic, V. Djukic, B. Stefanovic, B. Stefanovic, G. Milosevic-Zbutega, *et. al.*, *D-dimer in acute pancreatitis: a new approach for an early assessment of organ failure*, *Pancreas* **38** (2009), no. 6 655–660.

[118] S. Wei, *The assessment of factor viii-related antigen in endothelial cells of pulmonary blood vessels in multiple organ failure*, *Zhonghua jie he he hu xi za zhi= Zhonghua jiehe he huxi zazhi= Chinese journal of tuberculosis and respiratory diseases* **13** (1990), no. 6 346–8.

[119] L. Del Sorbo and A. S. Slutsky, *Acute respiratory distress syndrome and multiple organ failure*, *Current opinion in critical care* **17** (2011), no. 1 1–6.

[120] S. C.-X. Li, B. Jiang, and B. Marlin, *Misgan: Learning from incomplete data with generative adversarial networks*, *arXiv preprint arXiv:1902.09599* (2019).

[121] K. Brown, S. Brain, J. Pearson, J. Edgeworth, S. Lewis, and D. Treacher, *Neutrophils in development of multiple organ failure in sepsis*, *The Lancet* **368** (2006), no. 9530 157–169.

[122] A. Ulvik, R. Kvåle, T. Wentzel-Larsen, and H. Flaatten, *Multiple organ failure after trauma affects even long-term survival and functional status*, *Critical Care* **11** (2007), no. 5 R95.

[123] T. Gustot, *Multiple organ failure in sepsis: prognosis and role of systemic inflammatory response*, *Current opinion in critical care* **17** (2011), no. 2 153–159.

[124] Q. Qiu, Y.-j. Nian, Y. Guo, L. Tang, N. Lu, L.-z. Wen, B. Wang, D.-f. Chen, and K.-j. Liu, *Development and validation of three machine-learning models for predicting multiple organ failure in moderately severe and severe acute pancreatitis*, *BMC gastroenterology* **19** (2019), no. 1 1–9.

[125] R. Kamaleswaran, O. Akbilgic, M. A. Hallman, A. N. West, R. L. Davis, and S. H. Shah, *Applying artificial intelligence to identify physiomarkers predicting severe sepsis in the picu*, *Pediatric Critical Care Medicine— Society of Critical Care Medicine* **19** (2018), no. 10 e495–e503.

[126] B. J. Wells, K. M. Chagin, A. S. Nowacki, and M. W. Kattan, *Strategies for handling missing data in electronic health record derived data*, *Egems* **1** (2013), no. 3.

[127] W. Wothke, *Longitudinal and multigroup modeling with missing data.*, .

[128] P. E. McKnight, K. M. McKnight, S. Sidani, and A. J. Figueredo, *Missing data: A gentle introduction*. Guilford Press, 2007.

[129] J. W. Graham, *Missing data analysis: Making it work in the real world*, *Annual review of psychology* **60** (2009) 549–576.

[130] M. Kantardzic, *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons, 2011.

[131] F. E. Harrell Jr, *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis*. Springer, 2015.

[132] C. Zhang, V. Maroufy, B. Chen, and H. Wu, *Missing data issues in ehr*, *Statistics and Machine Learning Methods for EHR Data: From Data Extraction to Data Analytics* (2020) 149.

[133] C. for Disease Control, Prevention, *et. al.*, "Wisqars data visualization."

[134] Y. Zhang, R. Jiang, and L. Petzold, *Survival topic models for predicting outcomes for trauma patients*, in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pp. 1497–1504, IEEE, 2017.

[135] D. R. Cox, *Regression models and life-tables*, *Journal of the Royal Statistical Society: Series B (Methodological)* **34** (1972), no. 2 187–202.

[136] D. Faraggi and R. Simon, *A neural network model for survival data*, *Statistics in medicine* **14** (1995), no. 1 73–82.

[137] J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger, *Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network*, *BMC medical research methodology* **18** (2018), no. 1 1–12.

[138] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, *arXiv preprint arXiv:1810.04805* (2018).

[139] H. R. Darabi, D. Tsinis, K. Zecchini, W. F. Whitcomb, and A. Liss, *Forecasting mortality risk for patients admitted to intensive care units using machine learning*, *Procedia Computer Science* **140** (2018) 306–313.

[140] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, *Biobert: a pre-trained biomedical language representation model for biomedical text mining*, *Bioinformatics* **36** (2020), no. 4 1234–1240.

[141] L. Rasmy, Y. Xiang, Z. Xie, C. Tao, and D. Zhi, *Med-bert: pre-trained contextualized embeddings on large-scale structured electronic health records for disease prediction*, *arXiv preprint arXiv:2005.12833* (2020).

[142] J. Shang, T. Ma, C. Xiao, and J. Sun, *Pre-training of graph augmented transformers for medication recommendation*, *arXiv preprint arXiv:1906.00346* (2019).

[143] E. Alsentzer, J. R. Murphy, W. Boag, W.-H. Weng, D. Jin, T. Naumann, and M. McDermott, *Publicly available clinical bert embeddings*, *arXiv preprint arXiv:1904.03323* (2019).

[144] K. Huang, J. Altosaar, and R. Ranganath, *Clinicalbert: Modeling clinical notes and predicting hospital readmission*, *arXiv preprint arXiv:1904.05342* (2019).

[145] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-Wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, *Mimic-iii, a freely accessible critical care database*, *Scientific data* **3** (2016), no. 1 1–9.

[146] W. A. Knaus, D. P. Wagner, E. A. Draper, J. E. Zimmerman, M. Bergner, P. G. Bastos, C. A. Sirio, D. J. Murphy, T. Lotring, A. Damiano, *et. al.*, *The apache iii prognostic system: risk prediction of hospital mortality for critically iii hospitalized adults*, *Chest* **100** (1991), no. 6 1619–1636.

[147] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et. al.*, *Pytorch: An imperative style, high-performance deep learning library*, *arXiv preprint arXiv:1912.01703* (2019).

[148] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et. al.*, *Google's neural machine translation system: Bridging the gap between human and machine translation*, *arXiv preprint arXiv:1609.08144* (2016).

[149] J. P. Klein and M. L. Moeschberger, *Survival analysis: techniques for censored and truncated data*. Springer Science & Business Media, 2006.

[150] J. D. Kalbfleisch and R. L. Prentice, *The statistical analysis of failure time data*, vol. 360. John Wiley & Sons, 2011.

[151] R. Prentice and N. Breslow, *Retrospective studies and failure time models*, *Biometrika* **65** (1978), no. 1 153–158.

[152] H. Hung and C.-T. Chiang, *Estimation methods for time-dependent auc models with survival data*, *Canadian Journal of Statistics* **38** (2010), no. 1 8–26.

[153] L. E. Chambless, C. P. Cummiskey, and G. Cui, *Several methods to assess improvement in risk prediction models: extension to survival analysis*, *Statistics in medicine* **30** (2011), no. 1 22–38.

[154] F. E. Harrell Jr, K. L. Lee, R. M. Califf, D. B. Pryor, and R. A. Rosati, *Regression modelling strategies for improved prognostic prediction*, *Statistics in medicine* **3** (1984), no. 2 143–152.

[155] A. Avati, K. Jung, S. Harman, L. Downing, A. Ng, and N. H. Shah, *Improving palliative care with deep learning*, *BMC medical informatics and decision making* **18** (2018), no. 4 122.

[156] J. Dahlhamer, J. Lucas, C. Zelaya, R. Nahin, S. Mackey, L. DeBar, R. Kerns, M. Von Korff, L. Porter, and C. Helmick, *Prevalence of chronic pain and high-impact chronic pain among adultsunited states, 2016*, *Morbidity and Mortality Weekly Report* **67** (2018), no. 36 1001.

[157] D. J. Gaskin and P. Richard, *The economic costs of pain in the united states*, *The Journal of Pain* **13** (2012), no. 8 715–724.

[158] M. M. van der Miesen, M. A. Lindquist, and T. D. Wager, *Neuroimaging-based biomarkers for pain: state of the field and current directions*, *Pain reports* **4** (2019), no. 4.

[159] M. C. Reddan and T. D. Wager, *Brain systems at the intersection of chronic pain and self-regulation*, *Neuroscience letters* **702** (2019) 24–33.

[160] M. C. Reddan and T. D. Wager, *Modeling pain using fmri: from regions to biomarkers*, *Neuroscience bulletin* **34** (2018), no. 1 208–215.

[161] R. Cowen, M. K. Stasiowska, H. Laycock, and C. Bantel, *Assessing pain objectively: the use of physiological markers*, *Anaesthesia* **70** (2015), no. 7 828–847.

[162] Y. L. Yang, H. S. Seok, G.-J. Noh, B.-M. Choi, and H. Shin, *Postoperative pain assessment indices based on photoplethysmography waveform analysis*, *Frontiers in physiology* **9** (2018) 1199.

[163] L. S. Prichep, J. Shah, H. Merkin, and E. M. Hiesiger, *Exploration of the pathophysiology of chronic pain using quantitative eeg source localization*, *Clinical EEG and neuroscience* **49** (2018), no. 2 103–113.

[164] H. Liu, Y. Wang, and L. Wang, *A review of non-contact, low-cost physiological information measurement based on photoplethysmographic imaging*, in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 2088–2091, IEEE, 2012.

[165] J. A. Bargh and E. Morsella, *The unconscious mind*, *Perspectives on psychological science* **3** (2008), no. 1 73–79.

[166] R. Kanawade, S. Tewary, H. Sardana, *et. al.*, *Photoplethysmography based arrhythmia detection and classification*, in *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*, pp. 944–948, IEEE, 2019.

[167] R. Poplin, A. V. Varadarajan, K. Blumer, Y. Liu, M. V. McConnell, G. S. Corrado, L. Peng, and D. R. Webster, *Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning*, *Nature Biomedical Engineering* **2** (2018), no. 3 158.

[168] H. R. Darabi, D. Tsinis, K. Zecchini, W. F. Whitcomb, and A. Liss, *Forecasting mortality risk for patients admitted to intensive care units using machine learning*, *Procedia Computer Science* **140** (2018) 306–313.

[169] B. Jin, H. Yang, L. Sun, C. Liu, Y. Qu, and J. Tong, *A treatment engine by predicting next-period prescriptions*, in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1608–1616, ACM, 2018.

[170] M.-H. Hsieh, M.-J. Hsieh, C.-M. Chen, C.-C. Hsieh, C.-M. Chao, and C.-C. Lai, *An artificial neural network model for predicting successful extubation in intensive care units*, *Journal of clinical medicine* **7** (2018), no. 9 240.

[171] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, *Deep learning for healthcare: review, opportunities and challenges*, *Briefings in bioinformatics* **19** (2017), no. 6 1236–1246.

[172] P. Rodriguez, G. Cucurull, J. Gonzàlez, J. M. Gonfaus, K. Nasrollahi, T. B. Moeslund, and F. X. Roca, *Deep pain: Exploiting long short-term memory networks for facial expression classification*, *IEEE transactions on cybernetics* (2017).

[173] D. Liu, F. Peng, A. Shea, R. Picard, *et. al.*, *Deepfacelift: interpretable personalized models for automatic estimation of self-reported pain*, *arXiv preprint arXiv:1708.04670* (2017).

[174] Y. Chu, X. Zhao, J. Han, and Y. Su, *Physiological signal-based method for measurement of pain intensity*, *Frontiers in neuroscience* **11** (2017) 279.

[175] J. Lötsch and A. Ultsch, *Machine learning in pain research*, *Pain* **159** (2018), no. 4 623.

[176] M. H. Pitcher, M. Von Korff, M. C. Bushnell, and L. Porter, *Prevalence and profile of high-impact chronic pain in the united states*, *The Journal of Pain* **20** (2019), no. 2 146–160.

[177] A. E. Johnson, M. M. Ghassemi, S. Nemati, K. E. Niehaus, D. A. Clifton, and G. D. Clifford, *Machine learning and decision support in critical care*, *Proceedings of the IEEE. Institute of Electrical and Electronics Engineers* **104** (2016), no. 2 444.

[178] Z. M. Hira and D. F. Gillies, *A review of feature selection and feature extraction methods applied on microarray data*, *Advances in bioinformatics* **2015** (2015).

[179] N. Ben-Israel, M. Kliger, G. Zuckerman, Y. Katz, and R. Edry, *Monitoring the nociception level: a multi-parameter approach*, *Journal of clinical monitoring and computing* **27** (2013), no. 6 659–668.

[180] U. Rubins, Z. Marcinkevics, I. Logina, A. Grabovskis, and E. Kviesis-Kipge, *Imaging photoplethysmography for assessment of chronic pain patients*, in *Optical Diagnostics and Sensing XIX: Toward Point-of-Care Diagnostics*, vol. 10885, p. 1088508, International Society for Optics and Photonics, 2019.

[181] A. Johansson, P. Å. Öberg, and G. Sedin, *Monitoring of heart and respiratory rates in newborn infants using a new photoplethysmographic technique*, *Journal of clinical monitoring and computing* **15** (1999), no. 7-8 461–467.

[182] A. Bonissi, R. D. Labati, L. Perico, R. Sassi, F. Scotti, and L. Sparagino, *A preliminary study on continuous authentication methods for photoplethysmographic biometrics*, in *2013 IEEE Workshop on Biometric Measurements and Systems for Security and Medical Applications*, pp. 28–33, IEEE, 2013.

[183] `pulseSensor.com` Accessed May 26, 2020.

[184] I. Zuzarte, P. Indic, D. Sternad, and D. Paydarfar, *Quantifying movement in preterm infants using photoplethysmography*, *Annals of biomedical engineering* **47** (2019), no. 2 646–658.